



รายงานวิจัยฉบับสมบูรณ์

โครงการ การเพิ่มสมรรถนะและการประยุกต์ขั้นตอนวิธีพัฒนาการสำหรับ การออกแบบโครงข่ายหมุนและโครงข่ายแข็ง

โดย
สุจินต์ บุรีรัตน์ และคณะ

15 กรกฎาคม 2558

รายงานวิจัยฉบับสมบูรณ์

โครงการ

การเพิ่มสมรรถนะและการประยุกต์ขั้นตอนวิธีวิวัฒนาการสำหรับการ
การออกแบบโครงข้อหมุนและโครงข้อแข็ง

ผู้วิจัย

นาย สุจินต์ บุรีรัตน์

ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์

มหาวิทยาลัยขอนแก่น

สนับสนุนโดย

สำนักงานคณะกรรมการการอุดมศึกษาและสำนักงานกองทุนสนับสนุนการวิจัย

กิตติกรรมประกาศ

ผู้วิจัยขอขอบคุณ สำนักงานกองทุนสนับสนุนการวิจัย สำหรับทุนสนับสนุนการวิจัย และขอขอบคุณหน่วยงานต้นสังกัดสำหรับการส่งเสริมให้กิจกรรมนี้ดำเนินไปได้ด้วยดี ขอขอบคุณนักศึกษามหาวิทยาลัยที่ทำหน้าที่ผู้ช่วยวิจัยที่ดียิ่ง ท้ายนี้ขอบคุณผู้ใกล้ชิดทั้งหลายที่คอยให้กำลังใจ

บทคัดย่อ

รหัสโครงการ: BRG5580017

**ชื่อโครงการ: การเพิ่มสมรรถนะและการประยุกต์ขั้นตอนวิธีวิวัฒนาการสำหรับการออกแบบโครงข้อ
หมุนและโครงข้อแข็ง**

**ชื่อนักวิจัย และสถาบัน: นายสุจินต์ บุรีรัตน์ ภาควิชาวิศวกรรมเครื่องกล
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น**

อีเมล: sujbur@kku.ac.th

ระยะเวลาโครงการ 3 ปี

โครงถัก หรือในที่นี้คือโครงข้อหมุน (truss) และโครงข้อแข็ง (frame) เป็นโครงสร้างที่มีการประยุกต์ใช้งานในทางวิศวกรรมหลายรูปแบบ การประยุกต์ใช้โครงถักมีข้อได้เปรียบคือ มีราคาถูกและทำการก่อสร้างได้ง่าย โดยเฉพาะการก่อสร้างในพื้นที่ที่ยากในการเข้าถึง เช่น บนภูเขา เป็นต้น โครงถักสามารถใช้ในงานโครงหลังคา โครงสะพาน เสาส่งสัญญาณ เสาไฟฟ้าแรงสูง เสากังหันลม โครงของยานยนต์ โครงสร้างทางวิศวกรรมโยธา และชิ้นส่วนยานยนต์อีกมากมาย ในการออกแบบโครงถักขั้นตอนวิธีวิวัฒนาการเป็นวิธีที่นิยมใช้มากที่สุด เนื่องจากการใช้งานง่ายและได้คำตอบที่ดี อย่างไรก็ตามขั้นตอนวิธีวิวัฒนาการมีจุดด้อยคือ มีอัตราการลู่เข้าต่ำ ด้วยเหตุดังกล่าว การเพิ่มสมรรถนะของขั้นตอนวิธีเหล่านั้นจึงเป็นวัตถุประสงค์หลักของโครงการวิจัยนี้

งานวิจัยนี้ เป็นการศึกษาการประยุกต์ขั้นตอนวิธีวิวัฒนาการเพื่อการออกแบบโครงถัก ในรายละเอียดของงานสามารถแยกออกเป็นสองหมวดย่อย คือการพัฒนากระบวนการออกแบบโครงถัก และการพัฒนาสมรรถนะของขั้นตอนวิธีวิวัฒนาการทั้งแบบเป้าหมายเดียวและหลายเป้าหมาย ในระหว่างการศึกษาได้ทำการขยายการประยุกต์ใช้งานออกไปยังการออกแบบระบบทางวิศวกรรมอื่นด้วย เพื่อให้การดำเนินการวิจัยเกิดประโยชน์สูงสุด

ผู้วิจัย ได้ทำการศึกษาการเพิ่มสมรรถนะของขั้นตอนวิธีวิวัฒนาการ ด้วยเทคนิคการผสมผสานสำหรับการออกแบบโครงถักและระบบทางกลอื่นๆ ได้พัฒนาวิธี PBIL ที่ใช้เลขจำนวนจริง และผสมกับตัวดำเนินการของวิธีวิวัฒนาการเชิงอนุพันธ์ ซึ่งจะได้อัลกอริทึมที่มีสมรรถนะสูงสำหรับการหาค่าเหมาะที่สุดแบบหลายเป้าหมายของโครงถัก นอกจากนี้ยังได้ทำการประยุกต์ RPBIL-DE กับการออกแบบระบบโครงข่ายท่อน้ำประปา ซึ่งทำให้ได้ผลเฉลยที่ดีและเป็นการยืนยันสมรรถนะของวิธีผสม RPBIL-DE นอกจากนี้สำหรับกรณีของ RPBIL ได้พัฒนา local search เพื่อใช้กับวิธี RPBIL ซึ่ง

สามารถช่วยเพิ่มสมรรถนะของ RPBIL ในการหาค่าเหมาะที่สุดแบบหลายเป้าหมายของโครงถัก ดังจะเห็นได้จาก ผลการทดสอบเมื่อเปรียบเทียบกับขั้นตอนวิธีวิวัฒนาการที่มีมาก่อน นอกจากนี้ ยังได้สาธิตการประยุกต์ใช้ approximate gradient เพื่อเพิ่มสมรรถนะ MOEAs อื่นๆ ซึ่งก็แสดงให้เห็นว่า approximate gradient เป็น local search ที่ช่วยเพิ่มสมรรถนะของ MOEAs ในการหาค่าเหมาะที่สุดแบบหลายเป้าหมายของโครงถักได้

ได้ทำการศึกษาการหาค่าเหมาะที่สุดแบบตรงข้ามของโครงถัก การออกแบบลักษณะนี้มีประโยชน์คือ เมื่อปัญหาการออกแบบจริงมีพารามิเตอร์ที่ไม่สามารถระบุค่าได้อย่างชัดเจน แต่อาจสามารถระบุช่วงของค่าของพารามิเตอร์เหล่านั้นได้ ได้ทำการพัฒนาขั้นตอนวิธีวิวัฒนาการสำหรับปัญหาการหาค่าเหมาะที่สุดแบบตรงข้าม จากผลการทดสอบพบว่าวิธีที่พัฒนาขึ้นมาใหม่ เป็นวิธีที่มีประสิทธิภาพสูง สามารถใช้ในการหาค่าเหมาะที่สุดแบบตรงข้ามของโครงถักได้อย่างมีประสิทธิภาพ ได้ทำการผสมผสานวิธี real code ant colony optimisation (ACOR) กับวิธี local search ชิมเพล็กซ์ลงเขา ผลที่ได้ทำให้ ACOR มีสมรรถนะในการหาผลเฉลยการหาค่าเหมาะที่สุดแบบมีเงื่อนไขบังคับของระบบทางกลสูงขึ้น ได้ทำการผสมผสานระหว่างขั้นตอนวิธีวิวัฒนาการสองวิธีที่มีแนวคิดแตกต่างกัน คือวิธี differential evolution และวิธี teaching-learning based optimisation เพื่อการหาค่าเหมาะที่สุดของการม้วนแผ่นโลหะซึ่งเป็นปัญหาขนาดใหญ่เพราะมีจำนวนตัวแปรออกแบบมาก ผลจากการทดสอบพบว่าวิธีที่พัฒนาขึ้นมามีสมรรถนะสูงกว่าวิธีวิวัฒนาการอื่นที่มีมาก่อน

ได้ทำการวิจัยเพื่อพัฒนาระบบการออกแบบอัตโนมัติของโครงข้อหมุนและโครงข้อแข็ง ระบบการออกแบบชนิดนี้คือ การหาทอพอโลยี รูปทรงและขนาดเหมาะที่สุดพร้อมกันของโครงสร้าง การดำเนินการออกแบบลักษณะนี้สามารถทำได้โดยการใช้เทคนิคเอลิเมนต์พื้น ซึ่งในงานวิจัยนี้ได้ศึกษาทั้งในส่วนของเอลิเมนต์พื้นแบบคงตัวและเอลิเมนต์พื้นแบบ adaptive ซึ่งจากผลการวิจัยพบว่า เอลิเมนต์พื้นแบบ adaptive จะมีประสิทธิภาพในการออกแบบที่ดีกว่า แต่ก็มีปัญหาเรื่องความยากและสลับซับซ้อนของกระบวนการดังกล่าวคืออุปสรรคที่ตามมา ประโยชน์ของระบบการออกแบบนี้คือ เป็นกระบวนการที่ช่วยให้ผู้ออกแบบที่มีประสบการณ์ในการออกแบบโครงสร้างน้อย สามารถออกแบบ โครงสร้างได้ใกล้เคียงหรือแม้กระทั่งดีกว่าผู้ออกแบบที่มีประสบการณ์สูง ทั้งนี้ผู้ออกแบบต้องมีความรู้พื้นฐานในด้านการวิเคราะห์โครงสร้างและการหาค่าเหมาะที่สุดที่เพียงพอ นอกจากนี้ องค์ความรู้ส่วนหนึ่งของงานวิจัยนี้ถูกพัฒนาเป็นหลักสูตรอบรมการหาค่าเหมาะที่สุดของโครงถัก

คำหลัก: ขั้นตอนวิธีวิวัฒนาการ, การหาค่าเหมาะที่สุดของโครงสร้าง, การวิเคราะห์ทางไฟไนต์เอลิเมนต์, ขั้นตอนวิธีผสมผสาน, เอลิเมนต์พื้นแบบปรับตัวได้

Abstract

Project Code: BRG5580017

Project Title: Performance Enhancement and Applications of Evolutionary Algorithms for Truss and Frame Design

**Investigator: Sujin Bureerat, Department of Mechanical Engineering
Faculty of Engineering, Khon Kaen University**

Email: sujbur@kku.ac.th

Project Period 3 years

Trusses and frames, also known as skeleton-like structures, are some of the most commonly used structures in daily life engineering systems. Using such structures is advantageous since they are inexpensive and simple to construct. It is also easier to build this type of structures in difficult-to-access areas. The structures can be used as roof structures, bridge structures, transmission and communication towers, wind turbine towers, vehicle frames, civil engineering structure, and mechanical parts etc. With the invention and development of computer hardware and software as well as overwhelming research work towards structural design, it has been found that the use of optimisation in combination with finite element analysis greatly enhance a design process of the structures. The most popular optimisers for truss and frame design are evolutionary algorithms (EAs) due to their simplicity, robustness, and ability to deal with discrete design variables. None-the-less, their inevitable drawbacks still remains, which are a low convergence rate and a lack of search consistency.

As a result, this work is conducted so as to improve search performance of EAs. The performance enhancement is investigated in two ways. Firstly, the hybridisation between EAs is investigated. The second type of hybridisation is using EAs as the main algorithm while a local search technique is used to accelerate its convergence. In this project, it has been shown that the integration between real-code population-based incremental learning and differential evolution is an efficient method for multiobjective optimisation of trusses. A local search technique called an approximated gradient was created for both unconstrained and constrained optimisation. It has been found that the approximate gradient is a powerful local search technique for multiobjective evolutionary algorithms (MOEAs) in solving multiobjective optimisation of trusses. The simplex downhill method has been integrated into the main procedure of real-code ant colony optimisation resulting in a powerful hybrid evolutionary algorithm for constrained mechanical design problems. The hybrid real-code population-based incremental learning and differential evolution has been extended and then used to solve simultaneous topology and sizing multiobjective optimisation of pipe network. It is shown that the hybrid algorithm is superior to other existing MOEAs. Further demonstration of hybrid approached has been made. The differential evolution algorithm and teaching-learning based optimisation have been hybridised for solving a strip coiling process where the hybrid method is superior to other EAs. The two-level population-based incremental learning using binary strings has been proposed for anti-optimisation of trusses. The method is proven effective for such a design type.

Moreover, an automate design process of trusses and frames has been studied. The automate design of truss also known as simultaneous topology, shape and sizing optimisation is a means to help inexperienced designers to be able to design structures as effectively as those who are highly experienced provided that the inexperienced ones have good backgrounds in structural analysis and optimisation. The automate design processes of two- and three- dimensional trusses and frames have been investigated. The automate design process is based on a ground element approach. It has been illustrated that the use of adaptive ground element is the better choice for the automate design. Finally, this project leads to a training course on truss optimisation.

Keywords: Evolutionary Algorithms; Structural Optimisation; Finite Element Analysis; Hybrid Algorithms; Adaptive Ground Elements

หน้าสรุปโครงการ (Executive Summary)

1. ชื่อโครงการ (ภาษาไทย) การเพิ่มสมรรถนะและการประยุกต์ขั้นตอนวิธีวิวัฒนาการสำหรับการออกแบบโครงข้อหมุนและโครงข้อแข็ง
(ภาษาอังกฤษ) Performance Enhancement and Applications of Evolutionary Algorithms for Truss and Frame Design

2. ชื่อหัวหน้าโครงการ หน่วยงานที่สังกัด ที่อยู่ หมายเลขโทรศัพท์ โทรสาร และ e-mail

หัวหน้าโครงการ (ชื่อ-สกุล ภาษาไทย) สุจินต์ บุรีรัตน์
(ชื่อ-สกุล ภาษาอังกฤษ) Sujin Bureerat

สถานที่ทำงาน

ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น

จังหวัด ขอนแก่น รหัสไปรษณีย์ 40002

โทรศัพท์ 043-202845 โทรสาร 043-202849

โทรศัพท์มือถือ 09-7114229 e-mail sujbur@kku.ac.th

3. สาขาวิชาที่ทำการวิจัย

Structural Optimisation

4. งบประมาณทั้งโครงการ 1,000,000.00 บาท

5. ระยะเวลาดำเนินงาน 3 ปี

6. ได้เสนอโครงการนี้ หรือโครงการที่มีส่วนเหมือนกับเรื่องนี้บางส่วนเพื่อขอทุนต่อแหล่งทุนอื่นที่ใดบ้าง

✓ ไม่ได้เสนอต่อแหล่งทุนอื่น

7. ปัญหาที่ทำการวิจัย และความสำคัญของปัญหา

โครงข้อหมุน (trusses) และ โครงข้อแข็ง (frames) หรือเรียกว่าโครงถัก เป็น โครงสร้างที่นิยมใช้ในงานโครงสร้างทางวิศวกรรมด้วยเหตุว่ามีความได้เปรียบในการประยุกต์ใช้ เช่น มีความง่ายในการก่อสร้างและมีราคาถูก ง่ายในการลำเลียงเพื่อก่อสร้างในพื้นที่ที่เข้าถึงได้ยากเช่น บนภูเขา ตัวอย่างของการใช้งานโครงถักที่สามารถเห็นได้ในชีวิตประจำวัน เช่น โครงสร้างป้ายโฆษณาขนาดใหญ่ โครงหลังคาของอาคารขนาดใหญ่ สะพาน เสาไฟฟ้า และชิ้นส่วนทางกลบางอย่าง เป็นต้น ในอดีต การออกแบบโครงสร้างเหล่านี้มักดำเนินการด้วยการลองผิดลองถูก (trial and error) และอาศัยประสบการณ์ของผู้ออกแบบเป็นหลัก อย่างไรก็ตาม เมื่อวิทยาการทางคอมพิวเตอร์อยู่ในขั้นสูง การ

ประยุกต์ใช้การหาค่าเหมาะที่สุดของโครงถัก (truss optimisation) ก็สามารทำได้ ซึ่งส่งผลให้ผู้ออกแบบสามารถออกแบบโครงถักได้อย่างมีประสิทธิภาพมากกว่าเดิม

วัตถุประสงค์ของโครงการวิจัยนี้มีสองส่วนคือ จะทำการเพิ่มสมรรถนะของขั้นตอนวิธีวิวัฒนาการ (evolutionary algorithms) ซึ่งเป็นวิธีหาค่าเหมาะที่สุดรูปแบบหนึ่งที่นิยมใช้กันมากในกลุ่มนักวิจัยทางด้านวิศวกรรม การเพิ่มประสิทธิภาพของขั้นตอนวิธีวิวัฒนาการจะเน้นไปที่หลักการผสมผสานขั้นตอนวิธีที่มีอยู่เดิม เพื่อให้ได้สิ่งใหม่ที่ดีขึ้น ส่วนวัตถุประสงค์ที่สองเป็นการประยุกต์ใช้ขั้นตอนวิธีวิวัฒนาการที่พัฒนาขึ้นมาเพื่อการออกแบบ หาค่าเหมาะที่สุดของโครงถักที่ประยุกต์ใช้งานจริงในทางวิศวกรรมในประเทศไทย ทั้งนี้จะทำการพัฒนากระบวนการออกแบบอัตโนมัติ (automate design) ซึ่งเป็นกระบวนการที่จะช่วยให้วิศวกรที่มีประสบการณ์น้อย สามารถออกแบบโครงสร้างลักษณะนี้ได้ดีใกล้เคียงกับนักออกแบบที่มีประสบการณ์ นอกจากนี้ยังนำเสนอการพัฒนาขั้นตอนวิธีวิวัฒนาการสำหรับการหาค่าเหมาะที่สุดแบบตรงข้าม (anti-optimisation) เพื่อใช้ออกแบบโครงสร้างในกรณีที่มีพารามิเตอร์บางตัวมีความไม่แน่นอน (uncertainties) ท้ายที่สุดโครงการนี้จะนำไปสู่การตั้งหน่วยวิจัย Applied Mathematics and Optimisation Research Unit (AMORU)

8. วัตถุประสงค์

วัตถุประสงค์หลักของงานวิจัยนี้ประกอบด้วย

- 8.1 ศึกษาการเพิ่มสมรรถนะของขั้นตอนวิธีวิวัฒนาการทั้งแบบเป้าหมายเดียวและหลายเป้าหมายโดยเน้นไปที่การประยุกต์ใช้กับการออกแบบโครงสร้าง
- 8.2 ศึกษาการผสมผสานค่าประมาณเกรเดียนต์ (approximate gradient) กับขั้นตอนวิธีวิวัฒนาการเพื่อค้นหาวิธีผสม (hybrid) ที่มีประสิทธิภาพสูง
- 8.3 ศึกษาการผสมผสานระหว่างตัวดำเนินการของขั้นตอนวิธีวิวัฒนาการแบบต่างๆ เพื่อให้เกิดวิธีที่มีสมรรถนะสูงขึ้นกว่าเดิม
- 8.4 ประยุกต์ใช้ขั้นตอนวิธีแบบผสมผสานที่คิดขึ้นมาใหม่กับการหาค่าเหมาะที่สุดของโครงถัก โดยเน้นไปที่การประยุกต์ใช้ในงานจริง

9. ระเบียบวิธีวิจัย

9.1 การเพิ่มสมรรถนะของขั้นตอนวิธีวิวัฒนาการสามารถทำได้หลายวิธี วิธีที่ง่ายที่สุดคือการปรับปรุงตัวดำเนินการเชิงวิวัฒนาการ (evolutionary operators) เช่น การครอสโอเวอร์ และมิวเทชันในขั้นตอนวิธีเชิงพันธุกรรมเพื่อให้มีประสิทธิภาพมากขึ้น นอกจากนี้อาจทำการคิดค้นขั้นตอนวิธีวิวัฒนาการขึ้นมาใหม่ โดยยังคงแนวคิดในการเลียนแบบการวิวัฒนาการของระบบทางธรรมชาติรูปแบบต่างๆ

9.2 จากงานวิจัยก่อนหน้านี้ ผู้วิจัยหลักได้ทำการนำเสนอตัวดำเนินการที่เรียกว่า ค่าประมาณของเกรเดียนต์ ซึ่งเป็นการสร้างเวกเตอร์เกรเดียนต์จากข้อมูลในแต่ละรอบของขั้นตอนวิธีวิวัฒนาการ การเพิ่มตัวดำเนินการนี้ในขั้นตอนวิธีวิวัฒนาการ จะทำให้เพิ่มประสิทธิภาพในการค้นหาแบบโกลบอล-โลคอล ซึ่งเรียกอีกชื่อหนึ่งว่า ขั้นตอนวิธีเมเมติก (memetic algorithms) [17 24 45 81 88 90] ในที่นี้จะทำการพัฒนาค่าประมาณเกรเดียนต์เพิ่มเติม สำหรับการหาค่าเหมาะที่สุดแบบมีเงื่อนไขบังคับ

9.3 การผสมผสานระหว่างขั้นตอนวิธีวิวัฒนาการมากกว่าหนึ่งวิธี เป็นอีกรูปแบบหนึ่งในการสร้างขั้นตอนวิธีวิวัฒนาการแบบใหม่ ในงานวิจัยนี้จะทำการศึกษาเรื่องนี้ โดยนำขั้นตอนวิธีวิวัฒนาการที่มีประสิทธิภาพสูงหลายๆตัวมาผสมกัน โดยเน้นไปที่การดึงเอาข้อดีของแต่ละวิธีออกมาใช้ร่วมกัน

9.4 นอกจากนี้การประยุกต์แบบขั้นตอนวิธีเมเมติก จะมีการนำวิธีหาค่าเหมาะที่สุดแบบตรง (direct methods) เช่น วิธีซิมเพล็กซ์ลงเขา (simplex downhill) และวิธีค้นหาเป็นรูปแบบ (pattern search) เพื่อนำมาเป็นตัวดำเนินการค้นหาแบบโลคอลสำหรับขั้นตอนวิธีวิวัฒนาการ

9.5 จากนั้นทำการประยุกต์กับการออกแบบหาค่าเหมาะที่สุดของโครงถัก ทั้งนี้จะเน้นไปที่ การสาธิตการออกแบบโครงข้อหมุนเป็นหลัก ปัญหาการออกแบบจะมีทั้งการหาทอพอโลยี รูปทรงและ ขนาดเหมาะที่สุดของโครงถัก การออกแบบอัตโนมัติ มีทั้งการหาค่าเหมาะที่สุดแบบเป้าหมายเดียวและ หลายเป้าหมาย

ช่วงเวลาที่ใช้ในการดำเนินการแต่ละหัวข้อแสดงในตารางข้างล่าง

No.	Plan	1 st year				2 nd year				3 rd year			
		1	2	3	4	1	2	3	4	1	2	3	4
1	สืบค้นงานวิจัยที่มีมาก่อน												
2	การจัดเตรียมซอฟต์แวร์และโปรแกรมโค้ด												
3	การเพิ่มสมรรถนะของขั้นตอนวิธีวิวัฒนาการ												
4	การหาค่าเหมาะที่สุดของโครงถัก												
5	รายงานวิจัย												
6	การประชุมประจำปีของ สกว.												

สารบัญ

กิตติกรรมประกาศ	3
บทคัดย่อ	4
Abstract	6
หน้าสรุปโครงการ	8
บทที่ 1 บทนำ	13
บทที่ 2 ขั้นตอนวิธีวิวัฒนาการ	19
บทที่ 3 ขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมาย	39
บทที่ 4 การเพิ่มสมรรถนะขั้นตอนวิวัฒนาการด้วยเทคนิคการผสมผสาน	55
บทที่ 5 การวิเคราะห์โครงข้อหมุนและโครงข้อแข็ง	63
บทที่ 6 กรณีศึกษา การหาค่าเหมาะที่สุดแบบอัตโนมัติของโครงข้อหมุน	73
บทที่ 7 กรณีศึกษา การหาค่าเหมาะที่สุดของโครงข้อแข็ง	97
บทที่ 8 การออกแบบโครงถักด้วยการหาค่าเหมาะที่สุดแบบตรงข้าม	111
บทที่ 9 การหาค่าเหมาะที่สุดของโครงถักด้วยขั้นตอนวิธีแบบผสม	127
บทที่ 10 สรุปและวิจารณ์	149
เอกสารอ้างอิง	151
Output จากโครงการวิจัย	159
ภาคผนวก	161

บทที่ 1 บทนำ

1.1 ที่มาและความสำคัญของปัญหา

โครงข้อหมุน (trusses) และโครงข้อแข็ง (frames) หรือเรียกว่าโครงถัก เป็นโครงสร้างที่นิยมใช้ในงานโครงสร้างทางวิศวกรรมด้วยเหตุว่ามีความได้เปรียบในการประยุกต์ใช้ เช่น มีความง่ายในการก่อสร้างและมีราคาถูก ง่ายในการลำเลียงเพื่อก่อสร้างในพื้นที่ที่เข้าถึงได้ยากเช่น บนภูเขา ตัวอย่างของการใช้งานโครงถักที่สามารถเห็นได้ในชีวิตประจำวัน เช่น โครงสร้างป้ายโฆษณาขนาดใหญ่ โครงหลังคาของอาคารขนาดใหญ่ สะพาน เสาไฟฟ้า และชิ้นส่วนทางกลบางอย่าง เป็นต้น ในอดีต การออกแบบโครงสร้างเหล่านี้มักดำเนินการด้วยการลองผิดลองถูก (trial and error) และอาศัยประสบการณ์ของผู้ออกแบบเป็นหลัก อย่างไรก็ตาม เมื่อวิทยาการทางคอมพิวเตอร์อยู่ในขั้นสูง การประยุกต์ใช้การหาค่าเหมาะที่สุดของโครงถัก (truss optimisation) ก็สามารถทำได้ ซึ่งส่งผลให้ผู้ออกแบบสามารถออกแบบโครงถักได้อย่างมีประสิทธิภาพมากกว่าเดิม

วัตถุประสงค์ของโครงการวิจัยนี้มีสองส่วนคือ จะทำการเพิ่มสมรรถนะของขั้นตอนวิธีวิวัฒนาการ (evolutionary algorithms) ซึ่งเป็นวิธีหาค่าเหมาะที่สุดรูปแบบหนึ่งที่นิยมใช้กันมากในกลุ่มนักวิจัยทางด้านวิศวกรรม การเพิ่มประสิทธิภาพของขั้นตอนวิธีวิวัฒนาการจะเน้นไปที่หลักการผสมผสานขั้นตอนวิธีที่มีอยู่เดิม เพื่อให้ได้สิ่งใหม่ที่ดีขึ้น ส่วนวัตถุประสงค์ที่สองเป็นการประยุกต์ใช้ขั้นตอนวิธีวิวัฒนาการที่พัฒนาขึ้นมาเพื่อการออกแบบ หาค่าเหมาะที่สุดของโครงถักที่ประยุกต์ใช้งานจริงในทางวิศวกรรมในประเทศไทย ทั้งนี้จะทำการพัฒนากระบวนการออกแบบอัตโนมัติ (automate design) ซึ่งเป็นกระบวนการที่จะช่วยให้วิศวกรที่มีประสบการณ์น้อย สามารถออกแบบโครงสร้างลักษณะนี้ได้ใกล้เคียงกับนักออกแบบที่มีประสบการณ์ นอกจากนี้ยังนำเสนอการพัฒนาขั้นตอนวิธีวิวัฒนาการสำหรับการหาค่าเหมาะที่สุดแบบตรงข้าม (anti-optimisation) เพื่อใช้ออกแบบโครงสร้างในกรณีที่มีพารามิเตอร์บางตัวมีความไม่แน่นอน (uncertainties) ท้ายที่สุดโครงการนี้จะนำไปสู่การตั้งหน่วยวิจัย Applied Mathematics and Optimisation Research Unit (AMORU)

1.2 การสืบค้นงานวิจัยที่มีมาก่อน

ในการออกแบบโครงสร้าง เราสามารถแยกกระบวนการออกแบบออกเป็น 3 ขั้นตอนคือ การออกแบบตามแนวคิด (concept design) การออกแบบขั้นต้น (preliminary design) และการออกแบบละเอียด (detailed design) ทั้งสามขั้นตอนนี้ เราสามารถประยุกต์ใช้วิธีหาค่าเหมาะที่สุดในการหาคำตอบหรือผลเฉลยเหมาะที่สุด [7 31 70] การออกแบบตามแนวคิดเป็นการออกแบบขั้นต้นแรกที่ผู้ออกแบบมักจะกำหนดแนวคิดของการใช้งานของโครงสร้าง แต่จะไม่ทราบข้อมูลรูปทรงหรือทอพอโลยี (topology) ของโครงสร้าง หลังจากการออกแบบขั้นต้นนี้แล้ว ผู้ออกแบบสามารถปรับปรุงโครงสร้างเพิ่มเติมด้วยการการหารูปร่างและขนาดเหมาะที่สุด (shape and sizing optimisation) ซึ่ง

จัดว่าเป็นการออกแบบขั้นต้น ในขั้นตอนที่สามหรือการออกแบบละเอียด จะเป็นการวิเคราะห์โครงสร้างอย่างละเอียดโดยคำนึงถึงปัจจัยที่จะทำให้โครงสร้างเกิดการเสียหายทุกรูปแบบซึ่งเป็นขั้นตอนสุดท้ายของการออกแบบ ในส่วนของงานวิจัยโดยมากจะเป็นการหาวิธีการหาค่าที่เหมาะสมที่สุดของโครงข่ายโครงข่าย อย่างไรก็ดีตาม เราสามารถประยุกต์ใช้กระบวนการออกแบบโครงข่ายสำหรับการออกแบบโครงข่ายได้

ในการหาค่าที่เหมาะสมที่สุดของโครงข่าย ปัญหาการออกแบบคือ การหาผลเฉลยของชุดตัวแปรออกแบบ (design variables) เพื่อให้ได้ค่าที่ดีที่สุดของฟังก์ชันเป้าหมาย (objective function) และผลเฉลยนั้นสอดคล้องกับเงื่อนไขบังคับ (constraints) ของการออกแบบ ตัวแปรออกแบบของโครงสร้างสามารถจำแนกเป็นตัวแปรออกแบบทอพอโลยี ตัวแปรออกแบบกำหนดรูปร่าง และตัวแปรออกแบบกำหนดขนาด ตัวแปรทอพอโลยี คือเซตของค่าพารามิเตอร์ที่ใช้กำหนดโครงข่ายเบื้องต้นของโครงข่าย โดยทั่วไปประยุกต์ใช้ในการออกแบบเชิงแนวคิด [2-5 9 12 14 33 39-41 43 58 73 75 83 103 106 109 114 125-127] ส่วนตัวแปรออกแบบกำหนดรูปร่างและขนาด นิยมใช้ในการออกแบบขั้นต้น [10 15 21 25 28 30 34-38 46-55 57 59 62 64-66 68 74 86 88-89 92-93 98-99 104 113 116-117 119-121 123 134] การออกแบบทอพอโลยีของโครงสร้างสามารถดำเนินการได้หลายวิธี แต่หนึ่งในวิธีที่นิยมกันมากที่สุดคือการประยุกต์ใช้เอลิเมนต์พื้น (ground elements) [39 79] สำหรับโครงข่าย ตัวแปรออกแบบกำหนดรูปร่างคือตำแหน่งของจุดต่อ (nodes) ของโครงข่าย ส่วนตัวแปรออกแบบขนาดคือการกำหนดขนาดของชิ้นส่วนของโครงข่าย การประยุกต์ใช้ตัวแปรออกแบบหลายชนิดพร้อมกันในการหาค่าที่เหมาะสมที่สุด 1 ครั้ง ได้มีการศึกษาและวิจัย [1 24 76-80 82 91 97 108 111 118 128-129] และได้มีการแสดงให้เห็นว่า การหาค่าทอพอโลยี รูปร่างและขนาดที่เหมาะสมที่สุดพร้อมกัน ทำให้ได้ผลเฉลยที่เหมาะสมที่สุดที่ดีกว่าการหาค่าที่เหมาะสมที่สุดเป็นขั้นตอนตามลำดับดังที่กล่าวในข้างต้น [77-80] การออกแบบชนิดนี้ เรียกอีกชื่อหนึ่งว่า การออกแบบอัตโนมัติ (automate design) ซึ่งเป็นกระบวนการออกแบบที่ต้องการพัฒนาให้มีประสิทธิภาพสูงและเป็นโจทย์ที่ท้าทายสำหรับนักวิจัยทางการหาค่าที่เหมาะสมที่สุดของโครงข่าย

ฟังก์ชันเป้าหมายที่นิยมใช้มากที่สุดในการออกแบบโครงข่าย คือ น้ำหนักของโครงข่ายซึ่งอาจใช้บ่งชี้ถึงราคาส่วนหนึ่งของโครงข่าย ส่วนฟังก์ชันเป้าหมายรูปแบบอื่นที่นิยมประยุกต์ใช้คือ ค่าความอ่อนตัว (compliance) ของโครงข่าย การหาค่าความอ่อนตัวต่ำสุดมีความสมมูลกับการหาค่าความแข็งตึง (stiffness) สูงสุดของโครงข่าย ซึ่งเป็นพารามิเตอร์ที่ใช้บ่งชี้ความแน่นอน (reliability) ของโครงข่าย นอกจากนี้เราสามารถกำหนดค่าฟังก์ชันเป้าหมายเป็น ค่าปัจจัยการโก่งตัว (buckling factor) ค่าความถี่ธรรมชาติ และค่าฟังก์ชันการตอบสนองเชิงความถี่ (frequency response function) ส่วนเงื่อนไขบังคับประกอบด้วย ความเค้น การกระจัด ค่าความถี่ธรรมชาติและปัจจัยการโก่งตัว ค่าเหล่านี้ถูกกำหนดขอบเขตเพื่อให้โครงข่ายมีความปลอดภัยภายใต้ภาระที่กระทำกับตัวโครงข่าย ในส่วนของปัญหาการออกแบบ เราสามารถกำหนดเป็นปัญหาการหาค่าที่เหมาะสมที่สุดแบบเป้าหมายเดียว

(single-objective optimisation) หรือการหาค่าเหมาะที่สุดแบบหลายเป้าหมาย (multiobjective optimisation) ส่วนวิธีหรือขั้นตอนวิธีหาค่าเหมาะที่สุดสามารถจำแนกได้เป็น 2 แบบคือ วิธีหาค่าเหมาะที่สุดแบบใช้อนุพันธ์และวิธีที่ไม่ใช้อนุพันธ์ วิธีที่ไม่ใช้อนุพันธ์ที่นิยมใช้กันมากที่สุดคือ กลุ่มของวิธี เมตะฮิวริสติกส์และขั้นตอนวิธีวิวัฒนาการ

การประยุกต์ใช้ขั้นตอนวิธีวิวัฒนาการมีข้อได้เปรียบหลายอย่างเช่น ง่ายในการทำความเข้าใจ และใช้งาน มีความเสถียรเชิงตัวเลขสูง เนื่องจากการสุ่ม (randomisation) เป็นกลไกหลัก ทำให้วิธีการเหล่านี้สามารถใช้ในการหาผลเฉลยของปัญหาการหาค่าเหมาะที่สุดได้เกือบทุกรูปแบบ สามารถหาผลเฉลยเหมาะที่สุดแบบโกลบอล (global optimum) ได้ นอกจากนี้ ความสามารถที่โดดเด่นที่สุดของขั้นตอนวิธีวิวัฒนาการคือ สามารถหาขอบหน้าพาเรโต (Pareto front) ของปัญหาการหาค่าเหมาะที่สุดแบบหลายเป้าหมายได้ภายในการรันเพียงครั้งเดียว ซึ่งความสามารถที่โดดเด่นนี้ไม่มีในวิธีการหาค่าเหมาะที่สุดแบบใช้อนุพันธ์ อย่างไรก็ตาม ถึงแม้จะมีข้อได้เปรียบมากมาย ขั้นตอนวิธีวิวัฒนาการจะมีข้อด้อยสองอย่างคือ มีอัตราการลู่เข้าต่ำและไม่มีความคงเส้นคงวา (ไม่สามารถหาผลเฉลยเหมาะที่สุดชุดเดิมได้จากการรันหลายๆครั้ง) การประยุกต์ใช้ขั้นตอนวิธีวิวัฒนาการและเมตะฮิวริสติกส์มักจะมีปัญหาเสมอเมื่อการหาค่าฟังก์ชันในปัญหาการออกแบบใช้เวลานานมาก และเมื่อจำนวนของตัวแปรออกแบบมีมากเกินไป (large-scale problems) ด้วยเหตุนี้ จึงมีความต้องการในการพัฒนาสมรรถนะของวิธีการเหล่านี้อยู่เสมอ トラバโดปัญหาหรือข้อด้อยที่กล่าวถึงในข้างต้นยังไม่หมดไป การพัฒนาขั้นตอนวิธีใหม่ๆ มีการนำเสนออย่างต่อเนื่อง เริ่มต้นที่ขั้นตอนวิธีเชิงพันธุกรรม (genetic algorithm) จากนั้นได้มีการพัฒนาขั้นตอนวิธีวิวัฒนาการมากกว่าหนึ่งพันวิธี ทั้งในส่วนของวิธีการสำหรับการหาค่าเหมาะที่สุดแบบฟังก์ชันเป้าหมายเดียว [13 16 17 42 44 56 60 87 89 94 100 102 107 110 112] และหลายเป้าหมาย [6 11 19-20 26 27 29 32 61 69 94 101 105 131-133] แต่ก็ไม่มีวิธีการไหนที่สามารถกล่าวได้ว่าเป็นวิธีที่ดีที่สุด และวิธีการเหล่านี้จะมีความโดดเด่นสำหรับปัญหาการออกแบบเฉพาะบางชนิดเท่านั้น ในช่วงแรกการพัฒนาขั้นตอนวิธีวิวัฒนาการจะเน้นไปที่ความสามารถในการใช้งานได้ [29 61 87 95 105 133] แต่ในปัจจุบัน นักวิจัยจะเน้นที่การเพิ่มสมรรถนะในการหาผลเฉลย หรือการเพิ่มอัตราการลู่เข้า [6 11 17 26 32 42 56 131] อย่างไรก็ตาม อัตราการลู่เข้าของวิธีวิวัฒนาการยังไม่สามารถเทียบกับวิธีหาค่าเหมาะที่สุดแบบใช้อนุพันธ์ได้และยังคงต้องมีการพัฒนาต่อไป

ในเชิงการวิจัย ได้มีการประยุกต์ใช้ขั้นตอนวิธีวิวัฒนาการในการหาค่าเหมาะที่สุดของโครงถัก และมีการนำเสนอในบทความวิจัยมากกว่าหนึ่งพันเรื่องต่อปี การเพิ่มสมรรถนะในการหาผลเฉลยของขั้นตอนวิธีวิวัฒนาการนั้นสามารถทำได้หลายวิธีเช่น การใช้วิธีผสมผสาน [50-51 53-54 81 88 124] การใช้แบบจำลองทดแทน (surrogate models) และการผสมขั้นตอนวิธีวิวัฒนาการกับแนวคิดของการหาค่าเหมาะที่สุดแบบใช้อนุพันธ์ [17 24 45 90] การหาค่าเหมาะที่สุดของโครงข้อหมุนและโครงข้อแข็ง สามารถนำไปประยุกต์ใช้จริงได้หลากหลายงานเช่น เสาส่งสัญญาณ เสาไฟฟ้า ป้ายโฆษณา โครงหลังคา สะพาน โครงของยานพาหนะ และชิ้นส่วนทางกล ในปัจจุบันได้มีการพัฒนาขั้นตอนวิธี

วิวัฒนาการแบบใหม่และการเพิ่มสมรรถนะของวิธีที่มีอยู่แล้ว [16-20] และมีการสาธิตการประยุกต์ใช้ในงานจริงเช่นใน [19 47 79-80]

1.3 วัตถุประสงค์

วัตถุประสงค์หลักของงานวิจัยนี้ประกอบด้วย

- ศึกษาการเพิ่มสมรรถนะของขั้นตอนวิธีวิวัฒนาการทั้งแบบเป้าหมายเดียวและหลายเป้าหมาย โดยเน้นไปที่การประยุกต์ใช้กับการออกแบบโครงสร้าง
- ศึกษาการผสมผสานค่าประมาณเกรเดียนต์ (approximate gradient) กับขั้นตอนวิธีวิวัฒนาการเพื่อค้นหาวิธีผสม (hybrid) ที่มีประสิทธิภาพสูง
- ศึกษาการผสมผสานระหว่างตัวดำเนินการของขั้นตอนวิธีวิวัฒนาการแบบต่างๆ เพื่อให้เกิดวิธีที่มีสมรรถนะสูงขึ้นกว่าเดิม
- ประยุกต์ใช้ขั้นตอนวิธีแบบผสมผสานที่คิดขึ้นมาใหม่กับการหาค่าเหมาะสมที่สุดของโครงถัก โดยเน้นไปที่การประยุกต์ใช้ในงานจริง

1.4 ระเบียบวิธีวิจัย

- การเพิ่มสมรรถนะของขั้นตอนวิธีวิวัฒนาการสามารถทำได้หลายวิธี วิธีที่ง่ายที่สุดคือการปรับปรุงตัวดำเนินการเชิงวิวัฒนาการ (evolutionary operators) เช่น การครอสโอเวอร์ และ มิวเทชัน ในขั้นตอนวิธีเชิงพันธุกรรมเพื่อให้มีประสิทธิภาพมากขึ้น นอกจากนี้อาจทำการคิดค้นขั้นตอนวิธีวิวัฒนาการขึ้นมาใหม่ โดยยังคงแนวคิดในการเลียนแบบการวิวัฒนาการของระบบทางธรรมชาติรูปแบบต่างๆ
- จากงานวิจัยก่อนหน้านี้ ผู้วิจัยหลักได้ทำการนำเสนอตัวดำเนินการที่เรียกว่าค่าประมาณเกรเดียนต์ ซึ่งเป็นการสร้างเวกเตอร์เกรเดียนต์จากข้อมูลในแต่ละรอบของขั้นตอนวิธีวิวัฒนาการ การเพิ่มตัวดำเนินการนี้ในขั้นตอนวิธีวิวัฒนาการ จะทำให้เพิ่มประสิทธิภาพในการค้นหาแบบโกลบอล-โลคอล ซึ่งเรียกอีกชื่อหนึ่งว่า ขั้นตอนวิธีเมมติก (memetic algorithms) [17 24 45 81 88 90] ในที่นี้จะทำการพัฒนาค่าประมาณเกรเดียนต์เพิ่มเติม สำหรับการหาค่าเหมาะสมที่สุดแบบมีเงื่อนไขบังคับ
- การผสมผสานระหว่างขั้นตอนวิธีวิวัฒนาการมากกว่าหนึ่งวิธี เป็นอีกรูปแบบหนึ่งในการสร้างขั้นตอนวิธีวิวัฒนาการแบบใหม่ ในงานวิจัยนี้จะทำการศึกษาเรื่องนี้ โดยนำขั้นตอนวิธีวิวัฒนาการที่มีประสิทธิภาพสูงหลายๆตัวมาผสมกัน โดยเน้นไปที่การดึงเอาข้อดีของแต่ละวิธีออกมาใช้ร่วมกัน

บทที่ 2 ขั้นตอนวิธีวิวัฒนาการ

2.1 ความนำ นิยาม และแนวคิดเบื้องต้น

ขั้นตอนวิธีวิวัฒนาการในที่นี้ คือวิธีหาค่าเหมาะที่สุดที่มีกระบวนการคำนวณเลียนแบบหรือคล้ายคลึงกับพฤติกรรมการวิวัฒนาการของระบบบางอย่างในธรรมชาติ ขั้นตอนวิธีวิวัฒนาการที่เป็นที่รู้จักและนิยมใช้กันมากที่สุดคือ ขั้นตอนวิธีเชิงพันธุกรรม (genetic algorithms) อย่างไรก็ตาม เมื่อมีการค้นคว้าวิจัยในสาขาวิชานี้แพร่หลายไปทั่วโลก ได้ก่อให้เกิดวิธีวิวัฒนาการขึ้นมาหลากหลายรูปแบบ ซึ่งนอกจากจะเลียนแบบพฤติกรรมทางธรรมชาติแล้ว ยังขยายไปยังกระบวนการหาค่าเหมาะที่สุดที่เลียนแบบพฤติกรรมของระบบอื่นๆ ทั้งที่มีชีวิตและไม่มีชีวิต ด้วยเหตุนี้ จึงมีการกำหนดชื่อของวิธีหาค่าเหมาะที่สุดรูปแบบนี้ขึ้นใหม่เป็น เมตะฮิวริสติกส์ (meta-heuristics) อย่างไรก็ตาม คำว่าขั้นตอนวิธีวิวัฒนาการก็ยังเป็นที่นิยมเรียกกันอยู่เหมือนเดิม และในบทนี้ หากมีการกล่าวถึงขั้นตอนวิธีวิวัฒนาการ จะรวมไปถึงเมตะฮิวริสติกส์ทุกรูปแบบ นอกจากนี้ วิธีหาค่าเหมาะที่สุดชนิดนี้ยังนิยมเรียกอีกชื่อหนึ่งคือ วิธีหาค่าเหมาะที่สุดโดยอาศัยประชากร (population-based optimisers) ซึ่งคำว่าประชากรหมายถึงกลุ่มของผลเฉลยที่ใช้ในแต่ละวงวนของกระบวนการหาค่าเหมาะที่สุด

ขั้นตอนวิธีวิวัฒนาการเมื่อเทียบกับวิธีหาค่าเหมาะที่สุดที่ใช้อนุพันธ์จะมีทั้งข้อดีและข้อด้อย ในภาพรวม วิธีวิวัฒนาการมีข้อได้เปรียบคือ การใช้งานง่าย ไม่ต้องคำนวณค่าอนุพันธ์ของฟังก์ชันในการหาผลเฉลย สามารถหาผลเฉลยเหมาะที่สุดของทั้งหมด (global optimum) ได้ เป็นวิธีที่ทนทาน (robust) กล่าวคือ มีโอกาสเกิดความคลาดเคลื่อนหรือหยุดการคำนวณในระหว่างกระบวนการหาค่าเหมาะที่สุดน้อยมาก และที่สำคัญอีกข้อหนึ่งคือ เมื่อไม่มีการใช้อนุพันธ์ของฟังก์ชัน วิธีหาค่าเหมาะที่สุดเหล่านี้ สามารถประยุกต์ใช้กับปัญหาการออกแบบได้ทุกรูปแบบ ค่าพารามิเตอร์ย่อยของวิธีการเหล่านี้สามารถปรับเปลี่ยนได้ง่าย ไม่เป็นค่าเฉพาะเจาะจงเหมือนวิธีหาค่าเหมาะที่สุดแบบใช้อนุพันธ์ อย่างไรก็ตามวิธีวิวัฒนาการมีข้อด้อยคือ เนื่องจากทำการประยุกต์ใช้กระบวนการสุ่มในการหาค่าเหมาะที่สุด ทำให้วิธีการนี้มีสมรรถนะในการหาผลเฉลยต่ำเมื่อเทียบกับวิธีที่ใช้อนุพันธ์ นั่นคือมีอัตราการลู่เข้า (convergence rate) ต่ำ และไม่มี ความคงเส้นคงวา (consistency) ในการหาผลเฉลย (หมายความว่าเมื่อใช้วิธีวิวัฒนาการใดๆ หาคำตอบของปัญหาการออกแบบเดิมหลายครั้ง ผลเฉลยที่ได้จะไม่เหมือนเดิม) นอกจากนี้ เนื่องจากไม่มีค่าอนุพันธ์ของฟังก์ชัน การตรวจสอบเงื่อนไขความเหมาะที่สุดไม่สามารถทำได้ วิธีการนี้ใช้จำนวนครั้งในการหาค่าฟังก์ชันสูงมากเมื่อเทียบกับวิธีใช้อนุพันธ์ เนื่องจากมีอัตราการลู่เข้าต่ำและไม่สามารถรับประกันได้ว่าผลที่ได้เป็นผลเฉลยเหมาะที่สุดจริง ดังนั้น เรามักเรียกคำตอบที่ได้จากการประยุกต์ใช้วิธีเหล่านี้ว่า ผลเฉลยใกล้เคียงความเหมาะที่สุด (near optimum)

2.1.1 ปัญหาการออกแบบ

ปัญหาการออกแบบที่จะทำการประยุกต์ขั้นตอนวิธีวิวัฒนาการเพื่อหาผลเฉลยคือ ปัญหาการออกแบบที่มีเงื่อนไขขอบเขตหรือขอบเขตของตัวแปรออกแบบ หรือที่เรียกว่า เงื่อนไขขอบเขตแบบกล่อง (box constraints) ปัญหาการออกแบบชนิดนี้เรียกว่า การหาค่าเหมาะที่สุดที่มีเงื่อนไขขอบเขตแบบกล่อง (box-constrained optimisation) ซึ่งสามารถแสดงได้ดังนี้

$$\begin{aligned} \text{จงหาค่าต่ำสุด } f(\mathbf{x}) \\ \mathbf{L} \leq \mathbf{x} \leq \mathbf{U} \end{aligned} \quad (2.1)$$

เมื่อ $f(\mathbf{x})$ คือ ฟังก์ชันเป้าหมาย

$\mathbf{x}$ คือ เวกเตอร์ตัวแปรออกแบบขนาด $n \times 1$

$\mathbf{L}$ คือ ขอบเขตล่างของตัวแปรออกแบบ

$\mathbf{U}$ คือ ขอบเขตบนของตัวแปรออกแบบ

ขั้นตอนวิธีวิวัฒนาการโดยทั่วไป สามารถหาผลเฉลยของปัญหาการหาค่าเหมาะที่สุดนี้ได้โดยตรง อย่างไรก็ตาม เมื่อมีเงื่อนไขขอบเขตชนิดอื่น จำเป็นต้องเพิ่มกระบวนการเชิงตัวเลขเข้ามาเพื่อสามารถดัดแปลงปัญหาการออกแบบให้อยู่ในรูปแบบ (2.1) เช่น การประยุกต์ใช้เทคนิคฟังก์ชันปรับแก้ (penalty function techniques) เป็นต้น

กระบวนการหาผลเฉลยเหมาะที่สุดของวิธีการที่นำเสนอในที่นี้ ส่วนมากจะใช้กลุ่มของเวกเตอร์ตัวแปรออกแบบซึ่งนิยมเรียกว่า ประชากร (population) ในการหาคำตอบ ในกรณีของตัวแปรออกแบบจำนวนจริง ประชากรของเวกเตอร์ตัวแปรออกแบบ สามารถเขียนแทนด้วย

$$\mathbf{P} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \quad (2.2)$$

เมื่อ N คือ ขนาดของประชากรหรือจำนวนเวกเตอร์ตัวแปรออกแบบที่เป็นสมาชิกของประชากร วิธีวิวัฒนาการใช้หลักการคำนวณแบบกระทำซ้ำ (iterative method) ในการหาผลเฉลย รอบของการคำนวณหรือวงวน นิยมเรียกว่า รุ่น (generation) ของประชากรของเวกเตอร์ตัวแปรออกแบบ

2.1.2 การเข้ารหัสและถอดรหัสตัวแปรออกแบบ

รูปแบบของตัวแปรออกแบบที่ใช้สำหรับวิวัฒนาการมีได้หลายรูปแบบ เช่น เวกเตอร์จำนวนจริง เวกเตอร์จำนวนเต็ม หรือแนวของเลขฐานสอง (binary string) การเข้ารหัสของตัวแปรออกแบบหมายถึง การที่ผู้ออกแบบกำหนดชนิดของตัวแปรออกแบบเพื่อใช้กับปัญหาการออกแบบใดๆ ส่วนการถอดรหัสตัวแปรออกแบบคือ การแปลงเวกเตอร์ตัวแปรออกแบบให้อยู่ในรูปแบบของตัวแปรที่เป็นตัวเลข

เพื่อใช้คำนวณค่าของฟังก์ชันเป้าหมายและเงื่อนไขบังคับ เนื่องจากวิธีวิวัฒนาการเช่น ขั้นตอนวิธีเชิงพันธุกรรม และวิธีการเรียนรู้แบบเพิ่มพูนโดยอาศัยกลุ่มประชากร (Population-Based Incremental Learning, PBIL) เข้ารหัสตัวแปรออกเป็นแนวของเลขฐานสอง ดังนั้น เมื่อทำการหาค่าฟังก์ชันของตัวแปรเหล่านั้น จำเป็นต้องถอดรหัสจากตัวเลขฐานสองมาเป็นตัวเลขฐานสิบหรือจำนวนจริง เพื่อสามารถคำนวณค่าฟังก์ชันเป้าหมายและเงื่อนไขบังคับได้

ในกรณีทั่วไป อย่างเช่น ปัญหาการออกแบบ (2.1) การแปลงค่าเวกเตอร์แนวของเลขฐานสอง $\mathbf{b}$ ขนาด $1 \times N_b$ ไปเป็นค่าของตัวแปรออกแบบจำนวนจริง x_i ที่อยู่ในขอบเขตของตัวแปร $[L_i, U_i]$ สามารถทำได้โดยการใช้สมการ

$$x_i = L_i + (U_i - L_i) \frac{\mathbf{b}t}{2^{N_b} - 1} \quad (2.3)$$

เมื่อ $\mathbf{t}$ คือ เมตริกซ์การแปลงเลขฐานสองขนาด $N_b \times 1$ สมาชิกของ $\mathbf{t}$ มีค่าเป็น $t_i = 2^{(N_b-i)}$

ในกรณีที่ตัวแปรออกแบบในปัญหาดังต้นมีมากกว่าหนึ่งตัว ในการเข้ารหัสตัวแปรออกแบบ ใช้เวกเตอร์ของแนวของเลขฐานสองเพียงหนึ่งชุดเท่านั้นสำหรับเวกเตอร์ $\mathbf{x} = \{x_1, x_2\}^T$ ผู้ออกแบบต้องกำหนดว่า ต้องการใช้เวกเตอร์ $\mathbf{b}$ ขนาดเท่าใดเพื่อเป็นตัวแทนของ $\mathbf{x}$ เช่น ถ้าใช้ความยาวของเลขฐานสองเป็น $N_b = 10$ หลัก โดย 5 หลักแรก แปลงค่าเป็น x_1 และ 5 หลักที่เหลือเป็นตัวแทนของ x_2 ตัวอย่างเช่น ถ้าเรามี $\mathbf{b} = \{1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\}$ เลขฐานสอง 10001 ใช้ถอดรหัสเป็นค่าของ x_1 และ 00111 เป็นค่าของ x_2

ตัวแปรออกแบบที่ใช้แนวของเลขฐานสองจะเป็นตัวแปรแบบดิสครีทหรือวิยุต (discrete) ความละเอียดของผลเฉลยบนโดเมนของตัวแปรออกแบบขึ้นอยู่กับค่าของ N_b ถ้ามีค่ามาก ความละเอียดของตัวแปรออกแบบจะสูงมากขึ้น

2.1.3 กระบวนการคำนวณในรูปแบบทั่วไป

กระบวนการค้นหาผลเฉลยของขั้นตอนวิธีวิวัฒนาการโดยทั่วไปเริ่มต้นด้วยประชากรตั้งต้น (initial population) หรือกลุ่มของเวกเตอร์ออกแบบเริ่มต้น จากนั้นทำการประยุกต์ใช้กลไกที่เลียนแบบวิวัฒนาการทางธรรมชาติ หรือระบบอื่นๆ เพื่อสร้างประชากรลูก (offspring) ขึ้นมา ทั้งนี้ กระบวนการสุ่ม (randomisation) มักถูกใช้เป็นเครื่องมือหลักในขั้นตอนนี้ จากนั้นทำการคัดเลือก (selection) ผลเฉลยเพื่อเป็นประชากรรุ่นถัดไป (next generation) โอกาสในการถูกคัดเลือกจะขึ้นอยู่กับค่าฟังก์ชันเป้าหมาย หรือในบางวิธีวิวัฒนาการ จะแปลงค่าฟังก์ชันเป้าหมายเป็นค่าความเหมาะสม (fitness) ของผลเฉลยที่เป็นสมาชิกของประชากรเหล่านั้น กระบวนการคำนวณโดยทั่วไปสามารถแสดงได้ดังนี้

ขั้นตอนวิธีที่ 2.1 กระบวนการคำนวณของขั้นตอนวิธีวิวัฒนาการ

ขั้นตอนเริ่มต้น กำหนดพารามิเตอร์ที่จำเป็นทั้งหมด และประชากรเริ่มต้นพร้อมค่าฟังก์ชันเป้าหมายของสมาชิกในประชากร

ขั้นตอนคำนวณหลัก

- 1: ถ้าเงื่อนไขเหมาะสมที่สุดเป็นจริง หยุดการค้นหาผลเฉลย ถ้าไม่เป็นจริง ดำเนินการต่อในขั้นตอนที่ 2:
- 2: ผลิตประชากรลูกด้วยหลักการเฉพาะของขั้นตอนวิธีเหล่านั้น
- 3: หาค่าฟังก์ชันเป้าหมายและค่าความเหมาะสม (สำหรับบางวิธีการ)
- 4: ทำการคัดเลือกประชากรรุ่นใหม่
- 5: กลับไปขั้นตอนที่ 1:

เนื่องจากขั้นตอนวิธีวิวัฒนาการและเมตะฮิวริสติกส์อื่น ๆ มีจำนวนมากกว่า 100 วิธี ในบทนี้จะนำเสนอเฉพาะวิธีการที่เป็นที่รู้จัก ใช้งานง่าย มีรูปแบบการคำนวณที่ง่าย และในขณะเดียวกันมีสมรรถนะในการหาผลเฉลยสูง

2.2 ขั้นตอนวิธีเชิงพันธุกรรม

ขั้นตอนวิธีเชิงพันธุกรรมหรือ GA คือวิธีวิวัฒนาการที่เป็นที่รู้จักและนิยมใช้งานมากที่สุด ปัจจุบันได้มีการประยุกต์ใช้และมีการปรับปรุงดัดแปลง GA เพื่อปัญหาการออกแบบจริง ทำให้มีขั้นตอนวิธีของ GA มากกว่า 1000 วิธี GA ถูกพัฒนาขึ้นมาโดยเลียนแบบหลักการวิวัฒนาการ และการคัดเลือกทางธรรมชาติของกลุ่มของสิ่งมีชีวิต ตามทฤษฎีวิวัฒนาการของ ชาร์ล ดาร์วิน ในที่นี้ จะอธิบายกระบวนการคำนวณค้นหาค่าที่เหมาะสมที่สุดของ GA ในรูปแบบของกระบวนการคำนวณเชิงตัวเลข เพื่อความกระชับและเข้าใจง่าย

กระบวนการค้นหาผลเฉลยของ GA ในที่นี้เป็นรูปแบบอย่างง่ายสำหรับผู้เรียนรู้ใหม่ เริ่มต้นด้วยประชากรตั้งต้น จากนั้นทำการผลิตประชากรลูกโดยการคัดเลือกคู่ของพ่อ-แม่ (parents) จากสมาชิกของประชากรรุ่นปัจจุบัน (current generation) ทั้งนี้สมาชิกผลเฉลยของตัวแปรออกแบบที่ให้ค่าฟังก์ชันเป้าหมายดีกว่า จะมีโอกาสถูกคัดเลือกมากกว่า การผลิตประชากรลูกทำได้โดยตัวดำเนินการครอสโอเวอร์ (crossover) และการมิวเทชัน (mutation) เมื่อได้ประชากรลูก ทำการคัดเลือกประชากรสำหรับรุ่นหรือวงวนถัดไป ดำเนินการแบบทำซ้ำไปเรื่อยๆจนครบเงื่อนไขการหยุดค้นหาผลเฉลยที่เหมาะสมที่สุด รายละเอียดของตัวดำเนินการของ GA มีดังนี้

2.2.1 การคัดเลือก

การดำเนินการคัดเลือก ในที่นี้เป็นการคัดเลือกโดยใช้วงล้อรูเลตต์ (roulette) ซึ่งเป็นพื้นที่วงกลมที่แบ่งออกเป็นเซกเตอร์ย่อยจำนวนเท่ากับจำนวนผลเฉลยที่จะถูกเลือก ขนาดของเซกเตอร์จะแตกต่างกัน

กันขึ้นอยู่กับค่าโอกาสในการถูกเลือกของผลเฉลยต่างๆในประชากรปัจจุบัน ในรูปแบบที่ง่ายที่สุด ค่าโอกาสในการถูกเลือกสามารถคำนวณได้จาก ค่าความเหมาะสมของสมาชิกในประชากร กำหนดให้ค่าความเหมาะสมที่มีค่ามากกว่าหมายถึงผลเฉลยที่ดีกว่าและควรมีโอกาสถูกเลือกมากกว่า ดังนั้นโอกาสในการถูกเลือกของผลเฉลยที่ s -th สามารถคำนวณได้จาก

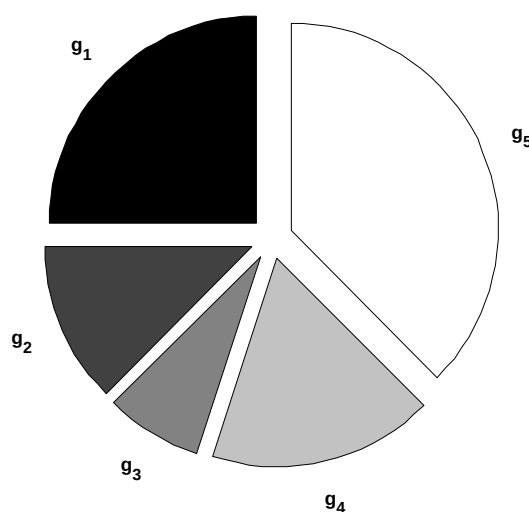
$$w_s = \frac{f_s}{\sum_{i=1}^N f_i} \quad (2.4)$$

เมื่อ N คือ ขนาดของประชากรตัวอย่าง กำหนดให้สมาชิกผลเฉลยในประชากรคือ g_1, g_2, g_3, g_4 และ g_5 โดยผลเฉลยทั้ง 5 นี้ มีค่าความเหมาะสมเป็น $f_1 = 10, f_2 = 5, f_3 = 3, f_4 = 7$ และ $f_5 = 15$ ตามลำดับ ดังนั้น โอกาสในการถูกเลือกของสมาชิกในประชากรชุดนี้ คำนวณได้ดังนี้ ($P(g_i)$ คือโอกาสในการถูกเลือกของ g_i)

สมาชิก	g_1	g_2	g_3	g_4	g_5	
ค่าความเหมาะสม	10	5	3	7	15	$\sum f_i = 40$
$P(g_i)$	0.25	0.125	0.075	0.175	0.375	

วงล้อรูเลตของโอกาสในการถูกเลือกของประชากรชุดนี้ แสดงในรูปที่ 2.1

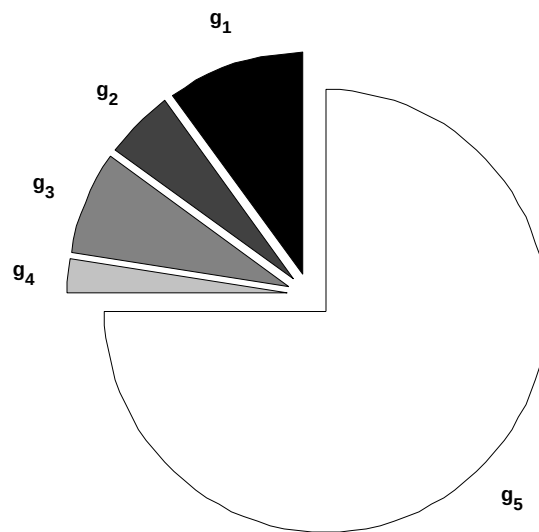
กระบวนการคัดเลือก สามารถทำได้โดยการตั้งเข็มชี้ตำแหน่งที่ด้านใดด้านหนึ่งของวงล้อ จากนั้นทำการหมุนวงล้อโดยใช้แรงที่ไม่เฉพาะเจาะจงหรือแบบสุ่ม เมื่อวงล้อหยุด เซกเมนต์ที่อยู่ตรงที่เข็มชี้จะได้รับการคัดเลือก จากหลักการนี้ สามารถกล่าวได้ว่า สมาชิกในประชากรที่มีค่าความเหมาะสมมากกว่า (ในที่นี้คือ g_5) ย่อมมีโอกาที่จะถูกเลือกมากกว่า เพราะมีแถบเซกเมนต์บนวงล้อใหญ่กว่าสมาชิกตัวอื่น



รูปที่ 2.1 วงล้อรูเลตสำหรับการดำเนินการคัดเลือกของ GA

ข้อดีของการสร้างวงล้อด้วยสมการที่ (2.4) คือ เมื่อมีสมาชิกตัวใดตัวหนึ่งในประชากรมีค่าความเหมาะสมมากกว่าสมาชิกตัวอื่นอย่างมีนัยสำคัญ หรืออาจเรียกว่า เป็นสมาชิกที่ครอบงำ (dominate) เหนือตัวอื่น ดังตัวอย่าง

สมาชิก	g_1	g_2	g_3	g_4	g_5	
ค่าความเหมาะสม	4	2	3	1	30	$\sum f_i = 40$
$P(g_i)$	0.100	0.050	0.075	0.025	0.750	



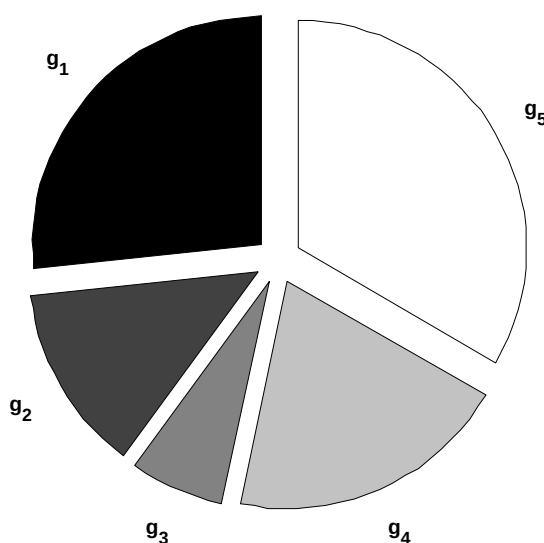
รูปที่ 2.2 วงล้อรูเลตสำหรับการดำเนินการคัดเลือกของ GA

รูปที่ 2.2 แสดงวงล้อรูเลตของประชากรชุดนี้ จากรูปจะเห็นได้ว่า เซกเมนต์ของสมาชิกตัวที่ดีที่สุด g_5 จะมีแถบกว้างเป็น $3/4$ เท่าของวงกลม นั้นหมายความว่า ในการหมุนวงล้อเพื่อคัดเลือกผลเฉลย g_5 มีโอกาสที่จะถูกเลือกซ้ำแล้วซ้ำอีกซึ่งไม่เป็นผลดีต่อกระบวนการคำนวณของ GA เพราะจะทำให้ขั้นตอนวิธีนี้ ถูเข้าสู่ผลเฉลยก่อนเวลาอันควร (premature convergence) การแก้ปัญหาดังนี้อันสามารถทำได้โดยการประยุกต์ใช้หลักการคำนวณโอกาสของการถูกเลือกรูปแบบใหม่ โดยทำการจัดอันดับของสมาชิกด้วยค่าความเหมาะสม กำหนดให้ r_s คืออันดับของสมาชิกตัวที่ s -th สมาชิกที่มีค่าความเหมาะสมมากที่สุดจะมีอันดับสูงสุด ($r_s = N$) ส่วนสมาชิกตัวที่มีค่าความเหมาะสมน้อยที่สุดจะได้เลขอันดับน้อยที่สุด ($r_s = 1$) ค่าโอกาสของการถูกเลือก สามารถคำนวณได้ดังนี้

$$w_s = \frac{2r_s}{N(N+1)} \quad (2.5)$$

จากวงล้อรูปที่ 2.2 ถ้าเราใช้สมการ (2.5) ในการคำนวณโอกาสของการถูกเลือก วงล้อรูปที่จะมีรูปแบบดังแสดงในรูปที่ 2.3 ค่ามัธยฐานของโอกาสในการถูกเลือกมีค่าประมาณ $1/(N+1)$ ส่วนผลเฉลี่ยที่ดีที่สุดจะมีโอกาสในการถูกเลือกที่ $2/(N+1)$ ซึ่งมีค่าเป็นสองเท่าของค่ามัธยฐานโดยประมาณ

สมาชิก	g1	g2	g3	g4	g5
ค่าความเหมาะสม	4	2	3	1	30
r_i	4	2	3	1	5
$P(g_i) (2r_i/N/(N+1))$	4/15	2/15	3/15	1/15	5/15



รูปที่ 2.3 วงล้อรูปสำหรับการดำเนินการคัดเลือกของ GA โดยใช้ (4.5)

นอกจากการคัดเลือกโดยใช้วงล้อรูปแล้ว เรายังสามารถประยุกต์ใช้กระบวนการคัดเลือกแบบการแข่งขัน (tournament selection) การคัดเลือกแบบนี้สามารถทำได้โดยกำหนดการแข่งขันขึ้นมาหลายๆ ครั้ง ในการแข่งขันแต่ละครั้ง สมาชิกในประชากร n_T ตัวจะถูกคัดเลือกแบบสุ่มเพื่อมาแข่งกัน ผู้ชนะหรือผลเฉลี่ยที่อยู่ในการแข่งขันที่ให้ค่าความเหมาะสมหรือฟังก์ชันเป้าหมายดีที่สุดคือผู้ถูกเลือกเพื่อเข้าสู่กระบวนการ คrossover ต่อไป การดำเนินการคัดเลือกแบบนี้เป็นกระบวนการที่ง่ายกว่าการใช้วงล้อรูป ถ้าค่า n_T ต่ำ โอกาสในการถูกเลือกของผลเฉลี่ยที่มีค่าความเหมาะสมต่ำจะมีสูง ในทางกลับกัน ถ้าค่า n_T สูง โอกาสในการถูกเลือกของสมาชิกที่มีค่าความเหมาะสมต่ำจะมีค่าน้อย

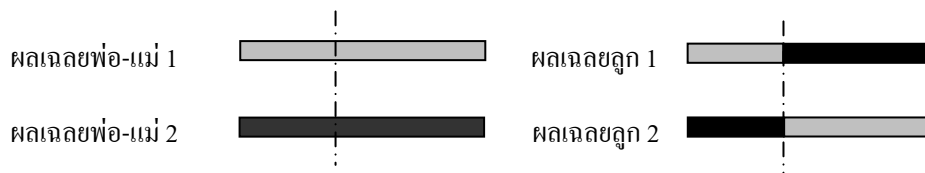
2.2.2 การดำเนินการครอสโอเวอร์

ครอสโอเวอร์เป็นตัวดำเนินการเชิงพันธุกรรมหลักของขั้นตอนวิธีเชิงพันธุกรรม เมื่อเลือกคู่ของผลเฉลี่ยพ่อแม่ได้ตามจำนวนที่ต้องการแล้ว จากนั้น ทำการดำเนินการแบบครอสโอเวอร์ การ

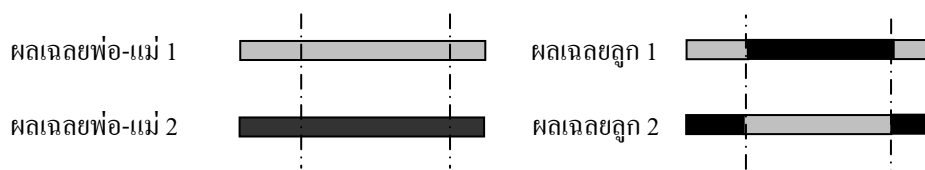
ดำเนินการเชิงตัวเลขนี้มีรูปแบบคล้ายกับการผสมพันธุ์ของคู่พ่อแม่ของสิ่งมีชีวิต **รูปที่ 2.4** แสดงรูปเสมือนโครโมโซม (chromosome) ของคู่ผลเฉลยพ่อแม่ การดำเนินการครอสโอเวอร์ คือการเลือกตำแหน่งจุดตัดโครโมโซมโดยการสุ่มเพื่อแยกโครโมโซมเหล่านั้นออกเป็นสองส่วน จากนั้น ผลเฉลยลูก (offspring) ชุดที่ 1 คือผลเฉลยที่มีส่วนแรกของโครโมโซมของผลเฉลยพ่อแม่ชุดที่ 1 และมีส่วนหลังของผลเฉลยพ่อแม่ชุดที่ 2 เช่นเดียวกัน ผลเฉลยลูกชุดที่ 2 จะมีโครโมโซมส่วนแรกเป็นของผลเฉลยพ่อแม่ชุดที่ 2 และมีโครโมโซมส่วนหลังเป็นของผลเฉลยพ่อแม่ชุดที่ 1 การดำเนินการนี้ เรียกว่าการครอสโอเวอร์แบบ 1 จุด

รูปที่ 2.5 แสดงการครอสโอเวอร์แบบ 2 จุด จากรูป ทำการตัดโครโมโซมของคู่พ่อแม่ 2 ตำแหน่งโดยการสุ่ม จากนั้นผลเฉลยลูก 1 และ 2 จะมีรูปแบบโครโมโซมเหมือนกับของผลเฉลยพ่อแม่ 1 และ 2 ตามลำดับ สำหรับโครโมโซมส่วนหัวและท้าย แต่มีการสลับตำแหน่งส่วนกลางของโครโมโซมพ่อแม่ทั้งสองดังแสดง ด้วยแนวคิดเดียวกันนี้ เราสามารถดำเนินการครอสโอเวอร์แบบหลายจุด ดังแสดงใน**รูปที่ 2.6** ในกระบวนการคำนวณของ GA เราใช้แนวของเลขฐานสองเป็นตัวแทนของโครโมโซมของผลเฉลย ส่วนการถอดรหัสเลขฐานสองเป็นตัวแปรออกแบบ นำเสนอในหัวข้อ 2.1.2 ตัวอย่างของการครอสโอเวอร์แบบ 1 จุด เมื่อมีโครโมโซมเป็นเลขฐานสอง แสดงได้ดังนี้

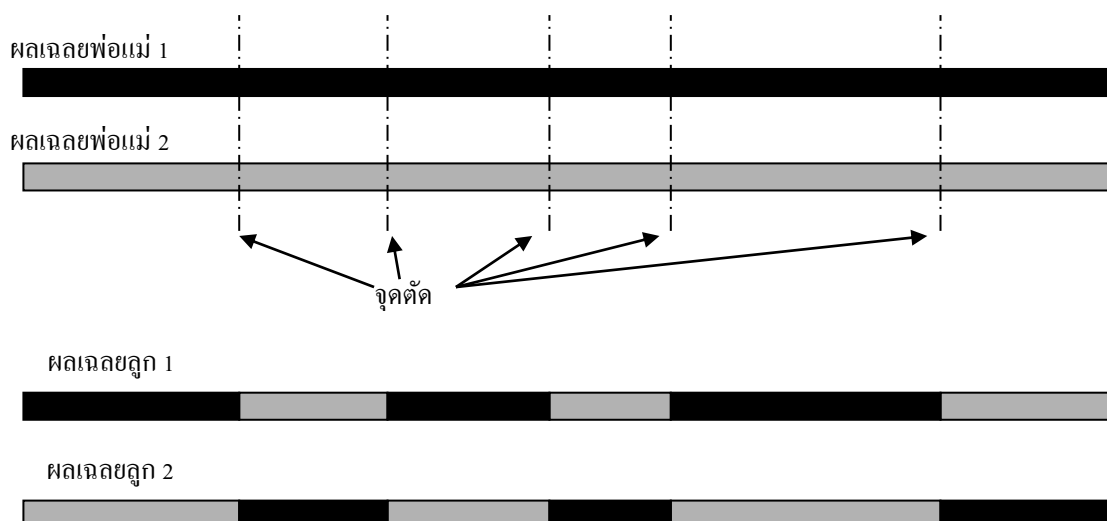
ผลเฉลยพ่อแม่ 1: 1000|11110 ผลเฉลยลูก 1: 100001010
 ผลเฉลยพ่อแม่ 2: 1011|01010 ผลเฉลยลูก 2: 101111110



รูปที่ 2.4 ครอสโอเวอร์แบบ 1 จุด



รูปที่ 2.5 ครอสโอเวอร์แบบ 2 จุด



รูปที่ 2.6 คrossover แบบหลายจุด

2.2.3 การดำเนินการมิวเทชัน

การดำเนินการมิวเทชันหรือการกลายพันธุ์ เป็นตัวดำเนินการเชิงพันธุกรรมที่เลียนแบบธรรมชาติของสิ่งมีชีวิตกล่าวคือ เมื่อมีการผสมพันธุ์ระหว่างคู่ของพ่อ-แม่ จะมีโอกาสที่ผลผลิตที่ได้คือประชากรลูกจะมียีน (gene) บางส่วนมีความแตกต่างไปจากอินทรีย์ของพ่อ-แม่ ในทางการคำนวณเชิงตัวเลข การดำเนินการนี้ถูกใช้เพื่อป้องกันการลู่เข้าสู่คำตอบก่อนเวลาอันควร (premature convergence) ที่เกิดจากการประยุกต์ใช้ ตัวดำเนินการ crossover เพียงอย่างเดียว การเกิดมิวเทชันในผลเฉลยแบบเลขฐานสองสามารถทำได้โดย เลือกตำแหน่งที่ต้องการเปลี่ยนแปลงยีนโดยการสุ่ม จากนั้นทำการเปลี่ยนเอลิเมนต์ที่เลือกนั้นในรูปแบบดังนี้ ถ้าเอลิเมนต์นั้นมีค่าเป็น 1 ให้เปลี่ยนค่าเป็น 0 หรือในทางกลับกันให้เปลี่ยนจากค่าเดิมที่เป็น 0 ให้เป็นเลข 1

ผลเฉลยพ่อแม่: 1000**1**1110

ผลเฉลยลูก: 1000**0**1110 (ตัวเข้ม คือเอลิเมนต์ที่ถูกเลือก)

โดยทั่วไปแล้ว การดำเนินการเชิงพันธุกรรม crossover และมิวเทชันจะได้รับอนุญาตให้ดำเนินการด้วยค่าความน่าจะเป็น หรือเรียกว่าอัตราการ crossover และมิวเทชันตามลำดับ สำหรับวิธีเชิงพันธุกรรม โดยทั่วไปจะมีอัตราการ crossover สูงและมีอัตราการมิวเทชันต่ำ กระบวนการคำนวณของขั้นตอนวิธีเชิงพันธุกรรมแสดงใน **ขั้นตอนวิธีที่ 2.2** กระบวนการเริ่มต้นด้วย การระบุขนาดของประชากร N ความยาวของแนวตัวเลขฐานสองต่อหนึ่งตัวแปรออกแบบ n_b อัตราการเกิด crossover p_c และมิวเทชัน p_m และจำนวนวงวน (รุ่น) สูงสุด G_{max} จากนั้นสุ่มสร้างประชากรเริ่มต้นซึ่งเป็นเซตของผลเฉลยตัวเลขฐานสองขนาด $N \times (n_b)$ ในที่นี้ ผลเฉลยคือแถวของเมตริกซ์ประชากร เมื่อได้ประชากร ทำการหาค่าฟังก์ชันเป้าหมาย และค่าความเหมาะสม จากนั้นคัดเลือกคู่ของผลเฉลยพ่อแม่ให้

ได้ตามจำนวนที่ต้องการ ทำการครอสโอเวอร์และมิวเทชัน ทำการคัดสรรประชากรสำหรับรุ่นถัดไป
ดำเนินการคำนวณซ้ำจนได้ผลเฉลยเหมาะสมที่สุด

ขั้นตอนวิธีที่ 2.2 กระบวนการคำนวณของขั้นตอนวิธีพันธุกรรม

ขั้นตอนเริ่มต้น กำหนดพารามิเตอร์ที่จำเป็นทั้งหมด และสร้างประชากรเริ่มต้นพร้อมค่าฟังก์ชัน
เป้าหมายของสมาชิกในประชากร

ขั้นตอนคำนวณหลัก

- 1: ถ้าเงื่อนไขการหยุดคำนวณเป็นจริง หยุดการค้นหาผลเฉลย
- ถ้าไม่เป็นจริง ดำเนินการต่อในขั้นตอนที่ 2:
- 2: ดำเนินการเลือกคู่ของผลเฉลยพ่อ-แม่
- 3: ทำการครอสโอเวอร์คู่พ่อ-แม่ที่เลือกมา ตามเงื่อนไขของอัตราการครอสโอเวอร์
- 4: ดำเนินการมิวเทชันตามเงื่อนไขของอัตราการมิวเทชัน จะได้ชุดประชากรลูก
- 5: หาค่าฟังก์ชันเป้าหมายและค่าความเหมาะสมของประชากรลูก
- 6: ทำการคัดเลือกประชากรรุ่นใหม่จากประชากรพ่อ-แม่ และประชากรลูก
- 7: กลับไปขั้นตอนที่ 1:

2.3 วิธีการเรียนรู้แบบเพิ่มพูนโดยอาศัยกลุ่มประชากร (Population-Based Incremental Learning, PBIL)

วิธี PBIL เป็นวิธีหาค่าเหมาะสมที่สุดเชิงวิวัฒนาการหรือเมตาสิวริสติกส์ที่ประยุกต์ใช้ขั้นตอนวิธีการประมาณการกระจาย (estimation of distribution algorithm) อย่างง่าย เป็นวิธีที่ประยุกต์ใช้แนวของเลขฐานสองแทนเวกเตอร์ตัวแปรออกแบบเช่นเดียวกับ GA อย่างไรก็ตาม PBIL จะไม่ใช่ประชากรของผลเฉลยเลขฐานสองในการหาค่าเหมาะสมที่สุดเหมือน GA แต่จะใช้เวกเตอร์ความน่าจะเป็น (probability vector) เวกเตอร์ความน่าจะเป็น คือเวกเตอร์ที่มีขนาดเท่ากับขนาดของแนวตัวเลขฐานสองที่ใช้แทนเวกเตอร์ตัวแปรออกแบบ ตัวอย่างเช่น ถ้าเวกเตอร์ตัวแปรออกแบบถูกแทนด้วยแนวของตัวเลขฐานสองที่มีจำนวนทั้งหมด 5 หลัก เวกเตอร์ของความน่าจะเป็นก็จะมีสมาชิกทั้งหมด 5 ตัว โดยสมาชิกแต่ละตัวคือค่าความน่าจะเป็นของการมีอยู่ของเลข “1” ในแนวคอลัมน์ของเมตริกซ์ประชากรเลขฐานสอง (สำหรับกรณีแถวของประชากรคือผลเฉลย)

เพื่อง่ายต่อการทำความเข้าใจ กำหนดให้ เวกเตอร์ของตัวแปรออกแบบถูกแทนด้วยแนวของเลขฐานสองที่มีสมาชิก 4 ตัว และในชุดประชากรเลขฐานสอง มีสมาชิกทั้งหมด 5 ผลเฉลย ดังนั้นเมตริกซ์ของประชากรเลขฐานสองจะมีขนาดเป็น 5×4 เมื่อแต่ละแถวของเมตริกซ์คือผลเฉลยของปัญหาการออกแบบ จากนั้นเวกเตอร์ความน่าจะเป็น เวกเตอร์จะมีขนาดเป็น 1×4 เมื่อสมาชิกตัวที่ i -th ของเวกเตอร์คือค่าความน่าจะเป็นในการมีเลข “1” ในแนวคอลัมน์ที่ i -th ของเมตริกซ์ของประชากรเลขฐานสอง **รูปที่ 2.8** แสดงตัวอย่างของเวกเตอร์ของความน่าจะเป็นและประชากรที่สร้างมาจาก

เวกเตอร์เหล่านั้น จากรูปเราสามารถสรุปได้ว่า เวกเตอร์ของความน่าจะเป็นหนึ่งเวกเตอร์สามารถสร้างเป็นประชากรเลขฐานสองได้หลากหลายรูปแบบ และนี่คือจุดแข็งของวิธี PBIL โดยเฉพาะเมื่อทำการประยุกต์ใช้ในการหาผลเฉลยพาเรโต ในบทถัดไป

ประชากร 1				ประชากร 2				ประชากร 3			
0	0	1	1	0	1	0	0	0	1	0	1
1	1	0	0	1	1	1	1	1	0	0	1
0	0	1	1	1	0	0	0	0	1	0	1
1	1	0	0	0	1	0	0	0	1	0	0
0	1	0	0	0	0	1	1	0	1	0	0

เวกเตอร์ของความน่าจะเป็น

2/5, 3/5, 2/5, 2/5 2/5, 3/5, 2/5, 2/5 1/5, 4/5, 0/5, 3/5

รูปที่ 2.8 เวกเตอร์ของความน่าจะเป็นและประชากรเลขฐานสอง

กระบวนการหาผลเฉลยของวิธี PBIL เริ่มต้นด้วย เวกเตอร์ของความน่าจะเป็นตั้งต้นที่มีค่าของสมาชิกทุกตัวเป็น 0.5 จากนั้น ทำการสร้างประชากรเลขฐานสองจากเวกเตอร์ของความน่าจะเป็นนี้ ทำการหาค่าฟังก์ชันเป้าหมาย เมื่อได้ผลเฉลยเลขฐานสองที่ดีที่สุดสำหรับประชากรชุดนี้ นำผลเฉลยนั้นมาปรับปรุงค่าของเวกเตอร์ของความน่าจะเป็นสำหรับวงวนถัดไป ดังนี้

$$P'_i = P_i(1 - L_R) + (b_i)L_R \quad (2.6)$$

เมื่อ

P'_i คือ สมาชิกตัวที่ i -th ของเวกเตอร์ของความน่าจะเป็นหลังจากปรับปรุงค่า

P_i คือ สมาชิกตัวที่ i -th ของเวกเตอร์ของความน่าจะเป็นในวงวนปัจจุบัน

b_i คือ สมาชิกตัวที่ i -th ของผลเฉลยเลขฐานสองที่ดีที่สุดที่ค้นหาได้

$L_R \in [0, 1]$ คือ ค่าอัตราการเรียนรู้

การเปลี่ยนแปลงค่าของเวกเตอร์ของความน่าจะเป็น ในสมการ (2.6) ขึ้นอยู่กับผลเฉลยเลขฐานสองที่ดีที่สุด b_i ถ้า $b_i = 1$ ค่าของ P_i จะเพิ่มขึ้น ทั้งนี้ ขึ้นอยู่กับค่าอัตราการเรียนรู้ ในทางกลับกัน ถ้า $b_i = 0$ ค่าของ P_i ที่ถูกปรับปรุงจะมีค่าลดลง นอกจากการปรับปรุงค่าของเวกเตอร์ของความน่าจะเป็นแล้ว เราสามารถประยุกต์การมิวเทชันกับเวกเตอร์นี้ โดยใช้สมการ

$$P''_i = P'_i(1 - m_s) + \text{rand}(0 \text{ or } 1)m_s \quad (2.7)$$

เมื่อ

$m_s \in [0, 1]$ คือ ค่าพารามิเตอร์ของการมิวเทชันที่ผู้ออกแบบต้องระบุ

$rand(0 \text{ or } 1) = round(rand)$ คือ การสุ่มตัวเลขที่มีโอกาสเท่ากันที่จะเป็น “0” หรือ “1”

$round(.)$ คือ ฟังก์ชันการปัดเศษเลขจำนวนจริงใดๆ ไปเป็นเลขจำนวนเต็มทีใกล้ที่สุด

เช่นเดียวกับวิธี GA ตัวดำเนินการมิวเทชันถูกใช้เพื่อป้องกันการลู่เข้าก่อนเวลาอันควรของกระบวนการหาผลเฉลยของ PBIL การมิวเทชันจะถูกอนุญาตให้ดำเนินการด้วยค่าของความน่าจะเป็นหรืออัตรามิวเทชัน p_m ซึ่งโดยทั่วไปจะมีค่าต่ำ ค่าของอัตราการเรียนรู้ L_R สามารถใช้ค่าคงที่ (โดยทั่วไปใช้ค่า $L_R = 0.5$) หรือเป็นแบบปรับเปลี่ยนได้ตามจำนวนรอบของวงวน

ขั้นตอนวิธีของ PBIL แสดงดังต่อไปนี้

ขั้นตอนวิธีที่ 2.3 กระบวนการคำนวณของวิธีการเรียนรู้แบบเพิ่มพูนโดยอาศัยกลุ่มประชากร

ขั้นตอนเริ่มต้น กำหนดค่าของ $P_i = 0.5, L_R, m_s, p_m$

ขั้นตอนคำนวณหลัก

- 1: ถ้าเงื่อนไขการหยุดคำนวณเป็นจริง หยุดการค้นหาค่าผลเฉลย
- ถ้าไม่เป็นจริง ดำเนินการต่อในขั้นตอนที่ 2:
- 2: สร้างประชากรเลขฐานสองจากเวกเตอร์ของความน่าจะเป็น $\mathbf{P}$
- 3: หาค่าฟังก์ชันเป้าหมายของประชากร $\mathbf{f}$
- 4: เลือกผลเฉลยที่ดีที่สุด $\mathbf{b}$
- 5: ปรับปรุงค่าของเวกเตอร์ของความน่าจะเป็นตามสมการ (2.6)
- 6: ดำเนินการมิวเทชันตามสมการ (2.7)
- 7: กลับไปขั้นตอนที่ 1:

2.4 วิธีการหาค่าเหมาะที่สุดแบบอนุภาครวมกลุ่ม (Particle Swarm Optimisation, PSO)

วิธีหาค่าเหมาะที่สุดแบบอนุภาครวมกลุ่มหรือ PSO นั้น เป็นเมตะฮิวริสติกส์ที่เลียนแบบพฤติกรรมของสิ่งมีชีวิตที่ออกหากินเป็นฝูงเช่น นก และ ปลา โดยสัตว์เหล่านี้จะรวมกลุ่มกันหากินในบริเวณที่มีอาหารอุดมสมบูรณ์มากที่สุด แต่ก็อาจมีนก หรือ ปลาบางตัวที่หากินในแหล่งที่ไม่อุดมสมบูรณ์มากนัก ในเชิงเปรียบเทียบ ตำแหน่งของนก คือตำแหน่งของเวกเตอร์ตัวแปรออกแบบ ส่วนความอุดมสมบูรณ์ของแหล่งอาหารคือค่าของฟังก์ชันเป้าหมาย เราอาจเรียกตำแหน่งของนกหรือปลาซึ่งเป็นตำแหน่งของผลเฉลยในรูปแบบทั่วไปว่า ตำแหน่งของอนุภาค วิธี PSO เป็นหนึ่งในวิธีแรกของขั้นตอนวิธีการรวมกลุ่มแบบฉลาด (swarm intelligence) กระบวนการคำนวณของ PSO เริ่มต้นด้วยกลุ่มของผลเฉลย หรือกลุ่มของอนุภาคตั้งต้น หรือจะเรียกว่าประชากรตั้งต้นเหมือนกับวิธีวิวัฒนาการที่กล่าวถึงก่อนหน้านี้ก็ได้ นอกจากนี้ยังมีค่าความเร็วตั้งต้นของอนุภาคเหล่านั้นประกอบการคำนวณ จากนั้นทำการคำนวณหาเวกเตอร์ความเร็วของกลุ่มอนุภาคเหล่านั้น ทิศทางการเคลื่อนที่ของอนุภาคจะขึ้นอยู่กับความอุดมสมบูรณ์ของแหล่งอาหารหรือค่าฟังก์ชันเป้าหมาย

ที่วงวนที่ k ใดๆ การปรับปรุงทิศทางและตำแหน่งของอนุภาคตัวที่ i -th สามารถหาได้จาก

$$\mathbf{x}_i(k) = \mathbf{x}_i(k-1) + \mathbf{v}_i(k) \quad (2.8)$$

โดยที่ $\mathbf{v}_i(k)$ คือ เวกเตอร์ความเร็วของอนุภาค ซึ่งคำนวณได้ดังนี้

$$\mathbf{v}_i(k) = W\mathbf{v}_i(k-1) + C_1r_1(\mathbf{p}_i^{best} - \mathbf{x}_i(k-1)) + C_2r_2(\mathbf{g}_i^{best} - \mathbf{x}_i(k-1)) \quad (2.9)$$

เมื่อ

$\mathbf{x}_i(k)$ คือ ตำแหน่งของอนุภาคหรือเวกเตอร์ผลเฉลยที่ i -th ในวงวนที่ k

$\mathbf{v}_i(k)$ คือ ความเร็วของอนุภาค i -th ในวงวนที่ k

W คือ ค่าถ่วงน้ำหนักใช้สำหรับแสดงผลกระทบของ $\mathbf{v}_i(k-1)$ ที่มีต่อ $\mathbf{v}_i(k)$

$r_1, r_2 \in [0,1]$ คือ ค่าตัวเลขสุ่มที่กระจายแบบเอกรูป

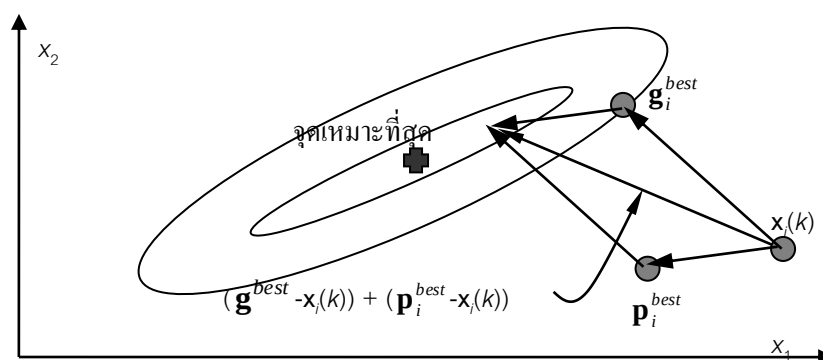
C_1 คือ ตัวประกอบการเรียนรู้ด้วยตนเอง

C_2 คือ ตัวประกอบการเรียนรู้ทางสังคม

$\mathbf{p}_i^{best}$ คือ ผลเฉลยที่ดีที่สุดที่อนุภาค $\mathbf{x}_i$ ค้นเจอ

$\mathbf{g}_i^{best}$ คือ ผลเฉลยที่ดีที่สุดที่อนุภาคทั้งหมดค้นเจอ

รูปที่ 2.9 แสดงทิศทางของเวกเตอร์ความเร็วของอนุภาค $\mathbf{x}_i(k)$ ในสองมิติ ในวงวนที่ k จากรูป ถ้ากำหนดให้ ค่าถ่วงน้ำหนัก $W = 0$ เวกเตอร์ความเร็ว คือเวกเตอร์ที่มีจุดเริ่มต้นที่ $\mathbf{x}_i(k)$ และมีจุดปลายที่จุดใดจุดหนึ่งในสี่เหลี่ยมด้านขนาน ดังแสดงในรูป ทั้งนี้ ขึ้นอยู่กับค่าของ r_1, r_2, C_1 และ C_2 ในกรณีที่ $W = 0$ และ $r_1 = r_2 = C_1 = C_2 = 1$ เวกเตอร์ความเร็วคือ $\mathbf{v}(k) = (\mathbf{g}^{best} - \mathbf{x}_i(k)) + (\mathbf{p}_i^{best} - \mathbf{x}_i(k))$ ดังแสดงในรูปที่ 2.9



รูปที่ 2.9 เวกเตอร์ความเร็วของอนุภาค

ขั้นตอนวิธีของ PSO แสดงในขั้นตอนวิธีที่ 2.4 กระบวนการคำนวณเริ่มต้นด้วยประชากรตั้งต้นหรือกลุ่มของอนุภาคตั้งต้น กำหนดให้เวกเตอร์ความเร็วของอนุภาคตั้งต้นทั้งหมดเป็นเวกเตอร์ศูนย์ (หมายความว่าอนุภาคเริ่มต้นเคลื่อนที่จากตำแหน่งหยุดนิ่ง) จากนั้นทำการหาค่าฟังก์ชันเป้าหมายของ

อนุภาคเหล่านั้น เมื่อทราบค่าของฟังก์ชันเป้าหมายอนุภาคแล้วทำการหาค่า p_i^{best} และ g^{best} จากนั้นคำนวณหาค่าความเร็วของอนุภาคสำหรับวงวนถัดไปโดยใช้สมการ (2.8) และ (2.9) จากนั้นทำการปรับปรุงตำแหน่งของอนุภาคทั้งหมดและปรับปรุง p_i^{best} และ g^{best} แบบทำซ้ำ จนกระทั่งเจอเงื่อนไขหยุดค้นหาผลเฉลย

ขั้นตอนวิธีที่ 2.4 กระบวนการคำนวณของวิธี PSO

ขั้นตอนเริ่มต้น กำหนดพารามิเตอร์ที่จำเป็นทั้งหมด และกลุ่มอนุภาคตั้งต้นพร้อมค่าฟังก์ชันเป้าหมายของสมาชิกในประชากร กำหนดความเร็วเริ่มต้นของอนุภาคเป็นศูนย์

ขั้นตอนคำนวณหลัก

- 1: ถ้าเงื่อนไขการหยุดคำนวณเป็นจริง หยุดการค้นหาผลเฉลย
ถ้าไม่เป็นจริง ดำเนินการต่อในขั้นตอนที่ 2
- 2: คำนวณเวกเตอร์ความเร็วของอนุภาคพร้อมปรับตำแหน่งของอนุภาคตามเวกเตอร์ความเร็ว
- 3: หาค่าฟังก์ชันเป้าหมายของอนุภาคใหม่
- 4: หาค่าของ p_i^{best} และ g^{best}
- 5: กลับไปขั้นตอนที่ 1

2.5 วิธีวิวัฒนาการเชิงอนุพันธ์ (Differential Evolution, DE)

วิธีวิวัฒนาการเชิงอนุพันธ์หรือ DE เป็นวิธีหาค่าเหมาะที่สุดเชิงวิวัฒนาการที่พัฒนาขึ้นมาในยุคใกล้เคียงกับ PBIL และ PSO จุดเด่นของวิธีนี้คือ มีขั้นตอนการคำนวณที่ง่าย ไม่สลับซับซ้อนแต่มีประสิทธิภาพในการค้นหาผลเฉลยเหมาะที่สุดสูง เมื่อเทียบกับขั้นตอนวิธีวิวัฒนาการอื่นๆ กระบวนการคำนวณของ DE จะใช้กลุ่มของเวกเตอร์ผลเฉลยหรือประชากรในการหาค่าตอบเช่นเดียวกับวิธีก่อนหน้านี้ ประชากรจะทำการวิวัฒนาการไปหาผลเฉลยเหมาะที่สุดด้วยผลต่างของเวกเตอร์ตัวแปรออกแบบ จึงเรียกชื่อว่าวิธีวิวัฒนาการเชิงอนุพันธ์ ตัวดำเนินการของ DE สามารถแบ่งได้เป็น การมิวเทชัน ทรอสโอเวอร์ และการคัดเลือก ในที่นี้จะป็นรายละเอียดของ DE/best/1/bin ซึ่งมีดังนี้

2.5.1 มิวเทชัน

ที่รอบของการคำนวณใดๆ กำหนดให้ประชากรวงวนปัจจุบันคือ $P = \{x_1, x_2, \dots, x_N\}$ การสร้างผลเฉลยลูก u_i เพื่อเปรียบเทียบกับผลเฉลยพ่อแม่ x_i โดยการมิวเทชัน สามารถทำได้โดยใช้สมการ

$$u_i = x_{i1} + F(x_{i2} - x_{i3}) \quad (2.10)$$

เมื่อ $F \in [0, 1]$ คือ ตัวประกอบมาตราส่วน (scaling factor)

u_i คือ ผลเฉลยจากการมิวเทชัน

x_{i1} คือ เวกเตอร์ผลเฉลยที่ดีที่สุดในสมาชิกของประชากรวงวนปัจจุบัน

x_{i2} และ x_{i3} คือ เวกเตอร์ผลเฉลยซึ่งเป็นสมาชิกของประชากรที่ได้มาจากการเลือกแบบสุ่ม
ทั้งนี้ $x_{i2} \neq x_{i3}$

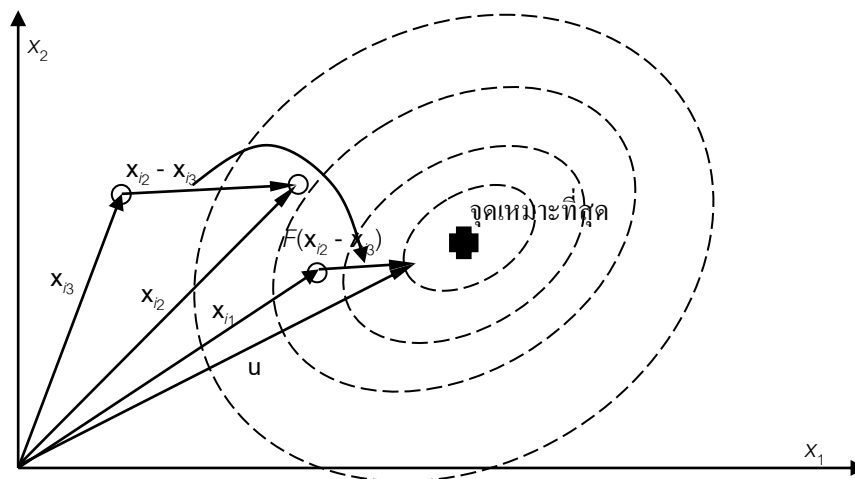
รูปที่ 2.10 แสดงการคำนวณหาเวกเตอร์ผลเฉลยลูก u_i สำหรับปัญหาการหาค่าเหมาะที่สุดในปริภูมิสองมิติ จากรูปจะเห็นได้ว่าตำแหน่งของ u_i ได้จากการเคลื่อนที่จากตำแหน่งของผลเฉลยที่ดีที่สุดไปยังทิศทางซึ่งเป็นผลต่างของเวกเตอร์ 2 เวกเตอร์ที่เลือกขึ้นมาแบบสุ่ม ข้อสังเกตเพิ่มเติมการเลือก x_{i2} และ x_{i3} จะมีผลต่อ u_i มาก เพราะจากตัวอย่างในรูป หากสลับตำแหน่งของเวกเตอร์ทั้งสองจะทำให้ u_i เคลื่อนที่ไปยังตำแหน่งที่แตกต่าง x_{i1}

2.5.2 คrossover

เมื่อได้ u_i จากการมิวเทชันแล้ว ผลเฉลยลูก u_i จะต้องนำมาทำการ crossover กับผลเฉลยพ่อแม่ x_i ซึ่งจะได้ผลเฉลยลูกสุดท้าย v_i ในที่นี้เป็นการ crossover แบบทวิภาค (binary crossover) เอลิเมนต์ที่ j -th ของ v_i เขียนแทนด้วย $v_{i,j}$ สามารถหาได้จาก

$$v_{i,j} = \begin{cases} u_{i,j}; \text{rand} \leq C_R \\ x_{i,j}; \text{rand} > C_R \end{cases} \quad (2.11)$$

เมื่อ $C_R \in [0, 1]$ คือ โอกาสในการถูกเลือกของเอลิเมนต์ของผลเฉลยลูก การ crossover จะถูกดำเนินการด้วยค่าความน่าจะเป็นของการ crossover p_c



รูปที่ 2.10 มิวเทชันของวิธีวิวัฒนาการเชิงอนุพันธ์

2.5.3 การคัดเลือก

เมื่อทำการผลิตเวกเตอร์ผลเฉลยที่เป็นสมาชิกของประชากรลูก v_i ด้วยการมิวเทชันและ crossover ต่อเนื่องกันแล้ว ทำการหาค่าฟังก์ชันเป้าหมายของผลเฉลยลูก จากนั้นเปรียบเทียบกับผลเฉลยพ่อแม่ตำแหน่งเดียวกัน x_i ผลเฉลยที่ดีกว่าจะถูกเลือกเป็นสมาชิกของประชากรในวงวนถัดไป จากนั้นทำ

การสลับตำแหน่งของสมาชิกในประชากรวงวนถัดไปแบบสุ่ม ค่าเงินการเข้าจนกระทั่งได้ค่าที่เหมาะสมที่สุด กระบวนการค้นหาผลเฉลยเหมาะที่สุดด้วยวิธี DE แสดงในขั้นตอนวิธีที่ 2.5

ขั้นตอนวิธีที่ 2.5 กระบวนการคำนวณของวิธี DE

ขั้นตอนเริ่มต้น กำหนดพารามิเตอร์ที่จำเป็นทั้งหมด และสร้างประชากรตั้งต้นพร้อมค่าฟังก์ชันเป้าหมายของสมาชิกในประชากร

ขั้นตอนคำนวณหลัก

- 1: ถ้าเงื่อนไขการหยุดคำนวณเป็นจริง หยุดการค้นหาผลเฉลย
ถ้าไม่เป็นจริง ดำเนินการต่อในขั้นตอนที่ 2
- 2: หาประชากรลูกด้วยการมิวเทชันและครอสโอเวอร์
- 3: หาค่าฟังก์ชันเป้าหมายของประชากรลูก
- 4: ทำการคัดเลือกสมาชิกของประชากรในวงวนถัดไประหว่างประชากรพ่อแม่และประชากรลูก
- 5: สลับตำแหน่งของประชากรชุดใหม่แบบสุ่ม และกลับไปขั้นตอนที่ 1

นับตั้งแต่มีการพัฒนาขั้นตอนวิวัฒนาการในยุคเริ่มต้นซึ่งประกอบด้วยวิธี GA และ (1+1)-ES เป็นต้น จนถึงปัจจุบัน ได้มีการพัฒนาขั้นตอนวิธีวิวัฒนาการหรือเมตาฮิวริสติกส์และนำเสนอในทางวิชาการประมาณหนึ่งพันวิธีหรือมากกว่านี้ ดังนั้น จึงเป็นไปได้ที่จะนำรายละเอียดหรือแม้กระทั่งกล่าวถึงแง่ข้อของวิธีการเหล่านั้นทุกวิธีในบทนี้ ขั้นตอนวิธีวิวัฒนาการที่กล่าวถึงในบทนี้ เป็นส่วนหนึ่งของขั้นตอนวิธีวิวัฒนาการที่มีแนวคิดและรูปแบบในการหาผลเฉลยเหมาะที่สุดที่แตกต่างกัน ซึ่งจะเป็นพื้นฐานที่ดีสำหรับผู้สนใจทางด้านนี้ทั้งในแง่ของการประยุกต์ใช้งานและการวิจัย

2.6 ฟังก์ชันปรับแก้สำหรับขั้นตอนวิธีวิวัฒนาการ

เนื่องจากขั้นตอนวิธีวิวัฒนาการส่วนใหญ่ เป็นวิธีหาค่าเหมาะที่สุดที่พัฒนาขึ้นมาสำหรับการหาค่าเหมาะที่สุดที่มีขอบเขตของตัวแปรออกแบบ ดังนั้น ในกรณีของการประยุกต์ใช้วิธีเหล่านี้กับปัญหาการออกแบบที่มีเงื่อนไขบังคับชนิดอื่น จำเป็นต้องมีการประยุกต์ใช้ฟังก์ชันปรับแก้เพื่อแปลงปัญหาที่มีเงื่อนไขบังคับไปเป็นปัญหาการหาค่าเหมาะที่สุดแบบไม่มีเงื่อนไขบังคับที่สมมูลกัน ได้มีการพัฒนาวิธีฟังก์ชันปรับแก้สำหรับขั้นตอนวิธีวิวัฒนาการหลากหลายรูปแบบ วิธีการที่ได้รับความนิยมมากที่สุดคือวิธีฟังก์ชันปรับแก้ภายนอก ในหัวข้อนี้จะนำเสนอวิธีฟังก์ชันปรับแก้ 3 รูปแบบที่ผู้วิจัยเห็นว่า มีประโยชน์จริง มีประสิทธิภาพและประสิทธิผลสำหรับการหาค่าเหมาะที่สุดด้วยขั้นตอนวิธีวิวัฒนาการ ซึ่งประกอบด้วยวิธีฟังก์ชันปรับแก้ภายนอก วิธีฟังก์ชันปรับแก้โดยใช้ทฤษฎีเซตคลุมเครือ (fuzzy set theory) และสุดท้ายเป็นวิธีฟังก์ชันปรับแก้สำหรับการหาค่าเหมาะที่สุดวงกว้าง (global optimisation) ซึ่งเรียกว่า กลไกการมีส่วนร่วม (sharing mechanism)

2.6.1 วิธีฟังก์ชันปรับแก้ภายนอก

เนื่องจากปัญหาการออกแบบที่มีเงื่อนไขบังคับส่วนมากเป็นปัญหาที่มีเฉพาะเงื่อนไขบังคับแบบอสมการ (หรือถ้ามีเงื่อนไขบังคับแบบสมการก็สามารถจัดการได้ด้วยแนวทางอื่นๆ) ดังนั้นการนำเสนอวิธีฟังก์ชันปรับแก้ในที่นี้จะพิจารณาเฉพาะปัญหาการออกแบบที่มีเฉพาะเงื่อนไขบังคับแบบอสมการเท่านั้น ซึ่งสามารถแสดงได้ดังนี้

จงหาค่าต่ำสุดของ

$$f(\mathbf{x}) \quad (2.12)$$

โดยมีเงื่อนไขบังคับ

$$\begin{aligned} g_i(\mathbf{x}) &\leq 0, \quad i = 1, \dots, m \\ L_i &\leq x_i \leq U_i, \quad i = 1, \dots, n \end{aligned}$$

แนวคิดของฟังก์ชันปรับแก้ คือการสร้างฟังก์ชันเป้าหมายสมมูลที่ไม่มีเงื่อนไขบังคับ เมื่อค่าของฟังก์ชันสมมูลลดและเพิ่ม ย่อมส่งผลให้ค่าฟังก์ชันเป้าหมายเดิมลดและเพิ่มตามไปด้วย ในขณะเดียวกัน หากผลเฉลยเป็นผลเฉลยที่เป็นไปได้ (feasible solution) ค่าของฟังก์ชันเป้าหมายสมมูลควรมีค่าเท่ากับหรือสัมพันธ์กับค่าฟังก์ชันเป้าหมายเดิม แต่ถ้าผลเฉลยอยู่นอกพื้นที่หาคำตอบได้ ค่าฟังก์ชันสมมูลควรมีค่าสูงขึ้นเพื่อให้วิธีหาค่าเหมาะที่สุดเคลื่อนย้ายผลเฉลยออกจากบริเวณนั้น ฟังก์ชันปรับแก้ภายนอกสามารถเขียนอยู่ในรูปของ

$$f'(\mathbf{x}) = f(\mathbf{x}) + p(\mathbf{x}) \quad (2.13)$$

เมื่อ f' คือ ฟังก์ชันสมมูลหรือฟังก์ชันปรับแก้ และ p คือเทอมปรับแก้ (penalty term) ซึ่งสามารถหาได้จาก

$$p(\mathbf{x}) = r \sum_{i=1}^m p'_i(\mathbf{x}) \quad (2.14)$$

เมื่อ

$$p'_i(\mathbf{x}) = \begin{cases} 0 & ; g_i(\mathbf{x}) \leq 0 \\ g_i^2(\mathbf{x}) & ; g_i(\mathbf{x}) > 0 \end{cases}$$

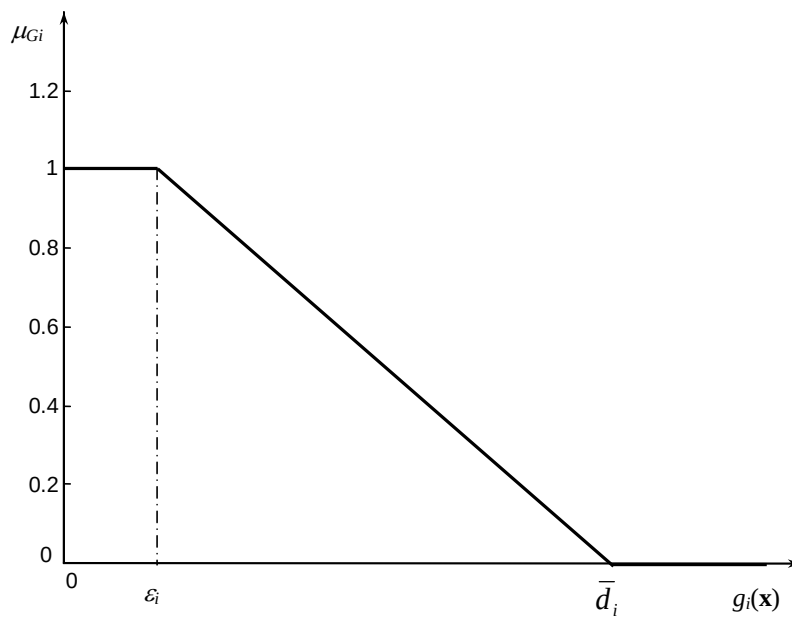
โดยทั่วไป ค่าพารามิเตอร์ปรับแก้ r จะตั้งค่าต่ำในช่วงเริ่มต้นหาค่าเหมาะที่สุด และจะมีค่าสูงขึ้นเมื่อจำนวนของวงวนเพิ่มมากขึ้น

2.6.2 วิธีฟังก์ชันปรับแก้โดยใช้ทฤษฎีเซตคลุมเครือ

สำหรับการประยุกต์ใช้ทฤษฎีเซตคลุมเครือ ที่จุด $\mathbf{x}$ ใดๆ เราสามารถคำนวณค่าฟังก์ชันภาวะสมาชิก (membership function) ของ $\mathbf{x}$ สำหรับเงื่อนไขบังคับ g_i เขียนแทนด้วย μ_{G_i} ซึ่งมีนิยามดังนี้

$$\mu_{G_i}(\mathbf{x}) = \begin{cases} 0; & g_i(\mathbf{x}) \geq \bar{d}_i \\ 1 - \frac{g_i(\mathbf{x}) - \varepsilon_i}{\bar{d}_i}; & \varepsilon_i < g_i(\mathbf{x}) < \bar{d}_i \\ 1; & g_i(\mathbf{x}) \leq \varepsilon_i \end{cases} \quad (2.15)$$

เมื่อ ε_i และ $\bar{d}_i$ คือ พารามิเตอร์ที่กำหนดเพื่อระบุพื้นที่หาคำตอบได้ของเงื่อนไขบังคับ g_i รูปที่ 4.13 แสดงรูปทรงของฟังก์ชันภาวะสมาชิกซึ่งจัดว่าเป็นฟังก์ชันต่อเนื่องเป็นช่วงๆ (piecewise continuous)



รูปที่ 2.11 ฟังก์ชันภาวะสมาชิก

เมื่อกำหนดค่าฟังก์ชันภาวะสมาชิกของเงื่อนไขบังคับทั้งหมดที่จุด $\mathbf{x}$ แล้ว จากนั้นหาค่าฟังก์ชัน μ_C สำหรับผลเฉลย $\mathbf{x}$ ได้จากความสัมพันธ์

$$\mu_C(\mathbf{x}) = \mu_1(\mathbf{x}) \cap \mu_2(\mathbf{x}) \cap \cdots \cap \mu_m(\mathbf{x}). \quad (2.16)$$

ฟังก์ชันปรับแก้โดยใช้ทฤษฎีเซตคลุมเครือ สามารถหาได้จาก

$$f'(\mathbf{x}) = \frac{f(\mathbf{x})}{c + f(\mathbf{x})} + (1 - \mu_C(\mathbf{x})) \cdot K_D. \quad (2.17)$$

เมื่อ K_D คือ ค่าสัมประสิทธิ์ปรับแก้ และ $c > 0$ คือ ตัวเลขที่ผู้ออกแบบต้องกำหนด

ผลเฉลยต่ำสุดของฟังก์ชัน $F_z = f(\mathbf{x})/(c + f(\mathbf{x}))$ จะมีค่าเดียวกันกับผลเฉลยต่ำสุดของ $f(\mathbf{x})$ ยกเว้นกรณีที่ $f(\mathbf{x}) = c$ ตัวเลข c ที่เป็นบวก ทำให้ค่าฟังก์ชันเป้าหมายสมมูลของผลเฉลยที่เป็นไปได้มีค่าน้อยกว่า 1 เสมอ เนื่องจากค่าฟังก์ชัน F_z มีค่าไม่เกิน 1 ทำให้ง่ายในการกำหนดค่า K_D เทียบกับวิธีฟังก์ชันปรับแก้ภายนอก ซึ่งมักมีปัญหาในการกำหนดค่าพารามิเตอร์ปรับแก้ r ส่งผลให้ ผลเฉลยเหมาะที่สุดที่ได้มักอยู่นอกพื้นที่หาคำตอบได้เสมอ

2.6.3 กลไกการมีส่วนร่วม (sharing mechanism)

กลไกการมีส่วนร่วม เป็นวิธีฟังก์ชันปรับแก้สำหรับการหาค่าเหมาะที่สุดที่มีผลเฉลยเหมาะที่สุดของทั้งหมด (global optimum) หลักการของฟังก์ชันปรับแก้คือการปรับเปลี่ยนค่าฟังก์ชันของผลเฉลยเดิมให้มีค่าสูงขึ้น (กรณีของการหาค่าต่ำสุด) หากผลเฉลยนั้นอยู่ในตำแหน่งที่ใกล้กับผลเฉลยตัวอื่น หากจำนวนผลเฉลยที่อยู่ใกล้กับผลเฉลย $\mathbf{x}$ ใดๆ มีจำนวนมาก ค่าของฟังก์ชันปรับแก้จะมีค่าเพิ่มขึ้นตามจำนวนเวกเตอร์ผลเฉลยใกล้เคียงเหล่านั้น กำหนดให้ประชากรของผลเฉลยของขั้นตอนวิธีวิวัฒนาการคือ $\mathbf{P} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ ฟังก์ชันการมีส่วนร่วมของจุด $\mathbf{x}_i$ เทียบกับ $\mathbf{x}_j$ คือ

$$sh(d_{ij}) = \begin{cases} 1 - (\frac{d_{ij}}{\sigma_s})^\alpha; d_{ij} < \sigma_s \\ 0; d_{ij} \geq \sigma_s \end{cases} \quad (2.18)$$

เมื่อ d_{ij} คือ ระยะห่างระหว่างจุด $\mathbf{x}_i$ กับจุด $\mathbf{x}_j$ ใดๆที่เป็นสมาชิกของ $\mathbf{P}$

σ_s คือ ระยะการปรับแก้

α คือ เลขยกกำลังของฟังก์ชันการมีส่วนร่วม

σ_s คือ ค่ากำหนดขอบเขตที่ยอมให้ผลเฉลย $\mathbf{x}_j$ เข้าใกล้ $\mathbf{x}_i$ ได้ หากระยะระหว่างผลเฉลยและ $\mathbf{x}_i$ มีค่าน้อยกว่า σ_s ค่าฟังก์ชันการมีส่วนร่วมของ $\mathbf{x}_i$ เทียบกับ $\mathbf{x}_j$ จะมีค่ามากกว่าศูนย์ คุณสมบัติอื่นของฟังก์ชันการมีส่วนร่วม มีดังนี้

- $0 \leq sh(d_{ij}) \leq 1$
- $sh(0) = 1$
- $\lim_{d_{ij} \rightarrow \infty} sh(d_{ij}) = 0$

เมื่อกำหนดค่าฟังก์ชันการมีส่วนร่วมของ $\mathbf{x}_i$ เทียบกับสมาชิกทุกตัวในเมตริกซ์ประชากร $\mathbf{P}$ แล้วจะได้ค่าการนับช่องว่าง (niche count) ของ $\mathbf{x}_i$ เป็น

$$\hat{m}_i = \sum_{j=1}^N sh(d_{ij}) \quad (2.19)$$

ค่าการนับช่องว่างนี้คือฟังก์ชันปรับแก้ ในการปรับปรุงฟังก์ชันเป้าหมายเดิมด้วยค่าฟังก์ชันปรับแก้นี้สามารถทำได้สองแบบคือ การมีส่วนร่วมโดยการคูณ

$$\hat{f}_i = \hat{m}_i f_i \quad (2.20)$$

หรือการมีส่วนร่วมโดยการบวก

$$\hat{f}_i = f_i + W(\hat{m}_i - 1) \quad (2.21)$$

เมื่อ W คือ ค่าสัมประสิทธิ์การปรับแก้

บทที่ 3 ขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมาย

3.1 ความนำ

ในบทที่ผ่านมา เป็นการนำเสนอการหาค่าเหมาะที่สุดที่มีฟังก์ชันเป้าหมายเพียงเป้าหมายเดียว ตัวอย่างเช่น การออกแบบเพื่อลดน้ำหนักหรือราคาของชิ้นงานให้น้อยที่สุด โดยมีเงื่อนไขคือโครงสร้างจะต้องไม่เสียหายจากแรงหรือภาระกระทำ อย่างไรก็ตามในทางปฏิบัติ ฟังก์ชันเป้าหมายไม่ได้มีเฉพาะน้ำหนักหรือราคาของชิ้นงานเท่านั้น ผู้ออกแบบอาจต้องการหาน้ำหนักน้อยสุด และในขณะเดียวกันยังต้องการให้โครงสร้างของชิ้นงานมีความแข็งแรงมากที่สุด และมีเงื่อนไขบังคับคือความปลอดภัยในการใช้งาน ปัญหาลักษณะนี้เรียกว่า การหาค่าเหมาะที่สุดแบบหลายเป้าหมาย (multiobjective optimisation) ซึ่งเป็นปัญหาที่เกิดขึ้นบ่อยครั้งในการออกแบบระบบทางวิศวกรรม

การหาผลเฉลยของปัญหาการหาค่าเหมาะที่สุดรูปแบบนี้ มีความแตกต่างจากการหาผลเฉลยของปัญหาการออกแบบที่มีฟังก์ชันเป้าหมายเดียว ในอดีต วิธีการหาค่าเหมาะที่สุดที่พัฒนาขึ้นมาส่วนใหญ่ใช้ในการหาผลเฉลยของปัญหาการออกแบบเป้าหมายเดียว ดังที่ได้กล่าวถึงในบทที่ผ่านมา อย่างไรก็ตาม เราสามารถประยุกต์ใช้วิธีเหล่านั้นกับปัญหาการออกแบบหลายเป้าหมาย โดยการเพิ่มเทคนิคเชิงตัวเลขพิเศษ เช่น วิธีผลรวมถ่วงน้ำหนัก (weighted sum method) ข้อเสียของกระบวนการออกแบบลักษณะนี้คือ ผู้ออกแบบจะต้องรันการหาค่าเหมาะที่สุดเป็นจำนวนหลายครั้งเท่ากับจำนวนผลเฉลยที่เราต้องการ

องค์ความรู้ทางด้านการหาผลเฉลยของปัญหาการหาค่าเหมาะที่สุดแบบหลายเป้าหมาย ได้มีการพัฒนาครั้งใหญ่เมื่อมีการพัฒนาวิธีการหาค่าเหมาะที่สุดแบบวิวัฒนาการ หรือขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมาย (multiobjective evolutionary algorithms, MOEAs) เพื่อใช้ในการหาผลเฉลยเหมาะที่สุดของปัญหาแบบหลายเป้าหมาย ข้อดีของการประยุกต์ใช้วิธีการเหล่านี้คือ เป็นวิธีที่ประยุกต์ใช้งานง่าย สามารถใช้หาผลเฉลยของปัญหาการหาค่าเหมาะที่สุดได้เกือบทุกรูปแบบ เป็นวิธีหาค่าเหมาะที่สุดที่สามารถรันได้โดยไม่ต้องมีการสะดุดหรือเกิดข้อผิดพลาดในระหว่างกระบวนการคำนวณและที่สำคัญที่สุด ขั้นตอนวิธีวิวัฒนาการที่พัฒนาขึ้นมาในปัจจุบันสามารถหากลุ่มของผลเฉลยพาเรโตได้ในการรันเพียงครั้งเดียวซึ่งความสามารถตรงนี้ทำให้วิธีวิวัฒนาการประเภทนี้ได้รับความนิยมสูง อย่างไรก็ตาม วิธีการเหล่านี้ยังมีข้อเสียที่หลีกเลี่ยงไม่ได้กล่าวคือ มีอัตราการลู่เข้า (convergence rate) ต่ำ และไม่มีความต้องกัน (consistency) ในการหาผลเฉลย (ไม่สามารถหาคำตอบชุดเดิมได้เสมอ ในการรันหลายๆครั้ง) ทั้งนี้เป็นเพราะว่า วิธีการเหล่านี้ใช้การสุ่มเป็นหลักในกระบวนการคำนวณ

3.2 แนวคิดเบื้องต้น

ปัญหาคณิตศาสตร์สำหรับการหาค่าเหมาะที่สุดแบบหลายเป้าหมาย สามารถแสดงได้ดังนี้

จงหาเวกเตอร์ตัวแปรออกแบบ $\mathbf{x}$ ที่ให้ค่าต่ำสุดของเซตของฟังก์ชันเป้าหมาย

$$\mathbf{f} = \{f_1(\mathbf{x}), \dots, f_p(\mathbf{x})\} \quad (3.1)$$

โดยมีเงื่อนไขบังคับ

$$\begin{aligned} g_i(\mathbf{x}) &\leq 0 \\ h_i(\mathbf{x}) &= 0 \end{aligned}$$

เมื่อ $\mathbf{x}$ คือ เวกเตอร์ตัวแปรออกแบบ

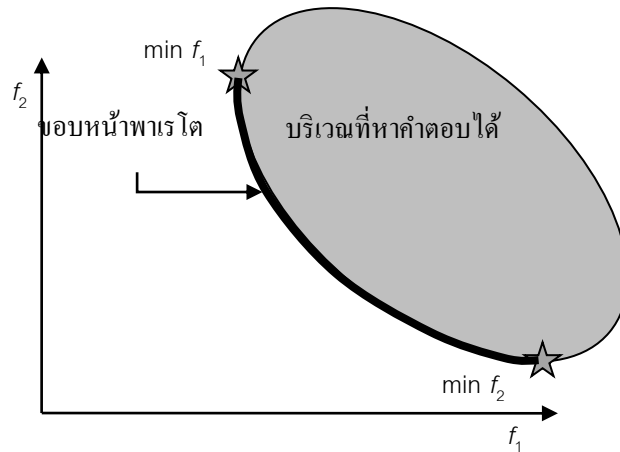
f_i คือ ฟังก์ชันเป้าหมาย p ค่า

g_i คือ เงื่อนไขบังคับอสมการ

h_i คือ เงื่อนไขบังคับสมการ

รูปที่ 3.1 แสดงปัญหาการออกแบบเหมาะที่สุดที่มีสองฟังก์ชันเป้าหมาย จากรูปเป็นการพล็อตผลเฉลยทั้งหมดลงบนโดเมนของฟังก์ชันเป้าหมาย เนื่องจากผลเฉลย $\mathbf{x}$ ใดๆ จะให้ค่าฟังก์ชันเป้าหมาย $\{f_1, f_2\}$ ดังนั้น ตำแหน่งของผลเฉลยใดๆ จะให้พิกัด (f_1, f_2) หรือจุดใดๆ บนระบบพิกัด $f_1 - f_2$ จากรูปแสดงกลุ่มของผลเฉลยที่เป็นไปได้ (feasible solutions) หรือผลเฉลยที่สอดคล้องกับเงื่อนไขบังคับทั้งหมดด้วยพื้นที่แรเงา ดังนั้น ผลเฉลยเหมาะที่สุดของปัญหาการหาค่าเหมาะที่สุดแบบสองเป้าหมายนี้จะอยู่ในพื้นที่แรเงาซึ่งเรียกว่า บริเวณหาค่าตอบได้ (feasible region) ดังที่ได้นิยามไว้แล้วในบทที่ผ่านมา

จากรูปที่ 3.1 จุด $\min f_1$ คือผลเฉลยเหมาะที่สุดในกรณีที่ทำกรหาค่าต่ำสุดเฉพาะฟังก์ชันเป้าหมาย f_1 ในทำนองเดียวกัน จุด $\min f_2$ คือผลเฉลยเหมาะที่สุดในกรณีที่ทำกรหาค่าต่ำสุดของฟังก์ชันเป้าหมาย f_2 จะเห็นได้ว่า ผลเฉลยของการหาค่าเหมาะที่สุดที่มีฟังก์ชันเป้าหมายเดียว คำตอบเหมาะที่สุดจะมีคำตอบเดียว แต่เมื่อพิจารณาการหาค่าต่ำสุดของฟังก์ชันเป้าหมายทั้งสองพร้อมกันพบว่า ผลเฉลยที่อยู่ขอบของบริเวณหาค่าตอบได้และอยู่บนเส้นโค้งที่ลากเชื่อมระหว่างจุด $\min f_1$ และจุด $\min f_2$ เป็นผลเฉลยที่ดีกว่าผลเฉลยอื่นๆ ในบริเวณหาค่าตอบได้ และกลุ่มผลเฉลยเหล่านี้ (จุดที่อยู่บนเส้นทึบในรูปที่ 3.1) จัดว่าเป็นผลเฉลยเหมาะที่สุดของการหาค่าเหมาะที่สุดแบบสองเป้าหมาย เราเรียกเซตของจุดผลเฉลยที่อยู่บนเส้นโค้งทึบในรูปนี้ว่า ขอบหน้าพारेโต (Pareto front) (ตั้งชื่อตาม Vilfredo Pareto นักเศรษฐศาสตร์ชาวอิตาลี) ส่วนเซตของผลเฉลยที่ให้ขอบหน้าพारेโตนี้เรียกว่า เซตของผลเฉลยเหมาะที่สุดแบบพारेโต (Pareto optimal set) ในกรณีที่ปัญหาการออกแบบมีสามฟังก์ชันเป้าหมาย ขอบหน้าพारेโตส่วนใหญ่จะเป็นรูปพื้นผิว (surfaces) แต่ก็มีบางกรณีที่เป็นเส้นโค้ง เหมือนกับการหาค่าเหมาะที่สุดแบบสองเป้าหมาย



รูปที่ 3.1 ขอบหน้าพาเรโตสำหรับปัญหาการออกแบบสองเป้าหมาย

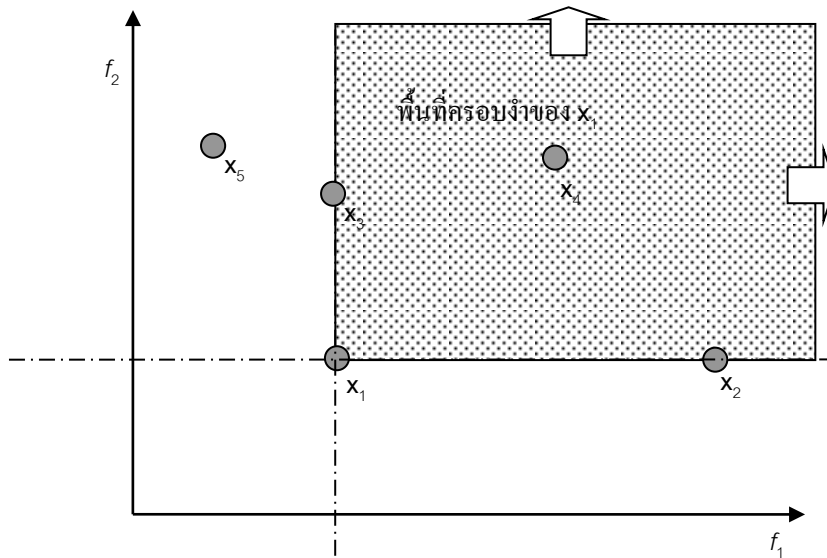
3.3 ขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมาย

วิธีผลรวมถ่วงน้ำหนักและวิธีเชิงตัวเลขในรูปแบบที่คล้ายกันมีจุดด้อยที่หลีกเลี่ยงไม่ได้เลยคือ ต้องทำการหาค่าเหมาะที่สุดหลายครั้งเพื่อหาขอบหน้าพาเรโตของปัญหาการออกแบบ อย่างไรก็ตาม เมื่อสองทศวรรษที่ผ่านมา ได้มีการพัฒนาขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมายขึ้นมา ทำให้เกิดกระบวนการหาค่าเหมาะที่สุดแบบหลายเป้าหมายในรูปแบบที่เปลี่ยนไป จุดเด่นที่สุดของขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมายคือ สามารถหาขอบหน้าพาเรโตของปัญหาการออกแบบได้ภายในการรันเพียงครั้งเดียว ความสามารถนี้มีในขั้นตอนวิธีวิวัฒนาการและเมตะฮิวริสติกส์อื่นๆที่ใช้กลุ่มของผลเฉลยหรือประชากรในการหาค่าเหมาะที่สุด เมื่อมีกลุ่มเวกเตอร์ผลเฉลย สิ่งที่เราเรียกว่าผลเฉลยที่ไม่ถูกครอบงำ (non-dominated solutions) สามารถหาได้จากกลุ่มผลเฉลยนี้ เมื่อขั้นตอนวิธีวิวัฒนาการดำเนินไป เซตของผลเฉลยที่ไม่ถูกครอบงำจะได้รับการปรับปรุงแบบทำซ้ำจนกระทั่งได้ขอบหน้าพาเรโต

นิยามสำคัญ สำหรับการค้นหาคำตอบด้วยหลักการคัดสรรผลเฉลยที่ไม่ถูกครอบงำ (non-dominated sorting) ในกรณีของการหาค่าต่ำสุด มีดังนี้

นิยาม 1 การครอบงำ (dominance) กำหนดให้ $f_i(\mathbf{x})$; $i = 1, \dots, p$ คือฟังก์ชันเป้าหมาย ถ้า $f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2)$ สำหรับทุกดัชนี $i \in \{1, \dots, p\}$ และมี j อย่างน้อยหนึ่งค่า โดยที่ $f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2)$ สามารถสรุปได้ว่า $\mathbf{x}_1$ ครอบงำ $\mathbf{x}_2$ เขียนแทนด้วย $\mathbf{x}_1 \prec \mathbf{x}_2$

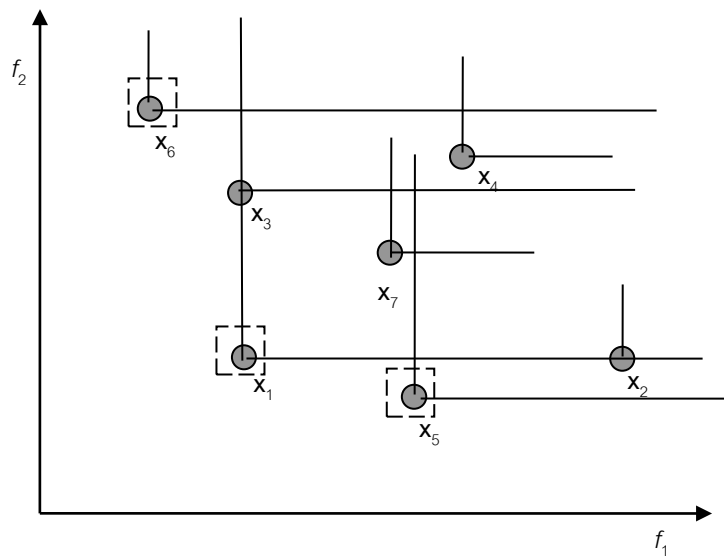
รูปที่ 3.2 แสดงผลเฉลยในโดเมนของฟังก์ชันเป้าหมายสองฟังก์ชัน จากรูปแสดงให้เห็นว่าผลเฉลย $\mathbf{x}_1$ ครอบงำผลเฉลย $\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$ ในขณะที่ $\mathbf{x}_5$ ไม่ถูกครอบงำโดย $\mathbf{x}_1$ ในรูปยังแสดงให้เห็นถึงพื้นที่ครอบงำของผลเฉลย $\mathbf{x}_1$ ซึ่งหมายความว่า ผลเฉลยใดๆ ที่อยู่ในขอบเขตพื้นที่ที่ครอบงำนี้จะถูกครอบงำโดย $\mathbf{x}_1$



รูปที่ 3.2 พื้นที่ครอบงำของผลเฉลย x_1

นิยาม 2 ผลเฉลยที่ไม่ถูกรอบงำ กำหนดให้ G คือเซตของผลเฉลยขนาด N , ผลเฉลย $x_e \in G$ เรียกว่า ผลเฉลยที่ไม่ถูกรอบงำใน G ถ้าไม่มีผลเฉลย $x \in G$ ใดๆ ที่ครอบงำ x_e

จากรูปที่ 3.3 ผลเฉลยที่ไม่ถูกรอบงำคือ x_1 , x_5 และ x_6



รูปที่ 3.3 การคัดสรรผลเฉลยที่ไม่ถูกรอบงำ

ในกระบวนการคำนวณเชิงตัวเลข การหาผลเฉลยที่ไม่ถูกรอบงำสามารถทำได้โดยใช้เมตริกซ์การครอบงำ (dominance matrix) A ซึ่งแสดงวิธีการหาในตารางที่ 3.1 เอลิเมนต์ A_{ij} สามารถหาได้ดังนี้

$$A_{ij} = \begin{cases} 1; x_i \prec x_j \\ 0; \text{otherwise} \end{cases} \quad (3.2)$$

เมื่อได้เมตริกซ์ของการครอบงำแล้ว คำนวณค่าการถูกครอบงำของผลเฉลย x_j ได้ดังนี้

$$B_j = \sum_{i=1}^N A_{ij} \quad (3.3)$$

ผลเฉลย x_j เรียกว่าผลเฉลยที่ไม่ถูกครอบงำถ้า $B_j = 0$

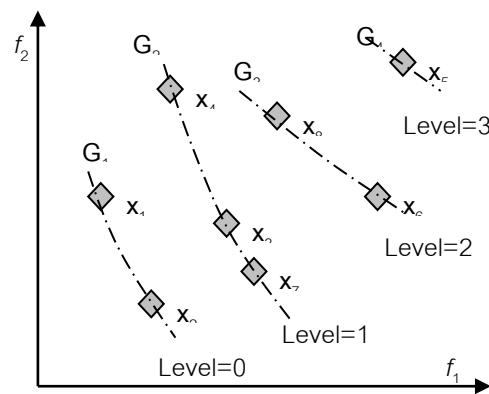
ตารางที่ 3.1 เมตริกซ์สำหรับคัดสรรผลเฉลยที่ไม่ถูกครอบงำ A

A	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	0	1	1	1	0	0	1
x_2	0	0	0	0	0	0	0
x_3	0	0	0	1	0	0	0
x_4	0	0	0	0	0	0	0
x_5	0	1	0	1	0	0	0
x_6	0	0	0	0	0	0	0
x_7	0	0	0	1	0	0	0
รวม (B)	0	2	1	4	0	0	1

นิยาม 3 ระดับของการถูกครอบงำ (level of being dominated) กำหนดให้ G_1 คือเซตของผลเฉลยที่ไม่ถูกครอบงำของ G และ G_2 คือเซตของผลเฉลยที่ไม่ถูกครอบงำของ $G - G_1$ และ G_n คือเซตของผลเฉลยที่ไม่ถูกครอบงำของ $G - \bigcup_{i=1}^{n-1} G_i$ เรากล่าวได้ว่า G_n มีระดับการถูกครอบงำเป็น $n-1$

รูปที่ 3.4 แสดงการคัดสรรผลเฉลยที่ไม่ถูกครอบงำของกลุ่มประชากร $G = \{x_1, \dots, x_8\} = \{G_1, G_2, G_3, G_4\}$ เมื่อ $G_1 = \{x_1, x_2\}$ มีระดับของการถูกครอบงำเป็น 0 $G_2 = \{x_3, x_4, x_7\}$ มีระดับของการถูกครอบงำเป็น 1 $G_3 = \{x_6, x_8\}$ มีระดับของการถูกครอบงำเป็น 2 และ $G_4 = \{x_5\}$ มีระดับของการถูกครอบงำเป็น 3 โดยนิยามแล้ว ถ้าเราลบเซตของผลเฉลย $G_1, \dots, G_n$ ออกจากประชากร เซตของผลเฉลย G_{n+1} จะเป็นผลเฉลยที่ไม่ถูกครอบงำของประชากรที่เหลืออยู่นั่นเอง

จากนิยาม ระดับการถูกครอบงำสามารถหาได้จากการเรียงลำดับค่า B_j โดยกลุ่มผลเฉลยที่มีค่า B_j เป็น 0 คือสมาชิกของเซต G_1 และกลุ่มผลเฉลยที่มีค่า B_j ลำดับถัดไปคือเซต G_2 ตามลำดับ จากตารางที่ 3.1 สรุปได้ว่า $G_1 = \{x_1, x_5, x_6\}$ มีระดับการถูกครอบงำเป็น 0 $G_2 = \{x_3, x_7\}$ มีระดับการถูกครอบงำเป็น 1 $G_3 = \{x_2\}$ มีระดับการถูกครอบงำเป็น 2 และ $G_4 = \{x_4\}$ มีระดับการถูกครอบงำเป็น 3



รูปที่ 3.4 ระดับของการถูกครอบงำ

กระบวนการค้นหาขอบหน้าพาเรโต ของขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมายในรูปแบบทั่วไป แสดงในขั้นตอนวิธีที่ 3.1 กระบวนการคำนวณเริ่มต้นด้วยประชากรและหน่วยเก็บผลเฉลยพาเรโตหรือผลเฉลยที่ไม่ถูกครอบงำ (Pareto archive) จากนั้นทำการผลิตประชากรลูก (offspring) จากประชากรพ่อแม่ เมื่อได้ประชากรลูกแล้วทำการหาค่าฟังก์ชันเป้าหมายของผลเฉลยในประชากรลูก จากนั้นทำการคัดสรรผลเฉลยที่ไม่ถูกครอบงำจากยูเนียนเซตของประชากรลูกและสมาชิกในหน่วยเก็บผลเฉลยที่ไม่ถูกครอบงำเดิม เมื่อได้กลุ่มของผลเฉลยที่ไม่ถูกครอบงำชุดใหม่ ใส่แทนที่ผลเฉลยที่ไม่ถูกครอบงำชุดเก่าในเมตริกซ์ ทำการปรับปรุงเมตริกซ์บรรจุผลเฉลยที่ไม่ถูกครอบงำไปเรื่อยๆจนกระทั่งหยุดการค้นหาคำตอบ ผลเฉลยพาเรโตจากการใช้วิธีวิวัฒนาการเหล่านี้คือสมาชิกในหน่วยเก็บผลเฉลยที่ไม่ถูกครอบงำนั่นเอง อย่างไรก็ตาม ผลเฉลยนี้เรียกว่า ค่าประมาณของขอบหน้าพาเรโต (approximate Pareto front) เท่านั้น เพราะไม่มีการรับประกันว่าวิธีวิวัฒนาการจะหาขอบหน้าพาเรโตที่แม่นยำได้

ในกระบวนการคำนวณของ MOEAs สิ่งที่เกิดขึ้นบ่อยคือ จำนวนของผลเฉลยที่ไม่ถูกครอบงำที่เก็บในหน่วยเก็บผลเฉลยพาเรโตจะมีจำนวนมากเกินไปและส่งผลให้คอมพิวเตอร์ใช้หน่วยความจำสูงมากเพื่อเก็บตัวแปรเมตริกซ์เหล่านี้ เพื่อแก้ปัญหาดังกล่าว ได้มีการประยุกต์ใช้วิธีเชิงตัวเลขเพื่อคัดกรองผลเฉลยบางส่วนทิ้ง โดยการคัดกรองนี้อยู่ภายใต้แนวคิดที่ว่า ผลเฉลยที่ยังคงอยู่ในหน่วยเก็บผลเฉลยต้องมีความหลากหลายมากที่สุดเท่าที่จะเป็นไปได้ จากขั้นตอนวิธีที่ 3.1 เราสามารถพัฒนา MOEAs ขึ้นมาใหม่หรือปรับปรุงวิธีการเดิมที่มีอยู่แล้วให้ดีขึ้น โดยสามารถปรับเปลี่ยนหรือนำแนวคิดใหม่ มาใช้ในขั้นตอนที่ 2 และ 4 ด้วยเหตุนี้ ในปัจจุบันจึงมี MOEAs ที่พัฒนาขึ้นมาใหม่เป็นจำนวนมาก ซึ่งมีทั้งที่ใช้แนวคิดหรือปรัชญาใหม่ การปรับปรุงของเดิมที่มีมาก่อน และการผสมแนวคิดที่แตกต่างกันหลายแนวคิดเพื่อให้ได้สิ่งที่ดีกว่า

ขั้นตอนวิธีที่ 3.1 กระบวนการค้นหาขอบหน้าพารโตของ MOEAs

ขั้นตอนเริ่มต้น กำหนดพารามิเตอร์ที่จำเป็นทั้งหมด หน่วยเก็บผลเฉลยพารโตตั้งต้น และประชากรตั้งต้นพร้อมค่าฟังก์ชันเป้าหมายของสมาชิกในประชากร

ขั้นตอนคำนวณหลัก

- 1: ถ้าเงื่อนไขการหยุดคำนวณเป็นจริง หยุดการค้นหาผลเฉลย
- ถ้าไม่เป็นจริง ดำเนินการต่อในขั้นตอนที่ 2
- 2: ผลิตประชากรลูกจากประชากรพ่อแม่
- 3: คัดสรรผลเฉลยที่ไม่ถูกครอบงำของกลุ่มผลเฉลยรวมกันระหว่าง สมาชิกในประชากรลูกและสมาชิกในหน่วยเก็บผลเฉลยพารโต
- 4: ถ้าผลเฉลยที่ไม่ถูกครอบงำในขั้นตอนที่ 3 มีจำนวนมากเกินไป ทำการลบผลเฉลยบางส่วนออกด้วยวิธีการของผลเฉลย
- 5: แทนที่สมาชิกในหน่วยเก็บผลเฉลยพารโตเดิม ด้วยกลุ่มผลเฉลยในขั้นตอนที่ 4
- 6: กลับไปขั้นตอนที่ 1

3.4 วิธีการเรียนรู้แบบเพิ่มพูนโดยอาศัยกลุ่มประชากรแบบหลายเป้าหมาย

วิธี PBIL สำหรับการหาค่าเหมาะที่สุดแบบเป้าหมายเดียวได้นำเสนอในบทที่ผ่านมา ในบทนี้ PBIL จะถูกดัดแปลงเพื่อเป็น MOEA โดยมีชื่อย่อคือ MPBIL ซึ่งย่อมาจากคำเต็มคือ multiobjective population-based incremental learning หลักการหาผลเฉลยของ MPBIL มีรูปแบบคล้ายกันกับ PBIL กล่าวคือ ใช้เวกเตอร์ความน่าจะเป็นเป็นตัวแทนของประชากรเลขฐานสอง ในกรณีของ MPBIL เราใช้เวกเตอร์ของความน่าจะเป็นมากกว่าหนึ่งเวกเตอร์ เพื่อความหลากหลาย (diversity) ในการหาผลเฉลยพารโต ดังนั้น เมื่อมีเวกเตอร์ความน่าจะเป็นมากกว่าหนึ่งเวกเตอร์จึงเรียกกลุ่มเวกเตอร์นั้นว่า เมตริกซ์ความน่าจะเป็น (probability matrix)

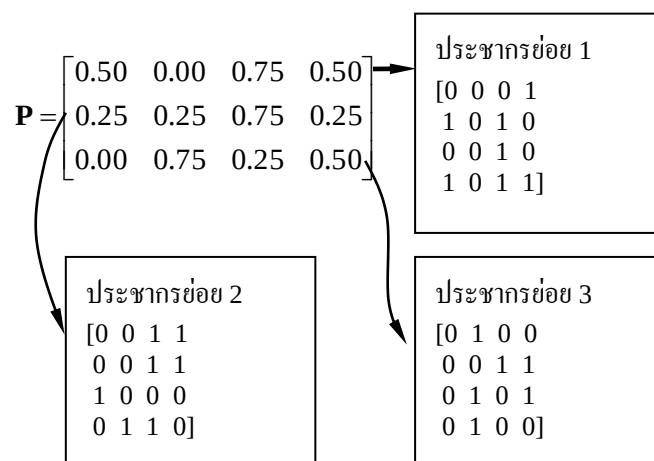
กระบวนการค้นหาคำตอบของ MPBIL เริ่มด้วยหน่วยเก็บผลเฉลยพารโตตั้งต้นและเมตริกซ์ความน่าจะเป็น ในขั้นตอนเริ่มต้น เอลิเมนต์ทุกเอลิเมนต์ของเมตริกซ์ความน่าจะเป็นจะถูกกำหนดค่าให้เป็น 0.5 ซึ่งหมายความว่ามีการกระจายของตัวเลข “0” และ “1” ในคอลัมน์ของเมตริกซ์ประชากรเลขฐานสองมีจำนวนเท่ากัน กำหนดให้ N คือจำนวนผลเฉลยในประชากร I คือจำนวนเวกเตอร์ของความน่าจะเป็นที่ต้องการใช้และ $n \times N_b$ คือความยาวของตัวเลขฐานสองสำหรับผลเฉลยใดๆ ดังนั้น เมตริกซ์ความน่าจะเป็นจะมีขนาดเป็น $I \times (n \times N_b)$ โดยในแต่ละแถวของเมตริกซ์คือเวกเตอร์ความน่าจะเป็นที่ใช้ในการผลิตประชากรย่อยเป็นจำนวน N/I ผลเฉลย การผลิตประชากรในแต่ละรอบของการคำนวณ ทำได้จากการรวมกลุ่มประชากรย่อย I กลุ่มเข้าด้วยกัน รูปที่ 3.5 แสดงเมตริกซ์ของความน่าจะเป็นขนาด 3×4 นั้นหมายความว่าใช้เลขฐานสอง 4 หลักสำหรับถอดรหัสเป็นตัวแปรออกแบบ กำหนดให้ประชากรเลขฐานสองมีจำนวนสมาชิกเป็น 12 ตัว ดังนั้นแต่ละแถวของเมตริกซ์ความน่าจะเป็น

เป็นจะต้องผลิตผลเฉลยเลขฐานสองจำนวน $12/3 = 4$ ผลเฉลย จากรูปแสดงประชากรย่อยที่ได้จากแต่ละแถวของเมตริกซ์ความน่าจะเป็น เมื่อทำการรวมผลเฉลยทั้งหมดเข้าด้วยกันก็จะได้ประชากรเลขฐานสองซึ่งเป็นเมตริกซ์ขนาด 12×4

เมื่อได้ประชากรเลขฐานสองแล้ว ทำการหาค่าฟังก์ชันเป้าหมายของผลเฉลยในกลุ่มประชากรที่สร้างขึ้นมา ขั้นตอนต่อมาคือการคัดสรรผลเฉลยที่ไม่ถูกรอรับจากการรวมกลุ่มของประชากรชุดใหม่ที่ได้กับสมาชิกในหน่วยเก็บผลเฉลยพาริโต และแทนที่หน่วยเก็บผลเฉลยพาริโตด้วยผลเฉลยที่ไม่ถูกรอรับชุดใหม่นี้ กลไกการค้นหาผลเฉลยของ MOEA ส่วนใหญ่มักจะค้นหาผลเฉลยที่ไม่ถูกรอรับได้เป็นจำนวนมาก ดังนั้น ถ้าจำนวนผลเฉลยที่ไม่ถูกรอรับมีจำนวนมากกว่าค่าที่เรากำหนดเอาไว้ (N_A) วิธีการของผลเฉลยบางส่วนออกจากหน่วยเก็บผลเฉลยจะถูกเรียกใช้งานเพื่อคัดผลเฉลยบางส่วนออกจากหน่วยเก็บ ส่วนเมตริกซ์ความน่าจะเป็น จะถูกปรับปรุงโดยใช้ผลเฉลยเลขฐานสองที่สุ่มมาจากหน่วยเก็บผลเฉลยพาริโต ซึ่งรูปแบบในการปรับปรุงแถวที่ i -th ของเมตริกซ์ความน่าจะเป็น จะคล้ายกับการปรับปรุงเวกเตอร์ความน่าจะเป็นในสมการ (2.6) ดังนี้

$$P_{i,j}^{new} = P_{i,j}^{old} (1 - L_R) + \bar{b}_j L_R \quad (3.4)$$

เมื่อ $\bar{b}_j$ หาได้โดย ทำการเลือกผลเฉลยตัวเลขฐานสองที่ไม่ถูกรอรับจากหน่วยเก็บผลเฉลยจำนวนหนึ่งโดยการสุ่ม ค่าของ $\bar{b}_j$ คือค่าเฉลี่ยของเอลิเมนต์ที่ j -th ของกลุ่มผลเฉลยที่เลือกมา นอกจากนี้เรายังสามารถประยุกต์มิวเทชันกับแถวที่ i -th ของเมตริกซ์ในรูปแบบเดียวกันกับสมการ (2.7) ทั้งนี้ขึ้นอยู่กับค่าความน่าจะเป็นของการเกิดมิวเทชัน เช่นเดียวกับกรณีของ PBIL ในบทที่ผ่านมา



รูปที่ 3.5 ตัวอย่างของเมตริกซ์ความน่าจะเป็นและประชากรที่เกี่ยวข้อง

ขั้นตอนการค้นหาผลเฉลยพาริโตของ MPBIL แสดงในขั้นตอนวิธีที่ 3.2 กระบวนการเริ่มต้นด้วยหน่วยเก็บผลเฉลยพาริโต และเมตริกซ์ความน่าจะเป็นตั้งต้น จากนั้นทำการสร้างประชากร

เลขฐานสองจากเมตริกซ์ความน่าจะเป็น ทำการคัดผลเฉลยที่ไม่ถูกรอบจำและทำการปรับปรุงหน่วยเก็บผลเฉลยที่ไม่ถูกรอบจำไปเรื่อยๆจนกระทั่งครบเงื่อนไขในการหยุดหาผลเฉลย

ขั้นตอนวิธีที่ 3.2 กระบวนการค้นหาขอบหน้าพารโตของ MPBIL

ขั้นตอนเริ่มต้น กำหนดพารามิเตอร์ที่จำเป็นทั้งหมด หน่วยเก็บผลเฉลยพารโตตั้งต้น **Pareto** และเมตริกซ์ความน่าจะเป็นตั้งต้น **P = 0.5**

ขั้นตอนคำนวณหลัก

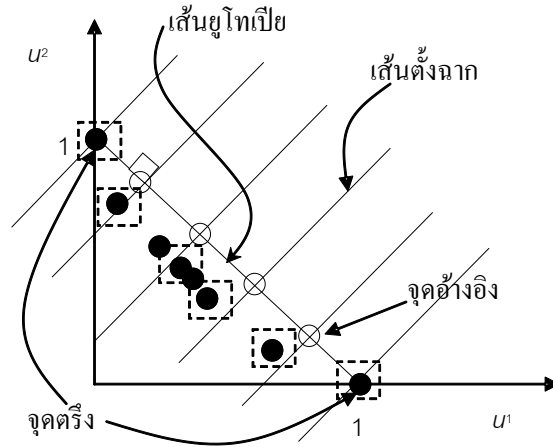
- 1: ถ้าเงื่อนไขการหยุดคำนวณเป็นจริง หยุดการค้นหาผลเฉลย
ถ้าไม่เป็นจริง ดำเนินการต่อในขั้นตอนที่ 2
- 2: ผลิตประชากรเลขฐานสอง **B** จาก **P**
- 3: คัดสรรผลเฉลยที่ไม่ถูกรอบจำ **B'** จากกลุ่มผลเฉลย $\mathbf{B} \cup \mathbf{Pareto}$ และแทนที่ **Pareto** ด้วย **B'**
- 4: ถ้าขนาดของ **Pareto** ใหญ่เกินไป ทำการลบผลเฉลยบางส่วนออกด้วยวิธีการกรองผลเฉลย
- 5: ปรับปรุงเมตริกซ์ความน่าจะเป็น **P** ด้วยสมการ (3.4)
- 6: กลับไปขั้นตอนที่ 1

ในขั้นตอนที่ 4 หากสมาชิกของหน่วยเก็บผลเฉลย **Pareto** มีจำนวนมากกว่าค่าที่ตั้งเอาไว้คือ N_A ต้องทำการกรองผลเฉลยที่ไม่ถูกรอบจำบางส่วนออกจาก **Pareto** สำหรับวิธี MPBIL ใช้วิธีเส้นตั้งฉาก (Normal Line Method) สำหรับขั้นตอนการทำงานของวิธีเส้นตั้งฉาก เริ่มต้นจากการปรับค่าฟังก์ชันเป้าหมายทั้งหมดของสมาชิกของ **Pareto** ดังนี้

$$u_i = \frac{(f_i - f_i^{\min})}{(f_i^{\max} - f_i^{\min})} \quad (3.5)$$

เมื่อ $f_i^{\min}$ คือ ค่าต่ำสุดของฟังก์ชันเป้าหมายที่ i -th ที่อยู่ใน **Pareto** และ $f_i^{\max}$ คือ ค่าสูงสุดของฟังก์ชันเป้าหมายที่ i -th ที่อยู่ใน **Pareto**

ขั้นตอนเชิงตัวเลขของวิธีเส้นตั้งฉากในกรณีของสองฟังก์ชันเป้าหมายแสดงในรูปที่ 3.6 ตำแหน่งของผลเฉลยที่ไม่ถูกรอบจำแสดงด้วยวงกลมสีดำ เริ่มต้นด้วย การระบุจุดตรึง (anchor points) จุดที่ให้ค่าต่ำสุดของแต่ละฟังก์ชันเป้าหมาย จากนั้นลากเส้นตรงเชื่อมระหว่างจุดทั้งสองเรียกว่า เส้นยูโทเปีย (Utopia lines) สร้างจุดอ้างอิงบนเส้นยูโทเปียและกระจายอยู่ระหว่างจุดตรึงทั้งสองโดยมีระยะห่างเท่ากันเป็นจำนวน N_A จุด ซึ่งแสดงในรูปด้วยวงกลมสีขาว จุดตรึงทั้งสองก็เป็นจุดอ้างอิงด้วย จากนั้นลากเส้นตั้งฉากจำนวน N_A เส้นซึ่งเป็นเส้นที่ลากผ่านจุดอ้างอิงและตั้งฉากกับเส้นยูโทเปีย ผลเฉลยที่จะถูกเลือกและเก็บในหน่วยเก็บผลเฉลย คือจุดที่อยู่ใกล้กับเส้นตั้งฉากแต่ละเส้นมากที่สุด จากรูปเส้นที่ล้อมรอบด้วยกรอบสี่เหลี่ยมเส้นประคือ ผลเฉลยที่ถูกเลือก



รูปที่ 3.6 วิธีเส้นตั้งฉาก

3.5 การหาค่าเหมาะที่สุดแบบอนุภาครวมกลุ่มหลายเป้าหมาย

การหาค่าเหมาะที่สุดแบบหลายเป้าหมายโดยใช้วิธีการหาค่าเหมาะที่สุดแบบอนุภาครวมกลุ่มเป็นการขยายผลจากวิธี PSO ในบทที่ 2 ในบทนี้จะเรียกว่า วิธีการหาค่าเหมาะที่สุดแบบอนุภาครวมกลุ่มหลายเป้าหมาย (Multiobjective Particle Swarm Optimisation, MPSO) หลักการของ MPSO จะคล้ายกันกับหลักการค้นหาคำตอบของวิธี PSO สำหรับการออกแบบที่มีฟังก์ชันเป้าหมายเดียว กล่าวคือ กระบวนการคำนวณเริ่มต้นด้วย ประชากรหรือกลุ่มของอนุภาคตั้งต้น ความเร็วเริ่มต้นของอนุภาค และหน่วยเก็บผลเฉลยพาริตี จากนั้นทำการปรับปรุงตำแหน่งของอนุภาคทั้งหมดด้วยสมการเดียวกันกับกรณีของการหาค่าเหมาะที่สุดแบบเป้าหมายเดียว

ที่วงวนที่ k ใดๆ การปรับปรุงทิศทางและตำแหน่งของอนุภาคตัวที่ i -th สามารถหาได้จาก

$$\mathbf{x}_i(k) = \mathbf{x}_i(k-1) + \mathbf{v}_i(k) \quad (3.6)$$

โดยที่ $\mathbf{v}_i(k)$ คือ เวกเตอร์ความเร็วของอนุภาคซึ่งคำนวณได้ดังนี้

$$\mathbf{v}_i(k) = W\mathbf{v}_i(k-1) + C_1r_1(\mathbf{p}_i^{best} - \mathbf{x}_i(k-1)) + C_2r_2(\mathbf{g}_i^{best} - \mathbf{x}_i(k-1)) \quad (3.7)$$

เมื่อ $\mathbf{x}_i(k)$ คือ ตำแหน่งของอนุภาคหรือเวกเตอร์ผลเฉลยที่ i -th ในวงวนที่ k

$\mathbf{v}_i(k)$ คือ ความเร็วของอนุภาค i -th ในวงวนที่ k

W คือ ค่าถ่วงน้ำหนักใช้สำหรับแสดงผลกระทบของเวกเตอร์ความเร็วในวงวนก่อนหน้า ($k-1$) ที่มีต่อ $\mathbf{v}_i(k)$

$r_1, r_2 \in [0,1]$ คือ ค่าตัวเลขสุ่มที่กระจายแบบคงตัว

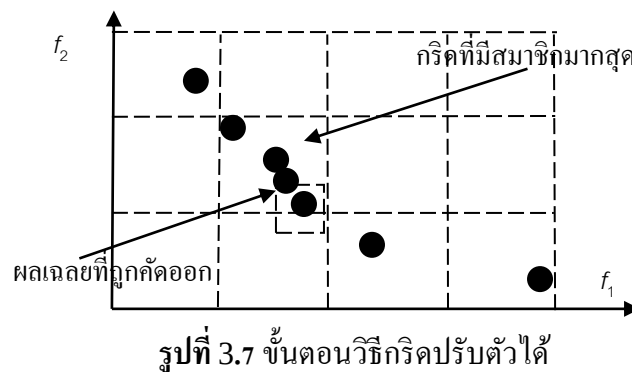
C_1 คือ ตัวประกอบการเรียนรู้ด้วยตนเอง

C_2 คือ ตัวประกอบการเรียนรู้ทางสังคม

$\mathbf{p}_i^{best}$ คือ ผลเฉลยที่ดีที่สุดที่อนุภาค $\mathbf{x}_i$ ค้นเจอ

$\mathbf{g}_i^{best}$ คือ ผลเฉลยที่เป็นสมาชิกในหน่วยเก็บผลเฉลยพารетоที่คัดเลือกมาแบบสุ่ม (กรณีที่มีมากกว่า 1 ชุด)

จะเห็นได้ว่า ข้อแตกต่างระหว่าง PSO และ MPSO คือส่วนของ $\mathbf{p}_i^{best}$ และ $\mathbf{g}_i^{best}$ ในกรณีของ $\mathbf{g}_i^{best}$ ทำการเลือกสมาชิกของหน่วยเก็บผลเฉลยพารетоแบบสุ่ม ส่วนการปรับปรุง $\mathbf{p}_i^{best}$ ถ้าอนุภาค $\mathbf{x}_i(k)$ ไม่ถูกรอบงำโดย $\mathbf{p}_i^{best}$ ของวงวนที่ $k-1$ ให้แทนที่ $\mathbf{p}_i^{best}$ ด้วย $\mathbf{x}_i(k)$ หากเงื่อนไขไม่เป็นจริงให้ใช้ $\mathbf{p}_i^{best}$ ตัวเดิม ในกรณีที่จำนวนผลเฉลยที่ไม่ถูกรอบงำในหน่วยเก็บมีจำนวนมากเกินไป สามารถประยุกต์ใช้วิธีเส้นตั้งฉากในการย้ายผลเฉลยออกบางตัวออกจากหน่วยเก็บ หรืออาจทำการประยุกต์ใช้วิธีคัดกรองผลเฉลยอีกรูปแบบหนึ่ง นั่นคือ ขั้นตอนวิธีกริดปรับตัวได้ (adaptive grid algorithm) รูปที่ 3.7 แสดงขั้นตอนวิธีกริดปรับตัวได้ วิธีการคือสร้างกริดให้ครอบคลุมผลเฉลยทั้งหมดบนโดเมนตัวแปรออกแบบ จากนั้น ทำการย้ายผลเฉลยที่อยู่ในกริดที่มีสมาชิกมากที่สุดและอยู่ไกลจากจุดศูนย์กลางของกริดนั้นมากที่สุดออกจากหน่วยเก็บ ทำการย้ายผลเฉลยออกทีละตัวด้วยหลักการดังกล่าว จนกระทั่งจำนวนผลเฉลยในหน่วยเก็บมีค่าเท่ากับ N_A กระบวนการค้นหาผลเฉลยของ MPSO แสดงในขั้นตอนวิธีที่ 3.3



รูปที่ 3.7 ขั้นตอนวิธีกริดปรับตัวได้

ขั้นตอนวิธีที่ 3.3 กระบวนการค้นหาขอบหน้าพารетоของ MPSO

ขั้นตอนเริ่มต้น กำหนดพารามิเตอร์ที่จำเป็นทั้งหมด หน่วยเก็บผลเฉลยพารетоตั้งต้น **Pareto** กลุ่มอนุภาคตั้งต้นพร้อมค่าฟังก์ชันเป้าหมายของสมาชิกในประชากร และกำหนดความเร็วเริ่มต้นของอนุภาคเป็นศูนย์

ขั้นตอนคำนวณหลัก

1: ถ้าเงื่อนไขการหยุดคำนวณเป็นจริง หยุดการค้นหาผลเฉลย

ถ้าไม่เป็นจริง ดำเนินการต่อในขั้นตอนที่ 2

2: คำนวณเวกเตอร์ความเร็วของอนุภาค $\mathbf{v}_i$ พร้อมปรับตำแหน่งของอนุภาคตามเวกเตอร์ความเร็ว

จะได้กลุ่มอนุภาค $\mathbf{X}$

3: หาค่าฟังก์ชันเป้าหมายของกลุ่มอนุภาคใหม่

- 4: หาค่าของ $\mathbf{p}_i^{best}$
- 5: คัดสรรผลเฉลยที่ไม่ถูกรอรับ $\mathbf{X}'$ จากกลุ่มผลเฉลย $\mathbf{X} \cup \mathbf{Pareto}$ และแทนที่ $\mathbf{Pareto}$ ด้วย $\mathbf{X}'$
- 6: ถ้าขนาดของ $\mathbf{Pareto}$ ใหญ่เกินไป ทำการลบผลเฉลยบางส่วนออกด้วยวิธีการของผลเฉลย
- 7: กลับไปขั้นตอนที่ 1

3.6 ขั้นตอนวิธีวิวัฒนาการที่ใช้พารโทเชิงกำลัง

ขั้นตอนวิธีวิวัฒนาการที่ใช้พารโทเชิงกำลัง (Strength Pareto Evolutionary Algorithm, SPEA) เป็นหนึ่งในวิธี MOEAs ที่พัฒนาในยุคเริ่มต้น และได้ทำการปรับปรุงเป็น SPEA2 เพื่อเพิ่มสมรรถนะ ในที่นี้จะกล่าวถึงรายละเอียดของ SPEA2 ซึ่งเป็นรุ่น (version) ที่มีประสิทธิภาพสูงกว่ารุ่นแรก กระบวนการคำนวณของ SPEA2 เริ่มต้นด้วยประชากร $\mathbf{X}$ และหน่วยเก็บผลเฉลยพารโทตั้งต้น $\mathbf{Pareto}$ จากนั้น รวมสมาชิกในประชากรและหน่วยเก็บผลเฉลยเข้าด้วยกันเพื่อทำการหาค่าความเหมาะสมหรือฟิตเนส (fitness) ค่าฟิตเนสของผลเฉลยที่ i th หาได้จาก

$$F_i = R_i + D_i \quad (3.8)$$

เมื่อ R_i เรียกว่า ค่าฟิตเนสดิบ (raw fitness) และ D_i คือค่าความหนาแน่น (density value) ของผลเฉลยที่ i th ค่า R_i สามารถคำนวณได้จากค่ากำลัง (strength value) ดังสมการ (3.9)

$$R_i = \sum_{j \in \mathbf{X} \cup \mathbf{Pareto}, j < i} S_j \quad (3.9)$$

เมื่อ S คือ ค่ากำลังของผลเฉลยต่างๆที่กำลังพิจารณา $j < i$ หมายถึงผลเฉลย j th ครอบงำผลเฉลย i th ค่ากำลัง S_j ของผลเฉลยที่ j th หรือ $\mathbf{x}_j$ คือ จำนวนของผลเฉลยใน $\mathbf{X} \cup \mathbf{Pareto}$ ที่ผลเฉลย $\mathbf{x}_j$ ครอบงำ สามารถแสดงด้วยสมการคณิตศาสตร์ดังนี้ เมื่อ $|x|$ หมายถึงจำนวนของสมาชิกใน x

$$S_j = |\{k | k \in \mathbf{X} \cup \mathbf{Pareto} \wedge j < k\}| \quad (3.10)$$

หากพิจารณาจากเมตริกซ์การครอบงำ สมการที่ (3.2) ค่ากำลังของผลเฉลยที่ j th คือผลรวมของตัวเลขในแถวที่ j th ของ $\mathbf{A}$ นั่นเอง

$$S_j = \sum_{k=1}^{N+N_A} A_{jk} \quad (3.11)$$

เมื่อ N และ N_A คือ ขนาดของ $\mathbf{X}$ และ $\mathbf{Pareto}$ ตามลำดับ ในส่วนของค่าความหนาแน่นสามารถหาได้จากความสัมพันธ์

$$D_j = \frac{1}{\sigma_j^k + 2} \quad (3.12)$$

เมื่อ σ_j^k คือ ระยะระหว่าง $\mathbf{x}_j$ กับผลเฉลยอื่นที่ใกล้ที่สุดกับ $\mathbf{x}_j$ เป็นอันดับที่ k th ค่าตัวเลข k ผู้ออกแบบสามารถกำหนดเองได้ ในที่นี้ กำหนดให้ $k = \text{round}(\sqrt{N + N_A})$ ค่าว่าระยะห่างในวิธี SPEA2 หมายถึง ระยะห่างบนโดเมนของฟังก์ชันเป้าหมาย

เมื่อได้ค่าฟิตเนสแล้ว ทำการบันทึกผลเฉลยที่ไม่ถูกรอบงำทั้งหมดลงในหน่วยเก็บผลเฉลย จากนิยามของ F_i , R_i และ D_i ผลเฉลยที่ให้ค่า F_i น้อยกว่า 1 คือผลเฉลยที่ไม่ถูกรอบงำ หากจำนวนผลเฉลยที่ไม่ถูกรอบงำมีมากกว่าค่าของ N_A ทำการลบผลเฉลยบางค่าออกจาก **Pareto** ด้วยวิธีย่านใกล้เคียงสุด (nearest neighborhood method) ผลเฉลยที่ i th จะถูกคัดเลือกให้ออกจาก **Pareto** ด้วยเงื่อนไข

$$\mathbf{x}_i \leq_d \mathbf{x}_j \text{ สำหรับ } \forall \mathbf{x}_j \in \mathbf{Pareto} \quad (3.13)$$

ทำการลบผลเฉลยออกจาก **Pareto** ทีละตัว จนกระทั่งจำนวนสมาชิกในหน่วยเก็บผลเฉลยมีค่าเป็น N_A ส่วนพจน์ $\mathbf{x}_i \leq_d \mathbf{x}_j$ มีนิยามดังนี้

$$\begin{aligned} \mathbf{x}_i \leq_d \mathbf{x}_j &\Leftrightarrow \forall 0 < k < |\mathbf{Pareto}| : \sigma_i^k = \sigma_j^k \vee \\ &\exists 0 < k < |\mathbf{Pareto}| : [(\forall 0 < l < k : \sigma_i^l = \sigma_j^l) \wedge \sigma_i^k < \sigma_j^k] \end{aligned} \quad (3.14)$$

นอกจากนี้ เราสามารถประยุกต์ใช้วิธีคัดกรองผลเฉลยอื่นที่กล่าวถึงมาแล้วในข้างต้น ในทางกลับกันในกรณีที่ผลเฉลยที่ไม่ถูกรอบงำมีจำนวนน้อยกว่า N_A ให้ทำการบวกผลเฉลยที่ถูกครอบงำบางส่วนเข้าไปใน **Pareto** จนกระทั่งสมาชิกของ **Pareto** มีจำนวนเท่ากับ N_A โดยผลเฉลยที่ถูกครอบงำที่บวกเข้าไปด้วยต้องมีค่าฟิตเนสต่ำสุดในกลุ่มของ $\mathbf{X} \cup \mathbf{Pareto}$ ถัดจากค่าฟิตเนสของผลเฉลยที่ไม่ถูกรอบงำ กระบวนการปรับปรุงหน่วยเก็บผลเฉลยพาเรโตนี้เรียกว่า การคัดเลือกตามสภาพแวดล้อม (environment selection)

จากนั้นทำการคัดเลือกผลเฉลยจาก **Pareto** ที่ทำการปรับปรุงแล้วในข้างต้น ด้วยวิธีการคัดเลือกแบบการแข่งขันทวิภาค (binary tournament selection) กล่าวคือ ผลเฉลยคู่ใดๆถูกเลือกมาแบบสุ่ม ผลเฉลยที่ถูกเลือกในท้ายที่สุด คือผลเฉลยที่มีค่าฟิตเนสต่ำกว่า เมื่อได้จำนวนผลเฉลยเพื่อสร้างประชากรลูกครบแล้ว ทำการสร้างประชากรลูกด้วยการดำเนินการรีคอมบิเนชันและมิวเทชันกับผลเฉลยที่ถูกเลือกเหล่านั้นจนได้ประชากรลูก ดำเนินการซ้ำจนกระทั่งครบเงื่อนไขการค้นหาค้นหาขอบหน้าพาเรโต กระบวนการคำนวณเพื่อค้นหาขอบหน้าพาเรโตของวิธี SPEA2 แสดงในขั้นตอนวิธีที่ 3.4

ขั้นตอนวิธีที่ 3.4 กระบวนการคำนวณของวิธี SPEA2

ขั้นตอนเริ่มต้น กำหนดพารามิเตอร์ที่จำเป็นทั้งหมด หน่วยเก็บผลเฉลยพารेटอเริ่มต้น **Pareto** และประชากรตั้งต้น **X** พร้อมค่าฟังก์ชันเป้าหมายของสมาชิกในประชากร

ขั้นตอนคำนวณหลัก

1: ถ้าเงื่อนไขการหยุดคำนวณเป็นจริง หยุดการค้นหาผลเฉลย

ถ้าไม่เป็นจริง ดำเนินการต่อในขั้นตอนที่ 2

2: หาค่าฟิตเนสของ $X \cup \text{Pareto}$

3: ปรับปรุงหน่วยเก็บผลเฉลย **Pareto** ด้วยกระบวนการคัดเลือกตามสภาพแวดล้อม

4: ทำการคัดเลือกผลเฉลยจาก **Pareto** ที่ปรับปรุงแล้ว ด้วยวิธีการคัดเลือกแบบแข่งขันทวิภาค

5: ดำเนินการรีคอมบินชันและมิวเทชันกับผลเฉลยที่ถูกคัดเลือก เพื่อสร้างประชากรชุดใหม่ **X**

6: กลับไปขั้นตอนที่ 1

3.7 ขั้นตอนวิธีเชิงพันธุกรรมที่ใช้การคัดสรรแบบไม่ถูกครอบงำ

ขั้นตอนวิธีเชิงพันธุกรรมที่ใช้การคัดสรรแบบไม่ถูกครอบงำ (non-dominated sorting genetic algorithm, NSGA) เป็นวิธีเชิงวิวัฒนาการที่พัฒนาขึ้นมาในยุคเดียวกันกับ SPEA และเช่นเดียวกันวิธีนี้ได้พัฒนารุ่นที่สองขึ้นมาเพื่อเพิ่มสมรรถนะในการหาผลเฉลยซึ่งเขียนแทน NSGA รุ่นที่สองด้วย NSGAII วิธีหาค่าเหมาะที่สุดนี้อาจกล่าวได้ว่าเป็นวิธี MOEA ที่นิยมใช้มากที่สุดในวงการวิจัยด้านขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมาย อย่างไรก็ตาม ในปัจจุบันได้มีวิธี MOEAs อื่นๆที่พัฒนาขึ้นมาและมีสมรรถนะสูงกว่า NSGAII

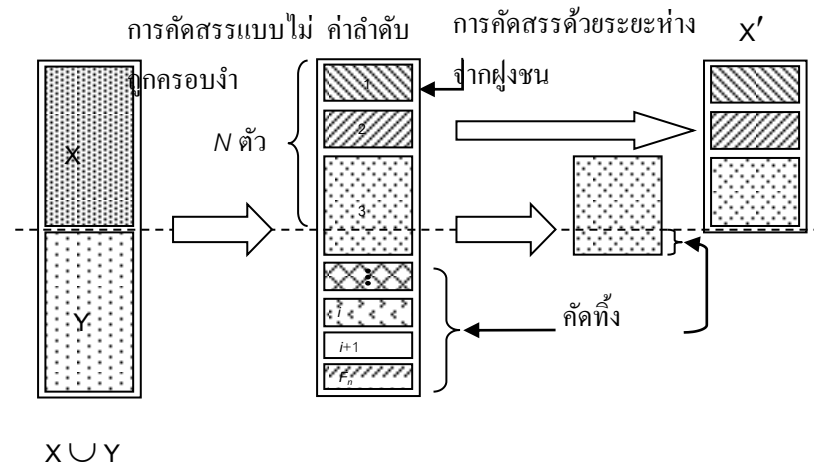
รูปแบบการค้นหาขอบหน้าพารेटอของ NSGAII จะแตกต่างจาก MOEAs ทั้ง 4 วิธีที่กล่าวมาข้างต้น คือ ไม่มีหน่วยเก็บผลเฉลยพารेटอ แต่จะเก็บผลเฉลยที่ไม่ถูกครอบงำไว้ในเมตริกซ์ประชากร กระบวนการคำนวณเริ่มต้นด้วย กลุ่มของผลเฉลยหรือประชากรตั้งต้น **X** จากนั้นทำการกำหนดค่าลำดับ (rank) ของผลเฉลย ค่าลำดับของผลเฉลย x_j ใดๆในประชากร เขียนแทนด้วย $rank_j$ สามารถคำนวณได้จากระดับของการถูกครอบงำ โดยกำหนดให้ $rank_j = 1$ ถ้าระดับของการถูกครอบงำของ x_j เป็น 0 หรือ $rank_j = 2$ ถ้าระดับของการถูกครอบงำเป็น 1 ไปเรื่อยๆ (ดูรายละเอียดของระดับของการถูกครอบงำในนิยาม 3) เมื่อได้ค่าลำดับของผลเฉลยทั้งหมดแล้ว ทำการคัดเลือกผลเฉลยเพื่อทำการผลิตประชากรลูก โดยการใช้การคัดเลือกแบบการแข่งขันทวิภาคและใช้ค่าลำดับของผลเฉลยเป็นเกณฑ์ ส่วนการผลิตประชากรลูกใช้กระบวนการรีคอมบินชันและมิวเทชันซึ่งมีรูปแบบที่หลากหลายมาก

เมื่อได้ประชากรลูก **Y** แล้ว ทำการคัดผลเฉลย N ตัวที่จะเป็นสมาชิกของประชากรในวงวนถัดไป **X'** จากสมาชิกของ $X \cup Y$ โดยพิจารณาค่าลำดับของผลเฉลยเป็นหลัก ปัญหาที่ต้องเจอเสมอในขั้นตอนนี้คือ กลุ่มผลเฉลยที่มีค่าลำดับดีกลุ่มสุดท้ายที่จะถูกเลือก มักมีจำนวนเกินขนาดของประชากร เมื่อรวมกลุ่มกับผลเฉลยที่ถูกเลือกก่อนหน้านี้ ดังแสดงในรูปที่ 3.8 กลุ่มผลเฉลยที่มีค่าลำดับเป็น 3 เมื่อ

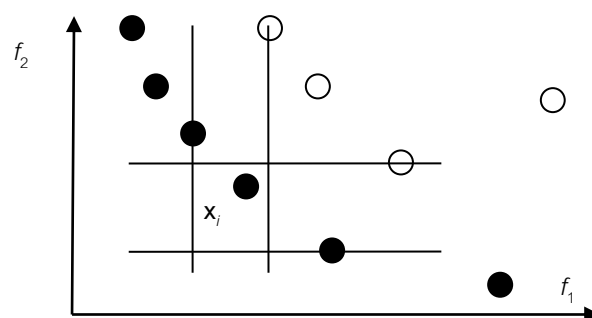
นำไปรวมกับกลุ่มผลเฉลยที่มีค่าลำดับเป็น 1 และ 2 ทำให้จำนวนผลเฉลยสำหรับประชากรในวงวนถัดไปมีค่ามากกว่า N นั้นหมายความว่า จะต้องคัดกรองผลเฉลยบางตัวออกจากกลุ่มผลเฉลยที่มีค่าลำดับเป็น 3 ดังแสดงในรูป ในการคัดผลเฉลยบางตัวในกลุ่มที่ 3 ที่ใช้ค่าระยะห่างจากฝูงชน (crowding distance) เป็นเกณฑ์ รูปที่ 3.9 แสดงการคำนวณค่าระยะห่างจากฝูงชนของผลเฉลย x_i บนโดเมนของฟังก์ชันเป้าหมาย จากรูป ระยะห่างจากฝูงชน คือผลรวมของด้านกว้างและด้านยาวของกล่องสี่เหลี่ยมที่เล็กที่สุดที่ครอบ x_i ซึ่งหมายความว่า ถ้าค่าระยะห่างจากฝูงชนสูง ผลเฉลยนั้นจะอยู่ไกลจากฝูงชน ย่อมสร้างความหลากหลายให้กับประชากรในวงวนถัดไปและสมควรถูกเลือกเป็นสมาชิกของประชากรในรุ่นต่อไป ดังนั้นผลเฉลยบางตัวในกลุ่มที่ 3 ในรูปที่ 3.8 ที่มีค่าระยะห่างจากฝูงชนต่ำต้องถูกตัดออกเพื่อให้ขนาดของประชากรคงที่ กระบวนการคำนวณดำเนินการแบบทำซ้ำจนกระทั่งครบเงื่อนไขในการหยุดค้นหา สิ่งที่ต้องทำเพิ่มเติมคือ ในการคัดเลือกแบบแข่งขันทวิภาคในวงวนปกติ จะใช้ตัวดำเนินการเปรียบเทียบฝูงชน (crowded-comparison operator) เขียนแทนด้วย $<_n$ เป็นเกณฑ์ในการคัดเลือกสำหรับผลเฉลย x_i และ x_j ใดๆในประชากร เรากล่าวว่า $x_i <_n x_j$ ก็ต่อเมื่อ

$$rank_i < rank_j \text{ หรือ } [rank_i = rank_j \text{ และ } x_i \text{ มีระยะห่างจากฝูงชนมากกว่า } x_j] \quad (3.15)$$

กระบวนการค้นหาขอบหน้าพาเรโตของวิธี NSGAIII แสดงในขั้นตอนวิธีที่ 3.5



รูปที่ 3.8 การคัดเลือกประชากรสำหรับวงวนถัดไป



รูปที่ 3.9 การคำนวณระยะห่างจากฝูงชนของผลเฉลย x_i

ขั้นตอนวิธีที่ 3.5 กระบวนการคำนวณของวิธี NSGAI

ขั้นตอนเริ่มต้น กำหนดพารามิเตอร์ที่จำเป็นทั้งหมด และประชากรตั้งต้น X พร้อมค่าฟังก์ชันเป้าหมายของสมาชิกในประชากร จากนั้นสร้างประชากรลูก Y โดยใช้ตัวดำเนินการ การคัดเลือกแบบแข่งขัน ทวิภาค รีคอมบิเนชัน มิวเทชัน

ขั้นตอนคำนวณหลัก

- 1: ถ้าเงื่อนไขการหยุดคำนวณเป็นจริง หยุดการค้นหาผลเฉลย
ถ้าไม่เป็นจริง ดำเนินการต่อในขั้นตอนที่ 2
- 2: คัดเลือกประชากรสำหรับวงวนใหม่จาก $X \cup Y$
- 3: หาค่าลำดับและระยะห่างจากฝูงชนของสมาชิกในประชากรใหม่ X
- 4: ทำการคัดเลือกสมาชิกของ X ด้วยการคัดเลือกแบบแข่งขันทวิภาค
และใช้ตัวดำเนินการเปรียบเทียบฝูงชน
- 5: สร้างประชากรลูก Y ด้วยตัวดำเนินการ รีคอมบิเนชัน และมิวเทชัน
- 6: กลับไปขั้นตอนที่ 1

บทที่ 4 การเพิ่มสมรรถนะขั้นตอนวิวัฒนาการด้วยเทคนิคการ

ผสมผสาน

4.1 ความนำ

เป็นที่ทราบกันดีว่าขั้นตอนวิวัฒนาการมีข้อดีคือ สามารถลู่เข้าใกล้บริเวณของจุดที่เหมาะสมได้อย่างรวดเร็ว แต่มักจะมีปัญหาในการหาผลเฉลยที่เหมาะสมที่สุดที่แท้จริง ด้วยเหตุนี้ ในบางครั้งเราเรียกคำตอบที่ได้จากการใช้ขั้นตอนวิวัฒนาการว่าเป็นจุดใกล้เคียงผลเฉลยที่เหมาะสมที่สุด (near optimum) การแก้ปัญหานี้ สามารถทำได้โดยการเพิ่มตัวดำเนินการพิเศษเข้าไปในกระบวนการคำนวณของขั้นตอนวิวัฒนาการ ในบทนี้จะเป็นการนำเสนอแนวคิดในการผสมผสานระหว่างขั้นตอนวิวัฒนาการแบบทั่วไป กับตัวดำเนินการสำหรับการค้นหาเฉพาะพื้นที่ (local search) ในที่นี้ วิธีค้นหาเฉพาะพื้นที่ที่นำเสนอ คือค่าประมาณเกรเดียนต์และวิธีซิมเพล็กซ์ลงเขา เนื้อหาในบทนี้จะเป็นการนำเสนอแนวคิดเบื้องต้นในการผสมผสานตัวดำเนินการพิเศษกับขั้นตอนวิวัฒนาการ ซึ่งในส่วนของ การประยุกต์ใช้ที่เป็นกรณีศึกษา จะนำเสนอในบทต่อไป

4.2 ค่าประมาณเวกเตอร์เกรเดียนต์สำหรับขั้นตอนวิวัฒนาการ

หัวข้อนี้นำเสนอการผสมผสานกระบวนการหาผลเฉลยที่เหมาะสมที่สุด ด้วยขั้นตอนวิวัฒนาการ กับแนวคิดของวิธีหาค่าเหมาะสมที่สุดที่ใช้เกรเดียนต์ กระบวนการค้นหาผลเฉลยที่เหมาะสมที่สุดของขั้นตอนวิวัฒนาการโดยทั่วไปใช้กลุ่มของผลเฉลยหรือประชากรในการหาคำตอบ แนวคิดของค่าประมาณเกรเดียนต์ (approximate gradient) คือการประมาณค่าของเกรเดียนต์ของฟังก์ชันจากข้อมูลของประชากรผลเฉลยในกระบวนการของขั้นตอนวิวัฒนาการ นั่นหมายความว่า เราสามารถประยุกต์ใช้วิธีหาค่าเหมาะสมที่สุดที่ใช้เกรเดียนต์เป็นตัวดำเนินการย่อยในขั้นตอนวิวัฒนาการ ซึ่งหากค่าประมาณของเกรเดียนต์มีความแม่นยำ จะทำให้กระบวนการถูกผสมนี้มีสมรรถนะในการหาผลเฉลยที่เหมาะสมที่สุดสูงขึ้น

4.2.1 การประมาณค่าเกรเดียนต์

การประมาณค่าเกรเดียนต์ที่ตำแหน่งเวกเตอร์ออกแบบ $\mathbf{x}_0$ ใดๆ จากข้อมูลของประชากรของผลเฉลยและค่าฟังก์ชันเป้าหมายของผลเฉลยเหล่านั้น สำหรับฟังก์ชันใน n มิติใดๆ สามารถคำนวณได้จากการประมาณค่าฟังก์ชันของจุด $\mathbf{x}$ ที่อยู่ใกล้เคียงกับ $\mathbf{x}_0$ ด้วยอนุกรมเทย์เลอร์อันดับที่ 1 ดังนี้

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0)^T \nabla f(\mathbf{x}_0) \quad (4.1)$$

ถ้าเรามีเวกเตอร์ผลเฉลยที่มีตำแหน่งใกล้เคียงกับ $\mathbf{x}_0$ เป็นจำนวน n ตัว $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ ซึ่งมีค่าฟังก์ชันเป็น $\{f_1, \dots, f_n\}$ เมื่อแทนค่า $\mathbf{x}$ ในสมการ (4.1) ด้วย $\mathbf{x}_1, \dots, \mathbf{x}_n$ และแทนค่า $f(\mathbf{x})$ ด้วย $f_1, \dots, f_n$ จะได้ระบบสมการเชิงเส้น n สมการ โดยที่ตัวแปรที่ไม่ทราบค่า n ตัวคือ เอลิเมนต์ของ $\nabla f(\mathbf{x}_0)$

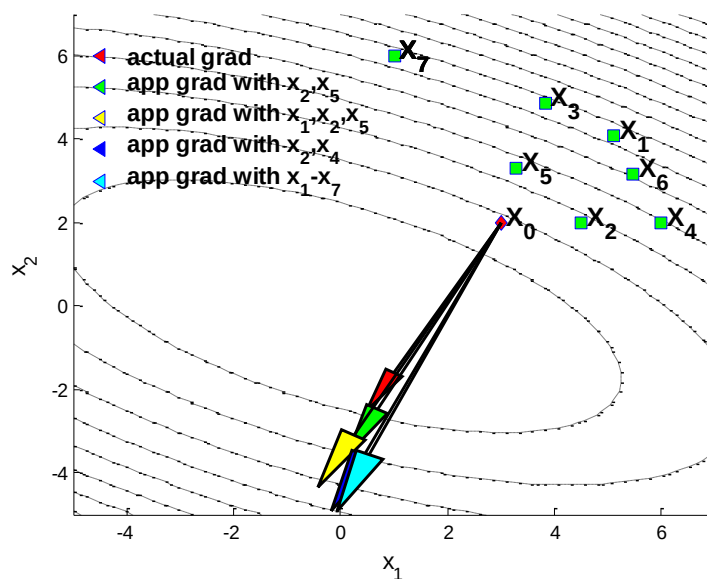
$$\begin{bmatrix} (\mathbf{x}_0 - \mathbf{x}_1)^T \\ \vdots \\ (\mathbf{x}_0 - \mathbf{x}_n)^T \end{bmatrix} \nabla f = \mathbf{A} \nabla f = \begin{Bmatrix} f(\mathbf{x}_0) - f_1 \\ \vdots \\ f(\mathbf{x}_0) - f_n \end{Bmatrix} = \mathbf{b}. \quad (4.2)$$

ในทางปฏิบัติ เนื่องจากตำแหน่งของผลเฉลยในประชากรของขั้นตอนวิธีวิวัฒนาการถูกสร้างขึ้นด้วยการสุ่ม หากใช้ผลเฉลย n ตัวเพื่อประมาณค่าของเกรเดียนต์ อาจส่งผลให้เมตริกซ์ $\mathbf{A}$ เป็นเมตริกซ์ซิงกูลาร์ (singular matrix) ได้ และการคำนวณค่าประมาณเกรเดียนต์ก็จะไม่สามารถทำได้ ดังนั้นโดยทั่วไปจะใช้ผลเฉลยจำนวน $R > n$ ตัวจากเมตริกซ์ประชากร โดยที่เวกเตอร์เหล่านี้มีระยะใกล้กับ $\mathbf{x}_0$ มากที่สุด ซึ่งจะได้ระบบสมการเชิงเส้น (4.2) เป็นระบบที่มีตัวแปร n ตัวและมี R สมการ ในการหาผลเฉลยของระบบสมการเชิงเส้นนี้สามารถทำได้โดยการใช้ตัวดำเนินการเมตริกซ์ผกผันเทียม (pseudo-inverse) หรืออาจใช้วิธีกำลังสองต่ำสุด

รูปที่ 4.1 แสดงการประมาณค่าของเวกเตอร์เกรเดียนต์ของฟังก์ชันในปริภูมิสองมิติที่จุด $\mathbf{x}_0$ จากรูป ค่าจริงของเวกเตอร์เกรเดียนต์คือเวกเตอร์ที่สั้นที่สุด ส่วนเวกเตอร์ตัวอื่นคือค่าประมาณของเกรเดียนต์ เช่น เวกเตอร์เกรเดียนต์ที่จุด $\mathbf{x}_0$ ที่ประมาณจาก จุดใกล้เคียงสองจุด $\mathbf{x}_2$ และ $\mathbf{x}_5$ สามารถคำนวณได้จาก

$$\begin{bmatrix} (x_1^0 - x_1^2) & (x_2^0 - x_2^2) \\ (x_1^0 - x_1^5) & (x_2^0 - x_2^5) \end{bmatrix} \nabla f(\mathbf{x}_0) = \begin{Bmatrix} f_0 - f_2 \\ f_0 - f_5 \end{Bmatrix} \quad (4.3)$$

จากในรูป แสดงค่าประมาณเกรเดียนต์ที่จุด $\mathbf{x}_0$ ที่คำนวณจากเซตของจุดที่ใกล้เคียงกับ $\mathbf{x}_0$ หลายเซต ซึ่งเมื่อเทียบกับเกรเดียนต์จริง จะมีค่าใกล้เคียงกันมาก



รูปที่ 4.1 การประมาณค่าเวกเตอร์เกรเดียนต์

4.2.2 การผสมผสานขั้นตอนวิธีวิวัฒนาการกับค่าประมาณเกรเดียนต์

การผสมตัวดำเนินการค่าประมาณเกรเดียนต์เข้าไปในกระบวนการหาค่าเหมาะที่สุดของขั้นตอนวิธีวิวัฒนาการสามารถทำได้โดย กำหนดให้กระบวนการคำนวณของวิธีวิวัฒนาการเป็นกระบวนการหลัก และให้ค่าประมาณเกรเดียนต์เป็นตัวดำเนินการย่อยของขั้นตอนวิธีวิวัฒนาการ ดังแสดงในขั้นตอนวิธีที่ 4.1 ในแต่ละวงวน เมื่อค้นหาประชากรลูกจากการดำเนินการรีคอมบิเนชันและมิวเทชันกับประชากรพ่อ-แม่ เราสามารถทำการหาผลเฉลยที่ดีที่สุด $\mathbf{x}^{best}$ และค่าฟังก์ชันเป้าหมาย f^{best} จากนั้น ทำการหาค่าประมาณเกรเดียนต์ของผลเฉลยนั้น ในการหาค่าเหมาะที่สุดด้วยวิธี ลดลงชันสุด (steepest decent method) เวกเตอร์ทิศทางในการหาค่าตอบคือ เวกเตอร์ที่มีทิศทางตรงข้ามกับ ∇f ดังนั้น เกรเดียนต์โดยประมาณที่ได้จากสมการ (4.2) สามารถนำมาใช้เป็นทิศทางในการหาผลเฉลยในรูปแบบคล้ายกัน กล่าวคือ $\mathbf{s} = -\nabla f$ เพื่อให้การประมาณค่าเกรเดียนต์นี้มีประโยชน์มากขึ้น ทำการประยุกต์การประมาณในช่วง (interpolation) ด้วยฟังก์ชันกำลังสองหนึ่งครั้งในการหาผลเฉลยด้วยเกรเดียนต์โดยประมาณ ทำการหาเวกเตอร์ออกแบบในแนวทิศทาง $\mathbf{s}$ เพิ่ม ดังนี้

$$\begin{aligned}\mathbf{z}_1 &= \mathbf{x}^{best} + \beta_1 \cdot \mathbf{s} \\ \mathbf{z}_2 &= \mathbf{x}^{best} + \beta_2 \cdot \mathbf{s}.\end{aligned}\tag{4.4}$$

เมื่อ $\beta_1 \neq \beta_2$ คือ พารามิเตอร์ที่สุ่มค่าขึ้นมาในช่วงของ (0,1) ขั้นตอนถัดไป ประยุกต์ใช้การประมาณค่าในช่วงเพื่อหาค่า β ที่ให้ค่าฟังก์ชันต่ำสุด ในกรณีของฟังก์ชันกำลังสอง

$$f(\mathbf{x}^{best} + \beta \cdot \mathbf{s}) = C_1 \beta^2 + C_2 \beta + C_3\tag{4.5}$$

จะได้ เมทริกซ์สัมประสิทธิ์ของฟังก์ชันกำลังสองจากการแก้ระบบสมการเชิงเส้น

$$\begin{bmatrix} 0 & 0 & 1 \\ \beta_1^2 & \beta_1 & 1 \\ \beta_2^2 & \beta_2 & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} f^{best} \\ f(\mathbf{z}_1) \\ f(\mathbf{z}_2) \end{bmatrix}\tag{4.6}$$

เมื่อ $f(\mathbf{z}_1)$ และ $f(\mathbf{z}_2)$ คือ ค่าฟังก์ชันเป้าหมายที่จุด $\mathbf{z}_1$ และ $\mathbf{z}_2$ ตามลำดับ จากนั้น เวกเตอร์ตัวแปรออกแบบตัวที่สาม หาได้จาก

$$\mathbf{z}_3 = \mathbf{x}^{best} + \beta_3 \cdot \mathbf{s}.\tag{4.7}$$

เมื่อ

$$\beta_3 = -C_2/2/C_1.$$

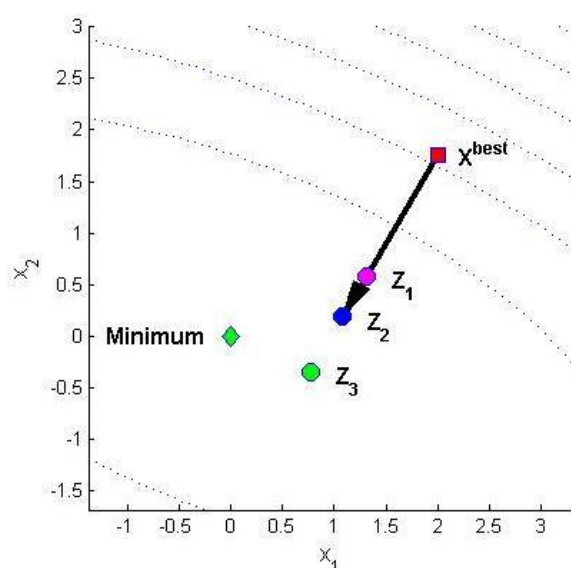
รูปที่ 4.2 แสดงกระบวนการหาผลเฉลยตามแนวของทิศทางตรงข้ามกับเวกเตอร์เกรเดียนต์ โดยการกำหนดเวกเตอร์ตัวแปรออกแบบสองจุดแรก z_1 และ z_2 จากนั้น ทำการหาจุด z_3 ด้วยสมการ (4.7)

ขั้นตอนวิธีที่ 4.1 การผสมระหว่างขั้นตอนวิธีวิวัฒนาการและค่าประมาณเกรเดียนต์

ขั้นตอนเริ่มต้น กำหนดพารามิเตอร์ที่จำเป็นทั้งหมด และประชากรเริ่มต้นพร้อมค่าฟังก์ชันเป้าหมายของสมาชิกในประชากร กำหนดหน่วยเก็บผลเฉลยสำหรับประมาณค่าเกรเดียนต์

ขั้นตอนคำนวณหลัก

- 1: ถ้าเงื่อนไขเหมาะสมที่สุดเป็นจริง หยุดการค้นหาผลเฉลย ถ้าไม่เป็นจริง ดำเนินการต่อในขั้นตอนที่ 2:
- 2: ผลิตประชากรลูกด้วยหลักการเฉพาะของขั้นตอนวิธีเหล่านั้น
- 3: ประมาณค่าเกรเดียนต์ของฟังก์ชันเป้าหมายที่ตำแหน่งของผลเฉลยที่ดีที่สุด
- 4: ค้นหาผลเฉลยด้วยเกรเดียนต์ (4.1) - (4.7) จะได้ z_1 , z_2 และ z_3 และกำหนดให้เป็นสมาชิกเพิ่มเติมของประชากรลูก
- 5: หาค่าฟังก์ชันเป้าหมาย และ/หรือ ค่าความเหมาะสม
- 6: ทำการคัดเลือกประชากรรุ่นใหม่
- 7: เก็บผลเฉลยบางส่วนในหน่วยเก็บผลเฉลยสำหรับประมาณค่าเกรเดียนต์ โดยใช้หลักการให้สมาชิกในหน่วยเก็บผลเฉลยกระจายไปทั่วโดเมนการออกแบบมากที่สุด
- 8: กลับไปขั้นตอนที่ 1:



รูปที่ 4.2 การหาความยาวก้าวของค่าประมาณเกรเดียนต์

4.2.3 ค่าประมาณเกรเดียนต์สำหรับการหาค่าเหมาะที่สุดที่มีเงื่อนไขบังคับ

การประยุกต์ใช้ค่าประมาณเกรเดียนต์กับปัญหาการหาค่าเหมาะที่สุดที่มีเงื่อนไขบังคับ ในที่นี้จะกล่าวถึงเฉพาะกรณีของเงื่อนไขบังคับแบบอสมการเท่านั้น ในกระบวนการค้นหาคำตอบของปัญหาการออกแบบชนิดนี้ เวกเตอร์ผลเฉลยหนึ่งชุดเราต้องทำการหาค่าฟังก์ชันเป้าหมายและเงื่อนไขบังคับ นั่นหมายความว่า เราสามารถประมาณค่าเกรเดียนต์ของฟังก์ชันเป้าหมายและฟังก์ชันเงื่อนไขบังคับได้พร้อมกันโดยใช้สมการ (4.2) อย่างไรก็ตาม หากเราใช้ค่าของทิศทางในการค้นหาผลเฉลยเป็น $\mathbf{s} = -\nabla f$ เช่นเดียวกับการหาค่าเหมาะที่สุดแบบไม่มีเงื่อนไขบังคับ ค่าประมาณเกรเดียนต์อาจไม่มีประสิทธิภาพสูงมาก หรืออาจไม่ช่วยเพิ่มสมรรถนะในการหาผลเฉลยของขั้นตอนวิธีวิวัฒนาการเลย ทั้งนี้เป็นเพราะว่าโดยทั่วไปแล้ว เมื่อผลเฉลยอยู่นอกบริเวณหาคำตอบได้ เวกเตอร์ทิศทาง $\mathbf{s} = -\nabla f$ มักจะนำผลเฉลยออกไปไกลจากบริเวณหาคำตอบได้ (feasible region) มากกว่าเดิม ดังนั้นเมื่อปัญหาการออกแบบมีเงื่อนไขบังคับแบบอสมการ ให้ทำการประยุกต์ใช้ฟังก์ชันปรับแก้ สมการที่ (2.13) ในบทที่ 2 จะได้ฟังก์ชันเป้าหมายสมมูลเป็น

$$F(\mathbf{x}) = f(\mathbf{x}) + r \sum_{i=1}^m (\max[0, g_i(\mathbf{x})])^2 \quad (4.8)$$

และเวกเตอร์ทิศทางในการหาผลเฉลยคือ $\mathbf{s} = -\nabla F$ ในส่วนของการหาค่าสูงสุดของระยะการก้าว เราสามารถประยุกต์ใช้สมการที่ (4.9) ดังนี้

$$\lambda_{\max} = \min_{\lambda_i > 0} \left[\lambda_i = -\frac{g_i(\mathbf{x}^k)}{\nabla g_i(\mathbf{x}^k)^T \mathbf{s}} \right] \quad (4.9)$$

รูปที่ 4.3 แสดงตัวอย่างของการประยุกต์ใช้ค่าประมาณเกรเดียนต์สำหรับการหาค่าเหมาะที่สุดที่มีเงื่อนไขบังคับ จากรูปปัญหาการออกแบบมีเงื่อนไขบังคับ 2 เงื่อนไข และทำการปรับปรุงผลเฉลย $\mathbf{x}_1$ โดยการประมาณค่าเกรเดียนต์ เนื่องจากผลเฉลย $\mathbf{x}_1$ อยู่ในพื้นที่หาคำตอบได้ ดังนั้น ทิศทางการค้นหาคำตอบคือ $\mathbf{s} = -\nabla F = -\nabla f$ ส่วนค่าระยะการก้าวสูงสุด $\lambda_{\max}$ สามารถหาได้จากสมการ (4.9) ในรูปแสดงถึงตำแหน่งถัดไปของ $\mathbf{x}_1$ ซึ่งก็คือผลเฉลย

$$\mathbf{x}_1' = \mathbf{x}_1 + \lambda_{\max} \mathbf{s} \quad (4.10)$$

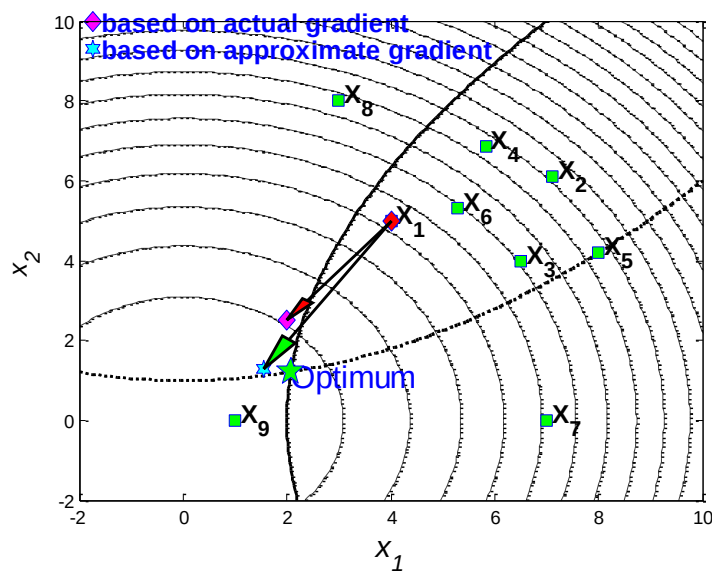
ในที่นี้แสดงตำแหน่งของ $\mathbf{x}_1'$ ที่ได้จากการคำนวณด้วยค่าเกรเดียนต์จริง (สัญลักษณ์ ♦) และค่าประมาณเกรเดียนต์ (สัญลักษณ์ ★) จากรูปจะเห็นได้ว่า ตำแหน่งทั้งสองมีค่าใกล้เคียงกันและอยู่นอกพื้นที่หาคำตอบได้ ทั้งนี้เป็นผลมาจากการที่สมการ (4.9) ใช้การกระจายอนุกรมเทย์เลอร์อันดับที่หนึ่ง และ

บริเวณหาคำตอบได้เป็นพื้นที่คอนเวกซ์ ฉะนั้น หากต้องการประยุกต์ใช้ ค่าประมาณเกรเดียนต์อย่างมีประสิทธิภาพ ควรปรับปรุงค่าของผลเฉลย $\mathbf{x}_1$ ดังนี้

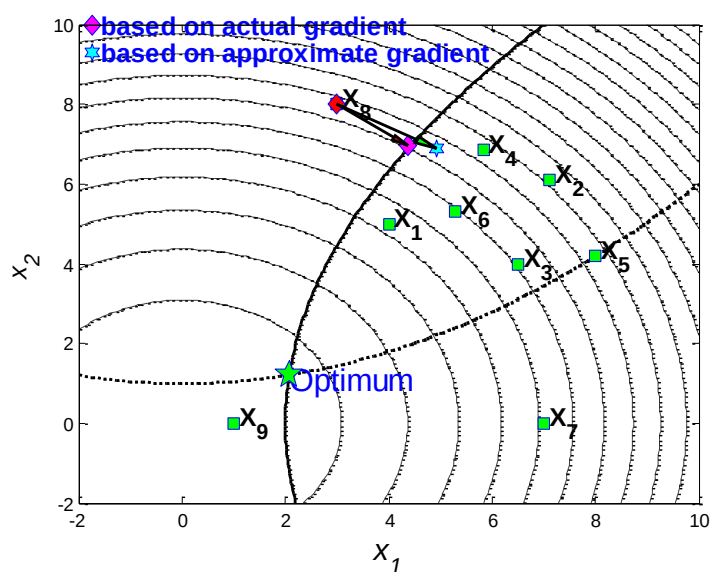
$$\mathbf{x}_1' = \mathbf{x}_1 + \text{rand} \cdot \lambda_{\max} \mathbf{s} \quad (4.11)$$

เมื่อ $\text{rand} \in [0, 1]$ คือ ค่าตัวเลขสุ่มที่กระจายแบบเอกรูป

รูปที่ 4.4 แสดงการปรับปรุงค่าของผลเฉลย $\mathbf{x}_8$ ซึ่งเป็นเวกเตอร์ผลเฉลยที่อยู่นอกพื้นที่หาผลเฉลยได้โดยใช้สมการ (4.11) (เปลี่ยนจาก $\mathbf{x}_1$ เป็น $\mathbf{x}_8$) จากรูป พบว่าผลเฉลยที่ถูกปรับปรุงค่าออกมาจากจะให้ค่าฟังก์ชันเป้าหมายลดลงแล้ว ยังนำผลเฉลยกลับเข้ามายังบริเวณหาคำตอบได้



รูปที่ 4.3 การปรับปรุงตำแหน่งของ $\mathbf{x}_1$ ด้วยค่าประมาณเกรเดียนต์



รูปที่ 4.4 การปรับปรุงตำแหน่งของ $\mathbf{x}_8$ ด้วยค่าประมาณเกรเดียนต์

เราสามารถประยุกต์ใช้ค่าประมาณเกรเดียนต์กับขั้นตอนวิธีวิวัฒนาการทั้งแบบเป้าหมายเดียวและหลายเป้าหมาย ในกรณีของขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมาย ประเด็นปัญหาที่เพิ่มขึ้นมาคือการหาค่าเวกเตอร์ทิศทาง เนื่องจากมีฟังก์ชันเป้าหมายมากกว่าหนึ่งค่า วิธีแก้ปัญหาก็ง่ายที่สุดคือ ทำการประยุกต์ใช้วิธีผลรวมถ่วงน้ำหนัก (weighted sum method) จะทำให้เราสามารถรวมค่าฟังก์ชันเป้าหมายเข้าเป็นฟังก์ชันเป้าหมายสมมูลเป้าหมายเดียว ตัวอย่างของกระบวนการคำนวณของวิธีลูกผสมระหว่างค่าประมาณเกรเดียนต์และขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมาย แสดงในขั้นตอนวิธีที่ 4.2 ซึ่งจะทำการประยุกต์ใช้ค่าประมาณเกรเดียนต์กับผลเฉลยที่ไม่ถูกรอบจำบางส่วน และค่าตัวประกอบถ่วงน้ำหนัก (weighting factors) สร้างด้วยการสุ่ม เมื่อได้ผลเฉลยใหม่แล้วนำเข้าสู่กระบวนการคัดสรรผลเฉลยที่ไม่ถูกรอบจำ ดังแสดงในกระบวนการคำนวณด้านล่าง

ขั้นตอนวิธีที่ 4.2 การผสมระหว่างขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมายและค่าประมาณเกรเดียนต์

ขั้นตอนเริ่มต้น กำหนดพารามิเตอร์ที่จำเป็นทั้งหมด หน่วยเก็บผลเฉลยพารетоตั้งต้น และประชากรตั้งต้นพร้อมค่าฟังก์ชันเป้าหมายของสมาชิกในประชากร กำหนดหน่วยเก็บผลเฉลยสำหรับประมาณค่าเกรเดียนต์

ขั้นตอนคำนวณหลัก

1: ถ้าเงื่อนไขการหยุดคำนวณเป็นจริง หยุดการค้นหาผลเฉลย

ถ้าไม่เป็นจริง ดำเนินการต่อในขั้นตอนที่ 2

2: ผลิตประชากรลูกจากประชากรพ่อแม่

3: ประมาณค่าเกรเดียนต์ของฟังก์ชันเป้าหมาย ที่ตำแหน่งของผลเฉลยที่ไม่ถูกรอบจำบางส่วน

4: สุ่มสร้างค่าตัวประกอบถ่วงน้ำหนักและค้นหาผลเฉลยใหม่ด้วยค่าประมาณเกรเดียนต์

5: คัดสรรผลเฉลยที่ไม่ถูกรอบจำของกลุ่มผลเฉลยรวมกัน ระหว่างสมาชิกในประชากรลูกในขั้นตอนที่ 2 และ 4 และสมาชิกในหน่วยเก็บผลเฉลยพารето

6: ถ้าผลเฉลยที่ไม่ถูกรอบจำในขั้นตอนที่ 5 มีจำนวนมากเกินไป ทำการลบผลเฉลยบางส่วนออกด้วยวิธีการรองผลเฉลย

7: แทนที่สมาชิกในหน่วยเก็บผลเฉลยพารетоด้วยกลุ่มของผลเฉลยในขั้นตอนที่ 6

8: เก็บผลเฉลยบางส่วนในหน่วยเก็บผลเฉลยสำหรับการประมาณค่าเกรเดียนต์ โดยใช้หลักการให้สมาชิกในหน่วยเก็บผลเฉลยกระจายไปทั่วโดเมนการออกแบบมากที่สุด

9: กลับไปขั้นตอนที่ 1:

6.6 การผสมผสานขั้นตอนวิธีวิวัฒนาการกับวิธีซิมเพล็กซ์ลงเขา

วิธีซิมเพล็กซ์ลงเขาเรียกอีกชื่อหนึ่งว่า วิธีเนลเดอร์และมิด (Nelder and Mead) ตามชื่อผู้พัฒนา วิธีการนี้ จัดว่าเป็นวิธีหาค่าเหมาะที่สุดที่ไม่ใช้อนุพันธ์ของฟังก์ชันในการหาผลเฉลยที่มีสมรรถนะในการหาค่าเหมาะที่สุดสูง อย่างไรก็ตาม วิธีการนี้ยังจัดว่าเป็นวิธีหาค่าเหมาะที่สุดวงแคบ (local optimiser) ดังนั้น การผสมวิธีนี้เข้ากับวิธีวิวัฒนาการจะเป็นการทำงานร่วมกันระหว่างการค้นหาผลเฉลยเหมาะที่สุดของทั้งหมด (global optimisation) ที่มีประสิทธิภาพของขั้นตอนวิธีวิวัฒนาการ และการค้นหาในวงแคบที่มีประสิทธิภาพสูงของวิธีซิมเพล็กซ์ลงเขา

การผสมผสานขั้นตอนวิธีวิวัฒนาการกับวิธีซิมเพล็กซ์ลงเขา สามารถทำได้ในรูปแบบคล้ายกับการผสมวิธีวิวัฒนาการกับตัวดำเนินการอื่นๆ ดังที่ผ่านมา นั่นคือ กระบวนการหลักสำหรับหาผลเฉลยเหมาะที่สุด คือ ขั้นตอนวิธีวิวัฒนาการ ส่วนวิธีซิมเพล็กซ์ลงเขาจะถูกใช้เป็นตัวช่วยในการผลิตสมาชิกของประชากรลูกจำนวนหนึ่งในแต่ละวงวนของขั้นตอนวิธีวิวัฒนาการนั้นๆ ส่วนกลุ่มของผลเฉลยเริ่มต้นที่ใช้ในการสร้างซิมเพล็กซ์เริ่มต้น สามารถคัดเลือกได้จากเวกเตอร์ตัวแปรออกแบบที่เป็นสมาชิกในประชากรที่รอบของการคำนวณนั้นๆ ($n + 1$ ค่า) ตัวอย่างกระบวนการคำนวณของวิธีลูกผสมระหว่างขั้นตอนวิธีวิวัฒนาการและวิธีซิมเพล็กซ์ลงเขาสำหรับการหาค่าเหมาะที่สุดที่มีเป้าหมายเดียวแสดงในขั้นตอนวิธีที่ 4.3

ขั้นตอนวิธีที่ 4.3 การผสมระหว่างขั้นตอนวิธีวิวัฒนาการและวิธีซิมเพล็กซ์ลงเขา

ขั้นตอนเริ่มต้น กำหนดพารามิเตอร์ที่จำเป็นทั้งหมด และประชากรเริ่มต้นพร้อมค่าฟังก์ชันเป้าหมายของสมาชิกในประชากร

ขั้นตอนคำนวณหลัก

- 1: ถ้าเงื่อนไขเหมาะที่สุดเป็นจริง หยุดการค้นหาผลเฉลย ถ้าไม่เป็นจริง ดำเนินการต่อในขั้นตอนที่ 2
- 2: ผลิตประชากรลูกด้วยหลักการเฉพาะของขั้นตอนวิธีเหล่านั้น
- 3: คัดเลือกผลเฉลยที่ดีที่สุด $n+1$ ค่า สำหรับเป็นผลเฉลยตั้งต้นของวิธีซิมเพล็กซ์ลงเขา
- 4: ค้นหาผลเฉลยด้วยวิธีซิมเพล็กซ์ลงเขาและกำหนดให้เป็นสมาชิกเพิ่มเติมของประชากรลูก
- 5: หาค่าฟังก์ชันเป้าหมาย และ/หรือ ค่าความเหมาะสม
- 6: ทำการคัดเลือกประชากรรุ่นใหม่
- 7: กลับไปขั้นตอนที่ 1

บทที่ 5 การวิเคราะห์โครงข้อหมุนและโครงข้อแข็ง

5.1 ความนำ

โครงข้อหมุนและโครงข้อแข็งหรือเรียกรวมกันว่าโครงถัก ในที่นี้ คือ โครงสร้างทางวิศวกรรมที่เกิดจากการนำชิ้นส่วนที่มีพื้นที่หน้าตัดคงที่ หรือชิ้นส่วนเอกรูป (uniform bars) ที่มีสองจุดต่อ (2 node) มาเชื่อมต่อเข้าด้วยกันด้วยข้อต่อหมุน (revolute joints) สำหรับโครงข้อหมุน และเชื่อมต่อด้วยข้อต่อแข็งเกร็งสำหรับโครงข้อแข็ง โครงข้อหมุนจึงมีคุณสมบัติพิเศษคือ ชิ้นส่วนเอกรูปที่ใช้ประกอบเป็นโครงข้อหมุนจะรับเฉพาะแรงดึงหรือกดในแนวแกนเท่านั้น ส่วนโครงข้อแข็งจะรับโมเมนต์ดัดและโมเมนต์บิดเพิ่มจากแรงในแนวแกน โครงถักสามารถแบ่งออกเป็นโครงถักในสองและสามมิติ โครงสร้างในชีวิตประจำวันที่จัดว่าเป็นโครงถักมีหลากหลายรูปแบบ เช่น โครงสร้างสะพาน เสาส่งสัญญาณ โทรคมนาคม หอดูดาว หุ่นยนต์ หุ่นยนต์ หุ่นยนต์ เสาไฟฟ้าแรงสูง โครงสร้างป้ายโฆษณา และโครงสร้างทางวิศวกรรมอื่นๆ อีกมากมาย ข้อดีของการเลือกใช้โครงถักคือ การก่อสร้างหรือการประกอบและการรื้อถอนทำได้ง่าย ราคาไม่แพงเกินไป สามารถนำไปก่อสร้างในบริเวณที่เข้าถึงได้ยาก เช่น บนภูเขา นอกจากนี้ การใช้งานโครงถักยังช่วยลดการสิ้นเปลืองที่เห็นยว่นาโดยลม (ถึงแม้ว่าอากาศพลวัตแบบไม่คงตัวที่ไหลผ่านชิ้นส่วนขนาดเล็กจะมีความถี่สูง แต่ก็มีความหนาของแรงกระทำต่ำ)

เนื้อหาในบทนี้ เป็นการวิเคราะห์โครงสร้างในแบบต่างๆ ไปของโครงถัก เพื่อความง่ายในการอธิบาย จะทำการนำเสนอการวิเคราะห์ทางไฟไนต์เอลิเมนต์ของโครงข้อหมุนเป็นหลัก อย่างไรก็ตาม แนวคิดของการวิเคราะห์โครงข้อหมุนสามารถขยายผลไปยังการวิเคราะห์ทางไฟไนต์เอลิเมนต์ของโครงข้อแข็งได้ เพราะใช้แนวคิดพื้นฐานเดียวกัน

5.2 การวิเคราะห์ทางกลของโครงถัก

ในการออกแบบโครงสร้างสำหรับรับภาระรูปแบบต่างๆ จำเป็นที่จะต้องรู้คุณสมบัติทางกายภาพของโครงสร้าง เช่น ความเค้น ความเครียด ระยะการโก่งตัว หรือพลังงานความเครียดของโครงสร้าง เป็นต้น เพื่อที่จะนำคุณสมบัติดังกล่าวมากำหนดเป็นฟังก์ชันเป้าหมายและเงื่อนไขบังคับในกระบวนการออกแบบ ในการวิเคราะห์ทางกลของโครงสร้างนั้น พฤติกรรมของโครงสร้างสามารถอธิบายด้วยแบบจำลองทางคณิตศาสตร์ ซึ่งแสดงในรูปของสมการเชิงอนุพันธ์ และสมการเหล่านี้สามารถหาผลเฉลยจากการวิเคราะห์โดยตรงได้เฉพาะกรณีของโครงสร้างที่มีรูปร่างไม่ซับซ้อนเท่านั้น แต่โครงสร้างที่ใช้ในงานจริงโดยส่วนมากจะมีลักษณะรูปร่างที่ซับซ้อน เช่น โครงสร้างที่มีส่วนเว้าส่วนโค้ง และมีรูปร่างแปลกๆ จึงเป็นผลทำให้ไม่สามารถหาผลเฉลยแม่นยำตรงจากสมการเชิงอนุพันธ์ได้ อย่างไรก็ตาม โดยการใช้วิธีคำนวณเชิงตัวเลข เราสามารถวิเคราะห์พฤติกรรมของโครงสร้างเหล่านี้ได้ การวิเคราะห์โครงสร้างด้วยวิธีคำนวณเชิงตัวเลขที่เป็นที่นิยมมากที่สุดรูปแบบหนึ่งคือ การวิเคราะห์ไฟไนต์เอลิเมนต์ (finite element analysis) ซึ่งเป็นหนึ่งในวิธีเชิงตัวเลขสำหรับหาผลเฉลยของสมการเชิงอนุพันธ์

การวิเคราะห์ทางไฟไนต์เอลิเมนต์ จะเกิดประโยชน์สูงสุดเมื่อนำมาประยุกต์ใช้แก้ปัญหาสมการเชิงอนุพันธ์ที่มีโดเมน (หรือรูปทรงของโครงสร้าง) สลับซับซ้อน หลักการเบื้องต้นคือการประมาณโดเมนนั้นด้วยโดเมนย่อยที่มีรูปทรงอย่างง่ายขนาดเล็กหลายๆอันซึ่งเรียกว่า เอลิเมนต์ (ในกรณีของโครงข้อหมุน เอลิเมนต์คือชิ้นส่วนเอกรูปของโครงสร้างนั่นเอง) ผลเฉลยโดยประมาณสำหรับโดเมนย่อยนั้นแสดงในรูปของฟังก์ชันอย่างง่าย เช่น ฟังก์ชันพหุนาม เมื่อทำการพิจารณาผลกระทบของเอลิเมนต์เหล่านั้นในภาพรวมต่อพลวัตของโครงสร้าง ทำให้ได้ผลเฉลยเชิงอนุพันธ์โดยประมาณจากการแก้ระบบสมการพีชคณิต

การวิเคราะห์โครงสร้างด้วยวิธีไฟไนต์เอลิเมนต์นำเสนอในที่นี้ คือ การวิเคราะห์ด้วยวิธีการกระจัด (displacement – based approach) โดยยึดตามกฎการเคลื่อนที่ของนิวตัน (Newton's laws of motion) และพิจารณาพฤติกรรมของวัสดุเป็นแบบการกระจัดขนาดเล็กเชิงเส้น (linear small-displacement) ซึ่งเมื่อประยุกต์ใช้กับระบบโครงสร้างที่ถูกแบ่งออกเป็นเอลิเมนต์ย่อยๆ สามารถเขียนอยู่ในรูปของสมการของลากรางจ์ ดังนี้

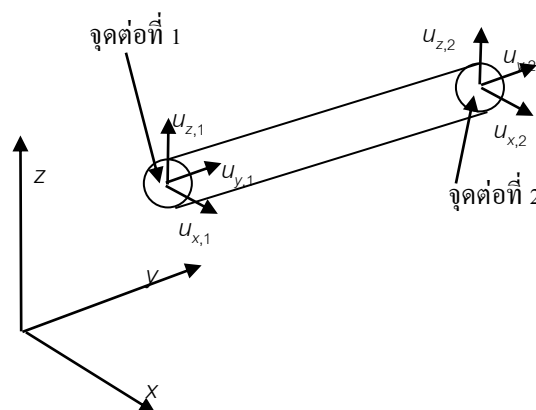
$$\frac{d}{dt} \frac{\partial T}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} + \frac{\partial U}{\partial q_i} - F_i = 0 \quad (5.1)$$

เมื่อ q_i คือ การกระจัดเสมือนของโครงสร้าง

T คือ พลังงานจลน์รวมของโครงสร้าง

U คือ พลังงานศักย์ยืดหยุ่นของโครงสร้าง

F_i คือ แรงภายนอกที่กระทำสอดคล้องกับการกระจัด q_i



รูปที่ 5.1 การกระจัดที่จุดต่อของเอลิเมนต์ของโครงข้อหมุนสามมิติ

เอลิเมนต์ใดๆจะมีองค์ประกอบพื้นฐานคือ รูปทรงของเอลิเมนต์ เช่น รูปทรงเส้นตรงสำหรับเอลิเมนต์ของโครงข้อหมุน รูปสามเหลี่ยมหรือสี่เหลี่ยมสำหรับ โครงสร้างแบบแผ่น เป็นต้น และส่วนประกอบอื่นคือ จุดต่อ (node) ของเอลิเมนต์ และขอบเขตของเอลิเมนต์ การคำนวณทางไฟไนต์เอลิ

เมนต์เป็นการทำนายค่าการกระจัดที่จุดต่อของเอลิเมนต์เหล่านั้น รูปที่ 5.1 แสดงภาพเอลิเมนต์แท่งเอกรูปชนิดสองจุดต่อ (2-node bar element) สำหรับโครงข้อหมุนในปริภูมิสามมิติ

จากรูป แสดงเวกเตอร์การกระจัดที่จุดต่อของเอลิเมนต์ใดๆ เวกเตอร์การกระจัดของเอลิเมนต์ใดๆ เขียนแทนด้วย $\mathbf{q}_e$ สามารถแสดงได้ดังนี้

$$\mathbf{q}_e = \begin{Bmatrix} \delta_1 \\ \delta_2 \end{Bmatrix} \quad (5.2)$$

เมื่อ $\delta = \{u_x \ u_y \ u_z\}^T$

δ_1, δ_2 คือ เวกเตอร์การกระจัดของจุดต่อที่ 1 และ 2 ตามลำดับ

การประมาณค่าของเวกเตอร์การกระจัดที่จุดต่อของเอลิเมนต์ สามารถทำได้โดย การประยุกต์ใช้การประมาณค่าในช่วง (interpolation) ด้วยฟังก์ชันพหุนาม ค่าการกระจัดที่จุด $\mathbf{x}$ ใดๆ ภายในเอลิเมนต์ สามารถแสดงได้ดังนี้

$$\delta(\mathbf{x}) = \{\mathbf{p}\}\{\mathbf{a}\} \quad (5.3)$$

เมื่อ $\mathbf{p}$ คือ เวกเตอร์แถวของฟังก์ชันพหุนาม และ $\mathbf{a}$ คือ เวกเตอร์คอลัมน์ของสัมประสิทธิ์ของฟังก์ชันพหุนาม เมื่อทราบพิกัดของจุดต่อ เราสามารถหาเวกเตอร์สัมประสิทธิ์ของฟังก์ชันพหุนาม ได้เป็น

$$\mathbf{q}_e = [\mathbf{C}]\{\mathbf{a}\} \quad (5.4)$$

ดังนั้น

$$\delta = \{\mathbf{p}\}[\mathbf{C}]^{-1}\mathbf{q}_e = [\mathbf{N}]\mathbf{q}_e \quad (5.5)$$

เนื่องจากเป็นแบบจำลองเชิงเส้น เราสามารถแสดงค่าเวกเตอร์ความเครียด $\{\boldsymbol{\varepsilon}\}$ ที่จุดใดๆ ในรูปของ

$$\boldsymbol{\varepsilon} = \boldsymbol{\mathcal{D}}\delta \quad (5.6)$$

เมื่อ $\boldsymbol{\mathcal{D}}$ คือ ตัวดำเนินการเชิงอนุพันธ์ ซึ่งจะได้ความสัมพันธ์

$$\boldsymbol{\varepsilon} = \boldsymbol{\mathcal{D}}\{\mathbf{p}\}[\mathbf{C}]^{-1}\mathbf{q}_e = [\mathbf{B}]\mathbf{q}_e \quad (5.7)$$

และสามารถแสดงความสัมพันธ์ระหว่างเวกเตอร์ความเค้น $\boldsymbol{\sigma}$ และความเครียด ดังนี้ (constitutive equation)

$$\{\boldsymbol{\sigma}\} = [\mathbf{D}]\{\boldsymbol{\varepsilon}\} \quad (5.8)$$

เมื่อ $[D]$ คือ เมตริกซ์สภาพยืดหยุ่น (elasticity matrix)

พลังงานความเครียดหรือพลังงานศักย์ยืดหยุ่นของเอลิเมนต์ สามารถคำนวณได้จากสมการ (5.9)

$$U_e = \frac{1}{2} \iiint_{\Omega} \boldsymbol{\sigma}^T \boldsymbol{\varepsilon} dV = \frac{1}{2} \mathbf{q}_e^T \left(\iiint_{\Omega} \mathbf{B}^T \mathbf{D} \mathbf{B} dV \right) \mathbf{q}_e = \frac{1}{2} \mathbf{q}_e^T [\mathbf{k}_e] \mathbf{q}_e \quad (5.9)$$

เมื่อ

$$\mathbf{k}_e = \iiint_{\Omega} \mathbf{B}^T \mathbf{D} \mathbf{B} dV \quad (5.10)$$

เรียกว่า เมตริกซ์ความแข็งตึง (stiffness matrix) ของเอลิเมนต์ ในขณะเดียวกัน เราสามารถหาความสัมพันธ์ระหว่างเวกเตอร์ของการกระจัดของเอลิเมนต์ $\mathbf{q}_e$ กับเวกเตอร์การกระจัดทั้งหมดของระบบโครงสร้าง $\mathbf{q}$ (เมื่อรวมทุกจุดต่อบนเอลิเมนต์) ดังนี้

$$\mathbf{q}_e = [\mathbf{R}_e] \mathbf{q} \quad (5.11)$$

ดังนั้น จะได้พลังงานศักย์ยืดหยุ่นของเอลิเมนต์ที่ e -th ดังนี้

$$U_e = \frac{1}{2} \mathbf{q}^T [\mathbf{R}_e^T \mathbf{k}_e \mathbf{R}_e] \mathbf{q} \quad (5.12)$$

ในทำนองเดียวกันจะได้พลังงานจลน์ของเอลิเมนต์ ดังนี้

$$T_e = \frac{1}{2} \dot{\mathbf{q}}_e^T \left(\rho \iiint_{\Omega} \mathbf{N}^T \mathbf{N} dV \right) \dot{\mathbf{q}}_e = \frac{1}{2} \dot{\mathbf{q}}_e^T [\mathbf{m}_e] \dot{\mathbf{q}}_e = \frac{1}{2} \dot{\mathbf{q}}^T [\mathbf{R}_e^T \mathbf{m}_e \mathbf{R}_e] \dot{\mathbf{q}} \quad (5.13)$$

เมื่อ $\mathbf{m}_e$ เรียกว่า เมตริกซ์มวลของเอลิเมนต์ที่ e -th (element mass matrix หรือ consistent mass matrix) พลังงานศักย์และพลังงานจลน์รวมของระบบ สามารถเขียนได้ดังนี้

$$U = \sum_{e=1}^N U_e = \frac{1}{2} \mathbf{q}^T \sum_{e=1}^N [\mathbf{R}_e^T \mathbf{k}_e \mathbf{R}_e] \mathbf{q} \quad (5.14)$$

และ

$$T = \sum_{e=1}^N T_e = \frac{1}{2} \dot{\mathbf{q}}^T \sum_{e=1}^N [\mathbf{R}_e^T \mathbf{m}_e \mathbf{R}_e] \dot{\mathbf{q}} \quad (5.15)$$

เมื่อแทนค่าพลังงานศักย์และพลังงานจลน์จาก (5.14) และ (5.15) ลงใน (5.1) จะได้ระบบสมการเชิงอนุพันธ์อันดับที่สอง ดังนี้

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{F} \quad (5.16)$$

$$\text{เมื่อ } \mathbf{K} = \sum_{e=1}^{N_e} [\mathbf{R}_e^T \mathbf{k}_e \mathbf{R}_e]$$

$$\mathbf{M} = \sum_{e=1}^{N_e} [\mathbf{R}_e^T \mathbf{m}_e \mathbf{R}_e]$$

$\mathbf{K}$ และ $\mathbf{M}$ คือ เมตริกซ์ความแข็งตึงและเมตริกซ์มวลของโครงสร้างตามลำดับ การรวมกันของเมตริกซ์ความแข็งตึงและเมตริกซ์มวลของเอลิเมนต์ เพื่อให้ได้เมตริกซ์ของโครงสร้างทั้งสอง เรียกว่า กระบวนการประกอบเมตริกซ์ (assembly process) ในการคำนวณเชิงไฟไนต์เอลิเมนต์ สมการ (5.16) คือสมการพลวัตหรือสมการการสั่นสะเทือนของระบบโครงสร้างโดยวิธีไฟไนต์เอลิเมนต์ซึ่งสามารถนำมาใช้ในการวิเคราะห์โครงสร้างในรูปแบบต่างๆ ดังนี้

8.2.1 การวิเคราะห์เชิงสถิต

การวิเคราะห์ทางกลศาสตร์ของโครงสร้างที่การกระจัดและค่าพลังงานต่างๆ ไม่เปลี่ยนแปลงตามเวลา หรืออาจกล่าวได้ว่า เป็นการวิเคราะห์พฤติกรรมของโครงสร้างกรณีที่ไม่มี การเคลื่อนที่ ซึ่งหมายถึง การลบเมตริกซ์มวลออกจากสมการ (5.16) จะได้

$$\mathbf{K}\mathbf{q} = \mathbf{F} \quad (5.17)$$

ในปัญหาการวิเคราะห์เชิงสถิต เวกเตอร์การกระจัดและแรงภายนอกจะมีขนาดเท่ากับระดับความเป็นอิสระ (degree of freedom) ของโครงสร้าง โดยทั่วไปมักจะรู้ค่าขอบเขตหรือรู้ค่าการกระจัดบางจุดต่อ $\mathbf{q}_b$ ซึ่ง ที่ตำแหน่งจุดต่อเหล่านี้จะไม่รู้ค่าแรงกระทำ ค่าแรงที่เป็นตัวแปรเหล่านี้คือแรงปฏิกิริยาที่จุดต่อเงื่อนไขขอบเขต $\mathbf{F}_b$ ในขณะเดียวกัน เราจะทราบค่าของแรงภายนอกที่จุดต่อส่วนที่เหลือ $\mathbf{F}_a$ และมีการกระจัดที่จุดต่อเหล่านั้น $\mathbf{q}_a$ เป็นตัวแปรที่จะต้องหาผลเฉลย ทำการจัดรูปสมการเมตริกซ์ใน (5.17) ให้สอดคล้องกับเวกเตอร์เหล่านี้ จะได้ระบบสมการเชิงเส้น

$$\begin{bmatrix} \mathbf{K}_{aa} & \mathbf{K}_{ab} \\ \mathbf{K}_{ba} & \mathbf{K}_{bb} \end{bmatrix} \begin{bmatrix} \mathbf{q}_a \\ \mathbf{q}_b \end{bmatrix} = \begin{bmatrix} \mathbf{F}_a \\ \mathbf{F}_b \end{bmatrix} \quad (5.18)$$

ทำการแก้สมการจะได้การกระจัดของโครงสร้าง

$$\mathbf{q}_a = \mathbf{K}_{aa}^{-1}(\mathbf{F}_a - \mathbf{K}_{ab}\mathbf{q}_b) \quad (5.19)$$

และแรงปฏิกิริยา

$$\mathbf{F}_b = \mathbf{K}_{ba}\mathbf{q}_a + \mathbf{K}_{bb}\mathbf{q}_b \quad (5.20)$$

เมื่อทราบค่าการกระจัดทุกจุดต่อแล้วสามารถคำนวณหาค่าความเครียดและความเค้นที่ทุกจุดของโครงสร้างได้โดยใช้สมการ (5.7) และ (5.8) การวิเคราะห์เชิงสถิติสามารถขยายผลไปยังการทำนายอายุการล้าของชิ้นงาน ซึ่งเป็นปัจจัยสำคัญประการหนึ่งในการออกแบบได้

5.2.2 การวิเคราะห์เชิงพลวัตบนโดเมนความถี่

การวิเคราะห์เชิงพลวัตหรือการวิเคราะห์การสั่นสะเทือนคือการแก้ระบบสมการ (5.16) โดยตรง แรงภายนอกในที่นี้หมายถึง แรงพลวัตที่มีการเปลี่ยนขนาดเมื่อเวลาเปลี่ยนไป ส่วนแรงสถิติจะไม่มีผลต่อการสั่นสะเทือนของระบบ เมื่อโครงสร้างมีเงื่อนไขขอบเขตเหมือนในกรณีการวิเคราะห์เชิงสถิติ สามารถจัดกลุ่มสมการได้เป็น

$$\begin{bmatrix} \mathbf{M}_{aa} & \mathbf{M}_{ab} \\ \mathbf{M}_{ba} & \mathbf{M}_{bb} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_a \\ \ddot{\mathbf{q}}_b \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{aa} & \mathbf{K}_{ab} \\ \mathbf{K}_{ba} & \mathbf{K}_{bb} \end{bmatrix} \begin{bmatrix} \mathbf{q}_a \\ \mathbf{q}_b \end{bmatrix} = \begin{bmatrix} \mathbf{F}_a \\ \mathbf{F}_b \end{bmatrix} \quad (5.21)$$

สมมติให้เงื่อนไขขอบเขต $\mathbf{r}_b = \mathbf{0}$ จะได้ระบบสมการดังนี้

$$\mathbf{M}_{aa}\ddot{\mathbf{q}}_a + \mathbf{K}_{aa}\mathbf{q}_a = \mathbf{F}_a \quad (5.22)$$

และ

$$\mathbf{M}_{ba}\ddot{\mathbf{q}}_a + \mathbf{K}_{ba}\mathbf{q}_a = \mathbf{F}_b \quad (5.23)$$

จากสมการ (5.22) ในกรณีที่ไม่มีแรงภายนอกกระทำหรือที่เรียกว่า การสั่นสะเทือนแบบอิสระ (free vibration) จะได้

$$\mathbf{M}_{aa}\ddot{\mathbf{q}}_a + \mathbf{K}_{aa}\mathbf{q}_a = \mathbf{0} \quad (5.24)$$

แทนค่า $\mathbf{q}_a = \bar{\mathbf{q}}_a e^{i\omega t}$ ผลเฉลยของ (5.24) จะอยู่ในรูปของปัญหาค่าลักษณะเฉพาะ (eigenvalue problem) ดังนี้

$$(\mathbf{K}_{aa} - \omega^2 \mathbf{M}_{aa})\bar{\mathbf{q}}_a = \mathbf{0} \quad (5.25)$$

เมื่อทำการหาคำตอบของ (5.25) จะได้ค่าความถี่ธรรมชาติของระบบ N_q ค่า

$$\boldsymbol{\omega} = \{\omega_1, \omega_2, \dots, \omega_{N_q}\}$$

และเวกเตอร์ลักษณะเฉพาะ (eigenvectors) N_q ค่าที่สอดคล้องกัน

$$\Phi = [\bar{\mathbf{q}}_1, \bar{\mathbf{q}}_2, \dots, \bar{\mathbf{q}}_{N_q}]$$

เมื่อ N_q คือ ขนาดของ $\mathbf{K}_{aa}$ ส่วนค่าลักษณะเฉพาะ สามารถเขียนได้เป็น $\lambda_i = \omega_i^2$ (รายละเอียดบางส่วน ของปัญหาค่าลักษณะเฉพาะอยู่ในบทที่ 1) ค่าลักษณะเฉพาะและเวกเตอร์ลักษณะเฉพาะ มีความสัมพันธ์เชิงตั้งฉาก (orthogonality) ดังนี้

$$\begin{aligned}\Phi^T \mathbf{M}_{aa} \Phi &= \text{diag}(\mu_i) \\ \Phi^T \mathbf{K}_{aa} \Phi &= \text{diag}(\mu_i \omega_i^2)\end{aligned}\quad (5.26)$$

เมื่อ μ_i คือ ค่าคงที่ใดๆ โดยการประยุกต์ใช้แบบจำลองการหน่วงแบบสัดส่วน (proportional damping) หรือ การหน่วงเรย์ไล (Rayleigh damping) เราสามารถใส่เมตริกซ์การหน่วงเข้าไปในสมการ ได้ดังนี้

$$\mathbf{M}_{aa} \ddot{\mathbf{q}}_a + \mathbf{C}_{aa} \dot{\mathbf{q}}_a + \mathbf{K}_{aa} \mathbf{q}_a = \mathbf{F}_a \quad (5.27)$$

เมื่อ $\mathbf{C}_{aa} = \alpha \mathbf{M}_{aa} + \beta \mathbf{K}_{aa}$

α และ β คือ สัมประสิทธิ์การหน่วงที่ต้องทำการระบุค่า

จากสมการที่ (5.27) แทนค่า $\mathbf{q}_a = \bar{\mathbf{q}}_a e^{i\omega t}$ และ $\mathbf{F}_a = \bar{\mathbf{F}}_a e^{i\omega t}$ ในสมการ จะได้เมตริกซ์ฟังก์ชันตอบสนองเชิงความถี่ (frequency response function, FRF) ซึ่งสามารถคำนวณได้จาก

$$\mathbf{H}(\omega) = [-\omega^2 \mathbf{M}_{aa} + j\omega \mathbf{C}_{aa} + \mathbf{K}_{aa}]^{-1} = \Phi \text{diag} \left\{ \frac{1}{\mu_i (\omega_i^2 - \omega^2 + 2j\xi_i \omega_i \omega)} \right\} \Phi^T \quad (5.28)$$

เมื่อ ω คือ ค่าความถี่ของแรงกระทำขาเข้าและการกระจัดของจุดต่อที่ตอบสนองต่อแรงนั้น

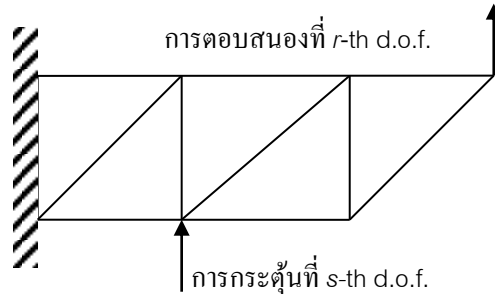
$$\xi_i = \frac{1}{2} \left(\frac{\alpha}{\omega_i} + \beta \omega_i \right) \text{ คือ อัตราส่วนการหน่วง (damping ratio)}$$

ค่า FRF อาจมองว่าเป็นอัตราส่วนระหว่างการตอบสนองแบบฮาร์มอนิกของข้อมูลขาออกต่อการกระตุ้นแบบฮาร์มอนิกที่ความถี่เดียวกันของข้อมูลขาเข้า ในที่นี้คืออัตราส่วนระหว่างการกระตุ้นที่ระดับความเป็นอิสระ (degree of freedom) r -th ต่อแรงกระตุ้นที่ระดับความเป็นอิสระ s -th ด้วยความถี่ ω ซึ่งสามารถเขียนได้เป็น

$$\bar{\mathbf{\delta}}_a = \mathbf{H}(\omega) \bar{\mathbf{F}}_a \quad (5.29)$$

รูปที่ 5.2 แสดงการวัดค่า H_{rs} ของโครงข้อหมุน ในกระบวนการออกแบบ เราสามารถประมาณค่าของ FRF เพื่อลดเวลาในการคำนวณได้ดังนี้

$$\mathbf{H}(\omega) \approx \sum_{i=1}^m \frac{\bar{\mathbf{q}}_i \bar{\mathbf{q}}_i^T}{\mu_i (\omega_i^2 - \omega^2 + 2j\zeta_i \omega_i \omega)} + \mathbf{K}_{aa}^{-1} - \sum_{i=1}^m \frac{\bar{\mathbf{q}}_i \bar{\mathbf{q}}_i^T}{\mu_i \omega_i^2} \quad (5.30)$$



รูปที่ 5.2 แสดงการหาฟังก์ชันตอบสนองเชิงความถี่ของโครงสร้าง

ถ้าเรานิยามการส่งถ่ายแรง (force transmissibility) เขียนแทนด้วย $\mathbf{T}(\omega)$ คืออัตราส่วนของแรงปฏิกิริยาแบบฮาร์มอนิกที่บริเวณเงื่อนไขขอบเขตของโครงสร้างต่อแรงกระตุ้นแบบฮาร์มอนิกที่กระทำที่จุดต่อใดๆของโครงสร้าง ซึ่งเขียนได้ดังนี้

$$\bar{\mathbf{F}}_b = \mathbf{T}(\omega) \bar{\mathbf{F}}_a \quad (5.31)$$

แทนค่า $\bar{\mathbf{F}}_b = \bar{\mathbf{F}}_b e^{i\omega t}$ สมการที่ (5.29) และ (5.31) ลงใน (5.23) จะได้ค่าของการส่งถ่ายแรงเป็น

$$\mathbf{T}(\omega) = (-\omega^2 \mathbf{M}_{ba} + \mathbf{K}_{ba}) \mathbf{H} \quad (5.32)$$

ในการออกแบบที่เกี่ยวข้องกับการสั่นสะเทือนของโครงข้อหมุน ค่า FRF และ FT เป็นค่าที่ใช้หาสมรรถนะของโครงสร้าง หากค่าของ FRF และ FT ในช่วงความถี่ที่พิจารณามีค่าต่ำ นั้นแสดงถึงการออกแบบเพื่อลดการสั่นสะเทือนที่ดี การวิเคราะห์โครงสร้างด้วยระเบียบวิธีทางไฟไนต์เอลิเมนต์ในหัวข้อนี้ เป็นการบรรยายหลักการเบื้องต้นและแนวคิดต่างๆ ไปของการวิเคราะห์เชิงสถิติและเชิงพลวัตของโครงสร้างที่สามารถประยุกต์ใช้ได้กับทั้งโครงข้อหมุน และโครงสร้างแบบต่อเนื่อง (continuous structures) ซึ่งจะนำเสนอในบทต่อไป

5.2.3 การวิเคราะห์เชิงพลวัตบนโดเมนเวลา

จากสมการที่ (5.27) เพื่อความสะดวกในการอธิบาย ทำการลบตัวห้อย a และ aa ออกจากสมการ จะได้สมการของการเคลื่อนที่ดังนี้

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{F}(t) \quad (5.33)$$

การหาผลเฉลยของ (5.33) สามารถทำได้โดยการประยุกต์ใช้วิธีการอินทิเกรตเชิงตัวเลข (numerical integration) ซึ่งสามารถประยุกต์ใช้ได้หลายวิธี ในที่นี้จะประยุกต์ใช้วิธีของนิวมาร์ค (Newmark time integration method หรือบางทีเรียก implicit time integration method) ซึ่งมีขั้นตอนการคำนวณดังนี้

ขั้นตอนก่อนกระบวนการคำนวณ

กำหนดค่า $\beta = 0.5$, $\gamma = 0.25$, Δt

หาค่าของ

$$\begin{aligned} b_1 &= 1/(\beta \Delta t^2) \\ b_2 &= 1/(\beta \Delta t) \\ b_3 &= (1-2\beta)/(2\beta) \\ b_4 &= \gamma/(\beta \Delta t) \\ b_5 &= (1-\gamma)/(\beta) \\ b_6 &= \Delta t (1-\gamma/(2\beta)) \end{aligned}$$

คำนวณหาเมทริกซ์ความแข็งเชิงประสิทธิผล (effective stiffness matrix)

$$\mathbf{K}_e = \mathbf{K} + b_1\mathbf{M} + b_4\mathbf{C} \quad (5.34)$$

หาค่าอินเวอร์สของ $\mathbf{K}_e$ ($\mathbf{K}_e^{-1}$)

กำหนดค่าเงื่อนไขเริ่มต้น (initial conditions) $\mathbf{q}_0, \dot{\mathbf{q}}_0, \mathbf{F}_0$

คำนวณค่า $\ddot{\mathbf{q}}_0 = \mathbf{M}^{-1}(\mathbf{F}_0 - \mathbf{C}\dot{\mathbf{q}}_0 - \mathbf{K}\mathbf{q}_0)$

ขั้นตอนการคำนวณหลัก

1. สำหรับ $n = 0$ ถึง N เมื่อ N คือ จำนวนจุดตัวอย่างที่ต้องการ
2. หาเวกเตอร์ของแรงเชิงประสิทธิผล (effective load vector)

$$\mathbf{F}_e = \mathbf{M}(b_1(\mathbf{q}_{n+1} - \mathbf{q}_n) - b_2\dot{\mathbf{q}}_n - b_3\ddot{\mathbf{q}}_n) + \mathbf{C}(b_4(\mathbf{q}_{n+1} - \mathbf{q}_n) + b_5\dot{\mathbf{q}}_n + b_6\ddot{\mathbf{q}}_n) + \mathbf{K}\mathbf{q}_{n+1} \quad (5.35)$$

3. คำนวณหาค่าของ

$$\begin{aligned} \mathbf{q}_{n+1} &= \mathbf{K}_e^{-1}\mathbf{F}_e \\ \dot{\mathbf{q}}_{n+1} &= b_4(\mathbf{q}_{n+1} - \mathbf{q}_n) + b_5\dot{\mathbf{q}}_n + b_6\ddot{\mathbf{q}}_n \end{aligned} \quad (5.36)$$

4. นับ n ตัวต่อไป (กลับไปขั้นตอนที่ 2)

จากการประยุกต์ใช้การอินทิเกรตเชิงตัวเลข ทำให้เราทราบการตอบสนองของการกระจัด $\mathbf{q}$ เมื่อมีการกระตุ้นขาเข้าของระบบโครงสร้างคือ $\mathbf{F}(t)$ ค่าการตอบสนองของการกระจัดที่เวลาใดๆ

สามารถนำไปกำหนดเป็นค่าฟังก์ชันเป้าหมายในการออกแบบได้เหมือนกันกับกรณีของการตอบสนองเชิงความถี่ ตัวอย่างเช่น การประยุกต์ใช้ในการออกแบบการควบคุมการสั่นสะเทือนเหมาะสมที่สุด

บทที่ 6 กรณีศึกษา การหาค่าเหมาะที่สุดแบบอัตโนมัติของโครงข้อ

หมูน

6.1 ความนำ

โดยทั่วไปในการออกแบบโครงถัก ผู้ออกแบบมักทำการกำหนดทอพอโลยี (topology) ของโครงถักขึ้นมาก่อน จากนั้นค่อยดำเนินการหารูปทรง (ตำแหน่งของจุดต่อบางส่วน) และขนาดของเอลิเมนต์ เพื่อให้ได้โครงสร้างที่เหมาะสมที่สุด โครงสร้างที่ได้จากการดำเนินการในลักษณะนี้จะขึ้นอยู่กับประสบการณ์ของผู้ออกแบบเป็นหลัก การหาค่าเหมาะที่สุดเป็นอีกทางเลือกหนึ่งของการประยุกต์ใช้คอมพิวเตอร์เพื่อช่วยในการออกแบบ ในหัวข้อนี้จะนำเสนอการออกแบบที่เรียกว่าการออกแบบอัตโนมัติหรือการหาค่าทอพอโลยี รูปร่างและขนาดของโครงถักพร้อมกัน หรือหาค่าเหล่านี้ด้วยการรันวิธีเหมาะที่สุดเพียงครั้งเดียว ข้อดีของกระบวนการออกแบบลักษณะนี้คือ สามารถช่วยให้นักออกแบบที่ประสบการณ์ไม่สูงมาก สามารถออกแบบได้ดีทัดเทียมกับวิศวกรผู้ออกแบบที่มีประสบการณ์สูงกว่า

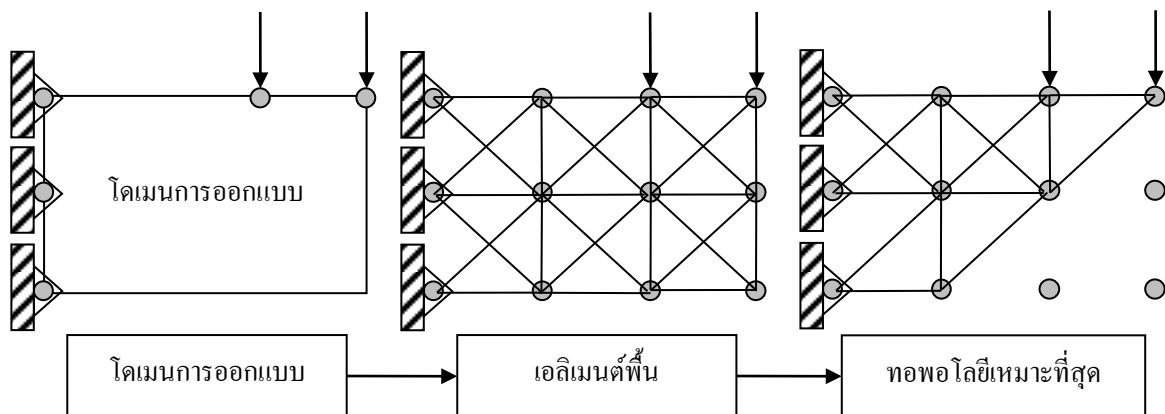
6.2 การหาขนาด รูปทรง และทอพอโลยีเหมาะที่สุดพร้อมกันของโครงข้อหมูนสองมิติ

ในการออกแบบโครงข้อหมูน ผู้ออกแบบอาจทำการหาทอพอโลยีเหมาะที่สุดในขั้นต้น เมื่อได้โครงร่างที่ดีแล้ว ทำการหาขนาดและรูปทรงเหมาะที่สุด แต่จากงานวิจัยของผู้แต่งและนักวิจัยหลายคนพบว่า การหาทอพอโลยี รูปทรง และขนาดเหมาะที่สุดพร้อมกัน จะให้ผลเฉลยการออกแบบที่มีประสิทธิภาพสูงกว่าการออกแบบหลายระดับ (หรือการออกแบบทีละขั้นตอน) ดังนั้น ในการสาธิตการออกแบบในหัวข้อนี้จะเป็นการหาค่าเหมาะที่สุดของโครงข้อหมูน โดยพิจารณาตัวแปรออกแบบกำหนดทอพอโลยี รูปทรง และขนาด พร้อมกันในปัญหาการออกแบบเดียวกัน เพื่อสามารถใช้ในการออกแบบจริงได้ ปัญหาการออกแบบจะเป็นปัญหาแบบหลายฟังก์ชันเป้าหมาย กล่าวคือ ทำการหามวลต่ำสุดและความน่าเชื่อถือสูงสุดของโครงสร้างพร้อมกัน เพื่อสามารถแก้ปัญหาการเชื่อมต่อตัวแปรออกแบบทั้งสามชนิดเข้าด้วยกัน และสามารถหาขอบหน้าพารโธของปัญหาการออกแบบได้ด้วยการรันวิธีหาค่าเหมาะที่สุดเพียงครั้งเดียว จะทำการประยุกต์ใช้ขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมายในการค้นหาผลเฉลยเหมาะที่สุดพารโธ

6.2.1 การหาทอพอโลยีเหมาะที่สุดของโครงข้อหมูนด้วยวิธีเอลิเมนต์พื้น

ในการหาทอพอโลยีเหมาะที่สุดของโครงข้อหมูน ความยุ่งยากของกระบวนการออกแบบคือการกำหนดเวกเตอร์ตัวแปรออกแบบทอพอโลยี รูปที่ 6.1 แสดงกระบวนการหาทอพอโลยีเหมาะที่สุดของโครงข้อหมูนโดยใช้วิธีเอลิเมนต์พื้น (ground elements) ในขั้นตอนเริ่มต้น ผู้ออกแบบกำหนดโดเมนการออกแบบ ตำแหน่งที่รับภาระของโครงสร้าง และตำแหน่งที่มีการยึดกับพื้นของโครงสร้าง (หรือเงื่อนไขขอบเขตการกระจัด) โจทย์ของการออกแบบคือ โครงข้อหมูนควรมีโครงร่างอย่างไรถึงจะ

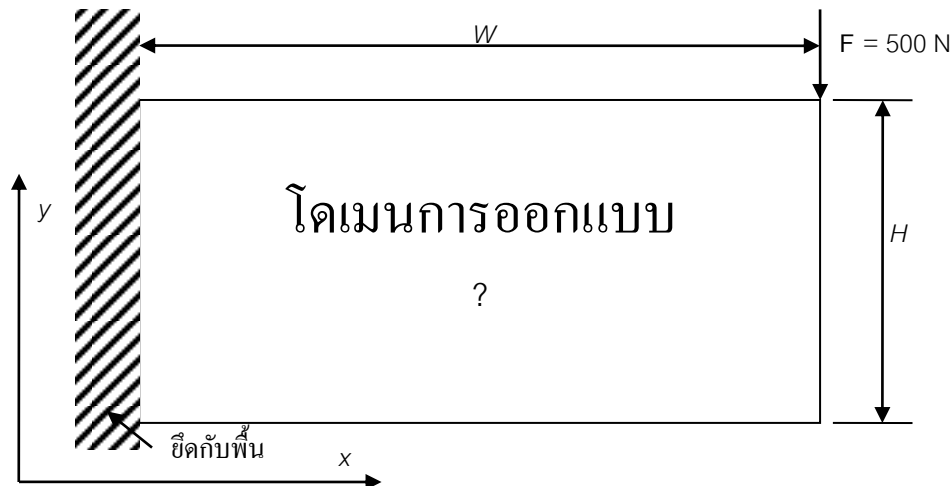
มีความแข็งแรงที่สุดเพื่อใช้งานตามเงื่อนไขการออกแบบและการใช้งานที่กำหนดให้ โดยทั่วไป ฟังก์ชันเป้าหมายที่นิยมประยุกต์ใช้คือ การหาความหย่อนของโครงสร้างต่ำสุด วิธีการหาทอพอโลยีที่เหมาะสมที่สุดที่นิยมใช้วิธีหนึ่งคือ การประยุกต์ใช้เอลิเมนต์พื้น คำว่า เอลิเมนต์พื้น หมายถึง โครงร่างของโครงข้อหมุนที่มีจำนวนจุดต่อ (nodes) และเอลิเมนต์มากที่สุดที่ผู้ออกแบบยอมรับได้หรือกำหนดไว้ เอลิเมนต์ทั้งหมดนี้เรียกว่า เอลิเมนต์พื้น ดังแสดงในโครงสร้างตรงกลางในรูปที่ 6.1 ตัวแปรออกแบบทอพอโลยีคือขนาดของเอลิเมนต์พื้นเหล่านั้น เมื่อสิ้นสุดกระบวนการหาค่าที่เหมาะสมที่สุด เอลิเมนต์ที่มีขนาดเล็กมากๆ จะถูกตัดออกจากโครงสร้าง ส่วนเอลิเมนต์ที่มีขนาดใหญ่ จะถูกกำหนดเป็นเอลิเมนต์หลักของโครงสร้าง ด้วยแนวคิดนี้ จะทำให้ได้โครงร่างหรือทอพอโลยีที่เหมาะสมที่สุด ดังโครงสร้างด้านขวามือในรูปที่ 6.1



รูปที่ 6.1 การหาทอพอโลยีของโครงข้อหมุนด้วยวิธีเอลิเมนต์พื้น

6.2.2 การหาทอพอโลยี รูปทรง และขนาดที่เหมาะสมที่สุดพร้อมกันด้วยเอลิเมนต์พื้นแบบปรับตัวได้

การหาทอพอโลยี รูปทรงและขนาดที่เหมาะสมที่สุดพร้อมกัน ในที่นี้ จะทำการประยุกต์ใช้เอลิเมนต์พื้นแบบปรับตัวได้ (adaptive ground elements) รูปที่ 6.2 แสดงโดเมนการออกแบบสำหรับสาธิตกระบวนการออกแบบนี้ จากรูปโดเมนมีขนาดเป็น $W \times H$ เมื่อ $W = 2$ m คือความกว้างที่มีค่าคงที่ ส่วนความสูงของโดเมนออกแบบสามารถปรับค่าได้ในช่วง $H \in [H_{min}, H_{max}] = [0.25, 1]$ m แฉทางด้านซ้ายมือของโครงสร้างถูกยึดกับพื้น และโครงสร้างรับแรงกระทำ $F = 500$ N ที่ขอบบนด้านขวาของโดเมนออกแบบ ดังแสดง

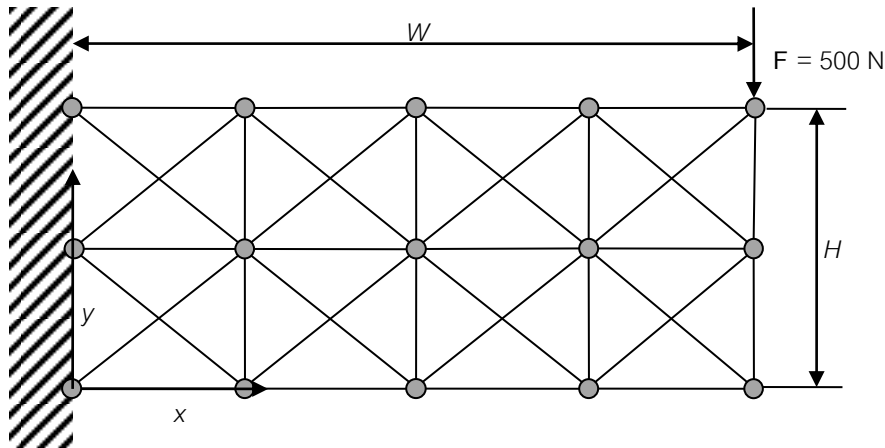


รูปที่ 6.2 โดเมนการออกแบบ

ในการสร้างเอลิเมนต์พื้น กำหนดให้ n_x คือจำนวนจุดต่อในแนวแกน x และ n_y คือจำนวนจุดต่อในแนวแกน y สำหรับเอลิเมนต์พื้นที่ปรับขนาดได้ ค่าของ n_x และ n_y สามารถเปลี่ยนแปลงได้ดังนี้ $n_x \in \{3, 4, 5\}$ และ $n_y \in \{2, 3, 4\}$ ในกระบวนการออกแบบ n_x และ n_y คือตัวแปรออกแบบที่กำหนดความละเอียดของเอลิเมนต์พื้น รูปที่ 6.3 แสดงตัวอย่างของเอลิเมนต์พื้นที่สร้างจาก $n_x = 5$ และ $n_y = 3$ จำนวนของเอลิเมนต์พื้น N_e สามารถคำนวณได้จาก

$$N_e = (n_x - 1)n_y + 3(n_x - 1)(n_y - 1) \quad (6.1)$$

เพื่อสามารถกำหนดทอพอโลยีได้ ตัวแปรออกแบบคือขนาดของเส้นผ่าศูนย์กลางของเอลิเมนต์พื้นทั้งหมดร่วมกับ H , n_x และ n_y กำหนดให้ $n_{x,\max}$ และ $n_{y,\max}$ คือจำนวนสูงสุดของ n_x และ n_y ตามลำดับ ดังนั้นจำนวนเอลิเมนต์พื้นสูงสุดคือ $N_{e,\max} = (n_{x,\max} - 1)n_{y,\max} + 3(n_{x,\max} - 1)(n_{y,\max} - 1)$ เพื่อความง่ายในการอธิบาย ในการศึกษาี้ กำหนดตัวแปรออกแบบสองกลุ่มคือ กลุ่มแรกเป็นการหาทอพอโลยีและขนาดที่เหมาะสมที่สุดพร้อมกัน และกลุ่มที่สองคือการหาขนาด รูปทรง และทอพอโลยีที่เหมาะสมที่สุดพร้อมกัน สำหรับชุดแรกตัวแปรออกแบบคือ H , n_x และ n_y และขนาดของเอลิเมนต์พื้นทั้งหมด ดังนั้น จำนวนตัวแปรออกแบบทั้งหมดสำหรับกรณีนี้คือ $N_{e,\max} + 3$ และมีขอบเขตเป็น $x_i \in [0, 1]$ ขั้นตอนวิธีที่ 6.1 แสดงการถอดรหัสของตัวแปรออกแบบจากเวกเตอร์จำนวนจริงขนาด $(N_{e,\max} + 3) \times 1$ โดยมีข้อมูลขาออกคือ ตำแหน่งของจุดต่อ การเชื่อมต่อของเอลิเมนต์พื้น และข้อมูลอื่นๆที่จำเป็นสำหรับการวิเคราะห์ทางไฟไนต์เอลิเมนต์ของโครงข้อหมุน



รูปที่ 6.3 เอลิเมนต์พื้นแบบปรับตัวได้

ขั้นตอนวิธีที่ 6.1 แสดงการถอดรหัสของตัวแปรออกแบบกลุ่มที่หนึ่ง

ข้อมูลเข้า $\mathbf{x}$ ขนาด $(N_{e,\max}+3) \times 1$ เมื่อ $x_i \in [0, 1]$

ข้อมูลออก ตำแหน่งจุดต่อ การเชื่อมต่อกันของเอลิเมนต์พื้น เงื่อนไขขอบเขต เวกเตอร์แรงกระทำ และเส้นผ่านศูนย์กลางของเอลิเมนต์

1: กำหนดให้

$$H = H_{\min} + x_1(H_{\max} - H_{\min})$$

$$n_x = \text{round}(n_{x,\min} + x_2(n_{x,\max} - n_{x,\min}))$$

$$n_y = \text{round}(n_{y,\min} + x_3(n_{y,\max} - n_{y,\min}))$$

2: หาค่าของ $N_e = (n_x-1)n_y + 3(n_x-1)(n_y-1)$

3: สร้างจุดต่อจำนวน $n_x \times n_y$ จุดและเอลิเมนต์พื้นจำนวน N_e ตามรูปที่ 8.16

4: กำหนดเส้นผ่านศูนย์กลางของเอลิเมนต์พื้น ให้ $y_j = 4x_i$ สำหรับ $i = 4, \dots, N_e + 3$ และ $j = 1, \dots, N_e$

$$y_j \in [0, 4]$$

5: สำหรับ $j = 1$ ถึง N_e

5.1: ถ้า $\text{round}(y_j) = 0$ กำหนดเส้นผ่านศูนย์กลางของเอลิเมนต์ j -th เป็น $d_j = 0.000001$ m

5.2: ถ้า $\text{round}(y_j) = 2$ กำหนดเส้นผ่านศูนย์กลางของเอลิเมนต์ j -th เป็น $d_j = 0.01$ m

5.3: ถ้า $\text{round}(y_j) = 3$ กำหนดเส้นผ่านศูนย์กลางของเอลิเมนต์ j -th เป็น $d_j = 0.02$ m

5.4: ถ้า $\text{round}(y_j) = 4$ กำหนดเส้นผ่านศูนย์กลางของเอลิเมนต์ j -th เป็น $d_j = 0.03$ m

5.5: ถ้า $\text{round}(y_j) = 5$ กำหนดเส้นผ่านศูนย์กลางของเอลิเมนต์ j -th เป็น $d_j = 0.04$ m

6: ใส่เงื่อนไขขอบเขตสำหรับจุดต่อที่ตำแหน่ง $x = 0$

7: ใส่แรงกระทำที่จุดต่อจุดสุดท้าย

ฟังก์ชัน $\text{round}(x)$ คือการปัดเศษไปยังจำนวนเต็มที่อยู่ใกล้ที่สุดกับ x ถึงแม้ว่าจำนวนของตัวแปรออกแบบทั้งหมดคือ $(N_{e,\max}+3) \times 1$ แต่จำนวนตัวแปรที่ใช้งานจริงมีแค่ $N_e + 3$ ตัวเท่านั้น (ขึ้นอยู่กับ

ค่าของ n_x และ n_y ในขั้นตอนที่ 1) กลุ่มตัวแปรออกแบบนี้เรียกว่า ตัวแปรผสมแบบต่อเนื่อง/จำนวนเต็ม (mixed integer/continuous) กล่าวคือมีค่าของตัวแปรที่เป็นจำนวนจริง ($x_1 = H$) และตัวแปรออกแบบที่เหลือเป็นเลขจำนวนเต็มบวก ในส่วนของตัวแปรกำหนดขนาดของเส้นผ่าศูนย์กลางของเอลิเมนต์พื้น ค่าของเส้นผ่าศูนย์กลางมีดังนี้ $d_j \in \{0.000001, 0.01, 0.02, 0.03, 0.04\}$ m เมื่อทำการถอดรหัสตัวแปรออกแบบในขั้นตอนที่ 5 หากค่าของเส้นผ่าศูนย์กลางที่ได้คือ 0.000001 แสดงว่าต้องทำการลบเอลิเมนต์ดังกล่าวออกจากโครงสร้างเพื่อกำหนดทอพอโลยีของโครงข้อหมุนได้ สาเหตุที่ใช้ค่า $d_j = 0.000001$ เพื่อป้องกันการเกิดซิงกูลาริตีในเมตริกซ์ความแข็งตึง ซึ่งจะส่งผลให้กระบวนการคำนวณเกิดความคลาดเคลื่อนและไม่สามารถดำเนินการต่อเนื่องได้ ส่วนค่าของเส้นผ่าศูนย์กลางของเอลิเมนต์พื้นค่าอื่นคือตัวแปรกำหนดขนาดของชิ้นส่วนเอกรูป จะเห็นได้ว่าการเข้าและถอดรหัสในรูปแบบนี้ง่ายและมีประสิทธิภาพ สามารถประยุกต์ใช้ร่วมกับขั้นตอนวิธีวิวัฒนาการโดยไม่มีอุปสรรคใดๆ แต่จะมีความยากหรือแม้กระทั่งเป็นไปได้เลยในการประยุกต์ใช้ร่วมกับวิธีหาค่าเหมาะที่สุดที่ใช้อนุพันธ์

ตัวแปรออกแบบกลุ่มที่สองขยายเพิ่มเติมจากตัวแปรออกแบบกลุ่มที่หนึ่ง กล่าวคือ มีการเพิ่มส่วนของตัวแปรกำหนดรูปทรงของโครงข้อหมุนเข้าไปด้วย โดยทั่วไปแล้ว ตัวแปรกำหนดรูปทรงของโครงข้อหมุนคือตัวแปรกำหนดตำแหน่งของจุดต่อของโครงสร้างนั่นเอง จากรูปที่ 6.3 กำหนดให้ตำแหน่งในแนวแกน x และ y ของจุดต่อคือตัวแปรออกแบบ ยกเว้นจุดต่อในแถบด้านซ้ายมือ $x = 0$ ซึ่งใช้เป็นเงื่อนไขขอบเขตที่ติดกับพื้น และจุดต่อบริเวณด้านบนสุดของโครงสร้าง $y = H$ มีการเปลี่ยนแปลงตำแหน่งในแนวแกน x เท่านั้นเพื่อให้จุดต่อด้านบนอยู่ในระดับความสูงเดียวกันเพื่อประโยชน์ในการใช้งาน ดังนั้น จำนวนจุดต่อที่สามารถเปลี่ยนแปลงตำแหน่งได้คือ $(n_x - 1)n_y$ และจำนวนตัวแปรออกแบบกำหนดรูปทรงของโครงสร้างมีทั้งหมดคือ $N_{sh} = 2(n_x - 1)(n_y - 1) + (n_x - 1) = (n_x - 1)(2n_y - 1)$ เมื่อรวมกับตัวแปรกำหนดโดเมน ทอพอโลยี และขนาด จะได้จำนวนตัวแปรออกแบบทั้งหมดสำหรับการหาทอพอโลยี รูปทรงและขนาดเหมาะที่สุดของโครงข้อหมุนโดยใช้เอลิเมนต์พื้นแบบปรับตัวได้ มีจำนวนตัวแปรทั้งสิ้น $N_{e,max} + N_{sh,max} + 3$ ตัว เมื่อ $N_{sh,max} = (n_{x,max} - 1)(2n_{y,max} - 1)$ กระบวนการถอดรหัสตัวแปรออกแบบ $x_i \in [0, 1]; i = 1, \dots, N_{e,max} + N_{sh,max} + 3$ แสดงในขั้นตอนวิธีที่ 6.2

ขั้นตอนวิธีที่ 6.2 แสดงการถอดรหัสของตัวแปรออกแบบกลุ่มที่สอง

ข้อมูลขาเข้า x ขนาด $(N_{e,max} + N_{sh,max} + 3) \times 1$ เมื่อ $x_i \in [0, 1]$	
ข้อมูลขาออก ตำแหน่งจุดต่อ การเชื่อมต่อกันของเอลิเมนต์พื้น เงื่อนไขขอบเขต เวกเตอร์แรงกระทำ และเส้นผ่าศูนย์กลางของเอลิเมนต์	
1: กำหนดให้	$H = H_{min} + x_1(H_{max} - H_{min})$ $n_x = \text{round}(n_{x,min} + x_2(n_{x,max} - n_{x,min}))$ $n_y = \text{round}(n_{y,min} + x_3(n_{y,max} - n_{y,min}))$
2: หาจำนวนเอลิเมนต์พื้น $N_e = (n_x - 1)n_y + 3(n_x - 1)(n_y - 1)$	

- 3: หาจำนวนตัวแปรกำหนดตำแหน่งจุดต่อในแนวแกน x ทั้งหมด $N_{sh,x} = n_y(n_x - 1)$ และในแนวแกน y ทั้งหมด $N_{sh,y} = (n_x - 1)(n_y - 1)$
- 4: คำนวณค่าขอบเขตของการเคลื่อนที่ (move limits) ของจุดต่อในแนวแกน x โดยมีขอบเขตล่างเป็น
- $$\Delta x_{\min} = -(W/2/n_{x,\max} - 0.01) \text{ m}$$
- และขอบเขตบนคือ
- $$\Delta x_{\max} = W/2/n_{x,\max} - 0.01 \text{ m}$$
- 5: คำนวณค่าขอบเขตของการเคลื่อนที่ (move limits) ของจุดต่อในแนวแกน y โดยมีขอบเขตล่างเป็น
- $$\Delta y_{\min} = -(H_{\min}/2/n_{y,\max} - 0.01) \text{ m}$$
- และขอบเขตบนคือ
- $$\Delta y_{\max} = H_{\min}/2/n_{y,\max} - 0.01 \text{ m}$$
- 6: สร้างจุดต่อจำนวน $n_x \times n_y$ จุด โดยมีพิกัดของจุดต่อเริ่มต้นในแนวแกน x และ y เป็น $\mathbf{X}$ และ $\mathbf{Y}$ ตามลำดับ สร้างเอลิเมนต์พื้นจำนวน N_e เอลิเมนต์
- 7: กำหนดเส้นผ่านศูนย์กลางกลางของเอลิเมนต์พื้น ให้ $y_j = 4x_i$ สำหรับ $i = 4, \dots, N_e + 3$ และ $j = 1, \dots, N_e$
 $y_j \in [0, 4]$
- 8: สำหรับ $j = 1$ ถึง N_e
- 8.1: ถ้า $\text{round}(y_j) = 0$ กำหนดเส้นผ่านศูนย์กลางกลางของเอลิเมนต์ j -th เป็น $d_j = 0.000001 \text{ m}$
- 8.2: ถ้า $\text{round}(y_j) = 2$ กำหนดเส้นผ่านศูนย์กลางกลางของเอลิเมนต์ j -th เป็น $d_j = 0.01 \text{ m}$
- 8.3: ถ้า $\text{round}(y_j) = 3$ กำหนดเส้นผ่านศูนย์กลางกลางของเอลิเมนต์ j -th เป็น $d_j = 0.02 \text{ m}$
- 8.4: ถ้า $\text{round}(y_j) = 4$ กำหนดเส้นผ่านศูนย์กลางกลางของเอลิเมนต์ j -th เป็น $d_j = 0.03 \text{ m}$
- 8.5: ถ้า $\text{round}(y_j) = 5$ กำหนดเส้นผ่านศูนย์กลางกลางของเอลิเมนต์ j -th เป็น $d_j = 0.04 \text{ m}$
- 9: คำนวณค่า
- $$\Delta x_j = \Delta x_{\min} + (\Delta x_{\max} - \Delta x_{\min})x_i \text{ สำหรับ } i = (N_e + 4), \dots, (N_e + N_{sh,x} + 3) \text{ และ } j = 1, \dots, N_{sh,x}$$
- ปรับปรุงตำแหน่งในแนวแกน x ของจุดต่อ $X_j = X_j + \Delta x_j$ สำหรับจุดต่อที่อนุญาตให้มีการเปลี่ยนแปลงตำแหน่งในแนวแกน x ได้
- 10: คำนวณค่า
- $$\Delta y_j = \Delta y_{\min} + (\Delta y_{\max} - \Delta y_{\min})x_i \text{ สำหรับ } i = (N_e + N_{sh,x} + 4), \dots, (N_e + N_{sh,x} + N_{sh,y} + 3) \text{ และ } j = 1, \dots, N_{sh,y}$$
- ปรับปรุงตำแหน่งในแนวแกน y ของจุดต่อ $Y_j = Y_j + \Delta y_j$ สำหรับจุดต่อที่อนุญาตให้มีการเปลี่ยนแปลงตำแหน่งในแนวแกน y ได้
- 11: ใส่เงื่อนไขขอบเขตสำหรับจุดต่อที่ตำแหน่ง $x = 0$

12: ใส่แรงกระทำที่จุดต่อจุดสุดท้าย

เวกเตอร์ตัวแปรออกแบบมีขนาดเป็น $(N_{e,max} + N_{sh,max} + 3) \times 1$ แต่มีจำนวนตัวแปร $N_e + N_{sh} + 3$ เท่านั้น ที่ใช้ในการกำหนดข้อมูลของโครงสร้าง (ขึ้นอยู่กับค่าของ n_x และ n_y ในขั้นตอนที่ 1) เพื่อความสะดวกในการอธิบาย กำหนดให้ปัญหาการออกแบบที่ใช้ตัวแปรกลุ่มที่หนึ่งเรียกว่า OPT1 และปัญหาการออกแบบที่ใช้ตัวแปรกลุ่มที่สองเรียกว่า OPT2 โครงสร้างทำจากวัสดุที่มีค่า

- โมดูลัสของยัง $E = 214 \times 10^9$ Pa
- ค่าความเค้นคราก $\sigma_{yt} = 360 \times 10^6$ Pa
- ค่าความหนาแน่น $\rho = 7850$ kg/m³

ค่าตัวประกอบความปลอดภัย คือ $FoS = 1.5$ ปัญหาการหาค่าเหมาะที่สุดแบบหลายเป้าหมายของโครงข้อมุมในหัวข้อนี้คือปัญหา (8.57) ที่ไม่มีเงื่อนไขบังคับการกระจัดและเพิ่มเงื่อนไขบังคับ $c_p - c_{p,al} \leq 0$ เมื่อ $c_{p,al} = 0.04$ N-m คือค่าความหยุ่นต่ำสุดที่ยอมรับได้ ขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมายที่ทำการประยุกต์ใช้มี 7 ขั้นตอนวิธี ดังนี้ (ดูรายละเอียดของที่มาและการตั้งค่าเพิ่มเติมได้ในบทที่ 7)

- SPEA1: SPEA เวอร์ชันสอง ใช้ตัวแปรออกแบบจำนวนจริงที่มีอัตราการเกิดครอสโอเวอร์และมิวเทชันเป็น 1.0 และ 0.05 ตามลำดับ
- SPEA2: SPEA เวอร์ชันสอง ใช้ตัวแปรออกแบบจำนวนจริงที่มีอัตราการเกิดครอสโอเวอร์และมิวเทชันเป็น 1.0 และ 0.5 ตามลำดับ
- SPEA3: SPEA เวอร์ชันสอง ใช้ตัวแปรออกแบบจำนวนจริงที่มีอัตราการเกิดครอสโอเวอร์และมิวเทชันเป็น 1.0 และ 0.75 ตามลำดับ
- PBIL: ค่าอัตราการเรียนรู้ (learning rate) ค่าพารามิเตอร์ของการมิวเทชัน (mutation shift) และอัตราการเกิดมิวเทชัน (mutation rate) คือ 0.5, 0.2 และ 0.05 ตามลำดับ
- AMOSA: อุณหภูมิเริ่มต้นคือ 10 อุณหภูมิสุดท้ายคือ 0.001 ใช้วิธีชิมเพิลส์ลงเขาใน 3 วงวนแรก
- MPSO: ค่าถ่วงน้ำหนักสำหรับเวกเตอร์ความเร็ว $W \in [0.01, 0.5]$ ตัวประกอบการเรียนรู้ด้วยตนเอง $C_1 = 0.75$ และตัวประกอบการเรียนรู้ทางสังคม $C_2 = 0.75$ ใช้ขั้นตอนวิธีกริดปรับตัวได้ (adaptive grid algorithm) ในการคัดกรองผลเฉลยที่ไม่ถูกรอบงำ และเลือกผลเฉลย g^{best} แบบสุ่มจากหน่วยเก็บผลเฉลยที่ไม่ถูกรอบงำ
- UPSEMOA: ตัวประกอบมาตราส่วน อัตราการครอสโอเวอร์ และโอกาสในการถูกเลือกของเอลิเมนต์ของผลเฉลยถูกคือ 0.8, 0.7 และ 0.5 ตามลำดับ ใช้แผนการผลิตประชากรใหม่คือ DE/best/2/bin จำนวนผลเฉลยในประชากรพ่อแม่คือ 25

ขนาดของประชากรและจำนวนวงวนของขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมายนี้คือ 150 และ 250 ตามลำดับ สำหรับ OPT1 ส่วน OPT2 มีขนาดประชากรและจำนวนวงวนเป็น 200 และ 300

ตามลำดับ ขนาดของหน่วยเก็บผลเฉลยพารโตมีค่าเท่ากับขนาดของประชากร วิธีหาค่าเหมาะที่สุดทั้งหมดเริ่มต้นในการหาผลเฉลยเหมาะที่สุดด้วยประชากรตั้งต้นชุดเดียวกัน ทำการรันแต่ละวิธี ทั้งหมด 10 ครั้ง

6.2.3 ผลการออกแบบและการเปรียบเทียบสมรรถนะ

จากการรันวิธีหาค่าเหมาะที่สุดทั้ง 7 วิธี ในการหาขอบหน้าพารโตของปัญหาการออกแบบ OPT1 และ OPT2 10 ครั้งสำหรับทุกวิธี ในการประเมินสมรรถนะของขั้นตอนวิธีเหล่านี้ ใช้ตัวบ่งชี้ปริมาตรหลายมิติ (hypervolume, HV) เพื่อสามารถเปรียบเทียบวิธีการต่างๆที่ใช้ในการศึกษานี้ได้ ผลการเปรียบเทียบค่า HV แสดงในตารางที่ 6.1 จากข้อมูล เป็นการเปรียบเทียบค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานของ HV เท่านั้น โดยใช้ค่าเฉลี่ยในการวัดอัตราการลู่เข้าและค่าส่วนเบี่ยงเบนมาตรฐานในการวัดความต้องกัน (consistency) ในการหาคำตอบ

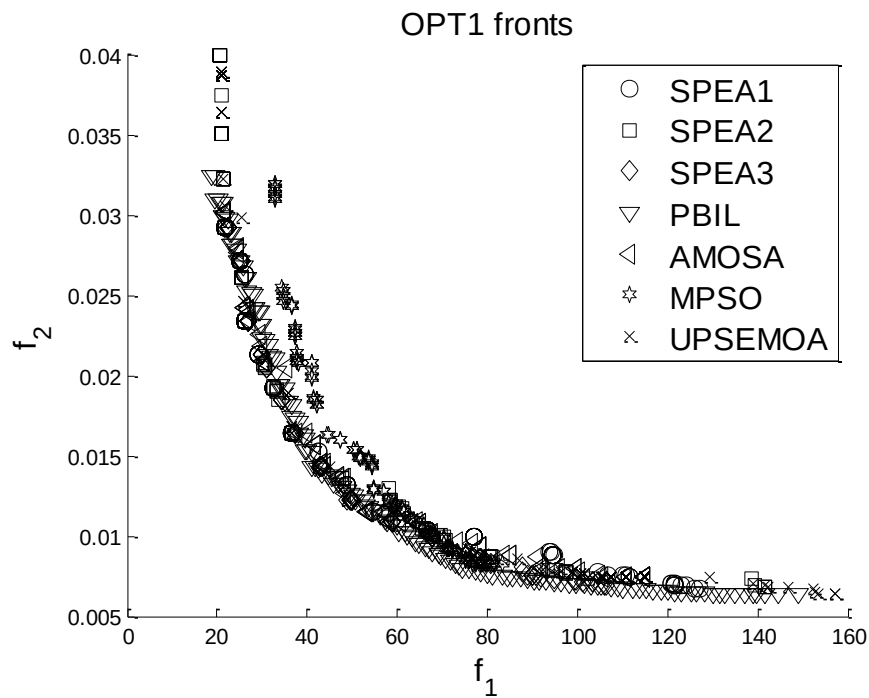
จากผลการเปรียบเทียบพบว่า PBIL คือวิธีที่ให้คำตอบได้ดีที่สุดทั้งปัญหาการออกแบบ OPT1 และ OPT2 วิธีที่ดีที่สุดอันดับสองสำหรับ OPT1 และ OPT2 คือ SPEA3 และ SPEA2 ตามลำดับ ส่วนวิธีที่ให้คำตอบแย่ที่สุดในกรณีของ OPT1 และ OPT2 คือ MPSO และ UPSEMOA ตามลำดับ วิธีที่ให้ความต้องกันมากที่สุดคือ SPEA2 อย่างไรก็ตาม ในการเปรียบเทียบขั้นตอนวิธีวิวัฒนาการ หากวิธีที่ให้อัตราการลู่เข้าดีที่สุดกับวิธีที่ให้ค่าความต้องกันที่สุดไม่ใช่วิธีเดียวกัน ให้ถือว่าวิธีที่มีอัตราการลู่เข้าดีที่สุดคือวิธีที่ดีที่สุด ทั้งนี้เป็นเพราะว่า เราต้องการหาขอบหน้าพารโตที่ดีที่สุดมากกว่านั่นเอง ดังนั้น เราสามารถสรุปได้ว่าวิธี PBIL คือวิธีที่ดีที่สุดสำหรับปัญหาการออกแบบลักษณะนี้ จากการเปรียบเทียบ SPEA ทั้งสามรูปแบบ สามารถสรุปได้ว่า SPEA ที่มีอัตราการเกิดมิวเทชันสูงกว่า ให้คำตอบที่ดีกว่าหรืออาจกล่าวได้ว่าตัวดำเนินการมิวเทชันช่วยเพิ่มสมรรถนะในการหาผลเฉลยของปัญหาการออกแบบนี้มากกว่าตัวดำเนินการ คrossover

ตารางที่ 6.1 การเปรียบเทียบค่า HV ของ OPT1 และ OPT2

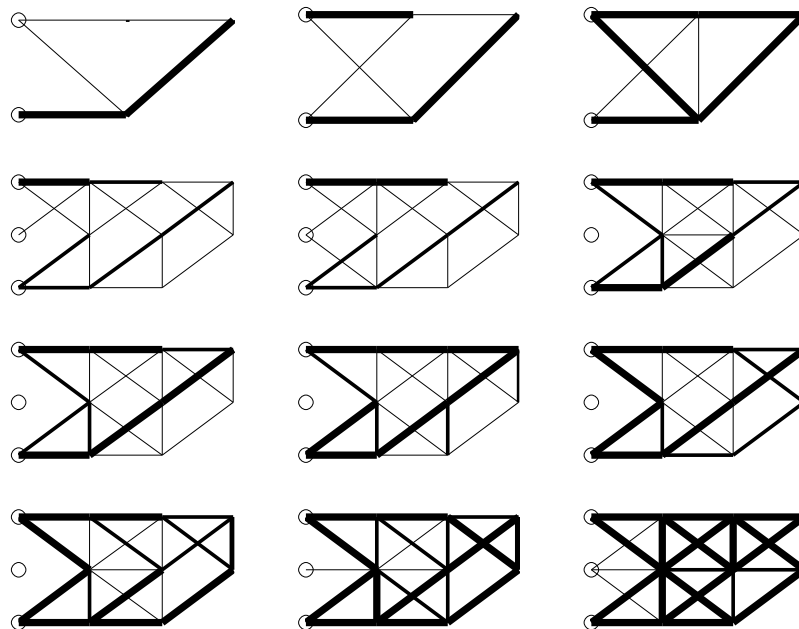
MOEA	OPT1		OPT2	
	AVG	STD	AVG	STD
SPEA1	0.7098	0.0998	0.8619	0.0267
SPEA2	0.8161	0.0343	0.9274	0.0184
SPEA3	0.8400	0.0564	0.9236	0.0219
PBIL	0.9330	0.0647	0.9519	0.0527
AMOS	0.6619	0.1143	0.7997	0.0277
MPSO	0.3043	0.1371	0.6699	0.0582
UPSEMOA	0.4334	0.2808	0.6136	0.3885

ขอบหน้าพารโตที่ดีที่สุดที่ได้จากการประยุกต์ใช้ MOEAs หลากหลายวิธีสำหรับ OPT1 แสดงในรูปที่ 6.4 จากรูปจะเห็นได้ว่า ขอบหน้าพารโตของวิธีทั้ง 7 วิธี วางอยู่ในตำแหน่งใกล้เคียงกันโดยมีขอบหน้าพารโตจากวิธี PBIL ล้ำหน้าไปได้ไกลที่สุด โครงสร้างผลเฉลยบางส่วนของขอบหน้าพารโต

ที่ได้จากวิธี PBIL แสดงในรูปที่ 6.5 ซึ่งจะได้โครงสร้างที่มีทอพอโลยีและมีขนาดหลากหลายรูปแบบจากรูป จำนวนจุดต่อของเอลิเมนต์พื้นมีอยู่ 2 ชุด คือ $n_x \times n_y = 2 \times 1$ และ $n_x \times n_y = 4 \times 2$

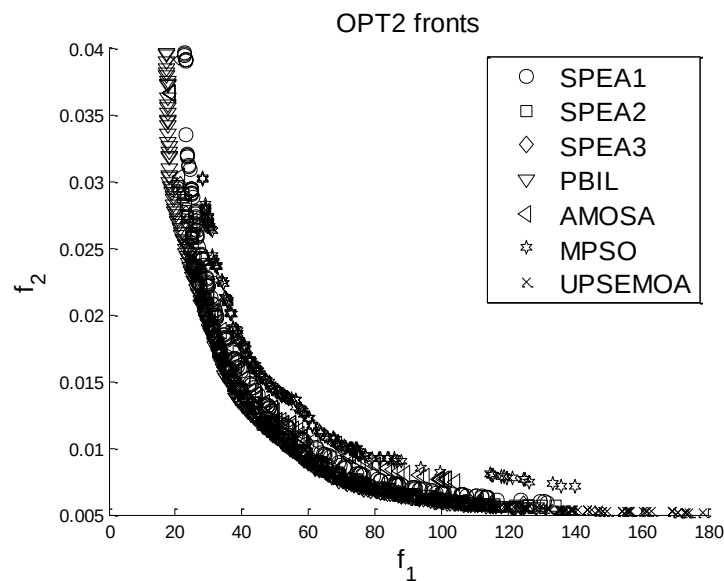


รูปที่ 6.4 ขอบหน้าพารโตของ OPT1

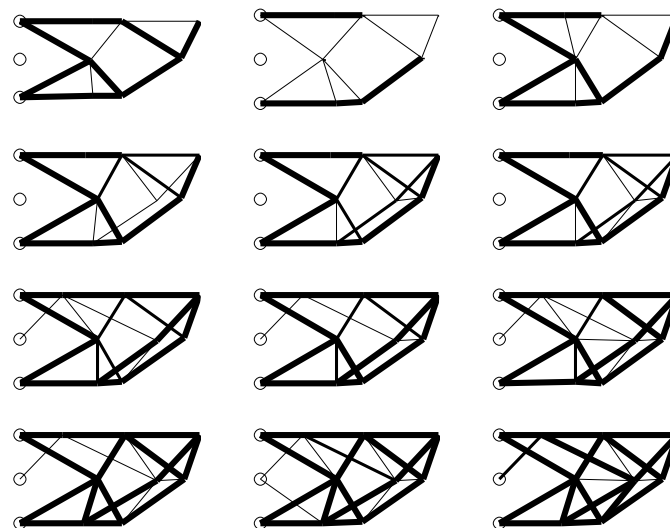


รูปที่ 6.5 โครงสร้างบางส่วนของขอบหน้าพารโต OPT1

ขอบหน้าพารेटอที่ดีที่สุดที่ได้จากการประยุกต์ใช้ MOEAs หลากหลายวิธีสำหรับปัญหาการออกแบบ OPT2 แสดงในรูปที่ 6.6 จากรูปจะเห็นได้ว่า ขอบหน้าพารेटอของวิธีทั้ง 7 วิธี วางอยู่ในตำแหน่งใกล้เคียงกันโดยมีขอบหน้าพารेटอจากวิธี PBIL ล้ำหน้าไปได้ไกลที่สุด โครงสร้างผลเฉลยบางส่วนของขอบหน้าพารेटอที่ได้จากวิธี PBIL แสดงในรูปที่ 6.7 ซึ่งจะได้โครงสร้างที่มีทอพอโลยีรูปทรง และขนาดหลากหลายรูปแบบ จากรูป จำนวนจุดต่อของเอลิเมนต์พื้นมีอยู่ 1 ชุดคือ $n_x \times n_y = 4 \times 2$ จากผลการออกแบบทั้งสองปัญหาการออกแบบพบว่า การประยุกต์ใช้เอลิเมนต์พื้นหลากหลายรูปแบบมีข้อดีคือ ช่วยให้ขั้นตอนวิธีวิวัฒนาการค้นหาเขตของเอลิเมนต์พื้นที่ที่ดีที่สุด ดังแสดงในรูปที่ 6.5 และ 6.7

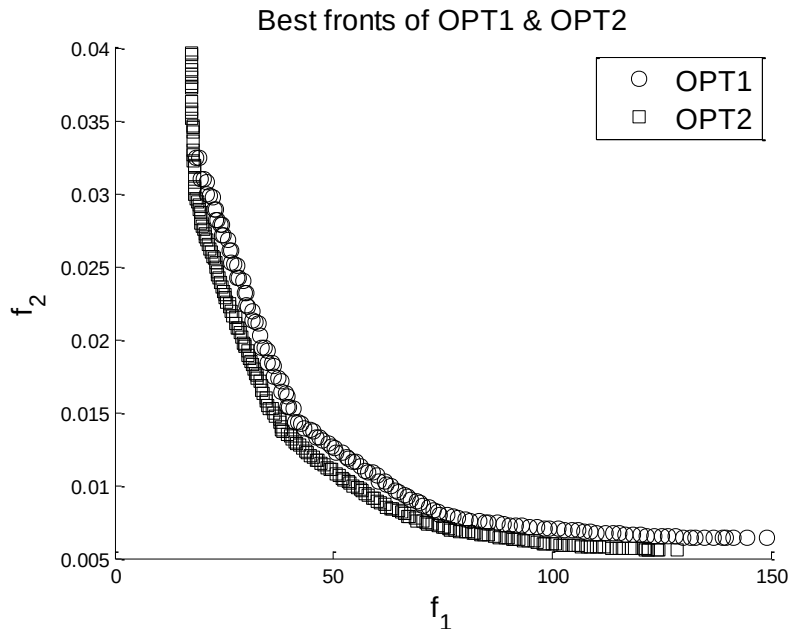


รูปที่ 6.6 ขอบหน้าพารेटอของ OPT2



รูปที่ 6.7 โครงสร้างบางส่วนของขอบหน้าพารेटอ OPT2

รูปที่ 6.8 แสดงการเปรียบเทียบขอบหน้าพารेटอที่ดีที่สุดที่ได้จากการรันวิธีหาค่าเหมาะที่สุดทั้ง 7 วิธี ของปัญหาการออกแบบ OPT1 และ OPT2 จากรูป พบว่าขอบหน้าพารेटอของ OPT2 ดีกว่า OPT1 ซึ่งแสดงให้เห็นว่า การหาทอพอโลยี รูปทรงและขนาดเหมาะที่สุดของโครงข้อหมุนพร้อมกัน ให้ผลเฉลยที่ดีกว่าการหาทอพอโลยีและขนาดพร้อมกัน ทั้งนี้เป็นเพราะว่า OPT2 มีปฏิภูมิในการหาผลเฉลยกว้างกว่า OPT1



รูปที่ 6.8 เปรียบเทียบขอบหน้าพารेटอ OPT1 และ OPT 2

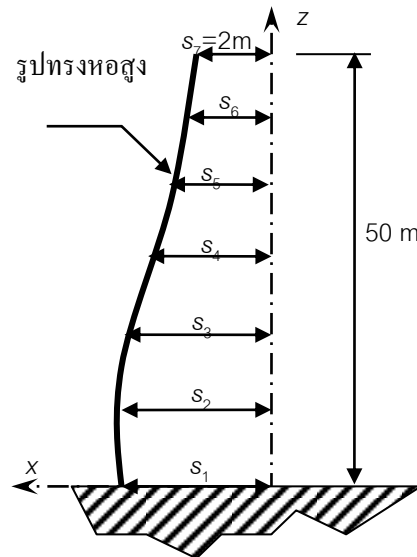
6.3 การหาขนาด รูปทรงและทอพอโลยีเหมาะที่สุดพร้อมกันของโครงข้อหมุนสามมิติ

หัวข้อนี้เป็นการขยายผลจากการออกแบบ โครงข้อหมุนสองมิติในหัวข้อที่ผ่านมา เพื่อการออกแบบโครงข้อหมุนในปริภูมิสามมิติ เนื่องจากได้ทำการแสดงให้เห็นแล้วว่า การหาทอพอโลยี รูปทรงและขนาดเหมาะที่สุดของโครงข้อหมุนพร้อมกัน เป็นกระบวนการออกแบบที่มีประสิทธิภาพสูงที่สุด ดังนั้นในหัวข้อนี้ จึงเป็นการออกแบบทอพอโลยี รูปทรงและขนาดเหมาะที่สุดของโครงข้อหมุนเท่านั้น ในที่นี้เป็นการออกแบบโครงสร้างของหอสูง ซึ่งนิยมใช้โครงข้อหมุนสามมิติในการสร้าง เช่น เสาไฟฟ้าแรงสูง เสาส่งสัญญาณโทรคมนาคม และ หอถังหั่นลม รายละเอียดของกระบวนการออกแบบมีดังต่อไปนี้

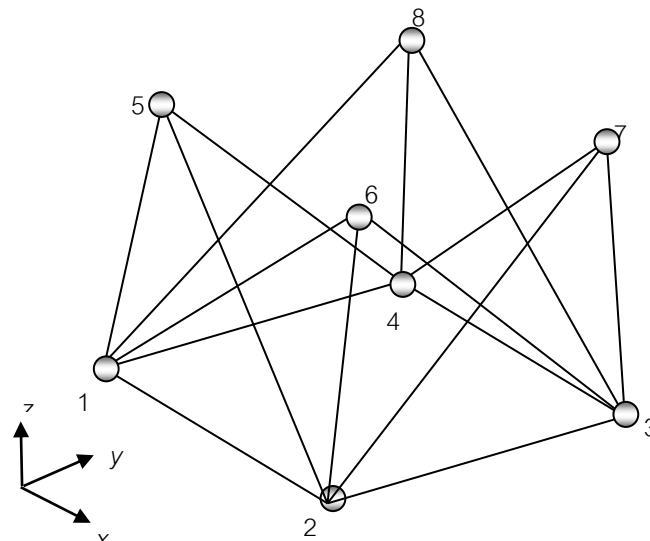
6.3.1 การเข้ารหัสและถอดรหัสตัวแปรออกแบบ

การเข้าและถอดรหัสของตัวแปรออกแบบทอพอโลยี รูปทรงและขนาดของโครงสร้างสามมิติพร้อมกัน จะมีความสลับซับซ้อนกว่ากรณีของการออกแบบโครงข้อหมุนสองมิติ ในกรณีของการสาธิตการออกแบบหอสูง กำหนดให้โครงสร้างมีความสูงเป็น $H = 50$ m และมีหน้าตัดเป็นรูปสี่เหลี่ยมจัตุรัส กำหนดให้หอสูงมีจำนวนชั้นอยู่ระหว่าง 10-20 ชั้น และสามารถปรับเปลี่ยนจำนวนชั้นได้ใน

กระบวนการออกแบบ ส่วนตัวแปรกำหนดรูปทรงแสดงในรูปที่ 6.9 จากรูปพารามิเตอร์ที่กำหนดรูปทรงของหอสถูคือ $s_1 - s_7$ โดยในการออกแบบ กำหนดให้ $s_2 - s_7$ เป็นตัวแปรออกแบบ การกำหนดรูปทรงสามารถทำได้โดยใช้การประมาณค่าในช่วง (interpolation) นอกจากนี้ ความสูงของแต่ละชั้นของหอนี้สามารถเปลี่ยนแปลงได้โดยให้พารามิเตอร์ $t_1 - t_4$ สำหรับกำหนดความสูงของชั้น



รูปที่ 6.9 พารามิเตอร์กำหนดรูปทรงของหอสถู



รูปที่ 6.10 เอลิเมนต์พื้นของชั้นของหอคอย

การหาทอพอโลยีของโครงสร้างในที่นี้เป็นการประยุกต์ใช้วิธีเอลิเมนต์พื้นแบบปรับตัวได้ รูปที่ 8.10 แสดงเอลิเมนต์พื้นที่ใช้สำหรับชั้นใดๆ ของหอสถู ในส่วนของตัวแปรกำหนดขนาดของเอลิเมนต์พื้นที่ชั้นใดๆ ใช้ตัวแปรวิชุด $y_1 - y_6 \in \{1, 2, 3, 4, 5\}^6$ ซึ่งในกระบวนการถอดรหัสให้เป็นขนาดของเอลิเมนต์พื้น เพื่อให้ได้โครงสร้างที่สมมาตร การกำหนดค่าตัวแปรให้เป็นขนาดพื้นที่หน้าตัดของเอลิ

เมนต์พื้นในแต่ละชั้นของหอสถู่ง แสดงในตารางที่ 6.2 (ดูรูปที่ 6.10 ประกอบ) ส่วนค่าพื้นที่หน้าตัดที่ใช้แสดงในตารางที่ 6.3 จากค่าพื้นที่หน้าตัดของเอลิเมนต์ในตาราง ถ้าเอลิเมนต์ใดมีพื้นที่หน้าตัดเป็น 1×10^{-8} m แสดงว่าเอลิเมนต์นั้นจะถูกกลบออกจากโครงสร้างซึ่งเป็นหลักการเดียวกันกับกระบวนการออกแบบหัวข้อ 8.5 ในการถอดรหัสตัวแปรออกแบบเพื่อเป็นข้อมูลของโครงข้อหมุน สามารถแบ่งตัวแปรออกเป็น 4 กลุ่มดังนี้ กลุ่มที่หนึ่ง ตัวแปรกำหนดจำนวนชั้นของหอสถู่ง (n_{level}) กลุ่มที่สอง ตัวแปรกำหนดรูปทรงของหอสถู่ง ($s_2 - s_7$) กลุ่มที่สาม ตัวแปรกำหนดความสูงของชั้นของหอสถู่ง ($t_1 - t_4$) และกลุ่มสุดท้าย ตัวแปรกำหนดพื้นที่หน้าตัดของเอลิเมนต์พื้นของหอสถู่งทุกชั้น โดยในแต่ละชั้นใช้ตัวแปรออกแบบ 6 ค่า ($y_1 - y_6$) ในการกำหนดพื้นที่หน้าตัดของเอลิเมนต์พื้น 16 เอลิเมนต์

ตารางที่ 6.2 การกำหนดขนาดของเอลิเมนต์พื้นในแต่ละชั้น

การเชื่อมจุดต่อของเอลิเมนต์	ตัวแปรกำหนดพื้นที่หน้าตัด
1-2 3-4 2-3 4-1	y_1
1-5 2-6 3-7 4-8	y_2
1-6 3-8	y_3
2-5 4-7	y_4
1-8 3-6	y_5
4-5 2-7	y_6

ตารางที่ 6.3 ข้อมูลพื้นที่หน้าตัดของชิ้นส่วนเอกรูปของโครงข้อหมุน

จำนวนเต็ม	พื้นที่หน้าตัดรวม ยอด (gross area) (m^2)	พื้นที่หน้าตัดสุทธิ (net area) (m^2)	รัศมีไจเรชัน (m)	ค่าโมดูลัสของยัง (N/m^2)
1	1×10^{-8}	1×10^{-8}	1×10^{-8}	214×10^5
2	0.000581	0.000426	0.0118	214×10^9
3	0.00108	0.000837	0.0157	214×10^9
4	0.00232	0.001894	0.0236	214×10^9
5	0.01108	0.009448	0.0695	214×10^9

กระบวนการถอดรหัสตัวแปรออกแบบแสดงในขั้นตอนวิธีที่ 6.3 ในขั้นตอนวิธีดังกล่าวมีรายละเอียดของฟังก์ชันและพารามิเตอร์ต่างๆ ดังนี้

- $\mathbf{P} = \text{SplineInterp}(\mathbf{X}, \mathbf{Y})$ คือการประมาณค่าในช่วงโดยใช้ฟังก์ชันกำลังสาม (cubic spline interpolation) ค่าของ $\mathbf{P}$ คือค่าสัมประสิทธิ์ของการประมาณค่าในช่วง ใช้สำหรับทำนายค่าฟังก์ชัน ส่วน $\mathbf{X}$ และ $\mathbf{Y}$ คือเซตของตัวแปรต้นและตัวแปรตามตามลำดับ

- $\mathbf{Y} = \text{SplineEval}(\mathbf{P}, \mathbf{X})$ เป็นการทำนายค่าฟังก์ชัน $\mathbf{Y}$ เมื่อค่าตัวแปรต้นคือ $\mathbf{X}$ และสัมประสิทธิ์ของการประมาณค่าในช่วงคือ $\mathbf{P}$

$n_{\text{level}} =$ จำนวนชั้นของหอสถู

$h_i =$ ความสูงของชั้นที่ i -th

$W_i =$ ครึ่งของความกว้างของชั้นที่ i -th

เนื่องจาก กำหนดให้จำนวนชั้นสูงสุดของหอสถูคือ 20 ชั้น ดังนั้น จำนวนตัวแปรออกแบบสูงสุดคือ $n = 6(20) + 12 = 132$ ในกรณีที่จำนวนชั้นของหอสถูน้อยกว่า 20 ($n_{\text{level}} < 20$) จำนวนตัวแปรที่ใช้ในการกำหนดทอพอโลยี รูปทรงและขนาดของโครงสร้างคือ $n = 6n_{\text{level}} + 12$ ในการถอดรหัสตัวแปรออกแบบ ขั้นที่ 1 เป็นการหาจำนวนชั้นของหอสถู จากนั้น ขั้นตอนที่ 2 – 5 คือการหาการกระจายความสูงของชั้นของโครงสร้าง ขั้นตอนที่ 5 เป็นการดำเนินการเพื่อให้ได้เงื่อนไข $h_{i+1} > h_i$ ขั้นตอนที่ 6 – 8 เป็นการคำนวณความกว้างของชั้นต่างๆของหอสถู ขั้นตอนที่ 9 เป็นการสร้างจุดต่อจำนวน $4(n_{\text{level}} + 1)$ จุดจาก h_i และ w_i ในขั้นตอนที่ 10 – 11 เป็นการสร้างเอลิเมนต์พื้น (16 เอลิเมนต์) สำหรับแต่ละชั้นของหอสถู ดังรายละเอียดในตารางที่ 6.2 – 6.3 ส่วนขั้นตอนที่ 12 – 14 เป็นการกำหนดเอลิเมนต์และคุณสมบัติของเอลิเมนต์ด้านบนสุด (4 เอลิเมนต์) ของโครงสร้าง ซึ่งต้องใช้ตัวแปรออกแบบกำหนดขนาดในการระบุคุณสมบัติของเอลิเมนต์

ขั้นตอนวิธีที่ 6.3 การถอดรหัสตัวแปรออกแบบ

ข้อมูลขาเข้า $\mathbf{x}_{132 \times 1}$

ข้อมูลขาออก ตำแหน่งจุดต่อ การเชื่อมต่อกันของเอลิเมนต์พื้น เงื่อนไขขอบเขต เวกเตอร์แรงกระทำ และคุณสมบัติทางกายภาพของเอลิเมนต์ของโครงสร้างหอสถู

1: กำหนดให้ $n_{\text{level}} = \text{round}(x_1)$, $\{s_1, \dots, s_6\} = \{x_2, \dots, x_7\}$, $s_7 = 2$, $\{t_1, \dots, t_4\} = \{x_8, \dots, x_{11}\}$

2: กำหนดค่า $\{Z_1, Z_2, Z_3, Z_4\} = \{0, H/3, 2H/3, H\}$

3: หาค่าสัมประสิทธิ์ของการประมาณค่าในช่วง $\mathbf{P}_1 = \text{SplineInterp}(\{Z_1, \dots, Z_4\}, \{t_1, \dots, t_4\})$

4: หาค่าพารามิเตอร์ สำหรับ กำหนด ความ สูง ของ ชั้น หอสถู $\{a_1, \dots, a_{n_{\text{level}}}\} = \text{SplineEval}(\mathbf{P}_1, \{H/n_{\text{level}}, 2H/n_{\text{level}}, \dots, H\})$

5: คำนวณค่าความสูงของชั้นหอสถู $h_i = H \sum_{j=1}^i a_j / \sum_{j=1}^{n_{\text{section}}} a_j$

6: กำหนดค่า $\{Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7\} = \{0, H/6, 2H/6, 3H/6, 4H/6, 5H/6, H\}$

7: หาค่าสัมประสิทธิ์ของการประมาณค่าในช่วง $\mathbf{P}_2 = \text{SplineInterp}(\{Z_1, \dots, Z_7\}, \{s_1, \dots, s_7\})$

- 8: หาขนาดของครึ่งหนึ่งของความกว้างของชั้นหอสอง $\{w_0, \dots, w_{nlevel}\} = SplineEval(\mathbf{P}_2, \{0, H/n_{level}, 2H/n_{level}, \dots, H\})$
- 9: สร้างจุดต่อจำนวน $4(n_{level} + 1)$ จุดจาก $\{0, h_1, \dots, h_{nlevel}\}$ และ $\{w_0, \dots, w_{nlevel}\}$
- 10: สำหรับ $i = 1$ ถึง n_{level}
- 10.1 สร้างเอลิเมนต์พื้นสำหรับชั้นที่ i -th (รูปที่ 6.11)
- 10.2 กำหนดให้ $\{y_{6i-5}, \dots, y_{6i}\} = round(\{x_{11+6i-5}, \dots, x_{11+6i}\})$
- 10.3 กำหนดคุณสมบัติต่างๆของเอลิเมนต์พื้น (ตารางที่ 6.2 – 6.3)
- 11: จบการวนรอบในขั้นตอนที่ 10
- 12: กำหนดเอลิเมนต์ด้านบนสุดของหอสอง
- 13: ค่าตัวเลขจำนวนเต็มสำหรับกำหนดคุณสมบัติของเอลิเมนต์ทั้ง 4 ที่อยู่บนสุดคือ $y_{6nlevel+1} = round(x_{11+6nlevel+1})$ เมื่อ $y_{6nlevel+1} \in \{2, 3, 4, 5\}$
- 14: ใส่ค่าคุณสมบัติของเอลิเมนต์ตาม ตารางที่ 6.3

6.3.2 ปัญหาการออกแบบ

ปัญหาการออกแบบทอพอโลยี รูปทรงและขนาดของหอสองพร้อมกัน สามารถเขียนได้ดังนี้

$$\min \{f_1, f_2\} \quad (6.2)$$

โดยมีเงื่อนไขบังคับ

$$\begin{aligned} \sigma_{\max} &\leq \sigma_{al} \\ \lambda_i &\leq 1 \\ 10 &\leq round(x_1) \leq 20 \\ 4m &\leq x_i \leq 6m; i=2, \dots, 7 \\ 0 &\leq x_i \leq 1; i=8, \dots, 11 \\ round(x_i) &\in \{1, 2, 3, 4, 5\}; i=12, \dots, (11 + 6round(x_1)) \\ round(x_i) &\in \{2, 3, 4, 5\}; i=(12 + 6round(x_1)) \end{aligned}$$

เมื่อ

f_1, f_2 คือ ฟังก์ชันเป้าหมาย

$\sigma_{\max}$ คือ ค่าความเค้นสูงสุดที่เกิดขึ้นบนโครงสร้าง

σ_{al} คือ ค่าความเค้นสูงสุดที่ยอมให้เกิดขึ้น

λ_i คือ ค่าตัวประกอบการโก่งตัว

หอสองรับภาระ 3 รูปแบบ ในแต่ละกรณี จุดต่อ 4 จุดบนสุด จะรับภาระแบบจุด (point loads) โดยในกรณีที่ 1, 2 และ 3 แรงกระทำอยู่ในแนวแกน x, y และ z และมีขนาดของแรงเป็น 1×10^4 N, 1×10^4 N และ 2×10^4 N ตามลำดับ เวกเตอร์ของแรงทั้งสามกรณี จะรวมแรงเนื่องจากน้ำหนักของชิ้นส่วนของโครงสร้างเข้าไปด้วย โครงสร้างทำจากวัสดุที่มีค่าโมดูลัสของยังเป็น $E = 214 \times 10^9$ N/m² ค่าความหนาแน่น $\rho = 7850$ kg/m³ และค่าความเค้นคราก $\sigma_{yt} = 360 \times 10^6$ N/m² ค่าตัวประกอบความ

ปลอดภัยคือ $FoS = 1.5$ ในส่วนของเอลิเมนต์ที่มีค่า $y_i = 1$ ค่ายังโมดูลัสของเอลิเมนต์นั้นคือ $E = 214 \times 10^5 \text{ N/m}^2$ หากทำการแยกตามฟังก์ชันเป้าหมาย ปัญหาการออกแบบในการศึกษานี้มี 4 อย่าง คือ

$$\text{OPT1: } f_1 = \text{มวลของโครงสร้าง } f_2 = \mathbf{F}_1^T \mathbf{q}_1 + \mathbf{F}_2^T \mathbf{q}_2 + \mathbf{F}_3^T \mathbf{q}_3$$

$$\text{OPT2: } f_1 = \text{มวลของโครงสร้าง } f_2 = -(\omega_1 + \omega_2 + \omega_3)$$

$$\text{OPT3: } f_1 = \text{มวลของโครงสร้าง } f_2 = FRFCrest(\omega_1) + FRFCrest(\omega_2) + FRFCrest(\omega_3)$$

$$\text{OPT4: } f_1 = \text{มวลของโครงสร้าง } f_2 = FTCrest(\omega_1) + FTCrest(\omega_2) + FTCrest(\omega_3)$$

เมื่อ

$\mathbf{F}_i^T \mathbf{q}_i$ คือ ค่าความหยุ่นรวมของภาระกระทำกรณีที่ i -th

ω_i คือ ค่าความถี่ธรรมชาติของโครงสร้างโหมดที่ i -th (เรียงจากน้อยไปหามาก)

$FRFCrest(\omega_i)$ คือ ค่าพื้นที่ใต้กราฟ FRF ที่จุดยอดของความถี่ธรรมชาติ ω_i

$FTCrest(\omega_i)$ คือ ค่าพื้นที่ใต้กราฟ FT ที่จุดยอดของความถี่ธรรมชาติ ω_i

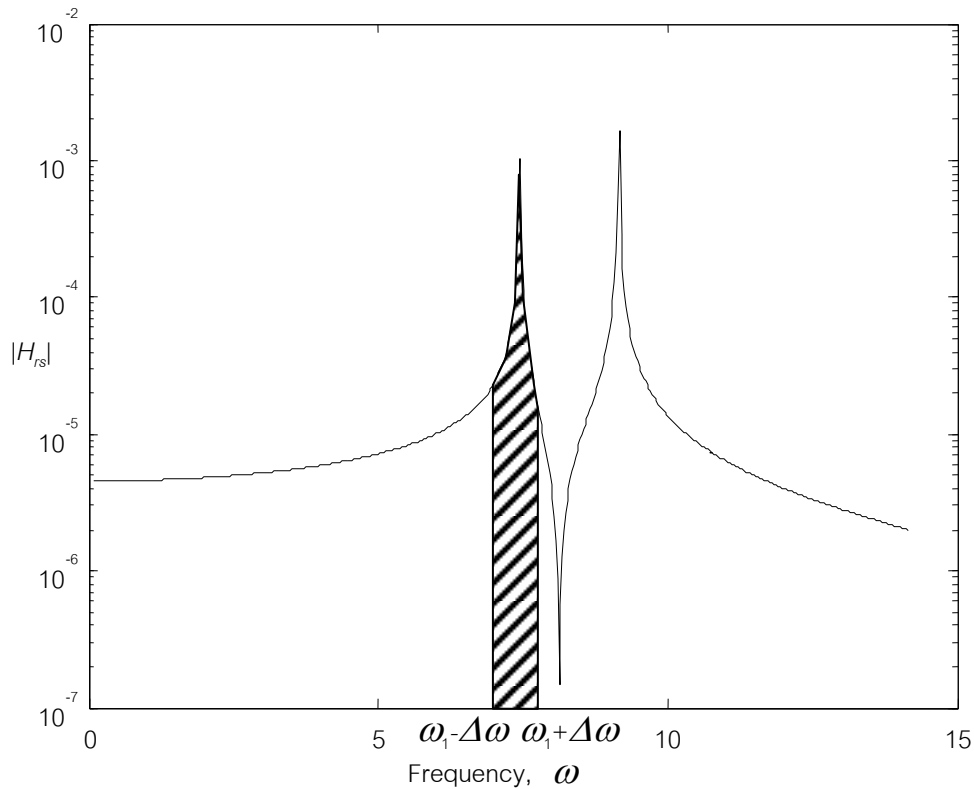
ฟังก์ชันเป้าหมายที่สองของปัญหาการออกแบบ OPT1 เป็นการหาค่าความน่าเชื่อถือสูงสุดของโครงสร้างซึ่ง ในที่นี้ เป็นการหาค่าความหยุ่นรวมของภาระสถิตทั้งสามกรณีต่ำสุด หรืออาจกล่าวได้ว่าเป็นการหาความแข็งแรงเชิงสถิตสูงสุดของโครงสร้าง ฟังก์ชันเป้าหมายที่สองของ OPT2 คือการหาค่าความถี่ธรรมชาติรวม 3 โหมดแรกสูงสุด ซึ่งสมมูลกับการหาค่าความแข็งแรงเชิงพลวัตสูงสุดของโครงสร้าง สำหรับ OPT3 รูปที่ 6.11 แสดงการคำนวณค่าของ $FRFCrest(\omega_1)$ ซึ่งเป็นการคำนวณหาพื้นที่ใต้กราฟ FRF ในช่วง $[\omega_1 - \Delta\omega, \omega_1 + \Delta\omega]$ การหาค่า $FRFCrest$ ต่ำสุด แสดงถึงการลดการสั่นสะเทือนของโครงสร้างนั่นเอง ในทำนองเดียวกันค่า $FTCrest$ ของปัญหาการออกแบบ OPT4 สามารถคำนวณในรูปแบบเดียวกันกับ $FRFCrest$ โดยที่การหาค่า $FTCrest$ ต่ำสุด คือ การลดการส่งผ่านแรงสั่นสะเทือนจากโครงสร้างไปยังฐานของหอสูงนั่นเอง ในงานนี้ ค่า FRF ที่ใช้เป็นค่ารีเซปแทนซ์แบบจุด (point receptance) วัดที่จุดยอดของหอสูง ส่วนค่าของ FT เป็นการส่งถ่ายแรงในแนวแกน x จากจุดยอดของหอสูงไปยังฐานรองรับโครงสร้าง

ขั้นตอนวิธีพัฒนาการแบบหลายเป้าหมายที่ใช้ในการค้นหาขอบหน้าพาเรโตของปัญหาการหาค่าเหมาะที่สุดของหอสูง พร้อมการตั้งค่า มีรายละเอียดดังต่อไปนี้

- SPEA: เวอร์ชันที่สองของ SPEA โดยใช้เลขจำนวนจริงเป็นตัวแปรออกแบบ และมีอัตราการเกิด ครอสโอเวอร์และมิวเทชันเป็น 1.0 และ 0.5 ตามลำดับ ขนาดของประชากรและจำนวนวงวนสูงสุดมีค่าเป็น 200 และ 300 ตามลำดับ ส่วนขนาดของหน่วยเก็บขอบหน้าพาเรโตมีค่าเป็น 200

- PBIL ค่าอัตราการเรียนรู้ (learning rate) ค่าพารามิเตอร์ของการมิวเทชัน (mutation shift) และอัตราการเกิดมิวเทชัน (mutation rate) คือ 0.5, 0.2 และ 0.1 ตามลำดับ จำนวนของเวกเตอร์ความน่าจะเป็นคือ 40 ขนาดของประชากรและจำนวนวงวนสูงสุดมีค่าเป็น 200 และ 300 ตามลำดับ ส่วนขนาดของหน่วยเก็บขอบหน้าพาเรโตมีค่าเป็น 200

- AMOSA: อุณหภูมิเริ่มต้นคือ 10 อุณหภูมิสุดท้ายคือ 0.001 ใช้วิธีซิมเพิล็กซ์ลงเขาใน 3 วงวนแรก จำนวนครั้งในการหาค่าฟังก์ชันคือ 200×300 ขนาดของหน่วยเก็บผลเฉลยที่ไม่ถูกรอรับจะถูกลดลงเป็น 200 หากสมาชิกในหน่วยเก็บมีจำนวนเกิน 400 ตัว



รูปที่ 6.11 $FRFCrest(\omega_1)$

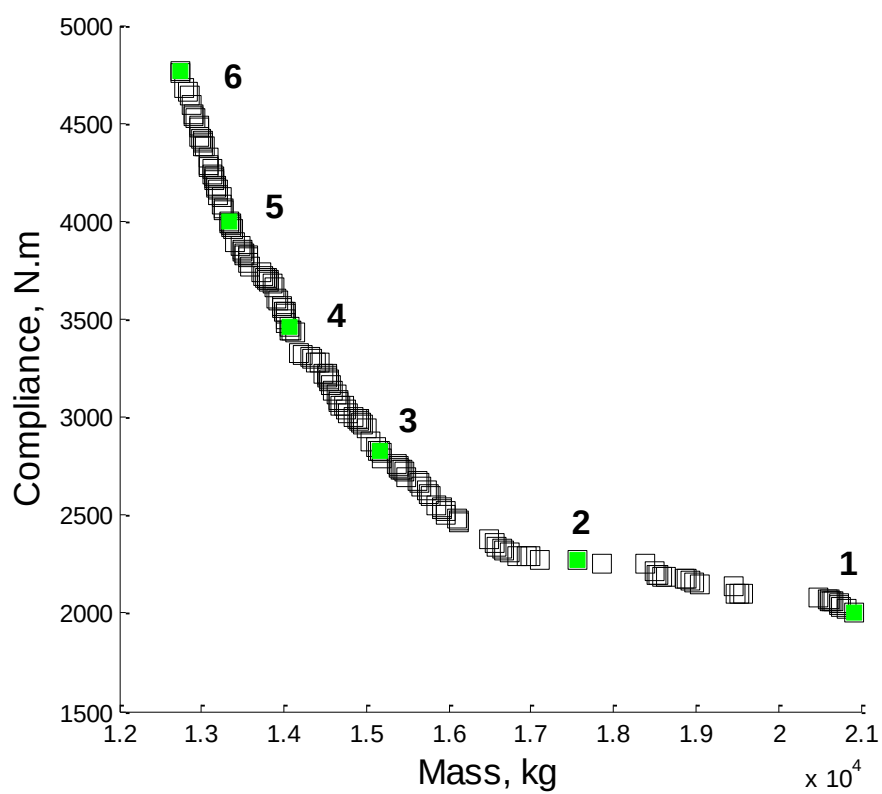
6.3.3 ผลการออกแบบ

ทำการรันวิธีหาค่าเหมาะที่สุดแต่ละวิธี 10 ครั้ง เมื่อได้ขอบหน้าพาเรโต 10 ชุดสำหรับแต่ละวิธี ทำการคำนวณค่าตัวบ่งชี้ HV และระยะตามรุ่น (generational distance, GD) ในที่นี้ ใช้ค่าเฉลี่ยและค่าส่วนเบี่ยงเบนมาตรฐานของ HV และ GD ในการประเมินสมรรถนะของ MOEAs ผลการเปรียบเทียบวิธีวิวัฒนาการทั้งสาม แสดงในตารางที่ 6.4 ในกรณีของ OPT1 เมื่อตรวจสอบค่าเฉลี่ยและค่าส่วนเบี่ยงเบนมาตรฐานของ HV และ GD พบว่า วิธีที่ดีที่สุดคือ PBIL สำหรับ OPT2 PBIL ให้ค่า HV ดีที่สุด ส่วน SPEA ให้ค่า GD ดีที่สุด ซึ่งแสดงว่าขอบหน้าพาเรโตที่ได้จาก PBIL มีการกระจายตัวกว้างกว่าของ SPEA แต่ขอบหน้าพาเรโตของ SPEA มีการลู่เข้าหาค่าตอบดีกว่า สำหรับปัญหาการออกแบบ OPT3 และ OPT4 SPEA คือวิธีที่ดีที่สุดในการหาขอบหน้าพาเรโต จากผลการเปรียบเทียบนี้ เราสามารถสรุปเป็นภาพรวมได้ว่า PBIL มีสมรรถนะสูงในกรณีของการหาค่าเหมาะที่สุดที่เกี่ยวกับการหาค่าความแข็งแรงเชิงสถิต (OPT1) และเชิงพลวัต (OPT2) ส่วน SPEA เป็นวิธีที่ดีที่สุดในการหาค่าเหมาะที่สุดที่เกี่ยวข้องกับการลดการสั่นสะเทือน (OPT2–4)

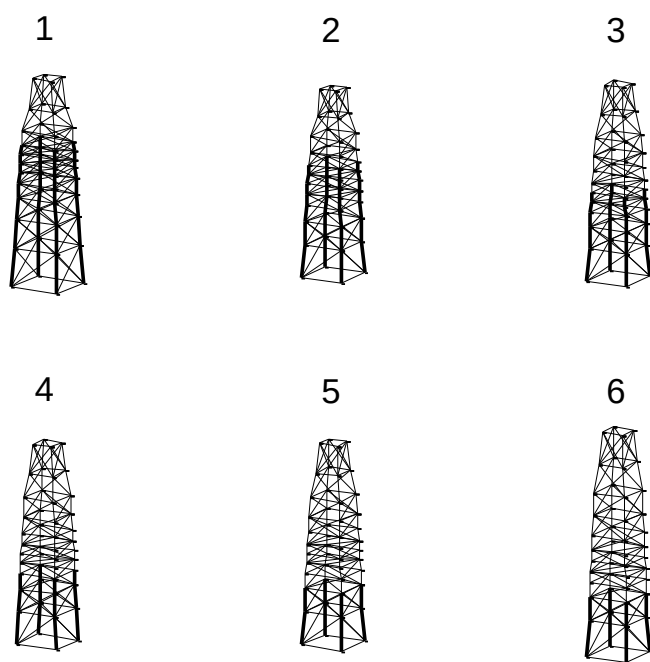
ตารางที่ 6.4 การเปรียบเทียบ HV และ GD

		HV		GD	
		AVG	STD	AVG	STD
OPT1	SPEA	0.8176	0.2008	0.0426	0.1348
	PBIL	0.9393	0.0300	0.0014	0.0045
	AMOSa	0.6020	0.2013	0.2944	0.4481
OPT2	SPEA	0.6453	0.1817	0.0083	0.0147
	PBIL	0.8055	0.0696	0.0245	0.0260
	AMOSa	0.4548	0.1985	0.3157	0.3007
OPT3	SPEA	0.9486	0.0235	0.0480	0.0378
	PBIL	0.8840	0.0524	0.3425	0.3676
	AMOSa	0.6632	0.1537	0.3012	0.2041
OPT4	SPEA	0.9767	0.0176	0.0295	0.0302
	PBIL	0.7672	0.1189	0.3203	0.2628
	AMOSa	0.8501	0.0743	0.3384	0.1609

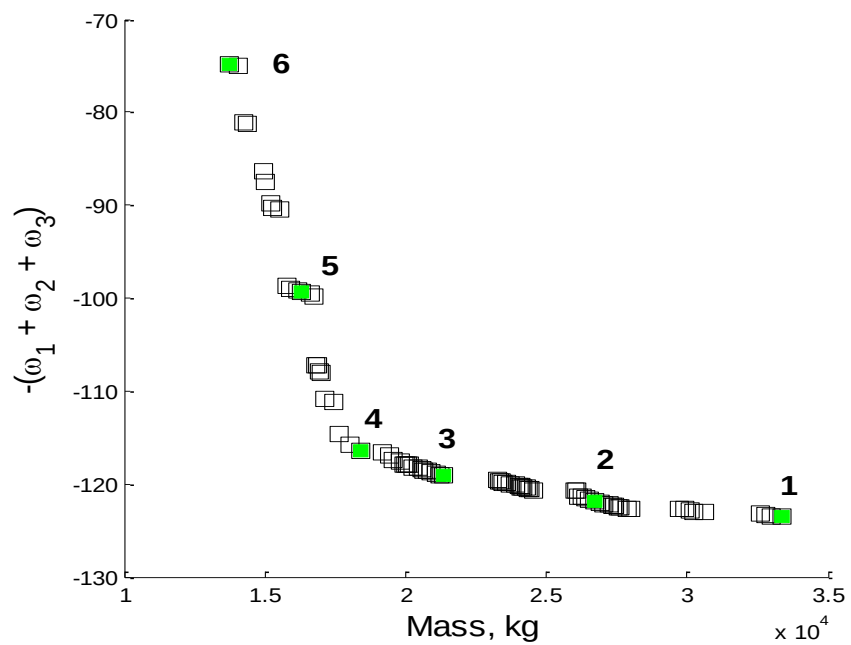
ขอบหน้าพารेटอที่ดีที่สุด (วัดจากค่า HV) ของปัญหาการออกแบบ OPT1–4 แสดงในรูปที่ 8.12, 8.14, 8.16 และ 8.18 ตามลำดับ ในแต่ละปัญหาการออกแบบ ทำการเลือกผลเฉลยจากขอบหน้าพารेटอมา 6 ค่าและแสดงรูปของโครงสร้างที่สอดคล้องกับผลเฉลยเหล่านั้น ดังแสดงในรูปที่ 8.13, 8.15, 8.17 และ 8.19 เราสามารถสรุปได้ว่า ในการใช้ MOEAs จะได้โครงสร้างของหอสูงหลากหลายรูปแบบเพื่อการตัดสินใจ จากการรันวิธีการหาค่าเหมาะที่สุดเพียงครั้งเดียว



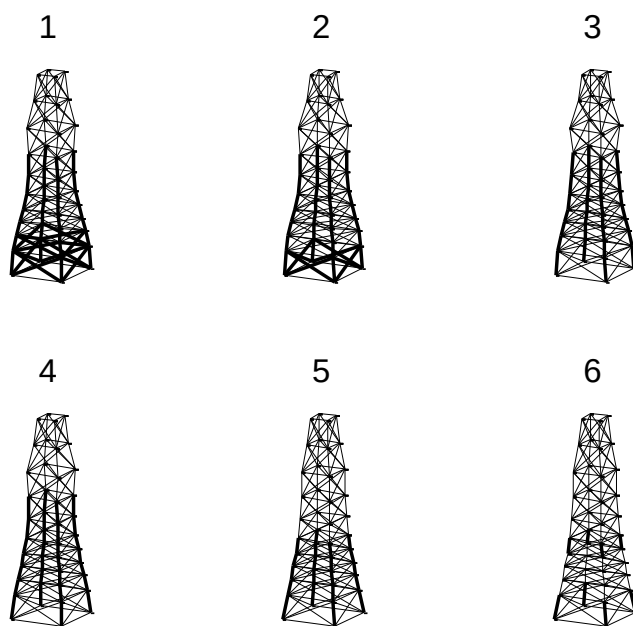
รูปที่ 6.12 ขอบหน้าพาเรโตที่ดีที่สุดของ OPT1



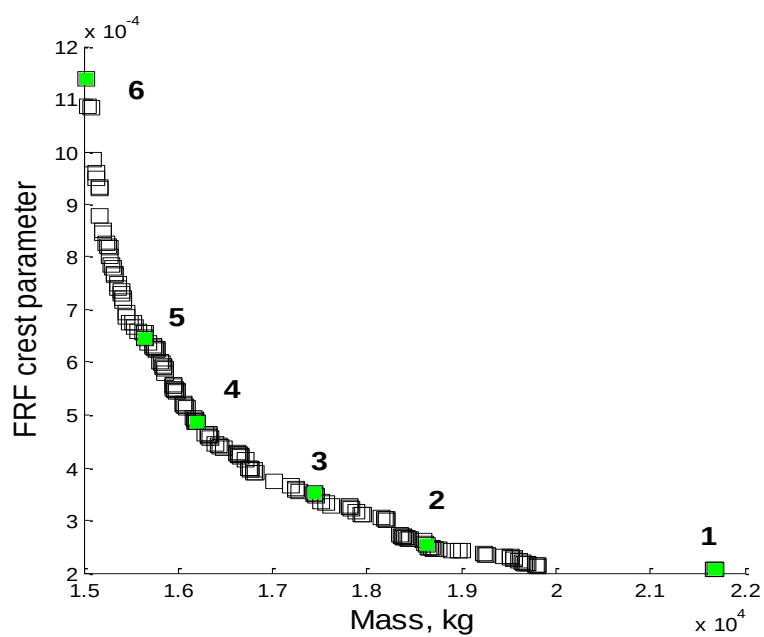
รูปที่ 6.13 โครงสร้างจากขอบหน้าพาเรโตในรูปที่ 6.12



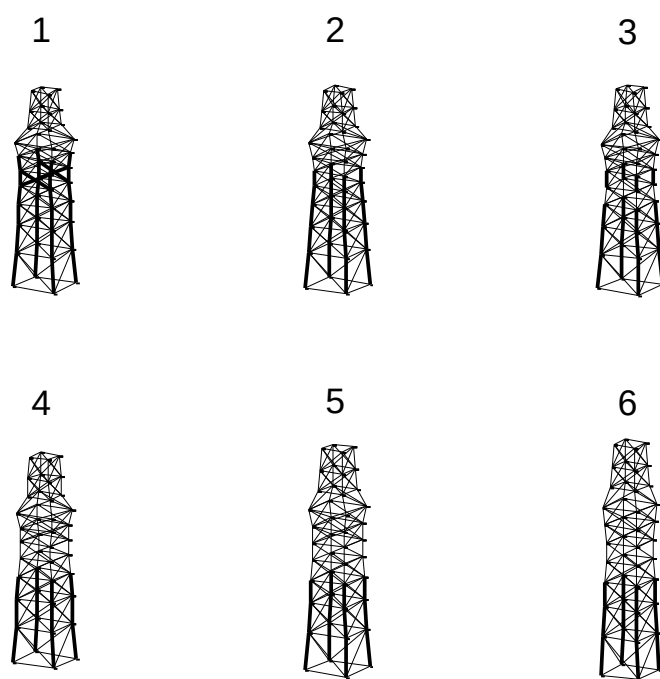
รูปที่ 6.14 ขอบหน้าพารेटโตที่ดีที่สุดของ OPT2



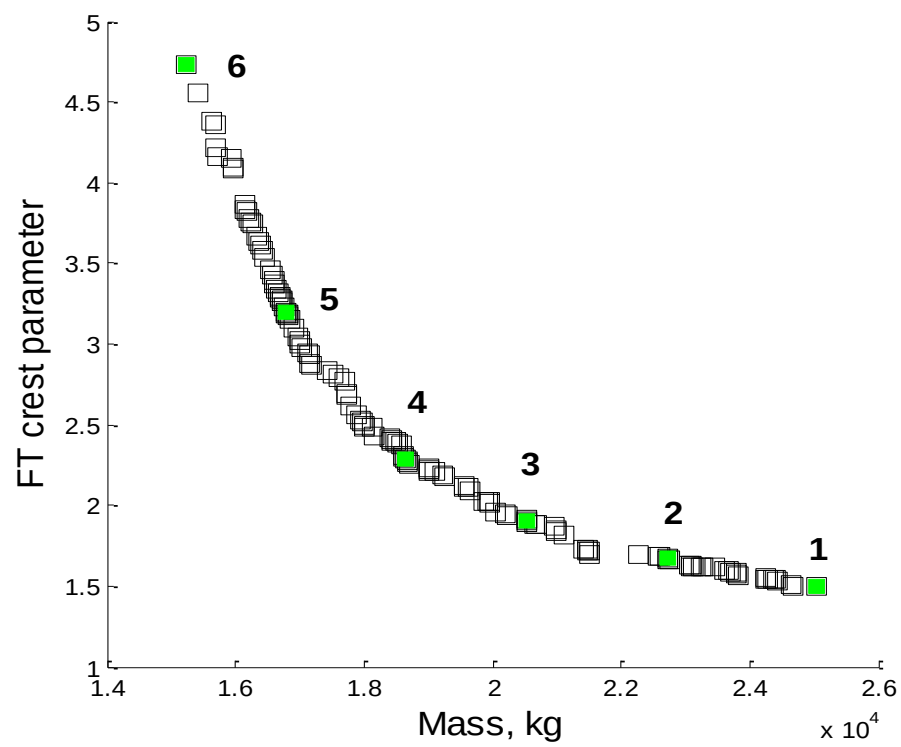
รูปที่ 6.15 โครงสร้างจากขอบหน้าพารेटโตในรูปที่ 6.14



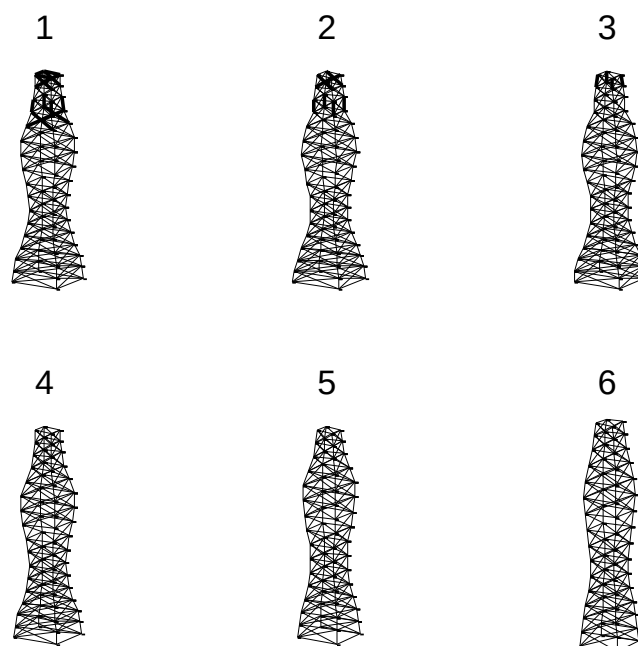
รูปที่ 6.16 ขอบหน้าพารโโตที่ดีที่สุดของ OPT3



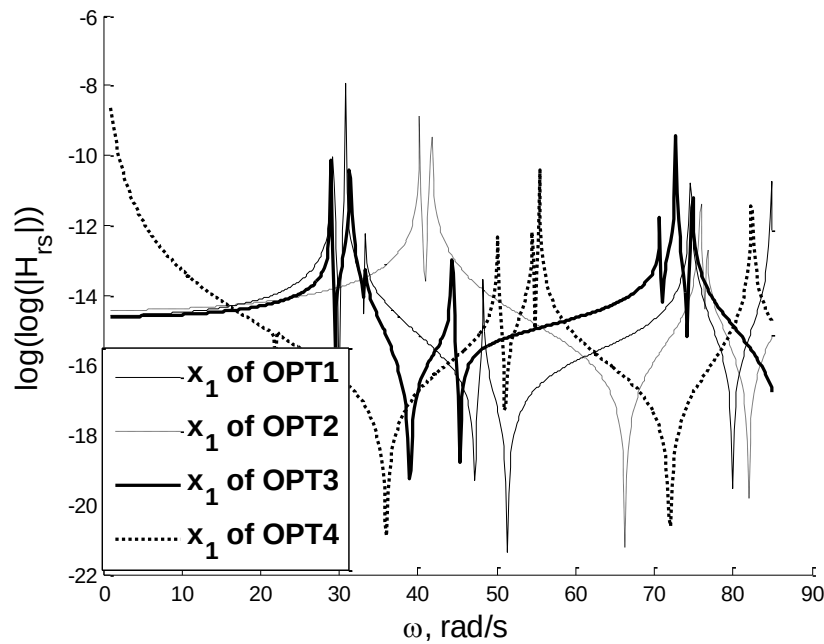
รูปที่ 6.17 โครงสร้างจากขอบหน้าพารโโตในรูปที่ 6.16



รูปที่ 6.18 ขอบหน้าพารโโตที่ดีที่สุดของ OPT4

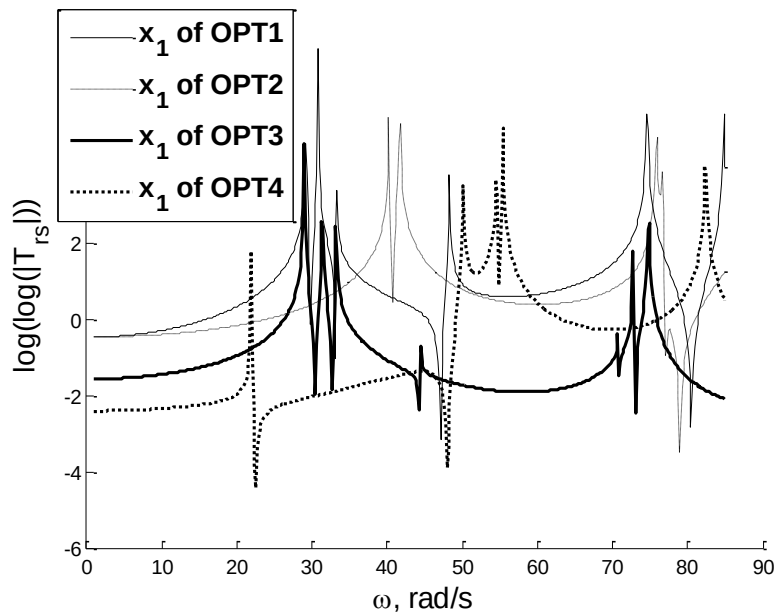


รูปที่ 6.19 โครงสร้างจากขอบหน้าพารโโตในรูปที่ 6.18



รูปที่ 6.20 FRF ของผลเฉลยจาก OPT1-4

เพื่อทำการตรวจสอบการใช้งานได้จริงของกระบวนการออกแบบนี้ รูปที่ 8.20 แสดงค่า FRF ของผลเฉลยที่ 1 ของ OPT1-4 ในรูปที่ 8.13, 8.15, 8.17 และ 8.19 ตามลำดับ จากรูปจะเห็นได้ว่า ผลเฉลยแต่ละตัว จะมีคุณสมบัติตามที่ได้กำหนดในฟังก์ชันเป้าหมายที่สองของปัญหาการออกแบบที่เกี่ยวข้อง ตัวอย่างเช่น ค่าความถี่ธรรมชาติสามค่าแรกของผลเฉลยเหมาะที่สุดที่ได้จาก OPT2 มีค่าสูงที่สุด ค่า $FRFCrest$ ของผลเฉลยเหมาะที่สุดที่ได้จากขอบหน้าพาเรโตของ OPT3 เป็นค่าที่ดีที่สุด รูปที่ 8.21 แสดงค่า FT ของทั้งสี่โครงสร้างที่กล่าวในข้างต้น จากรูปพบว่า ผลเฉลยจาก OPT4 ให้ค่า $FTCrest$ ดีที่สุด ซึ่งเป็นไปตามความคาดหมาย จากผลการตรวจสอบนี้แสดงให้เห็นว่า กระบวนการออกแบบที่นำเสนอนี้สามารถใช้ได้จริงในทางปฏิบัติ



รูปที่ 6.21 FT ของผลเฉลยจาก OPT1–4

6.4 สรุปและวิจารณ์

ในบทนี้ได้ทำการสาธิตการออกแบบโครงข่ายหมุนซึ่งเป็นการหาทอพอโลยี รูปทรงและขนาด พร้อมกัน จะให้ผลเฉลยเหมาะที่สุดที่มีสมรรถนะสูงกว่าการออกแบบที่ดำเนินการทีละขั้นตอน การประยุกต์ใช้ขั้นตอนวิธีวิวัฒนาการจะให้โครงสร้างเหมาะที่สุดที่ใกล้เคียงกับการนำไปใช้งานจริง การประยุกต์ใช้ขั้นตอนวิธีวิวัฒนาการทำให้ผู้ออกแบบสามารถหาค่ามูลต่ำสุดของโครงสร้าง ซึ่งหมายถึงราคาต่ำสุดของโครงสร้าง และในขณะเดียวกัน ก็สามารถเพิ่มความน่าเชื่อถือของโครงสร้างเป็นเป้าหมายที่สองในการออกแบบ การสาธิตการออกแบบในบทนี้ สามารถนำไปประยุกต์ใช้ในงานจริง หลากหลายรูปแบบ เช่น การออกแบบโครงสร้างสะพาน เสาไฟฟ้าแรงสูง และโครงสร้างป้ายโฆษณา เป็นต้น นอกจากนี้ แนวคิดในการออกแบบเหล่านี้ยังสามารถขยายผลไปยังการออกแบบโครงข่ายแข็ง เช่น โครงตัวถังของยานยนต์ขนาดเล็ก และ โครงสร้างคอนกรีตหลักของอาคาร เป็นต้น

บทที่ 7 กรณีศึกษา การหาค่าเหมาะที่สุดของโครงข้อแข็ง

7.1 ความนำ

เนื้อหาในบทนี้เป็นการสาธิตการประยุกต์ใช้ขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมาย เพื่อการออกแบบหาค่าเหมาะที่สุดของโครงข้อแข็ง (frames) ซึ่งมีพฤติกรรมทางกลแตกต่างจากโครงข้อหมุน (trusses) กล่าวคือ ชิ้นส่วนหรือเอลิเมนต์ของโครงสร้างชนิดนี้มีการรับโมเมนต์ดัดและโมเมนต์บิด โดยทั่วไปในการออกแบบโครงข้อหมุน ผู้ออกแบบจะต้องคำนึงถึงความเสถียรของโครงสร้างในเบื้องต้น โครงสร้างจะต้องจัดว่าเป็น statically determinate หรือ statically indeterminate เป็นอย่างน้อย แต่การออกแบบโครงข้อแข็งเราจะไม่คำนึงถึงปัจจัยนี้ ทำให้การออกแบบโครงข้อแข็งดำเนินการได้ง่ายกว่าโครงข้อหมุน

ในบทนี้จะนำเสนอการประยุกต์ใช้ขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมายเพื่อการออกแบบหาขนาดและรูปทรงเหมาะที่สุดของโครงสะพานสองมิติ และโครงสร้างของแขนของรถไถเดินตาม จากนั้นจะนำเสนอการหาค่าเหมาะที่สุดแบบอัตโนมัติหรือการหาทอพอโลยี รูปทรงและขนาดเหมาะที่สุดของโครงสร้างสะพานสองมิติ โดยขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมายที่ใช้ในการสาธิตการออกแบบคือ

- non-dominated sorting genetic algorithm (NSGAI)
- strength Pareto evolutionary algorithm (SPEA2)
- population-based incremental learning (PBIL)
- Pareto archive evolution strategy (PAES)
- multiobjective particle swarm optimisation (MPSO)

นอกจากนี้ยังได้นำเสนอตัวชี้วัดแบบใหม่เพื่อใช้ประเมินสมรรถนะของขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมายซึ่งเรียกว่าค่า C'

7.2 ปัญหาการออกแบบ

ปัญหาการออกแบบโครงข้อแข็ง (ในที่นี้อาจเรียกว่าโครงถัก) สำหรับกรณีศึกษาในบทนี้สามารถเขียนได้ดังนี้

$$\min_x \quad (7.1)$$

subject to

$$\sigma_i \leq \sigma_a, i = 1, \dots, N_e$$

เมื่อ σ_i คือ ค่าความเค้นบนเอลิเมนต์หรือชิ้นส่วนที่ i^{th}

σ_a คือ ค่าความเค้นสูงสุดที่วัสดุที่ใช้ทำเอลิเมนต์สามารถรับได้

N_e คือ จำนวนของชิ้นส่วนบนโครงสร้าง

ในส่วนของเวกเตอร์ฟังก์ชันเป้าหมาย $\mathbf{f}$ จะมีจำนวนสมาชิก 2 และ 3 ค่าเท่านั้น เนื่องจากเป็นโครงสร้างใช้งานอย่างง่ายจึงมีแรงกระทำที่ไม่ซับซ้อน ฟังก์ชันเป้าหมายที่ใช้ในการศึกษาจะมีรูปแบบคล้ายกันกับในบทที่ผ่านมา เพราะโครงสร้างต้องรับภาระทั้งแบบสถิตและพลวัต ซึ่งมีรายละเอียดดังนี้

f_1 : structural mass

f_2 : $1/\omega_1^2$

f_3 : $1/(\omega_1^2 + \omega_2^2 + \omega_3^2)$

f_4 : FRF crest parameter at ω_1

f_5 : mean value of FRF crest parameters at ω_1 , ω_2 and ω_3

f_6 : FT crest parameter at ω_1

f_7 : mean value of FT crest parameters at ω_1 , ω_2 and ω_3

เมื่อ ω_i คือ ค่าความถี่ธรรมชาติโหมดที่ i^{th} ของโครงสร้าง

ในที่นี้ จะใช้ f_1 ซึ่งสามารถบ่งบอกถึงราคาของโครงสร้างเป็นฟังก์ชันเป้าหมายแรกเสมอ เพราะราคาของโครงสร้าง ถือว่าเป็นปัจจัยที่สำคัญมากในโครงสร้างหรือชิ้นส่วนทางกลที่จะออกแบบนี้ ด้วยเหตุนี้ ทำให้เราสามารถตั้งปัญหาการออกแบบได้เป็น 7 ปัญหาดังนี้

P1: $\min f_1, \min f_2$

P2: $\min f_1, \min f_3$

P3: $\min f_1, \min f_4$

P4: $\min f_1, \min f_5$

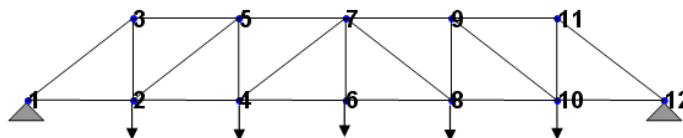
P5: $\min f_1, \min f_6$

P6: $\min f_1, \min f_7$

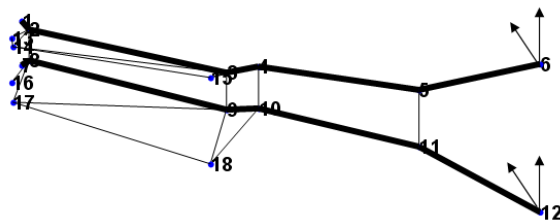
P7: $\min f_1, \min f_5, \min f_7$

โดยมีเงื่อนไขบังคับ ตามที่แสดงในปัญหาการออกแบบทั่วไป (7.1) โครงสร้างที่ใช้สาธิตการออกแบบทั้งสาม จะทำจากวัสดุที่มีค่าโมดูลัสของยังเป็น $E = 209 \times 10^9 \text{ N/m}^2$ ค่าความหนาแน่น $\rho = 7000 \text{ kg/m}^3$ ซึ่งมีรายละเอียดดังนี้

ST1: โครงถักสะพานสองมิติแสดงในรูปที่ 7.1 ประกอบด้วยจุดต่อ 12 จุด และมีชิ้นส่วนจำนวน 21 ชิ้น จุดต่อหมายเลข 1 และ 12 ถูกใส่เงื่อนไขขอบเขตเป็นการยึดกับพื้นและมีแรงในแนวดิ่งกระทำที่จุดต่อหมายเลข 2 4 6 8 และ 10 ตัวแปรออกแบบคือตำแหน่งของจุดต่อด้านบน และขนาดของเอลิเมนต์ ทั้งนี้ได้พิจารณาให้โครงสร้างมีความสมมาตรทั้งด้านซ้ายและขวา ค่า FRF คือค่า point-receptance ในทิศแนวดิ่งของจุดต่อหมายเลข 7 ส่วนค่า FT เป็นการคำนวณการส่งถ่ายแรงในแนวดิ่งจากจุดต่อหมายเลข 7 ไปยังจุดรองรับหมายเลข 1



รูปที่ 7.1 สะพานสองมิติ ST1

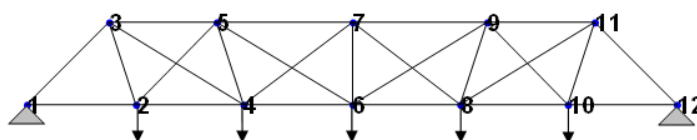


รูปที่ 7.2 โครงสร้างแขนรถไถเดินตาม ST2

ST2: โครงสร้างของชิ้นส่วนคันบังคับของรถไถเดินตามแสดงในรูปที่ 7.2 โครงสร้างประกอบด้วยจุดต่อ 16 จุด และมีชิ้นส่วนจำนวน 27 ชิ้น โดยมีจุดต่อด้านซ้ายสุดของโครงสร้างถูกใส่เงื่อนไขขอบเขตเป็นการยึดกับพื้นและมีแรงกระทำที่จุดต่อหมายเลข 6 และ 12 ทั้งนี้มีภาระกระทำสองรูปแบบ กล่าวคือ แรงกระทำเมื่อเลี้ยวรถไถและแรงกระทำเมื่อผู้ใช้งานกดปลายคันบังคับเพื่อให้รถอยู่ในสภาพสมดุลขณะทำงาน ตัวแปรออกแบบคือตำแหน่งของจุดต่อที่ไม่มีเงื่อนไขขอบเขตและขนาดของเอลิเมนต์ซึ่งมีทั้งหมด 24 ตัวแปร ทั้งนี้ได้พิจารณาให้โครงสร้างมีความสมมาตรทั้งด้านซ้ายและขวา ค่า FRF คือค่า point-receptance ในทิศแนวดิ่งของจุดต่อหมายเลข 6 ส่วนค่า FT เป็นการคำนวณการส่งถ่ายแรงในแนวดิ่งจากจุดต่อหมายเลข 6 ไปยังจุดรองรับหมายเลข 1

ST3: โครงถักสะพานสองมิติแสดงในรูปที่ 7.3 ซึ่งเป็นการดัดแปลงเพิ่มเติมจาก ST1 เพื่อใช้สาธิตการหาทอพอโลยี รูปทรง และขนาดที่เหมาะสมที่สุดพร้อมกันหรือที่เรียกว่าการออกแบบอัตโนมัติ โครงสร้างในรูปจะเป็นเอลิเมนต์พื้น (ground elements) ประกอบด้วยจุดต่อ 12 จุด และมีชิ้นส่วนจำนวน 25 ชิ้น จุดต่อหมายเลข 1 และ 12 ถูกใส่เงื่อนไขขอบเขตเป็นการยึดกับพื้นและมีแรงในแนวดิ่งกระทำที่จุดต่อหมายเลข 2 4 6 8 และ 10 ตัวแปรออกแบบคือตำแหน่งของจุดต่อด้านบน และขนาดของเอลิเมนต์ ทั้งนี้ได้พิจารณาให้โครงสร้างมีความสมมาตรทั้งด้านซ้ายและขวา ค่า FRF คือค่า point-receptance ในทิศแนวดิ่งของจุดต่อหมายเลข 7 ส่วนค่า FT เป็นการคำนวณการส่งถ่ายแรงในแนวดิ่งจากจุดต่อหมายเลข 7 ไปยังจุดรองรับหมายเลข 1 เมื่อได้ผลเฉลยที่เหมาะสมที่สุดแล้ว เอลิเมนต์ที่มีขนาดเล็กมากจะถูกตัดออกจากโครงสร้างซึ่งเป็นการกำหนดทอพอโลยีของโครงสร้างนั่นเอง

การวิเคราะห์โครงสร้างสามารถทำได้โดยการใช้เอลิเมนต์ของคานที่มี 12 d.o.f 2 จุดต่อ และวางตัวในสามมิติ



รูปที่ 7.3 โครงสร้างสองมิติ ST3

7.3 การทดสอบสมรรถนะ MOEAs

ขั้นตอนวิธีวิวัฒนาการที่ใช้ในการหาผลเฉลยของการออกแบบในที่นี้ประกอบด้วย

BPBIL: PBIL using binary codes with 0.05 mutation rate, and 0.2 mutation shift

BPAES: PAES using binary codes

BNSGA: NSGAII using binary codes with crossover and mutation rates as 1.0 and 0.5 respectively

BSPEA: SPEA2 using binary codes with crossover and mutation rates as 1.0 and 0.5 respectively

RSPEA: SPEA2 using real codes with crossover and mutation rates as 1.0 and 0.5 respectively

RNSGA: NSGAII using real codes with crossover and mutation rates as 1.0 and 0.5 respectively

RMPSO: MPSO using real codes with a starting inertia weight, an ending inertia weight, a cognitive learning factor, and a social learning factor as 0.5, 0.01, 0.5, and 0.5 respectively

เพื่อทำการทดสอบและเปรียบเทียบสมรรถนะของขั้นตอนวิธีวิวัฒนาการทั้ง 7 นี้อย่างเป็นระบบ จะทำการรันวิธีการเหล่านี้ในการหาผลเฉลยของปัญหาการออกแบบทั้ง 7 เป็นจำนวน 10 รอบและจะใช้ค่าทางสถิติในการเปรียบเทียบสมรรถนะ สำหรับทุกปัญหาการออกแบบ จะใช้จำนวนรอบในการค้นหาขอบหน้าพาเรโตของปัญหาการออกแบบเป็น 100 ลูป และใช้ขนาดของประชากรเป็น 50 ตัว กำหนดให้ขนาดของเมตริกซ์เก็บผลเฉลย (Pareto archive) มีค่าเป็น 100 ในที่นี้จะทดสอบ MOEAs กับการออกแบบโครงสร้าง ST1 และ ST2 เท่านั้น และเมื่อทราบวิธีที่มีประสิทธิภาพสูงสุด จะใช้วิธีการนั้นในการหาผลเฉลยของปัญหาการออกแบบโครงสร้าง ST3

สำหรับการเปรียบเทียบสมรรถนะของ MOEAs ในการศึกษาจะใช้ตัวชี้วัด C และตัวชี้วัดที่นำเสนอใหม่คือตัวชี้วัด C' ตัวชี้วัดทั้งสองนี้จะใช้ในการเปรียบเทียบขอบหน้าพาเรโตคู่ใดๆ (ที่ได้จากการประยุกต์ใช้ขั้นตอนวิธีวิวัฒนาการสองวิธีที่แตกต่างกัน) ถ้าเรามีเซตของผลเฉลยพาเรโตหรือขอบหน้าพาเรโตสองชุด $\{A\}$ และ $\{B\}$ ค่า $C_{A,B}$ แสดงถึงสัดส่วนระหว่างจำนวนผลเฉลยใน $\{B\}$ ที่ถูกรอบงำ (dominated) หรือเท่ากับกับผลเฉลยบางตัวใน $\{A\}$ หารด้วยจำนวนผลเฉลยทั้งหมดใน $\{B\}$ ในทางกลับกันค่าของ $C_{B,A}$ คือจำนวนผลเฉลยใน $\{A\}$ ที่ถูกรอบงำหรือเท่ากับกับ ผลเฉลยบางตัวใน $\{B\}$ หารด้วยจำนวนผลเฉลยทั้งหมดใน $\{A\}$ จากนิยามดังกล่าว ถ้า $C_{A,B} > C_{B,A}$ แสดงว่า $\{A\}$ ดีกว่า $\{B\}$ รูปที่ 7.4 แสดงเซตของขอบหน้าพาเรโต $\{A\}$ และ $\{B\}$ เมื่อ $C_{A,B} = 0.5000$ และ $C_{B,A} = 0.4000$ ซึ่งบ่งบอกว่าเซต $\{A\}$ ดีกว่า $\{B\}$ อย่างไรก็ตาม เราจะเห็นได้จากรูปว่าเซตทั้งสองไม่ได้ดีกว่ากันอย่างชัดเจน เมื่อเราสังเกตจากพื้นที่แรเงาในส่วนของ $\{A\}$ ดีกว่า $\{B\}$ จะพบว่ามีค่าน้อยกว่าส่วนของ $\{B\}$ ดีกว่า $\{A\}$ พื้นที่แรเงาทั้งสองส่วนนี้มีความสำคัญในการเปรียบเทียบเซตทั้งสองและควรนำมาพิจารณาเพื่อสามารถแสดงถึงผลกระทบของพื้นที่แรเงาในรูปที่ 7.4 เราได้ทำการนิยามตัวชี้วัดเพิ่มเติมซึ่งเรียกว่าตัวชี้วัด C' ดังนี้

$$C'_{A,B} = \frac{V_A}{V_A + V_B} \quad (7.2)$$

$$C'_{B,A} = 1 - C'_{AB}$$

เมื่อ

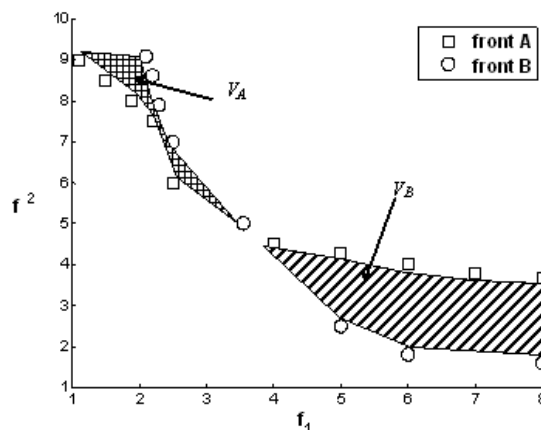
$$V_A = \sum_{a \in \hat{A}_1} \min \{ \|a - b\|; b \in \hat{B}_1 \}$$

$$V_B = \sum_{b \in \hat{B}_2} \min \{ \|b - a\|; a \in \hat{A}_2 \}.$$

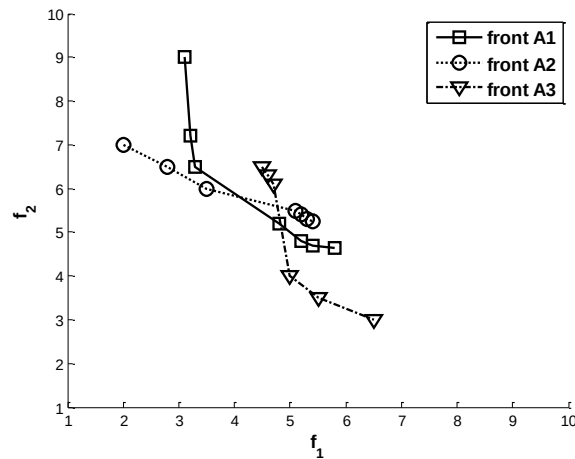
และถ้า $V_A = V_B = 0$ จะได้ $C'_{A,B} = C'_{B,A} = 0.5$

$\hat{A}_1$ คือ เซตของผลเฉลยบางส่วนใน $\{A\}$ ที่ครอบงำ (dominate) สมาชิกบางส่วนใน $\{B\}$ ซึ่งเขียนแทนด้วย $\hat{B}_1 \subseteq \{B\}$ และ $\hat{B}_2$ คือเซตของของผลเฉลยบางส่วนใน $\{B\}$ ที่ครอบงำสมาชิกบางส่วนของ $\{A\}$ ซึ่งเขียนได้เป็น $\hat{A}_2 \subseteq \{A\}$ ด้วยแนวคิดคล้ายๆกันถ้า $C'_{A,B} > C'_{B,A}$ หมายความว่า เซต $\{A\}$ ดีกว่าเซต $\{B\}$ จากนิยามพบว่าตัวชี้วัด C' รวมผลกระทบทั้งส่วนของความกว้าง การกระจายตัวของสมาชิกและการถ่วงน้ำหนักของขอบหน้าพารेटโต

จากรูปที่ 7.4 จะได้ $C'_{A,B} = 0.4556$ และ $C'_{B,A} = 0.5444$ ซึ่งหมายความว่าเซต $\{B\}$ ดีกว่าเซต $\{A\}$ จากตัวชี้วัดทั้งสองตัวนี้พบว่า เราไม่สามารถบ่งชี้ได้อย่างชัดเจนว่าเซตทั้งสองในรูปอันไหนดีกว่ากัน ซึ่งเป็นข้อสรุปที่เหมาะสมกว่าการใช้ตัวชี้วัดตัวใดตัวหนึ่งเพียงตัวเดียวในการเปรียบเทียบขอบหน้าพารेटทั้งสอง นอกจากนี้ ตัวชี้วัด C' ยังมีคุณสมบัติพิเศษ ดังแสดงในรูปที่ 7.5 จากรูปมีเซตของขอบหน้าพารेट 3 เซต เขียนแทนด้วย $\{A_1\}$ $\{A_2\}$ และ $\{A_3\}$ ส่วนการเปรียบเทียบตัวบ่งชี้ C และ C' แสดงในตารางที่ 7.1 ค่าที่แสดงในแถวที่ i^{th} และคอลัมน์ที่ j^{th} หมายถึง C_{A_i,A_j} จากการเปรียบเทียบด้วยตัวชี้วัด C เราสามารถสรุปได้ว่า เซต $\{A_3\}$ คือเซตที่ดีที่สุด รองลงมาคือเซต $\{A_1\}$ อย่างไรก็ตาม ในกรณีของตัวชี้วัด C' พบว่าเซต $\{A_2\}$ ดีกว่าเซต $\{A_1\}$ เซต $\{A_1\}$ ดีกว่าเซต $\{A_3\}$ และเซต $\{A_3\}$ ดีกว่าเซต $\{A_2\}$ สถานการณ์เช่นนี้ เรียกว่า กรรไกร-กระดาษ-ฉ้อน ซึ่งไม่สามารถสรุปได้ว่าใครดีที่สุด จากตัวอย่างนี้ทำให้เห็นได้ว่า การเปรียบเทียบขอบหน้าพารेटควรใช้ตัวชี้วัดมากกว่า 1 ตัว หากสามารถทำได้



รูปที่ 7.4 การเปรียบเทียบขอบหน้าพารेट



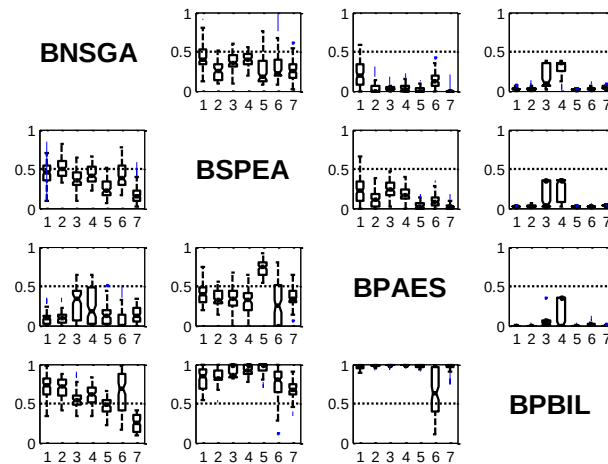
รูปที่ 7.5 สถานการณ์ กรรไกร-กระดาด-ฉ้อน

ตารางที่ 7.1 ค่า C และ C' ของ $\{A_1\}$, $\{A_2\}$, และ $\{A_3\}$

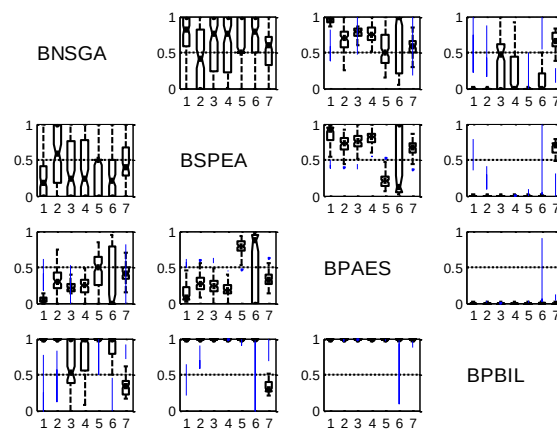
C-indicator			C'-indicator		
$\{A_1\}$	0.5714	0.1667	$\{A_1\}$	0.4929	0.5029
0.4286	$\{A_2\}$	0.5000	0.5076	$\{A_2\}$	0.4908
0.4286	0.5714	$\{A_3\}$	0.4971	0.5092	$\{A_3\}$

7.4 ผลการศึกษา

ผลการเปรียบเทียบสมรรถนะของ MOEAs ทั้ง 7 ด้วยตัวชี้วัด C และ C' จะแยกออกเป็นสองส่วนคือ ส่วนของวิธีที่ใช้เลขฐานสองและส่วนของวิธีที่ใช้เลขจำนวนจริงในกระบวนการหาผลเฉลย ผลการเปรียบเทียบทั้งหมดแสดงในรูปที่ 7.6 – 7.13 รูปที่ 7.6 แสดงผลการเปรียบเทียบค่า C ของวิธีหาค่าเหมาะที่สุดที่ใช้เลขฐานสองสำหรับโครงสร้าง ST1 ทั้งนี้จะแสดงในรูปแบบของ box-plot ในแต่ละ box-plot จะมีค่า C ของปัญหาการออกแบบทั้ง 7 ปัญหา box-plot ที่อยู่ที่แถวที่ i^{th} และคอลัมน์ที่ j^{th} แสดงค่า C_{ij} เมื่อ i คือขั้นตอนวิธีที่ i และ j คือขั้นตอนวิธีที่ j เนื่องจากการรันขั้นตอนวิธีวิวัฒนาการ 10 ครั้ง ค่า C ที่แสดงใน box-plot จะมีทั้งหมด 100 ค่าที่เกิดจากการเปรียบเทียบขอบหน้าพาเรโตที่ได้จากวิธี i 10 ชุด เทียบกับขอบหน้าพาเรโตที่ได้จากวิธี j 10 ชุด ส่วนรูปที่ 7.7 แสดงการเปรียบเทียบด้วยค่า C' ของวิธีการทั้ง 4 จากผลการศึกษาพบว่า BPBIL คือวิธีที่ดีที่สุด (วัดจากตัวชี้วัด C) สำหรับการออกแบบโครงสร้าง ST1 สำหรับปัญหาการออกแบบ P1-P6 ยกเว้นในปัญหาการออกแบบ P7 ที่ BNSGA ให้คำตอบที่ดีที่สุด ผลการเปรียบเทียบด้วยค่า C และ C' ให้ผลไปในทางเดียวกัน

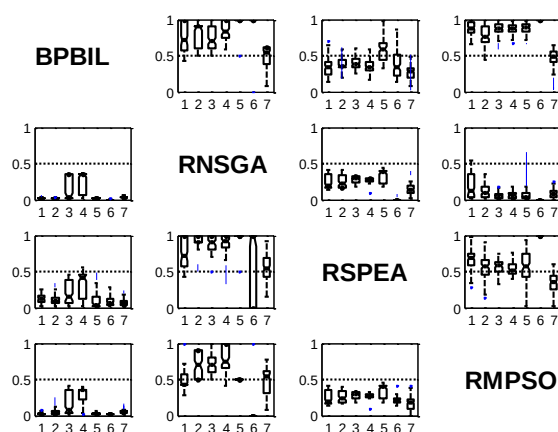


รูปที่ 7.6 การเปรียบเทียบค่า C ของ ST1-I

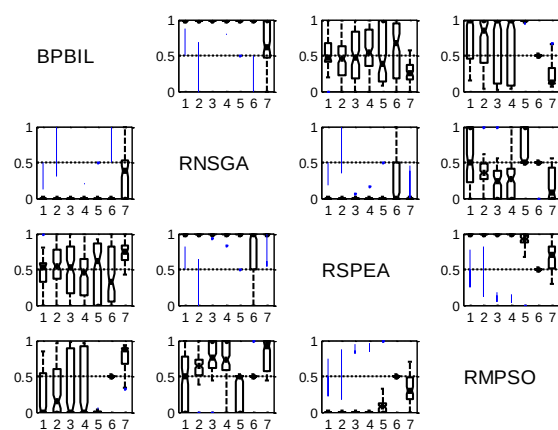


รูปที่ 7.7 การเปรียบเทียบค่า C' ของ ST1-I

เนื่องจาก BPBIL เป็นวิธีที่ดีที่สุดในการหาค่าของฟังก์ชันวัตถุประสงค์ จึงนำมาเปรียบเทียบกับ MOEAs ที่ใช้โค้ดแบบเลขจำนวนจริง ดังแสดงในรูปที่ 7.8 และ 7.9 สำหรับการเปรียบเทียบผลการออกแบบโครงสร้าง ST1 ด้วยตัวชี้วัด C และ C' ตามลำดับ จากผลการเปรียบเทียบพบว่า วิธีที่ดีที่สุดคือ BPBIL โดยมีวิธีที่ดีเป็นอันดับที่สอง คือ RSPEA จากรูปที่ 7.9 ซึ่งเป็นการเปรียบเทียบด้วยค่า C' เราจะเห็นว่า BPBIL และ RSPEA มีสมรรถนะใกล้เคียงกัน ซึ่งอาจเป็นปรากฏการณ์ที่คล้ายกับที่แสดงในรูปที่ 7.4 อย่างไรก็ตาม เมื่อพิจารณาทุกตัวชี้วัดแล้ว BPBIL จะเป็นวิธีที่ดีที่สุดโดยรวม



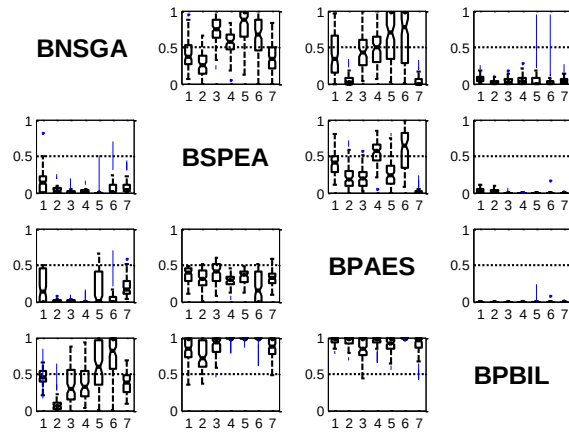
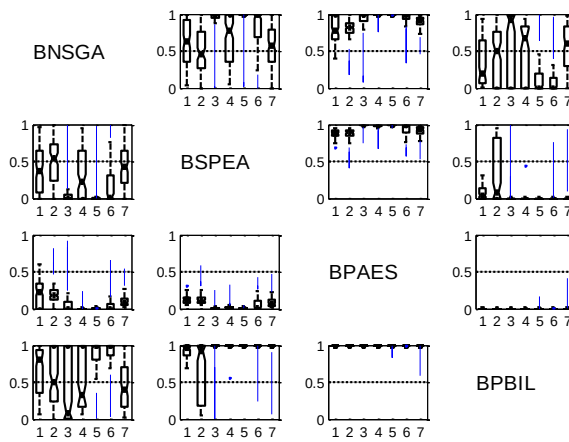
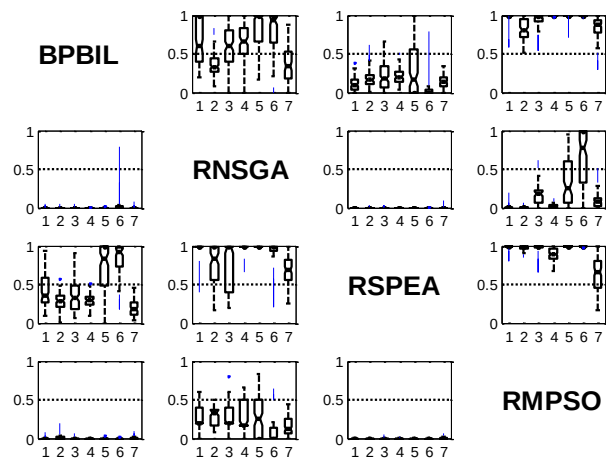
รูปที่ 7.8 การเปรียบเทียบค่า C ของ ST1-II

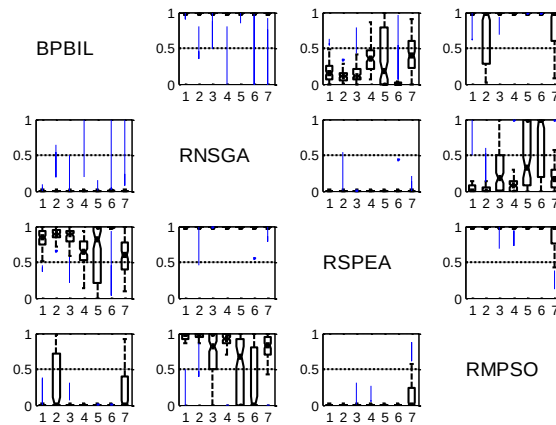


รูปที่ 7.9 การเปรียบเทียบค่า C' ของ ST1-II

ผลการเปรียบเทียบ MOEAs ที่ใช้เลขฐานสองสำหรับปัญหาการออกแบบทั้งเจ็ดสำหรับโครงสร้าง ST2 แสดงในรูปที่ 7.10 สำหรับการเปรียบเทียบด้วยค่า C และรูปที่ 7.11 สำหรับการเปรียบเทียบด้วยค่า C' จากรูปที่ 7.10 โดยภาพรวมวิธีที่ดีที่สุดคือ BPBIL และอันดับสองคือ BNSGA เมื่อเปรียบเทียบด้วยค่า C' วิธีที่ดีที่สุดโดยรวมยังคงเป็น BPBIL ในขณะที่ยันดับสองคือ BNSGA ทั้งนี้ BNSGA ดีกว่า BPBIL ในกรณีของปัญหา P3 และ P4 ซึ่งเป็นสถานการณ์คล้ายกันกับรูปที่ 7.4

เมื่อนำวิธีที่ดีที่สุดคือ BPBIL มาเปรียบกับ MOEAs ที่ใช้เลขจำนวนจริง ผลการเปรียบเทียบด้วย C และ C' แสดงในรูปที่ 7.12 และ 7.13 ตามลำดับ จากการเปรียบเทียบด้วยค่า C วิธีที่ดีที่สุดคือ RSPEA โดยมีอันดับสองคือ BPBIL และได้ผลสรุปเช่นเดียวกันในกรณีของการเปรียบเทียบด้วยค่า C' จากกรณีของโครงสร้าง ST1 และ ST2 เราสามารถสรุปได้ว่า วิธีที่ดีที่สุดมีสองวิธี คือ BPBIL สำหรับปัญหาการออกแบบที่มีจำนวนตัวแปรออกแบบน้อย และ RSPEA สำหรับปัญหาการออกแบบที่มีจำนวนตัวแปรออกแบบมาก

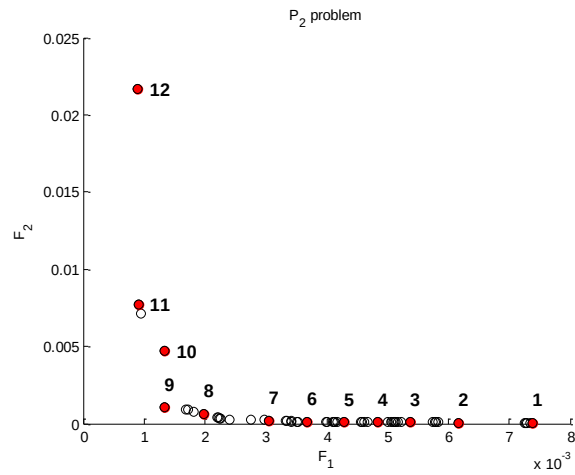
รูปที่ 7.10 การเปรียบเทียบค่า C ของ ST2-Iรูปที่ 7.11 การเปรียบเทียบค่า C' ของ ST2-Iรูปที่ 7.12 การเปรียบเทียบค่า C ของ ST2-II



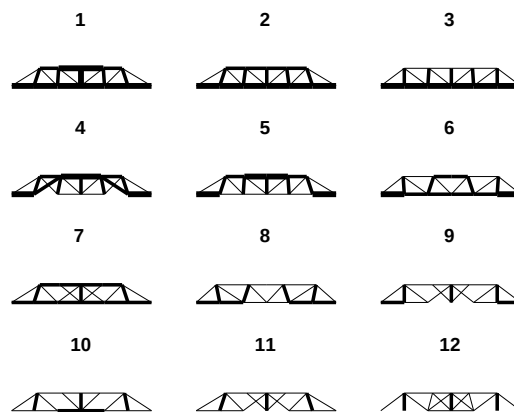
รูปที่ 7.13 การเปรียบเทียบค่า C' ของ ST2-II

ในการศึกษานี้ BPBIL ถูกเลือกเพื่อนำมาใช้ในการหาผลเฉลยของการออกแบบอัตโนมัติของโครงสร้าง ST3 ทั้งนี้เลือกหาผลเฉลยของปัญหาการออกแบบ P2 P4 และ P6 เท่านั้น ขอบหน้าพารโตของปัญหาการออกแบบ P2 ของการออกแบบอัตโนมัติของโครงสร้าง ST3 ที่ได้จากการใช้วิธี BPBIL แสดงในรูปที่ 7.14 โครงสร้างบางส่วนจากผลเฉลยพารโตในรูปที่ 7.14 แสดงในรูปที่ 7.15 ซึ่งจะเห็นได้ว่า เราจะได้โครงสร้างที่มีขนาดและรูปทรงที่แตกต่างกันหลากหลายรูปแบบ สำหรับการคัดเลือกไปใช้งานจากการรันขั้นตอนวิธีวิวัฒนาการเพียงครั้งเดียว

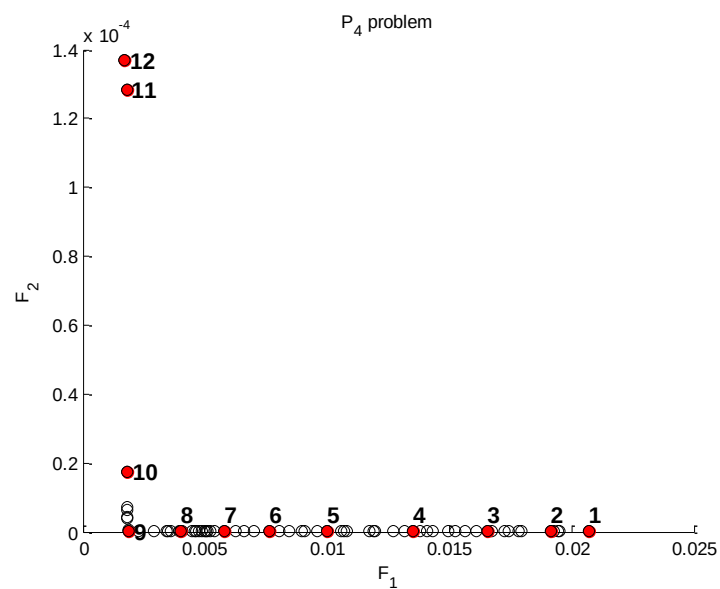
ขอบหน้าพารโตของปัญหาการออกแบบ P4 ของการออกแบบอัตโนมัติของโครงสร้าง ST3 ที่ได้จากการใช้วิธี BPBIL แสดงในรูปที่ 7.16 โครงสร้างบางส่วนจากผลเฉลยพารโตในรูปที่ 7.16 แสดงในรูปที่ 7.17 ส่วนขอบหน้าพารโตของปัญหาการออกแบบ P6 ของการออกแบบอัตโนมัติของโครงสร้าง ST3 ที่ได้จากการใช้วิธี BPBIL แสดงในรูปที่ 7.18 โครงสร้างบางส่วนจากผลเฉลยพารโตในรูปที่ 7.18 แสดงในรูปที่ 7.19 ทุกรูปแสดงให้เห็นถึงประโยชน์ของการประยุกต์ใช้ขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมายที่ให้ผลเฉลยที่เหมาะสมที่สุดจำนวนมากจากการรันวิธีหาค่าที่เหมาะสมที่สุดเพียงครั้งเดียว การออกแบบอัตโนมัติโครงสร้าง ST3 แสดงให้เห็นถึงข้อได้เปรียบในประเด็นที่ว่าผู้ออกแบบที่มีประสบการณ์ในการออกแบบน้อยก็สามารถออกแบบโครงสร้างได้ดีเท่าเทียมกับวิศวกรผู้ออกแบบที่มีประสบการณ์สูงได้ หากมีความรู้พื้นฐานทางด้านโครงสร้างและการหาค่าที่เหมาะสมที่สุดดีพอ



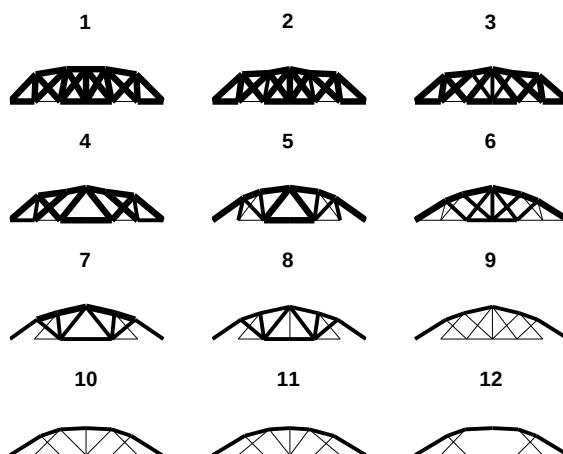
รูปที่ 7.14 ขอบหน้าพาเรโตของ ST3 ปัญหา P2



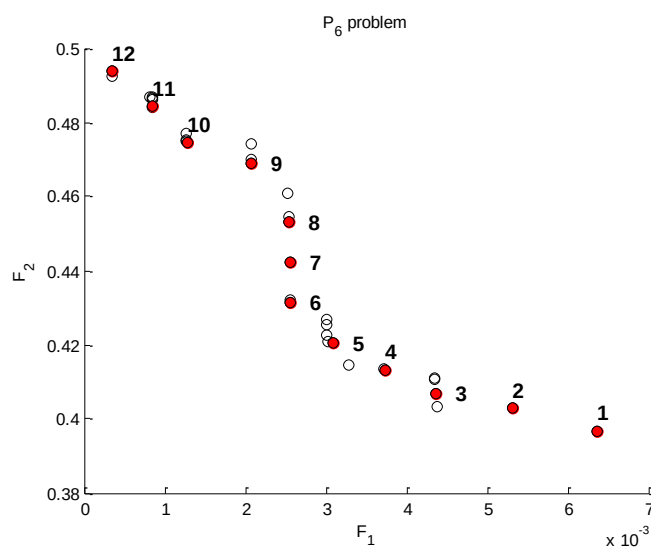
รูปที่ 7.15 โครงสร้างจาก รูปที่ 7.14



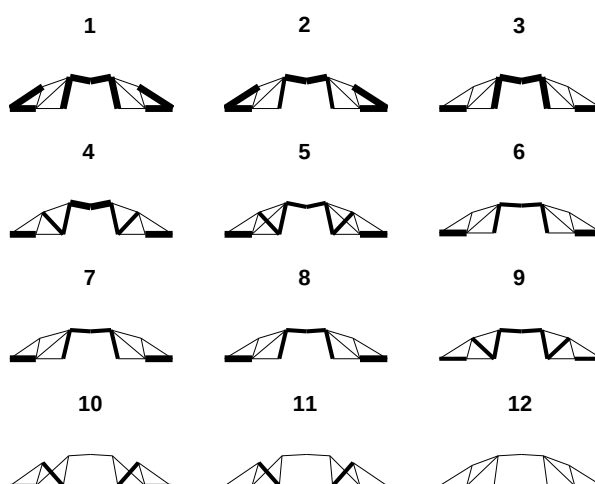
รูปที่ 7.16 ขอบหน้าพาเรโตของการออกแบบโครงสร้าง ST3 ปัญหาการออกแบบ P4



รูปที่ 7.17 โครงสร้างจาก รูปที่ 7.16



รูปที่ 7.18 ขอบหน้าพาเรโตของการออกแบบโครงสร้าง ST3 ปัญหาการออกแบบ P6



รูปที่ 7.19 โครงสร้างจาก รูปที่ 7.18

7.5 สรุปและวิจารณ์

ในการศึกษานี้แสดงให้เห็นถึง การประยุกต์ใช้ขั้นตอนวิธีพัฒนาการแบบหลายเป้าหมาย สำหรับการออกแบบโครงขั้หมุนอย่างมีประสิทธิภาพ ผลจากการเปรียบเทียบกับค่า C และ C' แสดงให้เห็นว่า (สำหรับวิธีที่ใช้ในการศึกษานี้) BPBIL เหมาะสำหรับการออกแบบที่มีจำนวนตัวแปร ออกแบบน้อย ในกรณีที่ปัญหาการออกแบบมีจำนวนตัวแปรออกแบบมาก RSPEA จะเป็นวิธีที่ดีที่สุด การประเมินสมรรถนะของ MOEAs ควรใช้ตัวชี้วัดมากกว่า 1 ตัว เพื่อความแม่นยำในการประเมิน

บทที่ 8 การออกแบบโครงสร้างด้วยการหาค่าที่เหมาะสมที่สุดแบบตรงข้าม

8.1 ความนำ

เนื้อหาในบทนี้นำเสนอการออกแบบโครงสร้างอีกรูปแบบหนึ่ง ซึ่งพิจารณาความไม่แน่นอน (uncertainties) ในปัญหาการออกแบบด้วย กระบวนการออกแบบลักษณะนี้เรียกอีกนัยหนึ่งว่า การออกแบบสำหรับความน่าเชื่อถือ (design for reliability) โดยทั่วไป ปัญหาการออกแบบโครงสร้างหรือโครงสร้างรูปแบบอื่นๆ มักจะตั้งค่าฟังก์ชันเป้าหมายเป็นมวลของโครงสร้าง โดยมีเงื่อนไขบังคับคือ โครงสร้างต้องไม่เสียหายเนื่องจากแรงกระทำที่ผู้ออกแบบกำหนดเอาไว้ ในทางปฏิบัติปัญหาที่มักเจอบ่อยๆ คือ เราไม่สามารถระบุค่าที่ถูกต้องของคุณสมบัติของวัสดุที่ใช้ทำโครงสร้างได้อย่างชัดเจน นอกจากนี้ เรายังไม่สามารถระบุขนาดและทิศทางของแรงกระทำได้อย่างชัดเจน ความไม่แน่นอนของปริมาณเหล่านี้ ส่งผลให้ผลการออกแบบมีความน่าเชื่อถือน้อยลง หากผู้ออกแบบตัดสินใจเลือกค่าเหล่านั้นเป็นตัวเลขที่ชัดเจนด้วยการสุ่ม

การแก้ปัญหานี้สามารถทำได้หลายวิธี เช่น การเพิ่มค่าของความปลอดภัย (factor of safety) เข้าไปในเงื่อนไขบังคับของปัญหาการออกแบบ หรืออีกวิธีหนึ่งคือการกำหนดฟังก์ชันเป้าหมายที่สองเพื่อใช้ในการเพิ่มความน่าเชื่อถือของโครงสร้าง ดังแสดงในบทที่ 6 และ 7 นอกเหนือไปจากนี้ เราสามารถประยุกต์ใช้วิธีการออกแบบสำหรับความน่าเชื่อถือเช่นวิธี จุดที่เป็นไปได้มากที่สุดของความเสียหาย (most probable point of failure) เป็นต้น นอกเหนือจากนี้เราสามารถประยุกต์ใช้วิธีการหาค่าที่เหมาะสมที่สุดแบบตรงข้าม (anti-optimisation, AO) เพื่อหาผลเฉลยของปัญหาการออกแบบลักษณะนี้

เนื้อหาในบทนี้เป็นการสาธิตการออกแบบที่เหมาะสมที่สุดแบบตรงข้ามของโครงข้อหมุน โดยพิจารณาความไม่แน่นอนในคุณสมบัติวัสดุที่ใช้ก่อสร้างโครงสร้าง และความไม่แน่นอนของขนาดและทิศทางของแรงกระทำ ผู้วิจัยได้ทำการพัฒนาขั้นตอนวิธีวิวัฒนาการเรียกว่า two-level PBIL เพื่อใช้ในการหาผลเฉลยของการหาค่าที่เหมาะสมที่สุดแบบตรงข้าม ผลการออกแบบของโครงสร้างกรณีศึกษาได้แสดงให้เห็นว่าวิธีหาค่าที่เหมาะสมที่สุดที่คิดค้นขึ้นนี้มีสมรรถนะสูง สามารถใช้หาผลเฉลยของปัญหาการออกแบบรูปแบบนี้ได้อย่างมีประสิทธิภาพ

8.2 ปัญหาการหาค่าที่เหมาะสมที่สุดแบบตรงข้าม

ปัญหาการออกแบบทั่วไปของการหาค่าที่เหมาะสมที่สุดของโครงสร้าง สามารถเขียนได้ดังนี้

$$\begin{aligned} & \min_{\mathbf{x}} f(\mathbf{x}) \\ & \text{Subject to} \\ & g_i(\mathbf{x}) \leq 0, i = 1, \dots, m \end{aligned} \tag{8.1}$$

เมื่อ $\mathbf{x}$ คือ เวกเตอร์การออกแบบขนาด $n_1 \times 1$

f คือ ค่าฟังก์ชันเป้าหมายโดยทั่วไปคือมวลของโครงสร้าง

g_i คือ เงื่อนไขบังคับแบบอสมการที่กำหนดเพื่อให้โครงสร้างมีความปลอดภัยภายใต้ภาระกระทำ

เมื่อมีความไม่แน่นอนในปัญหาการออกแบบ โจทย์การหาค่าเหมาะที่สุดของ โครงถักที่มีความไม่แน่นอนในระบบ สามารถเขียนได้ดังนี้

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}, \mathbf{y}) \\ \text{Subject to} \quad & g_i(\mathbf{x}, \mathbf{y}) \leq 0, i = 1, \dots, m \end{aligned} \quad (8.2)$$

เมื่อ $\mathbf{y}$ คือ เวกเตอร์ของตัวแปรที่มีความไม่แน่นอนขนาด $n_2 \times 1$ มีค่ากระจายอยู่ในช่วง $\mathbf{y} \in [\mathbf{y}_l, \mathbf{y}_u]$ จากปัญหาการออกแบบ (8.2) เพื่อทำการหาผลเฉลยด้วยวิธีหาค่าเหมาะที่สุดแบบตรงข้าม เราสามารถแยกปัญหาการออกแบบออกเป็นสองปัญหาย่อยคือ ปัญหาแรกคือการหาค่าเหมาะที่สุด (8.1) ซึ่งเป็นปัญหาการออกแบบปกติ ส่วนปัญหาย่อยที่สองคือ การหาค่าเหมาะที่สุดที่มี $\mathbf{y}$ ตัวแปรออกแบบและมีค่าฟังก์ชันเป้าหมายเป็น

$$\max_{\mathbf{y}} F(\mathbf{y}); \mathbf{y} \in [\mathbf{y}_l, \mathbf{y}_u] \quad (8.3)$$

เมื่อ F คือ ฟังก์ชันที่บ่งชี้สถานการณ์ที่เลวร้ายที่สุดของโครงสร้างภายใต้ตัวแปร $\mathbf{y}$ เพื่อความง่ายในการดำเนินการ เราสามารถกำหนดให้ฟังก์ชันเป้าหมายที่สองคือ

$$F(\mathbf{y}) = \max(g_i(\mathbf{x}, \mathbf{y})) \quad (8.4)$$

ทั้งนี้ในปัญหาการออกแบบย่อยที่สองนี้ จะไม่มีการเปลี่ยนแปลงค่าของเวกเตอร์ตัวแปรออกแบบดั้งเดิม $\mathbf{x}$ จากปัญหาการหาค่าเหมาะที่สุดแบบตรงข้าม เราสามารถกล่าวได้ว่าเป็นการหาค่าเหมาะที่สุดแบบสองระดับ (two-level optimisation) กล่าวคือเราหาค่าเหมาะที่สุดจากปัญหา (8.3) จากนั้นเซตคำตอบจาก (8.3) จะเป็น feasible region ให้ปัญหาการออกแบบ (8.1) การหาผลเฉลยของ (8.3) อาจเรียกว่าเป็นการหาผลเฉลยที่ทำให้สถานการณ์ของโครงสร้างเลวร้ายที่สุด (worst case scenario) ก่อนที่จะนำผลที่ได้นี้ไปหาผลเฉลยด้วย (8.1) เป็นระดับที่สอง ด้วยเหตุที่ผลเฉลยของระดับที่หนึ่งและสองมีความขัดแย้งกันเสมอ จึงนิยมเรียกกระบวนการออกแบบลักษณะนี้ว่า การหาค่าเหมาะที่สุดแบบตรงข้าม

ในการพัฒนาขั้นตอนวิธีวิวัฒนาการสำหรับหาผลเฉลยของการออกแบบชนิดนี้ เราสามารถแบ่งกระบวนการค้นหาคำตอบของ EAs ออกเป็นสองส่วนคือ ส่วนในการหาคำตอบของ (8.1) และ (8.2) ซึ่งมีแนวทางการดำเนินการได้หลายรูปแบบ ในบทนี้จะนำเสนอการดัดแปลง PBIL เป็น 2-level PBIL เพื่อหาผลเฉลยเหมาะที่สุดของ (8.2)

8.3 วิธี PBIL แบบสองชั้นสำหรับการหาค่าเหมาะที่สุดแบบตรงข้าม

วิธี PBIL เป็นวิธีหาค่าเหมาะที่สุดเชิงวิวัฒนาการที่ประยุกต์ใช้ขั้นตอนวิธีการประมาณการกระจาย (estimation of distribution algorithm) อย่างง่าย เป็นวิธีที่ประยุกต์ใช้แนวของเลขฐานสอง แทนเวกเตอร์ตัวแปรออกแบบเช่นเดียวกับ GA อย่างไรก็ตาม PBIL จะไม่ใช่ประชากรของผลเฉลย เลขฐานสองในการหาค่าเหมาะที่สุดเหมือน GA แต่จะใช้เวกเตอร์ความน่าจะเป็น (probability vector) เวกเตอร์ความน่าจะเป็น คือเวกเตอร์ที่มีขนาดเท่ากับขนาดของแนวตัวเลขฐานสองที่ใช้แทนเวกเตอร์ตัวแปรออกแบบ สำหรับตัวอย่างของเวกเตอร์ของความน่าจะเป็น กำหนดให้เวกเตอร์ของตัวแปรออกแบบถูกแทนด้วยแนวของเลขฐานสองที่มีสมาชิก 4 ตัว และในชุดประชากรเลขฐานสอง มีสมาชิกทั้งหมด 4 ผลเฉลย ดังนั้นเมตริกซ์ของประชากรเลขฐานสองจะมีขนาดเป็น 4×4 เมื่อแต่ละแถวของเมตริกซ์คือผลเฉลยของปัญหาการออกแบบ จากนั้นเวกเตอร์ความน่าจะเป็น เวกเตอร์จะมีขนาดเป็น 1×4 เมื่อสมาชิกตัวที่ i -th ของเวกเตอร์คือความน่าจะเป็นในการมีเลข “1” ในแนวคอลัมน์ที่ i -th ของเมตริกซ์ของประชากรเลขฐานสองดังแสดงใน รูปที่ 8.1 จากรูปเราสามารถสรุปได้ว่า เวกเตอร์ของความน่าจะเป็นหนึ่งเวกเตอร์สามารถสร้างเป็นประชากรเลขฐานสองได้หลากหลายรูปแบบ และนี่คือจุดแข็งของวิธี PBIL

กระบวนการค้นหาคำตอบของ PBIL แสดงในขั้นตอนวิธีที่ 8.1 PBIL เริ่มต้นกระบวนการด้วยเวกเตอร์ของความน่าจะเป็นที่มีสมาชิกทั้งหมดมีค่าเป็น $P_i = 0.5$ จากนั้นสร้างประชากรเลขฐานสองจากเวกเตอร์นี้ เมื่อได้ผลเฉลยที่ดีที่สุด $\mathbf{b}$ ใช้ผลเฉลยเลขฐานสองนี้ในการปรับปรุงค่าของสมาชิกในเวกเตอร์ของความน่าจะเป็นดังนี้

$$P_i^{new} = (1 - L_R)P_i^{old} + L_R b_i \quad (8.5)$$

เมื่อ P_i^{old} คือเอลิเมนต์ของเวกเตอร์ความน่าจะเป็นในวงวนก่อนหน้า P_i^{new} คือค่าของเอลิเมนต์ของเวกเตอร์ความน่าจะเป็นหลังจากปรับปรุงค่า และ b_i คือเอลิเมนต์ของ $\mathbf{b}$ ค่า L_R เรียกว่าอัตราการเรียนรู้ (learning rate) ซึ่งในที่นี้คือเลขที่ได้จากการสุ่ม (ทุกครั้งที่มีการใช้งาน) แบบ uniform ในช่วง $[0.4, 0.6]$ ในการศึกษาจะไม่มีการเกิดมิวเทชันในวิธี PBIL แต่ค่าของ P_i จะถูกปิดให้อยู่ในช่วงของ $[0.1, 0.9]$ หลังจากการประยุกต์ใช้ (8.5) เพื่อป้องกันการลู่เข้าก่อนเวลาอันควร กระบวนการคำนวณจะดำเนินการไปจนกระทั่งโปรแกรมถูกสั่งให้หยุดค้นหาผลเฉลยเหมาะที่สุด ซึ่งผลที่ได้สุดท้ายคือเวกเตอร์ $\mathbf{b}$ นั่นเอง

Population 1				Population 2				Population 3			
0	0	1	1	1	1	1	0	0	0	0	1
1	1	1	0	0	1	0	1	1	1	0	0
0	1	0	1	1	0	1	1	0	0	0	1
1	0	0	0	0	0	0	0	0	1	0	1
Probability vectors											
[0.5,0.5,0.5,0.5]				[0.5,0.5,0.5,0.5]				[0.25,0.5,0,0.75]			

รูปที่ 8.1 เวกเตอร์ของความน่าจะเป็น

ขั้นตอนวิธีที่ 8.1 วิธี PBIL

Initialisation $P_i = 0.5$, $k = 0$, $\mathbf{b} = \{\}$

Main procedure

1. Generate a binary population $\mathbf{B}$ according to P_i .
2. Perform function evaluations $f(\mathbf{B})$ and find the best solution $\mathbf{b}$ from $\mathbf{B} \cup \mathbf{b}$.
3. Update P_i using equation (5).
4. Set P_i to $[0.1, 0.9]$ in cases that it is out of the interval.
5. If the termination condition is met, stop the procedure; otherwise, set $k = k+1$ and go to 1.

ในกรณีของ PBIL สำหรับการหาค่าเหมาะที่สุดแบบตรงข้าม AO แสดงในขั้นตอนวิธีที่ 8.2 ซึ่งประกอบด้วยขั้นตอนกันสองกลุ่ม เนื่องจาก PBIL ใช้แนวของเลขฐานสอง $\mathbf{b}$ แทนเวกเตอร์ตัวแปร ออกแบบ ในที่นี้จะแบ่ง $\mathbf{b}$ ออกเป็นสองส่วนคือ $\mathbf{b}_x$ สำหรับตัวแปรออกแบบขั้นต้น $\mathbf{x}$ และ $\mathbf{b}_y$ สำหรับตัวแปรที่มีความไม่แน่นอน $\mathbf{y}$ จากนั้น ทำการสร้างประชากรเลขฐานสองเริ่มต้นด้วยการสุ่ม กำหนดให้แนวเลขฐานสองที่ดีที่สุดตามปัญหา (8.1) เป็นเวกเตอร์ $\mathbf{b}$ และผลเฉลยเลขฐานสองที่ให้ค่าฟังก์ชัน (8.3) แย่ที่สุดคือ $\mathbf{w} = \{\mathbf{w}_x \mathbf{w}_y\}$ PBIL นี้ให้เวกเตอร์ของความน่าจะเป็นสองชุดสำหรับ $\mathbf{x}$ และ $\mathbf{y}$ นั่นคือ $\mathbf{P}_x$ และ $\mathbf{P}_y$ ตามลำดับ นั้นหมายความว่าเซตของผลเฉลยเลขฐานสองสำหรับ $\mathbf{x}$ จะสร้างจาก $\mathbf{P}_x$ และเขียนแทนด้วย $\mathbf{B}_x$ ทั้งนี้ เมื่อทำการหาค่าฟังก์ชันจะแทนส่วนของ $\mathbf{y}$ ด้วย $\mathbf{w}_y$ ตัวแปรที่ดีที่สุดจาก $\mathbf{B}_x$ และ $\mathbf{w}_y$ เขียนแทนด้วย $\mathbf{b}$ ซึ่งประกอบด้วย $\mathbf{b}_x$ และ $\mathbf{b}_y$ จากนั้นใช้ $\mathbf{b}_x$ ในการปรับปรุงค่าของ $\mathbf{P}_x$ ด้วยสมการ (8.5)

ลูกภายในของ PBIL (ขั้นตอนที่ 7-13) เป็นการค้นหาตัวแปร $\mathbf{y}$ ที่ให้สถานการณ์ที่แย่ที่สุดของโครงสร้าง เริ่มต้นด้วยการสร้างเซตของผลเฉลยเลขฐานสองสำหรับตัวแปร $\mathbf{y}$ ซึ่งเขียนแทนด้วย $\mathbf{W}_y$ จากเวกเตอร์ $\mathbf{P}_y$ ในการหาค่าฟังก์ชันจะใช้ $\mathbf{w}_x$ เป็นส่วนของ $\mathbf{x}$ จากนั้นหาผลเฉลยที่ให้ค่าที่แย่ที่สุดตามฟังก์ชันใน (8.3) กำหนดให้ $\mathbf{w}$ คือผลเฉลยนั้น จากนั้นใช้ $\mathbf{w}_y$ ซึ่งเป็นส่วนหลังของ $\mathbf{w}$ ในการปรับปรุง $\mathbf{P}_y$ ตามสมการ (8.5) ลูกภายในจะหยุดทำงานเมื่อวนครบรอบ N_I ลูกภายในที่ผู้ใช้กำหนด ในทำนองเดียวกันการรันของ PBIL นี้จะหยุดเมื่อจำนวนลูกภายในนอกกรอบ N_O รอบ กำหนดให้ N_P คือขนาดประชากร ดังนั้น ในการรัน 1 ครั้งจะใช้จำนวนครั้งในการวิเคราะห์โครงสร้างทั้งหมด $N_O(N_I + 1)N_P$ ครั้ง

ขั้นตอนวิธีที่ 8.2 PBIL สำหรับการหาค่าเหมาะที่สุดแบบตรงข้าม

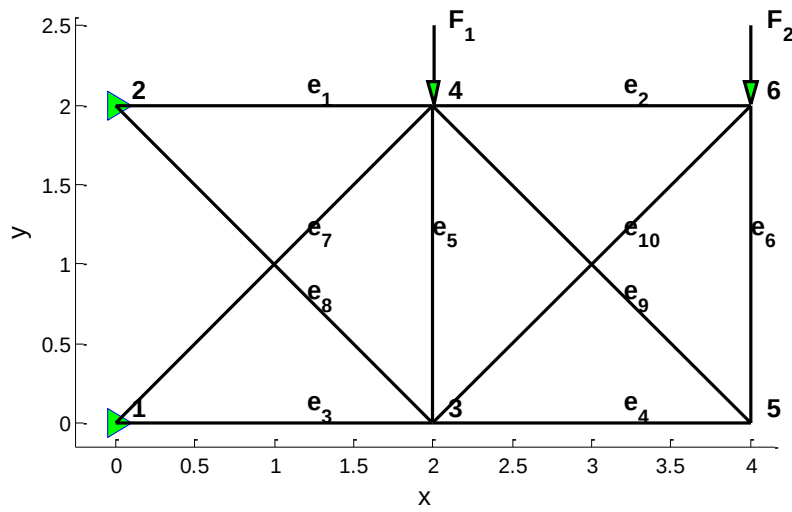
Initialisation $P_{x,i} = 0.5$, $P_{y,i} = 0.5$, and find $\mathbf{b} = \{\mathbf{b}_x \mathbf{b}_y\}$ and $\mathbf{w} = \{\mathbf{w}_x \mathbf{w}_y\}$ from an initial population.

Main procedure

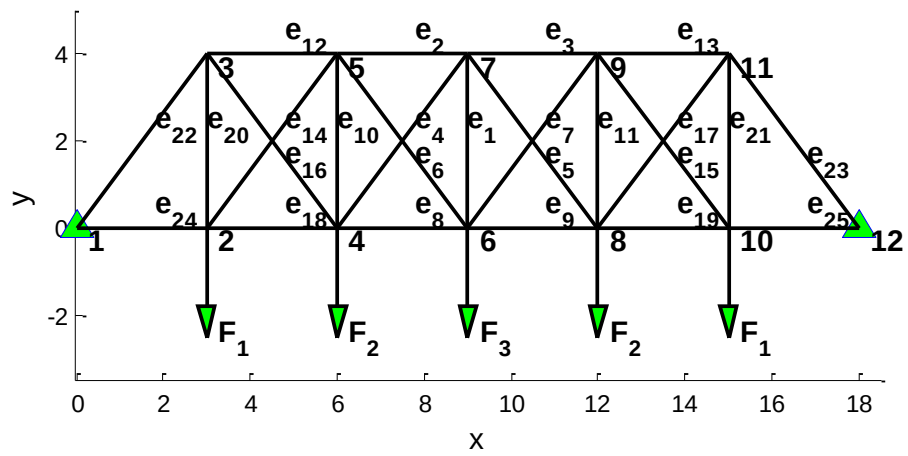
1. For $k = 1$ to N_O (outer loop)
2. Generate a binary population $\mathbf{B}_x$ according to $P_{x,i}$ and assign $\mathbf{w}_y$ as the second part of every solution in $\mathbf{B}_x$. The population is $\mathbf{B} = [\mathbf{B} \mathbf{w}_y]$.
3. Perform function evaluations $f(\mathbf{B})$.
4. Find the new best solution $\mathbf{b}$ from $\mathbf{B} \cup \mathbf{b}$ based on the objective function in (1).
5. Update $P_{x,i}$ with $\mathbf{b}_x$ using equation (8.5).
6. Set $P_{x,i}$ to $[0.1, 0.9]$ in cases that it is out of the interval.
7. For $j = 1$ to N_I (inner loop)
 8. Generate a binary population $\mathbf{W}_y$ according to $P_{y,i}$ and assign $\mathbf{w}_x$ as the first part of every solution in $\mathbf{W}_y$. The population is $\mathbf{W} = [\mathbf{w}_x \mathbf{W}_y]$.
 9. Perform function evaluations $f(\mathbf{W})$
 10. Find the new worst solution $\mathbf{w}$ from $\mathbf{W} \cup \mathbf{w}$ based on the objective function in (3).
 11. Update $P_{y,i}$ with $\mathbf{w}_y$ using equation (8.5).
 12. Set $P_{y,i}$ to $[0.1, 0.9]$ in cases that it is out of the interval.
13. Next j
14. Next k

8.4 การออกแบบโครงถักกรณีศึกษา

ในการศึกษานี้ ใช้ปัญหาการออกแบบ 4 ปัญหาในการทดสอบวิธีที่คิดค้นขึ้นมาใหม่นี้ โดยทั้ง 4 ปัญหาการออกแบบกำหนดมาจากการออกแบบโครงข้อหมุนสองชุดคือ 10-bar truss แสดงในรูปที่ 8.2 เป็นโครงถักสองมิติประกอบด้วยจุดต่อ 6 จุด และเอลิเมนต์ 10 ชิ้น โดยที่จุดต่อ 1 และ 2 ถูกยึดกับพื้น ส่วนจุดต่อ 4 และ 6 รับแรงในแนวตั้ง F_1 และ F_2 ตามลำดับ โครงสร้างที่สองเรียกว่า 25-bar truss ประกอบด้วย 16 จุดต่อและ 25 เอลิเมนต์ เมื่อจุดต่อ 1 และ 12 ถูกยึดกับพื้นและมีแรง $F_1 F_2 F_3 F_2$ และ F_1 กระทำที่จุดต่อ 2 4 6 8 และ 10 ตามลำดับ ปัญหาการออกแบบทั้งสี่มีรายละเอียดดังนี้



รูปที่ 8.2 10-bar truss



รูปที่ 8.3 25-bar truss

AOP1: sizing optimisation of the 10-bar truss

$$\min_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}), \max_{\mathbf{y}} F(\mathbf{x}, \mathbf{y}) \quad (8.6)$$

Subject to

$$\sigma_{\max}(\mathbf{x}, \mathbf{y}) - \sigma_{yt} \leq 0, \text{ N/m}^2$$

$$P_i - P_{cr} \leq 0, \text{ N}$$

$$U_{max} - 0.005 \leq 0, m$$

$$x_i = d_i \in \{0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10, 0.20\}, m$$

$$\{75, 150\}^T \leq \mathbf{y} = \{F_1, F_2\}^T \leq \{80, 175\}^T, N$$

เมื่อ d_i คือ ขนาดเส้นผ่าศูนย์กลางของเอลิเมนต์และ U_{max} คือค่าการกระจัดในแนวดิ่งสูงสุดที่เกิดขึ้น ตัวแปรบอกขนาดของเอลิเมนต์จะกำหนดเป็นตัวแปรแบบดิสครีตเพื่อให้ใกล้เคียงกับปัญหาการออกแบบจริง แรงกระทำ F_1 และ F_2 ถูกกำหนดเป็นตัวแปรที่มีความไม่แน่นอน ดังนั้น ปัญหาการออกแบบคือการหา $\mathbf{x}$ ที่ดีที่สุดและในขณะเดียวกันก็หา F_1 และ F_2 ที่ทำให้โครงสร้างอยู่ในสถานการณ์ที่แย่ที่สุด

AOP2: *sizing optimisation of the 25-bar truss*

$$\min_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}), \max_{\mathbf{y}} F(\mathbf{x}, \mathbf{y}) \quad (8.7)$$

Subject to

$$\sigma_{max}(\mathbf{x}, \mathbf{y}) - \sigma_{yt} \leq 0, N/m^2$$

$$P_i - P_{cr} \leq 0, N$$

$$U_{max} - 0.005 \leq 0, m$$

$$x_i \in \{0.06, 0.07, 0.08, 0.09, 0.10, 0.15, 0.20, 0.25\}m; i = 1, \dots, 13$$

$$\{150, 150, 150, 199.00 \times 10^9\}^T \leq \mathbf{y} = \{F_1, F_2, F_3, E\}^T \leq \{300, 300, 300, 201.00 \times 10^9\}^T.$$

ในกรณีนี้ เราเพิ่มค่ามอดูลัสของยังเป็นตัวแปรที่มีความไม่แน่นอนเพิ่มเติมจากส่วนของแรงกระทำ เพื่อให้เกิดความสมมาตร ตัวแปรกำหนดขนาดของเอลิเมนต์ของโครงถัก มีดังนี้

- x_1 for d_1
- x_2 for d_2 and d_3
- x_3 for d_4 and d_5
- x_4 for d_6 and d_7
- x_5 for d_8 and d_9
- x_6 for d_{10} and d_{11}
- x_7 for d_{12} and d_{13}
- x_8 for d_{14} and d_{15}
- x_9 for d_{16} and d_{17}
- x_{10} for d_{18} and d_{19}
- x_{11} for d_{20} and d_{21}
- x_{12} for d_{22} and d_{23}
- x_{13} for d_{24} and d_{25}

AOP3: *shape and sizing optimisation of the 10-bar truss*

ปัญหาการออกแบบที่สามนี้คือปัญหาการออกแบบ AOP1 โดยเพิ่มตัวแปรออกแบบกำหนดรูปทรงขึ้นมาอีก 4 ตัว ซึ่งก็คือตำแหน่งในแนวนอนและแนวดิ่งของจุดต่อ 3 และ 5 โดยกำหนดขอบเขตของตัวแปรกำหนดรูปทรงเป็น

$$-0.75m \leq x_i \leq 0.75m \text{ for } i = 11, \dots, 14.$$

AOP4: *shape and sizing optimisation of the 25-bar truss*

ปัญหาการออกแบบที่สี่คือปัญหาการออกแบบ AOP2 โดยเพิ่มตัวแปรออกแบบกำหนดรูปทรงขึ้นมาอีก 3 ตัว ซึ่งก็คือตำแหน่งในแนวดิ่งของจุดต่อ 3 5 7 9 และ 11 โดยที่ x_{14} คือตัวแปรของจุดต่อ 3

และ 11 โดยที่ x_{15} คือตัวแปรของจุดต่อ 5 และ 9 โดยที่ x_{16} คือตัวแปรของจุดต่อ 7 โดยกำหนดขอบเขตของตัวแปรกำหนดรูปทรงเป็น $-3.00 \text{ m} \leq x_i \leq 3.00 \text{ m}$ สำหรับ $i = 14, \dots, 16$

วิธีหาค่าเหมาะที่สุดที่ใช้หาผลเฉลยของปัญหาทั้ง 8 มีดังนี้

- PBILb5: This optimiser uses $\mathbf{b}_x$ from the best solution $\mathbf{b}$ instead of $\mathbf{w}_x$ from the worst solution $\mathbf{w}$ in step 8 of Algorithm 2. N_I is set to be 5.
- PBILw5: This is PBIL detailed in Algorithm 2 where N_I is set to be 5.
- PBILw5m: This is PBILw5 with GA mutation. Having generated binary populations in steps 2 and 8, The GA mutation is applied to the populations where each solution has the probability of being mutated as 0.25.
- PBILw1: This is PBIL detailed in Algorithm 2 where N_I is set to be 1.
- PBILw10: This is PBIL detailed in Algorithm 2 where N_I is set to be 10.

ขนาดของประชากรเลขฐานสองคือ 1000 ซึ่งหมายความว่า ในการรันแต่ละครั้งจะทำการวิเคราะห์โครงสร้าง 15,000 ครั้ง ใช้วิธี fuzzy set ในการแปลงเงื่อนไขบังคับไปเป็นส่วนหนึ่งของค่าฟังก์ชันเป้าหมาย สำหรับแต่ละปัญหาการออกแบบ จะใช้วิธีหาค่าเหมาะที่สุดใดๆหาคำตอบ 30 ครั้งเพื่อสามารถเปรียบเทียบสมรรถนะของวิธีเหล่านี้ด้วยวิธีการทางสถิติได้ นอกจากนี้ ยังได้ทำการประยุกต์ใช้ HybridGA ซึ่งเป็นวิธีที่มีอยู่ในวรรณกรรมเพื่อใช้วัดสมรรถนะของ PBIL ที่นำเสนอใหม่นี้

8.5 ผลการออกแบบเหมาะที่สุด

ผลการออกแบบของปัญหา AOP1 แสดงในตารางที่ 8.1 สำหรับปัญหาการออกแบบนี้ เราได้ทำการกำหนดปัญหาเพื่อให้ได้คำตอบว่าสถานการณ์ที่เลวร้ายที่สุด (worst case) ที่จะเกิดกับโครงสร้างคือเมื่อค่าของแรงกระทำมีค่าเป็น $\{F_1, F_2\}^T = \{80, 175\}^T \text{ N}$ จากตาราง จะเห็นได้ว่า HybridGA กับ PBIL5 ไม่สามารถหาค่าแรงที่ทำให้เกิดสถานะเลวร้ายที่สุด ในขณะที่ขั้นตอนวิธี PBIL อื่นที่ใช้ $\mathbf{w}_x$ ในการปรับปรุงค่าเวกเตอร์ความน่าจะเป็นในขั้นตอนที่ 8 ของขั้นตอนวิธีที่ 8.2 สามารถหาสถานการณ์นี้ได้ทุกวิธี เมื่อพิจารณาค่าฟังก์ชันเป้าหมายพบว่า PBILw1 ให้คำตอบที่ดีที่สุดในบรรดา PBILw1 ไม่สามารถหา worst case ได้ทั้ง 30 ครั้ง ทั้งนี้เป็นเพราะว่า ใช้จำนวนลูบในการหา worst case น้อยเกินไปนั่นเอง ซึ่งตรงข้ามกับ PBILw10 ซึ่งหา worst case ได้ทุกครั้ง แต่หาค่าฟังก์ชันเป้าหมายได้แย่มาก เนื่องจากใช้เวลาส่วนใหญ่ในการหา worst case นั่นเอง จากการเปรียบเทียบ PBILw5 และ PBILw5m พบว่า การเพิ่มมิวเทชันของ GA ในวิธี PBIL ไม่มีผลต่อสมรรถนะในการหาผลเฉลยมากนัก

ผลการออกแบบของปัญหา AOP2 แสดงในตารางที่ 8.2 สำหรับปัญหาการออกแบบนี้ ค่าของ worst case คือ $\{F_1, F_2, F_3, E\}^T = \{1000, 1000, 1000, 199.0 \times 10^9\}^T$ จากตารางจะเห็นได้ว่า HybridGA กับ PBIL5 ไม่สามารถหาค่าแรงที่ทำให้เกิดสถานะเลวร้ายที่สุด ในขณะที่ขั้นตอนวิธี PBIL อื่น สามารถหา worst case ได้ทุกวิธี เมื่อพิจารณาค่าฟังก์ชันเป้าหมายพบว่า PBILw1 ให้คำตอบที่ดีที่สุดในบรรดา PBILw5 ให้คำตอบดีกว่า PBILw5m ผลการออกแบบของ AOP3 แสดงในตารางที่

8.3 ซึ่งได้ผลสรุปคล้ายกันกับสองกรณีแรก ในปัญหานี้ PBILw ทุกวิธีสามารถหา worst case ได้ทั้ง 30 ครั้ง ผลการเปรียบเทียบของปัญหา AOP4 แสดงในตารางที่ 8.4 PBILw1 ให้คำตอบที่ดีที่สุดและ PBILw5 ให้คำตอบดีกว่า PBILw5m เล็กน้อย ตารางที่ 8.5 แสดงจำนวนครั้งที่วิธีหาค่าเหมาะที่สุดแบบต่างๆ สามารถหา worst case ได้จากการรัน 30 ครั้ง ซึ่งจะเห็นได้ว่าวิธี PBILw เป็นวิธีการเดียวที่สามารถหา worst case ได้และอยู่ในเกณฑ์ที่สามารถยอมรับได้

ผลเฉลี่ยที่ดีที่สุดที่ได้จาก PBILw5 PBILw5m PBILw1 และ PBILw10 จากการรัน 30 ครั้งของปัญหา AOP1 AOP2 AOP3 และ AOP4 แสดงในรูปที่ 8.4 8.5 8.6 และ 8.7 ตามลำดับ จากรูปจะเห็นว่า โครงสร้างที่ได้มีรูปทรงและขนาดใกล้เคียงกัน โดยที่ผลเฉลี่ยจาก PBILw1 มีค่าดีกว่าผลเฉลยชุดอื่นเล็กน้อย นั่นแสดงว่าการกำหนดค่า $N_I=1$ หรือการวนลูปภายใน 1 รอบเพื่อหา worst case เพียงพอที่จะหาคำตอบได้สำหรับวิธี PBILw

ตารางที่ 8.3 Optimum results of AOP4

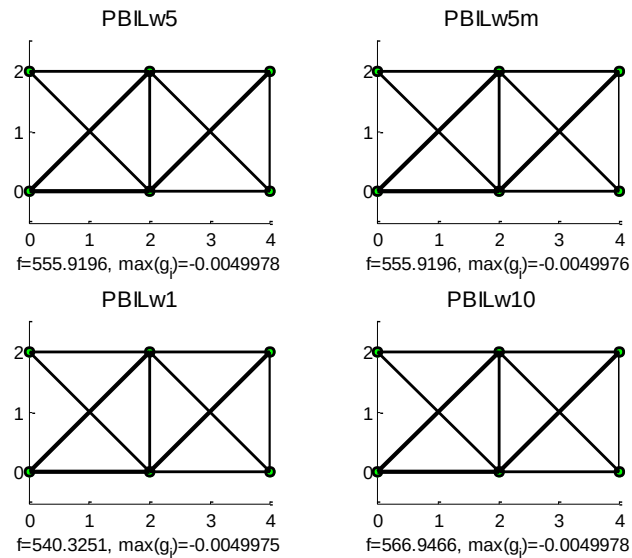
MHs	HybridGA		PBILb5		PBILw5		PBILw5m		PBILw1		PBILw10	
	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev
x_1	0.0467	0.0084	0.0517	0.0053	0.0520	0.0041	0.0513	0.0043	0.0490	0.0031	0.0533	0.0084
x_2	0.0477	0.0107	0.0510	0.0040	0.0503	0.0018	0.0513	0.0035	0.0507	0.0045	0.0543	0.0082
x_3	0.0833	0.0048	0.0867	0.0055	0.0863	0.0067	0.0867	0.0048	0.0860	0.0050	0.0873	0.0069
x_4	0.0603	0.0049	0.0577	0.0082	0.0607	0.0098	0.0577	0.0068	0.0563	0.0076	0.0610	0.0099
x_5	0.0570	0.0070	0.0543	0.0068	0.0590	0.0092	0.0533	0.0071	0.0553	0.0078	0.0583	0.0105
x_6	0.0613	0.0043	0.0603	0.0072	0.0617	0.0079	0.0603	0.0093	0.0577	0.0043	0.0653	0.0107
x_7	0.0720	0.0130	0.0783	0.0046	0.0773	0.0058	0.0797	0.0056	0.0760	0.0081	0.0793	0.0087
x_8	0.0457	0.0082	0.0507	0.0037	0.0500	0.0037	0.0507	0.0037	0.0490	0.0031	0.0523	0.0082
x_9	0.0440	0.0056	0.0510	0.0031	0.0507	0.0025	0.0497	0.0032	0.0483	0.0038	0.0507	0.0045
x_{10}	0.0787	0.0035	0.0787	0.0051	0.0797	0.0041	0.0770	0.0047	0.0780	0.0048	0.0810	0.0055
x_{11}	0.0518	0.1773	0.3132	0.2705	0.1478	0.3773	0.3319	0.3448	0.2337	0.3527	0.2345	0.4409
x_{12}	0.5991	0.1766	0.5610	0.1850	0.6211	0.1079	0.5072	0.2017	0.6345	0.1444	0.5174	0.1896
x_{13}	-0.6919	0.1381	-0.5828	0.1442	-0.5369	0.2001	-0.4271	0.2176	-0.6712	0.1039	-0.4691	0.2160
x_{14}	0.6804	0.1490	0.5697	0.1723	0.5550	0.1935	0.5924	0.1489	0.6494	0.1410	0.5549	0.1754
y_1	75.3793	1.2182	78.1703	1.5065	80.0000	0.0000	80.0000	0.0000	80.0000	0.0000	80.0000	0.0000
y_2	170.9637	7.2167	166.1746	7.4693	175.0000	0.0000	175.0000	0.0000	175.0000	0.0000	175.0000	0.0000
$f(x,y)$	479.4000	34.7275	529.8728	21.6285	535.8320	21.6881	534.5283	22.3252	501.9159	11.7035	574.2029	37.1328
$\max g_i$	-0.0047	0.0009	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000

ตารางที่ 8.4 Optimum results of AOP5

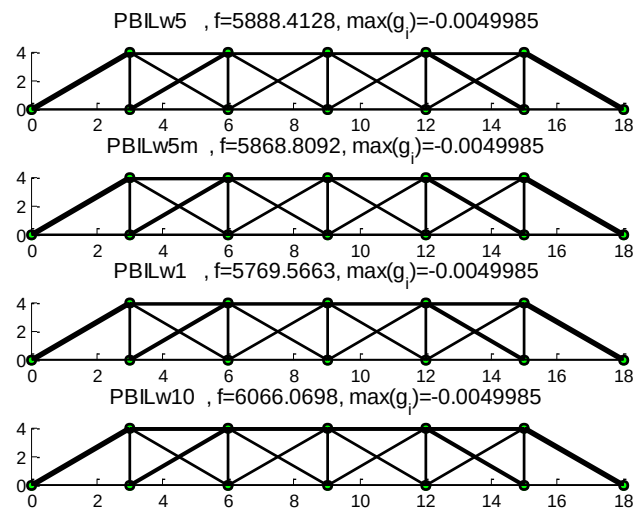
MHs	HybridGA		PBILb5		PBILw5		PBILw5m		PBILw1		PBILw10	
	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev
x_1	0.0787	0.0157	0.0787	0.0082	0.0813	0.0111	0.0800	0.0105	0.0730	0.0065	0.0870	0.0115
x_2	0.1233	0.0286	0.1533	0.0127	0.1483	0.0091	0.1500	0.0131	0.1500	0.0000	0.1577	0.0311
x_3	0.0830	0.0174	0.0780	0.0119	0.0833	0.0175	0.0810	0.0171	0.0717	0.0087	0.0903	0.0227
x_4	0.0770	0.0124	0.0810	0.0212	0.0787	0.0111	0.0747	0.0104	0.0727	0.0087	0.0883	0.0261
x_5	0.0807	0.0141	0.0770	0.0079	0.0803	0.0096	0.0803	0.0110	0.0717	0.0065	0.0847	0.0117
x_6	0.0793	0.0131	0.0740	0.0062	0.0783	0.0099	0.0770	0.0088	0.0710	0.0031	0.0857	0.0205
x_7	0.1200	0.0249	0.1483	0.0091	0.1533	0.0183	0.1567	0.0217	0.1517	0.0091	0.1557	0.0310
x_8	0.0893	0.0187	0.0857	0.0176	0.0840	0.0214	0.0840	0.0181	0.0750	0.0107	0.0963	0.0318
x_9	0.0780	0.0140	0.0807	0.0098	0.0827	0.0120	0.0777	0.0094	0.0747	0.0094	0.0833	0.0163
x_{10}	0.0810	0.0106	0.0783	0.0095	0.0783	0.0105	0.0777	0.0101	0.0713	0.0035	0.0817	0.0162
x_{11}	0.0817	0.0109	0.0773	0.0098	0.0823	0.0205	0.0767	0.0152	0.0707	0.0052	0.0850	0.0242
x_{12}	0.1500	0.0000	0.1533	0.0127	0.1500	0.0000	0.1500	0.0000	0.1500	0.0000	0.1540	0.0234
x_{13}	0.0857	0.0161	0.0823	0.0168	0.0860	0.0203	0.0823	0.0165	0.0737	0.0089	0.0940	0.0302
x_{14}	-1.3764	0.8037	-1.9818	0.7962	-1.8035	0.7824	-1.5169	0.9604	-1.8596	0.7010	-1.5277	1.0779
x_{15}	-0.6342	0.7655	-0.9292	1.0364	-0.6835	1.0393	-0.5185	0.9608	-0.7484	0.9619	-0.5791	1.3451
x_{16}	-0.4087	0.7922	-0.3812	1.1555	-0.3243	1.1576	-0.1742	1.0618	-0.4516	0.9760	-0.3193	1.3226
y_1	194.0161	59.2515	238.8221	48.0552	300.0000	0.0000	300.0000	0.0000	300.0000	0.0000	299.5406	2.5164
y_2	168.5592	35.6225	251.6178	31.7033	300.0000	0.0000	300.0000	0.0000	300.0000	0.0000	296.8964	16.9992
y_3	239.6624	57.7826	231.6178	47.0528	300.0000	0.0000	300.0000	0.0000	300.0000	0.0000	299.4868	2.8109
y_4	1.995E+1 1	6.119E+0 8	1.999E+1 1	5.890E+0 8	1.990E+1 1	0.000E+0 0	1.990E+1 1	0.000E+0 0	1.990E+1 1	0.000E+0 0	1.990E+1 1	1.902E+0 8
$f(x,y)$	5222.3215	813.5479	5440.5355	524.4721	5616.7374	429.4775	5679.4132	654.9014	4938.2933	266.2513	6585.0710	1474.1376
$\max_i g$	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000	68.9978	377.9440

ตารางที่ 8.5 Number of worst cases found by the optimisers

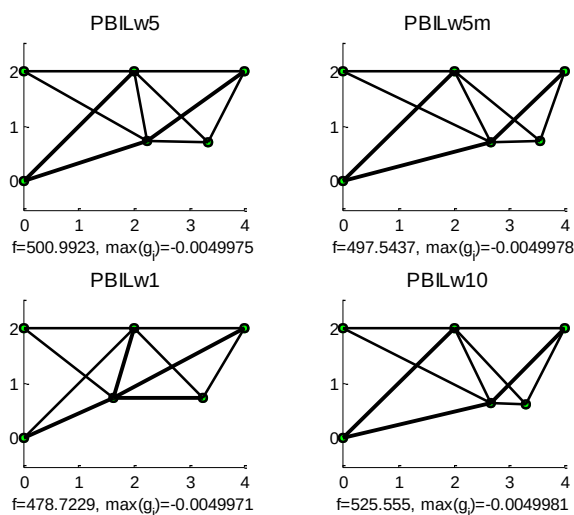
MHs	AOP1	AOP2	AOP3	AOP4
HybridGA	0	0	0	0
PBILb5	0	0	0	0
PBILw5	30	29	30	30
PBILw5m	30	27	30	30
PBILw1	29	24	30	30
PBILw10	30	29	30	29



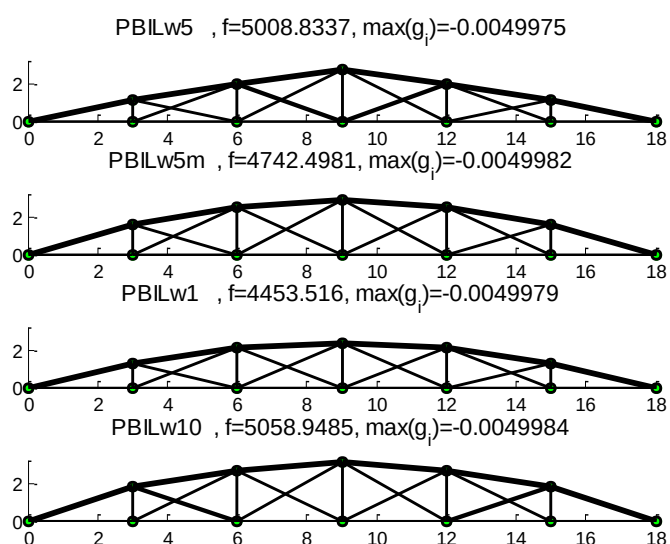
รูปที่ 8.4 ผลเฉลยเหมาะที่สุด AOP1



รูปที่ 8.5 ผลเฉลยเหมาะที่สุด AOP2



รูปที่ 8.6 ผลเฉลยเหมาะที่สุด AOP3



รูปที่ 8.7 ผลเฉลยเหมาะที่สุด AOP4

8.6 สรุปและวิจารณ์

วิธี PBILw ที่พัฒนาขึ้นมาเป็นขั้นตอนวิธีวิวัฒนาการที่มีประสิทธิภาพสูง สามารถใช้ในการหาผลเฉลยของปัญหาการหาค่าเหมาะที่สุดแบบตรงข้ามของโครงถักได้อย่างมีประสิทธิภาพ เป็นวิธีการที่สามารถหา worst case ของปัญหาการออกแบบได้และในขณะเดียวกันก็สามารถหาผลเฉลยเหมาะที่สุดของปัญหาการหาค่าเหมาะที่สุดเบื้องต้น (8.1) ได้พร้อมกันในการรัน 1 รอบ เมื่อเปรียบเทียบกับวิธีที่พัฒนามาก่อน PBILw มีประสิทธิภาพสูงกว่าอย่างมีนัยสำคัญ ส่วนวิธี fuzzy set theory สามารถนำมาใช้แก้ปัญหาเงื่อนไขบังคับในปัญหาการหาค่าเหมาะที่สุดแบบมีเงื่อนไขบังคับได้อย่างมีประสิทธิภาพ

บทที่ 9 การหาค่าเหมาะที่สุดของโครงข่ายขั้นตอนวิธีแบบผสม

9.1 ความนำ

เนื้อหาในบทนี้เป็นตัวอย่างการพัฒนาขั้นตอนวิธีวิวัฒนาการแบบผสมเพื่อการออกแบบโครงข่ายวิธี PBIL ซึ่งปกติจะใช้เลขฐานสองในการหาผลเฉลยจะถูกพัฒนามาใช้เลขจำนวนจริง ดังนั้นจึงเรียกว่า PBIL แบบใช้จำนวนจริงและเขียนแทนด้วย RPBIL จากนั้นนำเสนอการปรับปรุง RPBIL เพื่อหาค่าเหมาะที่สุดแบบหลายเป้าหมาย และทำการผสมระหว่าง RPBIL กับขั้นตอนวิธีวิวัฒนาการเชิงอนุพันธ์ DE เพื่อให้ได้วิธีหาค่าเหมาะที่สุด แบบหลายเป้าหมายที่มีสมรรถนะในการหาผลเฉลยที่ดีกว่า จากนั้นทำการผสม RPBIL แบบหลายเป้าหมายกับ approximate gradient เพื่อเพิ่มสมรรถนะของ RPBIL วิธีผสมทั้งสองแบบจะถูกนำมาใช้หาผลเฉลยของการหาค่าเหมาะที่สุดแบบหลายเป้าหมายของโครงข่าย และเปรียบเทียบผลที่ได้กับขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมายชนิดอื่นที่พัฒนามาก่อน เพื่อเป็นการยืนยันว่าวิธีผสมผสานที่คิดค้นขึ้นมาใหม่นี้ เป็นขั้นตอนวิธีวิวัฒนาการที่มีประสิทธิภาพสูง

9.2 RPBIL สำหรับการหาค่าเหมาะที่สุดแบบเป้าหมายเดียว

RPBIL ในที่นี้พัฒนาเพื่อใช้ในการหาผลเฉลยของปัญหาการออกแบบเหมาะที่สุด ที่มีขอบเขตของตัวแปรออกแบบ (bound-constrained optimisation) ซึ่งสามารถเขียนได้ดังนี้

$$\min_{\mathbf{x}} f(\mathbf{x}); \mathbf{L} \leq \mathbf{x} \leq \mathbf{U}$$

เมื่อ $\mathbf{x}$ คือ เวกเตอร์ตัวแปรออกแบบขนาด $n \times 1$ f คือฟังก์ชันเป้าหมายและ $\mathbf{L}$ และ $\mathbf{U}$ คือขอบเขตบนและล่างของ $\mathbf{x}$ ตามลำดับ แนวคิดของ RPBIL จะคล้ายกับ PBIL ดังเดิมคือใช้เวกเตอร์ของความน่าจะเป็นในการสร้างประชากรหรือกลุ่มของผลเฉลย ในกรณีของ PBIL แบบใช้แนวของเลขฐานสอง เราใช้เวกเตอร์ของความน่าจะเป็นเพื่อเป็นตัวแทนของการกระจายของประชากร แต่ในกรณีของ RPBIL เราใช้เมตริกซ์ความน่าจะเป็น (probability matrix) เพื่อเป็นตัวแทนของประชากรเลขจำนวนจริง กำหนดให้เมตริกซ์ความน่าจะเป็น มีขนาดเป็น $n \times n_I$ เมื่อ n_I คือจำนวนคอลัมน์ของเมตริกซ์ซึ่งผู้ออกแบบต้องระบุค่า พิจารณาปัญหาการหาค่าเหมาะที่สุดในปริภูมิสองมิติ

$$\min_{\mathbf{x}} f(\mathbf{x}) = x_1^2 + 2x_1x_2 + 3x_2^2$$

$$\text{subject to } -5 \leq x_i \leq 5; i = 1, 2$$

กำหนดให้เมตริกซ์ความน่าจะเป็นมีขนาดเป็น 2×4 ซึ่งหมายความว่า ขอบเขตของตัวแปรจะถูกแบ่งออกเป็น 4 ช่วงเท่าๆกัน ดังนี้ $[-5, -2.5]$ $[-2.5, 0]$ $[0, 2.5]$ และ $[2.5, 5]$ กำหนดให้เอลิเมนต์ของเมตริกซ์ความน่าจะเป็น P_{ij} แสดงในตารางที่ 9.1 ซึ่งแสดงให้เห็นว่าเอลิเมนต์ของ $\mathbf{x}$ จะสอดคล้องกับแถวของเมตริกซ์

ตารางที่ 9.1 ตัวอย่างของเมตริกซ์ความน่าจะเป็น

Subintervals	$-5 \leq x_i < -2.5$	$-2.5 \leq x_i < 0$	$0 \leq x_i < 2.5$	$2.5 \leq x_i \leq 5$
x_1	1/8	2/8	3/8	2/8
x_2	2/8	1/8	3/8	2/8

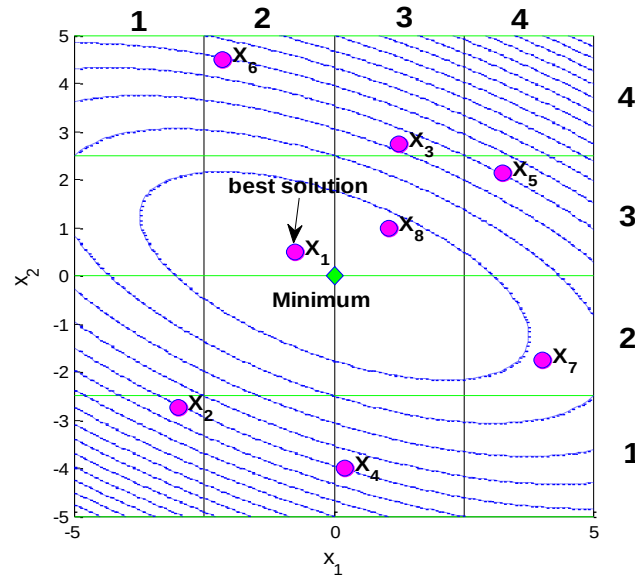
ตารางที่ 9.2 การสร้างประชากรเลขจำนวนจริงจากเมตริกซ์ความน่าจะเป็น

Subintervals	$-5 \leq x_i < -2.5$	$-2.5 \leq x_i < 0$	$0 \leq x_i < 2.5$	$2.5 \leq x_i \leq 5$
x_1	-3.00	-2.15, -0.75	0.20, 0.05, 1.25	3.25, 4.00
x_2	-4.00, -2.75	-1.75	1.00, 0.50, 2.15	2.75, 4.50
Randomly permutate the positions in each row				
x_1	-0.75, -3.00, 1.25, 0.20, 3.25, -2.15, 4.00, 0.05			
x_2	0.50, -2.75, 2.75, -4.00, 2.15, 4.50, -1.75, 1.00			

ค่าของ P_{ij} คือ ความน่าจะเป็นที่เอลิเมนต์ x_i จะถูกสุ่มสร้างขึ้นในช่วงที่ j กำหนดให้ขนาดของประชากรคือ $N_P = 8$ ดังนั้น จำนวนของ x_i ที่จะถูกสุ่มสร้างขึ้นในช่วงที่ j สามารถคำนวณได้จาก $\text{round}(N_P \cdot P_{ij})$ ตารางที่ 9.2 แสดงตัวอย่างการสร้างตัวแปรออกแบบจำนวนจริงด้วยหลักการนี้ ยกตัวอย่างเช่น จำนวนของ x_1 ที่จะถูกสร้างแบบสุ่มจากช่วงย่อยที่ 3 คือ $\text{round}(3/8 \times 8) = 3$ และได้ทำการสุ่มตัวอย่าง ดังแสดงในตาราง เมื่อได้ตัวเลขครบตามที่ต้องการ กล่าวคือค่าตัวเลขของ x_1 และ x_2 อย่างละ 8 ค่า จากนั้นทำการสลับตำแหน่งของค่าเหล่านั้นแบบสุ่มจะได้ประชากรเลขจำนวนจริงที่สร้างจากเมตริกซ์ความน่าจะเป็นนี้ ดังแสดงในตาราง (กำหนดให้ $\mathbf{R}$ คือเมตริกซ์ของประชากรเลขจำนวนจริงที่สร้างขึ้นมา)

$$\mathbf{R} = \{\mathbf{x}_1, \dots, \mathbf{x}_8\} =$$

$$\begin{Bmatrix} -0.75 \\ 0.50 \end{Bmatrix}, \begin{Bmatrix} -3.00 \\ -2.75 \end{Bmatrix}, \begin{Bmatrix} 1.25 \\ 2.75 \end{Bmatrix}, \begin{Bmatrix} 0.20 \\ -4.00 \end{Bmatrix}, \begin{Bmatrix} 3.25 \\ 2.15 \end{Bmatrix}, \begin{Bmatrix} -2.15 \\ 4.50 \end{Bmatrix}, \begin{Bmatrix} 4.00 \\ -1.75 \end{Bmatrix}, \begin{Bmatrix} 0.05 \\ 1.00 \end{Bmatrix}$$



รูปที่ 9.1 การกระจายตัวของสมาชิกในประชากรเลขจำนวนจริง

รูปที่ 9.1 แสดงตำแหน่งและการกระจายตัวของผลเฉลยที่สร้างขึ้นมาจากเมตริกซ์ความน่าจะเป็นในตารางที่ 9.1 ซึ่งจะเห็นได้ว่าจุด x_1 ให้ค่าฟังก์ชันเป้าหมายดีที่สุด ด้วยเหตุนี้ เราจะทำการปรับปรุงค่าของ P_{ij} ในรูปแบบที่ว่า P_{ij} ในรอบถัดไปจะสร้างประชากรที่มีสมาชิกส่วนใหญ่อยู่ในบริเวณใกล้เคียงกับ x_1 ด้วยแนวคิดนี้ ค่าของ P_{12} ควรจะถูกปรับให้สูงขึ้น เพราะ $x_{1,1} = -0.75$ อยู่ในช่วงย่อยที่สอง (อ้างอิงตารางที่ 9.2) ในทำนองเดียวกันค่าของ P_{23} ควรถูกปรับให้สูงขึ้น เพราะ $x_{1,2} = 0.50$ อยู่ในช่วงย่อยที่ 3 สมการที่ใช้ในการปรับค่าของ P_{ij} คือ

$$P'_{ij} = (1 - L_R(j))P_{ij}^{old} + L_R(j) \quad (9.1)$$

$$L_R(j) = L_{R0} \exp(-(j - r)^2) \quad (9.2)$$

เมื่อ L_{R0} คือ ค่าอัตราการเรียนรู้เริ่มต้น ซึ่งจะต้องกำหนดค่าโดยผู้ใช้ ค่า r คือ 2 เมื่อปรับค่าของ P_{12} และมีค่าเป็น 3 เมื่อปรับค่าของ P_{23} หลังจากนั้น ทำการปรับปรุงค่าของ P_{ij}

$$P''_{ij} = P'_{ij} / \sum_{j=1}^{n_i} P'_{ij} \quad (9.3)$$

เพื่อให้ได้เงื่อนไข

$$\sum_{j=1}^{n_i} P''_{ij} = 1 \quad (9.4)$$

เมื่อ P''_{ij} คือ ค่าของ P_{ij} หลังการปรับค่า

กระบวนการคำนวณของ RPBIL แสดงในขั้นตอนวิธีที่ 9.1 ค่าของเมตริกซ์ความน่าจะเป็นเริ่มต้นคือ $P_{ij} = 1/n_i$ หลังจากนั้นทำการสร้างประชากรจากเมตริกซ์นี้ จากนั้นทำการหาค่าฟังก์ชันเป้าหมายของผลเฉลยที่เป็นสมาชิกของประชากรที่สร้างขึ้นมานี้ จากนั้นหาผลเฉลยที่ดีที่สุดเพื่อนำไปปรับค่าของเมตริกซ์ความน่าจะเป็นโดยใช้สมการ (9.1-9.4) ดำเนินการซ้ำจนกระทั่งจำนวนรอบของการคำนวณครบตามที่ได้อ้างไว้ (N_G รอบ)

ขั้นตอนวิธีที่ 9.1 RPBIL สำหรับฟังก์ชันเป้าหมายเดียว

<u>Input:</u> N_G, n_I , objective function name (fun)
<u>Output:</u> $\mathbf{x}^{best}, f^{best}$
<u>Initialization:</u> $P_{ij} = 1/n_I, \mathbf{x}^{best} = \{\}$
1: For $i = 1$ to N_G
2: Generate a real code population $\mathbf{X}$ from P_{ij} (ดูตัวอย่างในตารางที่ 9.2) and find $\mathbf{f} = fun(\mathbf{X})$
3: Find new $\mathbf{x}^{best}$ from $\mathbf{X}$ and $\mathbf{x}^{best}$ from the previous generation
4: Update P_{ij} based on the new $\mathbf{x}^{best}$ (Equations (9.1), (9.2), (9.3))
5: End

9.3 การผสม RPBIL และ DE สำหรับการหาค่าเหมาะที่สุดแบบหลายเป้าหมาย

การขยายผลวิธี RPBIL เพื่อใช้ในการหาค่าเหมาะที่สุดแบบหลายเป้าหมาย สามารถทำได้ในรูปแบบเดียวกันกับการขยายผลของวิธี PBIL ที่ใช้เลขฐานสองจากวิธีหาค่าเหมาะที่สุดแบบเป้าหมายเดียวไปเป็นวิธีแบบหลายเป้าหมาย (หัวข้อที่ 3.4) ซึ่งในเบื้องต้นเราทำการขยายจากการใช้เวกเตอร์ความน่าจะเป็นตัวเดียวไปเป็นหลายเวกเตอร์เพื่อให้เกิดความหลากหลายในการสร้างประชากร (population diversity) ซึ่งเป็นหนึ่งในกลไกที่สำคัญที่สุดสำหรับขั้นตอนวิธีวิวัฒนาการแบบหลายเป้าหมาย ด้วยหลักการนี้ RPBIL สำหรับการหาค่าเหมาะที่สุดแบบหลายเป้าหมายจะใช้เมตริกซ์ความน่าจะเป็นหลายๆตัว เพื่อใช้ในการสร้างประชากรเลขจำนวนจริงในกระบวนการหาค่าเหมาะที่สุด เนื่องจากเป็นการใช้เมตริกซ์หลายๆตัวซ้อนกันอยู่ ดังนั้นเราอาจเรียกแต่ละเมตริกซ์ในกลุ่มของเมตริกซ์นี้ว่า ถาดของความน่าจะเป็น (probability tray) กำหนดให้จำนวนชั้นของเมตริกซ์ที่ใช้งานคือ N_T ดังนั้นเซตของเมตริกซ์ความน่าจะเป็นสำหรับ RPBIL แบบหลายเป้าหมาย (เขียนแทนด้วย MRPBIL) ก็คือเมตริกซ์สามมิติขนาด $n \times n_I \times N_T$ โดยในแต่ละชั้นของเมตริกซ์จะถูกใช้ในการสร้างประชากรย่อยจำนวน N_P/N_T ผลเฉลี่ย

กระบวนการหาค่าเหมาะที่สุดแบบหลายเป้าหมายด้วย MRPBIL แสดงในขั้นตอนวิธีที่ 9.2 โดยเริ่มต้นด้วย เมตริกซ์ความน่าจะเป็นสามมิติที่มีค่าของเอลิเมนต์เป็น $P_{ij} = 1/n_I$ ในแต่ละชั้นของเมตริกซ์ย่อย นอกจากนี้ต้องทำการสร้างเมตริกซ์เพื่อเก็บผลเฉลี่ยที่ไม่ถูกครอบงำ $\mathbf{A}$ ขึ้นมาเพื่อเก็บผลเฉลี่ยพารेटโตในแบบคู่ขนาน จากนั้นสร้างประชากรเลขจำนวนจริงจากเมตริกซ์ความน่าจะเป็นสามมิติในรูปแบบคล้ายๆกับ PBIL แบบหลายเป้าหมายในหัวข้อ 3.4 เมื่อได้ประชากรแล้วทำการหาค่าฟังก์ชันเป้าหมายและหาผลเฉลี่ยที่ไม่ถูกครอบงำ (non-dominated solutions) จากนั้นบันทึกผลเฉลี่ยที่ไม่ถูกครอบงำเหล่านี้ลงในเมตริกซ์ $\mathbf{A}$ ขั้นตอนถัดไปเป็นการปรับค่าของเมตริกซ์ความน่าจะเป็นสามมิติ ในขั้นตอนวิธีที่ 9.1 เราใช้ $\mathbf{x}^{best}$ ผลเฉลี่ยที่ดีที่สุดที่ค้นเจอเป็นตัวปรับค่าเมตริกซ์ความน่าจะเป็น ในกรณีของ MRPBIL เราจะใช้สมาชิกใน $\mathbf{A}$ เป็นตัวปรับค่าของแต่ละถาดในเมตริกซ์ความน่าจะเป็นสามมิตินั้นหมายความว่า เราต้องการผลเฉลี่ย N_T ตัวจาก $\mathbf{A}$ เพื่อมาปรับค่าของเมตริกซ์สามมิติ ซึ่งมีทั้งหมด N_T ถาด ในงานวิจัยนี้ เราใช้วิธี clustering เช่นวิธี k-mean เพื่อจัดกลุ่มสมาชิกใน $\mathbf{A}$ ออกเป็น N_T กลุ่ม

จากนั้นหาจุดศูนย์กลางของแต่ละกลุ่มซึ่งเขียนแทนด้วย $\mathbf{r}_G$ และเราจะใช้ $\mathbf{r}_G$ ในรูปแบบเดียวกันกับเราใช้ $\mathbf{x}^{\text{best}}$ ปรับปรุงค่าของเมตริกซ์ของความน่าจะเป็นในขั้นตอนวิธีที่ 9.1 ในกรณีที่จำนวนสมาชิกใน $\mathbf{A}$ มีมากเกินไป ซึ่งจะส่งผลให้คอมพิวเตอร์ใช้ความจำ (memory) มากเกินไป ส่งผลให้การรันเกิดความล่าช้า เราจะใช้วิธี clustering ในการกรองเอาสมาชิกบางส่วนออกจาก $\mathbf{A}$ เมื่อได้เมตริกซ์ความน่าจะเป็นสามมิติที่ปรับค่าใหม่ ทำการสร้างประชากรเลขจำนวนจริงขึ้นมาพร้อมหาค่าฟังก์ชันเป้าหมาย จากนั้นปรับปรุงเมตริกซ์ $\mathbf{A}$ โดยการแทนที่สมาชิกเดิมด้วยสมาชิกชุดใหม่ที่เป็นผลเฉลยที่ไม่ถูกรับรองว่าดีกว่าจากการรวมสมาชิกเดิมใน $\mathbf{A}$ และประชากรเลขจำนวนจริงที่สร้างขึ้นใหม่ กระบวนการดำเนินการซ้ำเป็นรอบๆ จนกระทั่งได้ข้อหน้าพาเรโตที่เหมาะสม

ขั้นตอนวิธีที่ 9.2 RPBIL สำหรับการหาค่าเหมาะที่สุดแบบหลายเป้าหมาย

Input: N_G, n_l, N_T , objective function name (fun), Pareto archive size (N_A)
Output: $\mathbf{x}^{\text{best}}, f^{\text{best}}$
Initialization: $P_{ij} = 1/n_l$ for each tray, a Pareto archive $\mathbf{A} = \{\}$
1: For $i = 1$ to N_G
2: Generate a real code population $\mathbf{X}$ from the probability trays and find $\mathbf{f} = fun(\mathbf{X})$
3: Find non-dominated solutions from $\mathbf{X} \cup \mathbf{A}$ and replace the members in $\mathbf{A}$ with these solutions.
4: Group the non-dominated solutions into N_T groups using a clustering technique, and find the centroid $\mathbf{r}_G$ of each group.
5: Update each tray P_{ij} based on $\mathbf{r}_G$.
6: If the number of archive members is larger than N_A , remove some of the members using a clustering technique.
7: End

ปัญหาที่มักเกิดขึ้นเมื่อรัน MRPBIL คือ ถึงแม้จะมีสรรถนะในการหาคำตอบเป็นที่ยอมรับได้ แต่เนื่องจากเมตริกซ์ความน่าจะเป็นสามมิติมักจะสร้างกลุ่มผลเฉลยที่อยู่บริเวณ $\mathbf{r}_G$ เท่านั้น ทำให้ความหลากหลายของประชากรมีน้อย ดังนั้นจึงต้องหาวิธีเพิ่มความหลากหลายของประชากร ซึ่งในที่นี้เราจะผสมผสานตัวดำเนินการของ DE เข้ามาสร้างความหลากหลายให้กับประชากรเลขจำนวนจริงที่สร้างมาจากเมตริกซ์ความน่าจะเป็นสามมิติ กระบวนการที่เพิ่มขึ้นมาจาก MRPBIL ในขั้นตอนวิธีที่ 9.2 แสดงในขั้นตอนวิธีที่ 9.3 เมื่อทำการสร้างประชากรผลเฉลยจำนวนจริงจากเมตริกซ์ความน่าจะเป็นสามมิติแล้ว นำผลเฉลยทั้งหมดในประชากรนี้มาทำการมิวเทชันและครอสโอเวอร์แบบ DE/best/1/bin โดยคำว่า best ในที่นี้คือ การเลือกสมาชิกใน $\mathbf{A}$ (เขียนแทนด้วย $\mathbf{p}$) และสมาชิกในประชากรใหม่มาแบบสุ่ม จากนั้นทำการมิวเทชันและครอสโอเวอร์ ดังแสดงในขั้นตอนที่ 5 ในขั้นตอนวิธีที่ 9.3 และวิธีวิวัฒนาการใหม่นี้ เรียกว่า RPBIL-DE

- SPEA2 using real codes with crossover and mutation rates of 1.0 and 0.1 respectively.
- MPSO using the starting inertia weight, ending inertia weight, cognitive learning factor, and social learning factor being 0.5, 0.01, 0.5 and 0.5 respectively. The best solution for computing a velocity vector is randomly selected from a Pareto archive. The adaptive grid algorithm is used as an archiving technique.
- NSGA-II using real codes with crossover mutation probabilities as 1.0 and 0.1 respectively.
- RPBIL using real codes with $N_I = 40$ where each probability tray produces 5 design solutions.
- RPBIL-DE using real codes with $N_I = 40$ where each probability tray produces 5 design solutions. Crossover probability, scaling factor and probability of choosing an element from an offspring in crossover for DE operators are set as 0.7, 0.8, and 0.5 respectively.
- DMOEA using real codes with temperature and mutation rate of 1000 and 1 respectively.
- DEMO using real codes with crossover probability, scaling factor and probability of choosing an element from an offspring in crossover for DE operators being 0.7, 0.8, and 0.5 respectively.
- MOEA/D using real codes with number of neighboring weight vectors, crossover and mutation probabilities being 6, 1.0, and 0.1 respectively.

วิธี MOEAs ทั้งหมด ยกเว้น UPS-EMOA ใช้ประชากรขนาด 100 และใช้จำนวนรอบในการหาผลเฉลยเป็น 250 รอบ สำหรับปัญหา ZTDs และใช้ประชากรขนาด 100 และใช้จำนวนรอบในการหาผลเฉลยเป็น 250 รอบ สำหรับปัญหา CFs ขนาดของเมตริกซ์เก็บผลเฉลยที่ไม่ถูกครอบงำคือ 500 ยกเว้นกรณีของ NSGA-II และ DEMO ที่เก็บผลเฉลยที่ไม่ถูกครอบงำในเมตริกซ์ประชากรผลเฉลยแต่ละวิธีหาค่าเหมาะที่สุดจะถูกใช้หาผลเฉลยของแต่ละปัญหาการออกแบบ 30 ครั้งเพื่อสามารถเปรียบเทียบสถิติได้ ส่วนตัวชี้วัดที่ใช้คือ ค่า hypervolume หรือ HV ซึ่งถ้าค่า HV สูงกว่าแสดงว่าสมรรถนะในการหาผลเฉลยพาเรโตดีกว่า

จากตารางที่ 9.3 วิธีที่ดีที่สุด (วัดจาก hypervolume) สำหรับปัญหา ZDT1 ZDT2 ZDT3 และ ZDT6 คือวิธี MOEA/D ส่วนวิธี NSGA-II ให้คำตอบดีที่สุดในการแก้ปัญหาทดสอบมาตรฐาน ZDT4 วิธีที่ดีอันดับสองสำหรับ ZDT1 และ ZDT2 คือ RPBIL ในขณะที่ RPBIL-DE เป็นวิธีที่ดีอันดับสองในการแก้ปัญหา ZDT3 และ ZDT6 สำหรับ ZDT4 วิธีที่ดีอันดับสองคือ MOEA/D วิธีที่ให้คำตอบแย่ที่สุดคือ MPSO สำหรับปัญหาการออกแบบ ZDT1-3 ส่วนอีกสองปัญหาที่เหลือ วิธีที่ให้คำตอบแย่ที่สุดคือ RPBIL ถ้าดูจากค่าเฉลี่ยของ HV เราสามารถสรุปได้ว่า วิธีที่ดีที่สุดคือ MOEA/D

ผลการเปรียบเทียบค่า HV ในการแก้ปัญหาทดสอบมาตรฐาน CFs แสดงในตารางที่ 9.4 จากตารางพบว่า วิธีที่ดีที่สุดสำหรับ CF2 และ CF5 คือ RPBIL-DE วิธีที่ดีที่สุดสำหรับ CF4 และ CF7 คือ DEMO ส่วนวิธีที่ดีที่สุดสำหรับ CF1 CF3 และ CF6 คือ SPEA2 NSGA-II และ MOEA/D

ตามลำดับ วิธีที่ดีอันดับสองสำหรับ CF1 และ CF5 คือ RPBIL-DE และ DEMO ตามลำดับ ส่วนวิธีที่ดีอันดับสองของปัญหาการออกแบบที่เหลือนั้น คือ SPEA2 ในชุดของปัญหาการทดสอบมาตรฐาน CFs วิธีที่ให้คำตอบที่ดีที่สุดยังคงเป็น RPBIL และ MPSO โดยภาพรวมแล้ว RPBIL-DE เป็นวิธีที่ดีที่สุดสำหรับปัญหาการทดสอบมาตรฐาน CF1-7 ซึ่งเป็นการหาค่าเหมาะที่สุดแบบหลายเป้าหมายที่มีเงื่อนไขบังคับ ผลการทดสอบทางสถิติ Friedman test แสดงให้เห็นว่าค่า HV ที่ได้จากวิธีต่างๆ มีความแตกต่างกันอย่างมีนัยสำคัญ

ตารางที่ 9.5 แสดงผลการจัดลำดับสมรรถนะของ MOEAs โดยการทดสอบทางสถิติ Wilcoxon rank sum test ซึ่งสามารถสรุปได้ว่า MOEA/D เป็นวิธีที่ดีที่สุดสำหรับปัญหาการทดสอบ ZTDs โดยมีวิธีที่ดีอันดับสองคือ BPBIL และวิธีที่ดีอันดับสามคือ SPEA2 และ RPBIL-DE ส่วนในกรณีของการหาค่าเหมาะที่สุดแบบมีเงื่อนไขบังคับกับ CFs วิธีที่ดีที่สุดคือ SPEA2 วิธีที่ดีอันดับสองและสามคือ RPBIL-DE และ NSGA-II ตามลำดับ โดยรวมแล้วเราสามารถสรุปได้ว่า RPBIL และ MPSO มีสมรรถนะในการหาผลเฉลยพาริตีไม่สูงมากนัก แต่เมื่อเพิ่ม DE เข้ามาใน RPBIL ซึ่งจะกลายเป็นวิธี RPBIL-DE ทำให้ประสิทธิภาพของขั้นตอนวิธีผสมนี้มีสมรรถนะในการหาคำตอบเพิ่มขึ้นอย่างมีนัยสำคัญ

ตารางที่ 9.3 Normalised hypervolumes of all MOEAs for all ZDT test problems.

MOEAs Test Problems		BPBI L	UPS- EMO A	SPEA 2	MPS O	NSG A-II	MRPBI L	RPBI L-DE	DEMOE A	DEM O	MOEA/ D	Friedma n test (p- value)
ZDT 1	Mea n	0.996 9	0.987 2	0.986 7	0.391 1	0.967 9	0.9481	0.9868	0.8727	0.977 8	0.9999	8.33 $\times 10^{-51}$
	STD	0.001 3	0.003 7	0.001 7	0.159 7	0.012 3	0.0022	0.0010	0.0191	0.006 3	0.0001	
	Max .	0.998 9	0.995 3	0.989 9	0.636 8	0.981 8	0.9516	0.9890	0.9178	0.987 4	1.0000	
	Min.	0.993 9	0.979 5	0.982 8	0.000 0	0.935 0	0.9433	0.9845	0.8418	0.959 7	0.9995	
ZDT 2	Mea n	0.997 7	0.942 1	0.987 6	0.446 3	0.978 2	0.9239	0.9857	0.8559	0.983 9	0.9992	1.41x 10^{-46}
	STD	0.001 2	0.098 7	0.001 3	0.175 9	0.007 2	0.0051	0.0015	0.0175	0.004 4	0.0002	
	Max .	1.000 0	0.997 5	0.991 3	0.699 9	0.985 8	0.9353	0.9898	0.8828	0.991 5	0.9995	
	Min.	0.994 2	0.698 1	0.985 3	0.000 0	0.954 5	0.9134	0.9828	0.8246	0.971 7	0.9987	
ZDT 3	Mea n	0.972 6	0.975 1	0.986 1	0.287 6	0.964 8	0.9639	0.9890	0.8272	0.970 4	0.9996	5.19x10 ⁻⁴⁴
	STD	0.018 4	0.024 7	0.001 8	0.107 4	0.025 7	0.0034	0.0012	0.0200	0.011 1	0.0004	
	Max .	0.990 9	0.995 0	0.990 0	0.466 0	0.987 6	0.9689	0.9921	0.8632	0.988 7	1.0000	
	Min.	0.901 3	0.886 9	0.983 1	0.000 0	0.865 3	0.9541	0.9867	0.7902	0.948 8	0.9981	
ZDT 4	Mea n	0.880 1	0.548 5	0.982 8	0.763 0	0.994 1	0.4026	0.6743	0.5138	0.846 3	0.9820	9.80x10 ⁻⁴⁹
	STD	0.045 3	0.189 6	0.012 7	0.073 4	0.003 8	0.1299	0.0772	0.0802	0.055 0	0.0078	
	Max .	0.944 5	0.905 3	0.999 2	0.931 0	1.000 0	0.6253	0.7987	0.6600	0.958 2	0.9957	
	Min.	0.807 0	0.000 0	0.956 4	0.611 0	0.985 1	0.1462	0.4743	0.3418	0.756 5	0.9685	
ZDT 6	Mea n	0.997 7	0.997 3	0.995 1	0.953 1	0.991 9	0.1803	0.9988	0.2209	0.989 3	0.9990	7.73x10 ⁻⁴²
	STD	0.000 1	0.001 1	0.004 5	0.125 0	0.004 1	0.0215	0.0010	0.1970	0.006 0	0.0007	
	Max .	0.997 8	0.998 6	0.999 5	1.000 0	0.997 4	0.2370	0.9998	0.7714	0.996 1	0.9996	
	Min.	0.997 6	0.992 7	0.978 4	0.491 0	0.980 0	0.1396	0.9958	0.0000	0.968 8	0.9962	

ตารางที่ 9.4 Normalised hypervolumes of all MOEAs for all ZDT test problems.

MOEAs Test Problems		BPBI L	UPS- EMO A	SPEA 2	MPS O	NSG A-II	MRPBI L	RPBI L-DE	DEMOE A	DEM O	MOEA/ D	Friedma n test (p- value)
CF 1	Mea n	0.891 5	0.4171	0.962 9	0.897 8	0.8639	0.4655	0.9200	0.6463	0.3745	0.8251	5.36x10 ⁻⁴²
	STD	0.027 9	0.2001	0.009 1	0.082 5	0.1878	0.1159	0.0234	0.0241	0.0849	0.0756	
	Max .	0.936 4	0.9315	0.977 7	1.000 0	0.9744	0.7123	0.9563	0.7240	0.5545	0.9137	
	Min.	0.812 5	0.0000	0.938 7	0.712 4	0.0489	0.2330	0.8572	0.5989	0.2631	0.5911	
CF 2	Mea n	0.978 5	0.8765	0.993 0	0.803 2	0.5303	0.3676	0.9980	0.7930	0.8572	0.9677	2.29x10 ⁻³⁹
	STD	0.019 7	0.1240	0.003 0	0.090 7	0.2644	0.2866	0.0013	0.0250	0.1492	0.0260	
	Max .	0.993 8	0.9998	0.999 0	0.936 2	0.9466	0.8980	1.0000	0.8350	0.9978	0.9911	
	Min.	0.901 7	0.5245	0.988 4	0.625 7	0.0216	0.0000	0.9955	0.7320	0.6228	0.9182	
CF 3	Mea n	0.600 0	0.7740	0.905 3	0.223 8	0.9262	0.4082	0.7909	0.5361	0.8811	0.7582	1.45x10 ⁻⁴²
	STD	0.087 8	0.1355	0.072 7	0.121 0	0.0507	0.1630	0.0659	0.0366	0.0322	0.1033	
	Max .	0.757 9	0.9262	1.000 0	0.537 0	1.0000	0.6822	0.9190	0.6301	0.9324	0.9169	
	Min.	0.444 1	0.3729	0.733 6	0.000 0	0.7555	0.0939	0.6425	0.4750	0.8078	0.5397	
CF 4	Mea n	0.915 6	0.9008	0.979 0	0.773 7	0.9332	0.5008	0.9750	0.9288	0.9813	0.9187	7.35x10 ⁻³⁵
	STD	0.057 4	0.0737	0.038 7	0.116 7	0.0421	0.2968	0.0497	0.0260	0.0167	0.0597	
	Max .	0.990 2	0.9958	0.999 4	0.920 8	0.9889	0.8721	0.9982	0.9698	1.0000	0.9803	
	Min.	0.817 5	0.7098	0.826 2	0.507 2	0.8023	0.0000	0.8206	0.8524	0.9411	0.7927	
CF 5	Mea n	0.531 5	0.4708	0.717 2	0.102 0	0.7296	0.7603	0.7638	0.6296	0.7503	0.5691	2.79x10 ⁻³⁰
	STD	0.163 9	0.2329	0.118 6	0.121 5	0.1470	0.0007	0.0004	0.1397	0.0181	0.1792	
	Max .	0.774 4	0.8214	0.914 0	0.337 6	1.0000	0.7618	0.7644	0.7565	0.7646	0.8578	
	Min.	0.064 7	0.0000	0.414 9	0.000 0	0.4042	0.7592	0.7628	0.3652	0.6572	0.2209	
CF 6	Mea n	0.867 0	0.6565	0.903 3	0.821 6	0.7367	0.5429	0.8595	0.7780	0.5848	0.9056	9.26x10 ⁻³⁹
	STD	0.035 6	0.1646	0.016 4	0.037 8	0.1698	0.0465	0.1003	0.0382	0.1554	0.0636	
	Max .	0.947 9	0.8924	0.940 3	0.885 2	0.8870	0.7041	0.9685	0.8522	0.8651	1.0000	
	Min.	0.790 0	0.3661	0.865 0	0.742 9	0.0000	0.4776	0.6421	0.7105	0.2672	0.6192	
CF 7	Mea n	0.795 9	0.7173	0.910 6	0.320 8	0.8769	0.6569	0.8800	0.8180	0.9433	0.8402	1.58x10 ⁻³¹
	STD	0.070 9	0.1746	0.049 3	0.163 1	0.0661	0.1788	0.0708	0.0397	0.0467	0.0777	
	Max .	0.882 9	0.9532	0.983 2	0.764 0	0.9768	0.8974	0.9786	0.8722	1.0000	0.9668	
	Min.	0.618 1	0.2648	0.781 7	0.000 0	0.7395	0.2056	0.7212	0.7378	0.8128	0.6437	

ตารางที่ 9.5 Ranking of MOEAs for all test problems based on Wilcoxon rank sum test.

	BPBI L (1)	UPS- EMO A (2)	SPEA 2 (3)	MPS O (4)	NSGA -II (5)	MRPBI L (6)	RPBIL -DE (7)	DEMOE A (8)	DEM O (9)	MOEA/ D (10)
ZDT 1	2	3	3	10	7	8	3	9	6	1
ZDT 2	2	6	3	10	6	8	4	9	5	1
ZDT 3	4	4	3	10	4	8	2	9	4	1
ZDT 4	4	8	2	6	1	10	7	8	5	2
ZDT 6	3	5	6	3	7	9	1	9	7	1
Σ	15	26	17	39	25	43	17	44	27	6
Rank	2	5	3	7	4	8	3	9	6	1
CF1	5	8	1	2	2	8	2	7	10	6
CF2	3	5	2	7	9	10	1	8	5	3
CF3	7	4	1	10	1	9	4	8	3	4
CF4	4	8	1	9	4	10	1	4	1	4
CF5	8	8	3	10	1	3	1	6	5	6
CF6	3	8	1	5	6	10	3	6	9	1
CF7	7	8	2	10	4	8	2	6	1	5
Σ	37	49	11	53	27	58	14	45	34	29
Rank	6	8	1	9	3	10	2	7	5	4

9.5 ปัญหาทดสอบสำหรับการหาค่าเหมาะที่สุดของโครงถัก

ในการศึกษานี้จะใช้ปัญหาการออกแบบเหมาะที่สุดแบบหลายเป้าหมายของโครงถักสองและสามมิติ 5 ปัญหา ดังรายละเอียดต่อไปนี้

9.5.1 Case I: A 2D simple supported 37 bar truss

โครงถักของปัญหาการออกแบบนี้แสดงในรูปที่ 9.2 ประกอบด้วยจุดต่อ 20 จุดและชิ้นส่วน 37 ชิ้น โครงสร้างรับแรงในแนวดิ่ง -10 kN ที่จุดต่อหมายเลข 2 ถึง 10 ตัวแปรออกแบบคือพื้นที่หน้าตัดของชิ้นส่วน และตำแหน่งในแนวนอนและแนวดิ่งของจุดต่อ 2-10 และ 12-20 ตำแหน่งในแนวดิ่งของจุดต่อ 6 และ 16 ตัวแปรออกแบบถูกกำหนดเพื่อให้ได้โครงสร้างที่สมมาตรเทียบกับแกนแนวดิ่ง ความหนาแน่นและมอดูลัสของยังของวัสดุที่ใช้ทำโครงถักนี้ คือ 7971.810 kg/m^3 และ 210 GPa ค่าความเค้นสูงสุดที่ยอมให้กระทำกับชิ้นส่วน คือ 400 MPa ปัญหาการหาค่าเหมาะที่สุด สามารถแสดงดังนี้

$$\text{Min: } [f_1(\mathbf{x}), f_2(\mathbf{x})]$$

Subject to

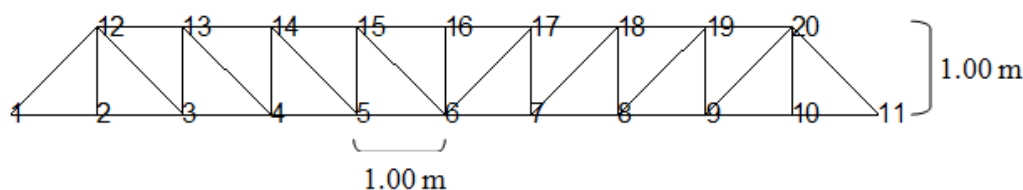
$$|\sigma_{\max}| \leq \sigma_y$$

$$1.77 \leq A_i \leq 113.09 \quad \text{cm}^2$$

$$-0.45 \leq X_i \leq 0.45 \text{ m}$$

$$-0.45 \leq Y_i \leq 0.45 \text{ m}$$

เมื่อ $\mathbf{x} = \{x_1, \dots, x_{37}\}^T$ คือ เวกเตอร์ตัวแปรออกแบบ $f_1(\mathbf{x})$ คือ มวลของโครงสร้าง $f_2(\mathbf{x})$ คือ ค่าความอ่อนตัวของโครงสร้าง $\sigma_{\max}$ คือค่าความเค้นสูงสุดที่เกิดขึ้นกับโครงถัก A_i คือพื้นที่หน้าตัดของเอลิเมนต์ X_i และ Y_i คือ ตำแหน่งของจุดต่อในแนวแกน x และแกน y ตามลำดับ



รูปที่ 9.2 2D simple supported 37 bar truss

9.5.2 Case II: Sizing design of a 2D 200 bar truss

โครงถักของปัญหาการออกแบบนี้แสดงในรูปที่ 9.3 ประกอบด้วยจุดต่อ 75 จุดและชิ้นส่วน 200 ชิ้น โครงสร้างรับแรงในแนวดิ่ง -45 kN ที่จุดต่อหมายเลข 1- 6 8 10 12 14-20 22 24-75 และแรงกระทำในแนวนอน 4.5 kN ที่จุดต่อหมายเลข 1 6 15 20 29 34 43 48 57 62 และ 7 ตัวแปรออกแบบคือพื้นที่หน้าตัดของชิ้นส่วนทุกเอลิเมนต์ ตัวแปรออกแบบถูกกำหนดเพื่อให้ได้โครงสร้างที่สมมาตรเทียบกับแกนแนวดิ่ง ความหนาแน่นและมอดูลัสของยังของวัสดุที่ใช้ทำโครงถักนี้คือ

7833 kg/m³ และ 206.9 GPa ตามลำดับ ค่าความเค้นสูงสุดที่ยอมให้กระทำกับชิ้นส่วนคือ 65.95 MPa ปัญหาการหาค่าที่เหมาะสมที่สุด สามารถแสดงดังนี้

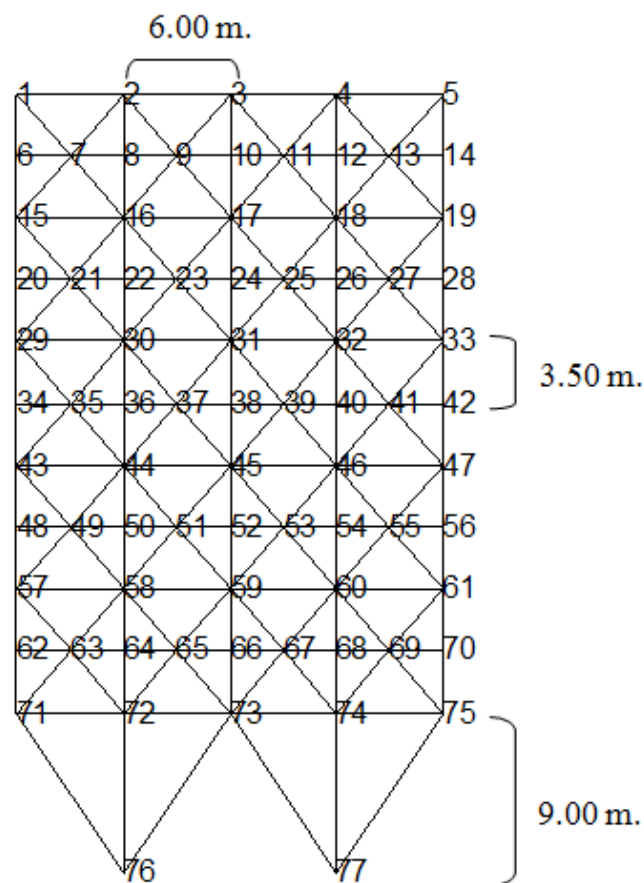
$$\text{Min: } [f_1(\mathbf{x}), f_2(\mathbf{x})]$$

Subject to

$$|\sigma_{\max}| \leq \sigma_y$$

$$0.636 \leq A_i \leq 132.73 \text{ cm}^2$$

เมื่อ $\mathbf{x} = \{x_1, \dots, x_{84}\}^T$ คือ เวกเตอร์ตัวแปรออกแบบ $f_1(\mathbf{x})$ คือ มวลของโครงสร้าง $f_2(\mathbf{x})$ คือ ค่าความอ่อนตัวของโครงสร้าง $\sigma_{\max}$ คือ ค่าความเค้นสูงสุดที่เกิดขึ้นกับโครงถัก A_i คือ พื้นที่หน้าตัดของเอलिเมนต์



รูปที่ 9.3 2D 200-bar truss

8.5.3 Case III: Shape and sizing design of a 2D 200 bar truss

ปัญหาการออกแบบนี้คือปัญหาของ Case II ที่มีตัวแปรกำหนดรูปทรงเพิ่มขึ้นมา ในปัญหาการออกแบบนี้ ตำแหน่งในแนวแกน y ของชั้นของโครงสร้างในรูปที่ 9.3 ถูกกำหนดเป็นตัวแปรออกแบบเพิ่มเติม ปัญหาการหาค่าที่เหมาะสมที่สุด สามารถเขียนได้ดังนี้

$$\text{Min: } [f_1(\mathbf{x}), f_2(\mathbf{x})]$$

Subject to

$$|\sigma_{\max}| \leq \sigma_y$$

$$0.636 \leq A_i \leq 132.73 \text{ cm}^2$$

$$-1.5 \leq Y_i \leq 1.5 \quad \text{m for } i > 1$$

$$-7 \leq Y_i \leq 1.5 \quad \text{m for } i = 1$$

เมื่อ Y_i คือ ค่าที่ใช้ปรับความสูงของชั้นต่างๆจากตำแหน่งเดิมของโครงสร้างในรูปที่ 9.3

9.5.4 Case IV: Sizing design of a 3D Dome 120 bar truss

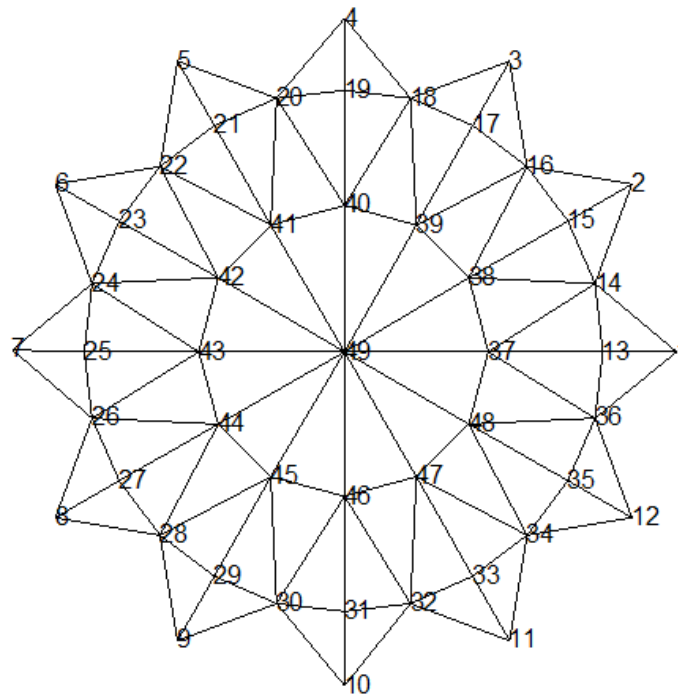
โครงถักของปัญหาการออกแบบนี้แสดงในรูปที่ 9.4 ประกอบด้วยจุดต่อ 49 จุดและชิ้นส่วน 120 ชิ้น โครงสร้างรับแรงในแนวตั้ง -10 kN ที่จุดต่อหมายเลข 13-36 ขนาด -30 kN ที่จุดต่อหมายเลข 37-49 และขนาด -60 kN ที่จุดต่อหมายเลข 49 ตัวแปรออกแบบคือพื้นที่หน้าตัดของชิ้นส่วนทุกเอलिเมนต์ ค่าความความหนาแน่นและมอดูลัสของยังของวัสดุที่ใช้ทำโครงถักนี้คือ 7971.810 kg/m^3 และ 210 GPa ตามลำดับ ค่าความเค้นสูงสุดที่ยอมให้กระทำกับชิ้นส่วนคือ 400 MPa ปัญหาการหาค่าเหมาะที่สุด สามารถเขียนได้ดังนี้

$$\text{Min: } [f_1(\mathbf{x}), f_2(\mathbf{x})]$$

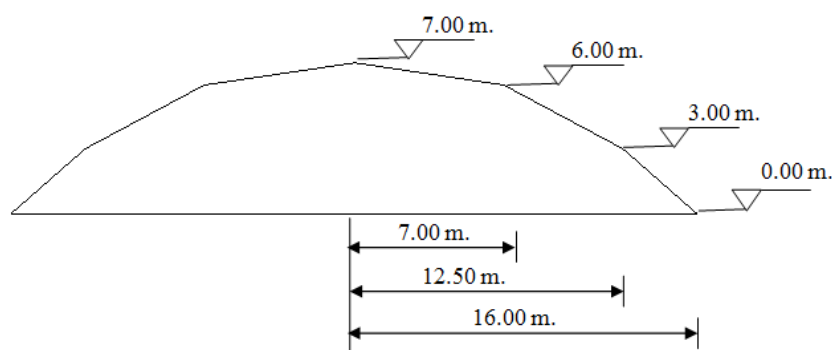
Subject to

$$|\sigma_{\max}| \leq \sigma_y$$

$$1.77 \leq A_i \leq 132.73 \text{ cm}^2$$



รูปที่ 9.4 ก. Top view of a 3D Dome 120 bar truss



รูปที่ 9.4 ข. Side view of a 3D Dome 120 bar truss

9.5.5 Case V: Shape and sizing design of a 3D Dome 120 bar truss

ปัญหาการออกแบบนี้คือ ปัญหาของ Case IV ที่มีตัวแปรกำหนดรูปทรงเพิ่มขึ้นมา ในปัญหาการออกแบบนี้ตำแหน่งในแนวแกน z ของชั้นของโครงสร้างในรูปที่ 9.4 ถูกกำหนดเป็นตัวแปรออกแบบเพิ่มเติม ยกเว้นชั้นที่เป็นพื้นที่มีจุดต่อ 1-12 ปัญหาการหาค่าเหมาะที่สุด สามารถเขียนได้ดังนี้

$$\text{Min: } [f_1(\mathbf{x}), f_2(\mathbf{x})]$$

Subject to

$$|\sigma_{\max}| \leq \sigma_y$$

$$1.77 \leq A_i \leq 132.73 \quad \text{cm}^2$$

$$-1.5 \leq Z_i \leq 1.5 \quad \text{m}$$

เมื่อ Z_i คือ ค่าที่ใช้ปรับความสูงของชั้นต่างๆจากตำแหน่งเดิมของโครงสร้างในรูปที่ 9.4 ข.

ขั้นตอนวิธีวิวัฒนาการแต่ละวิธีถูกใช้ในการหาผลเฉลยแต่ละปัญหา 30 รอบของการรัน ขนาดของเมตริกซ์เก็บผลเฉลยพารโตคือ 500 ขนาดของประชากรและจำนวนลูปกำหนดค่าใน ตารางที่ 9.6 ซึ่งแสดงการสรุปผลค่าที่สำคัญของปัญหาการออกแบบโครงสร้างทั้ง 5 ปัญหา

ตารางที่ 9.6 สรุปค่าพารามิเตอร์ของปัญหาทั้ง 5

Parameters	No. of design variables	Population size	No. of generations	Archive size
CASE I	37	30	30	500
CASE II	84	50	50	500
CASE III	94	50	50	500
CASE IV	120	50	60	500
CASE V	123	50	60	500

9.6 ผลการศึกษาเชิงเปรียบเทียบ

จากผลการเปรียบด้วยค่าเฉลี่ยในตารางที่ 9.7 และการจัดลำดับด้วยการทดสอบทางสถิติ Wilcoxon rank sum test ในตารางที่ 9.8 พบว่า วิธี RPBIL-DE ที่พัฒนาขึ้นมาใหม่เป็นวิธีที่ดีที่สุดในทุกกรณี ในขณะที่วิธีที่แย่ที่สุดคือ MPSO ยกเว้นปัญหา Case V ซึ่ง MOEA/D ให้ผลคำตอบที่แย่

ที่สุด วิธีที่ดีอันดับสองสำหรับ Case I และ Case IV คือ RPBIL สำหรับ Case II และ Case III คือ DEMO และสำหรับ Case V คือ UPS-EMOA จากการจัดอันดับด้วยการทดสอบทางสถิติในตารางที่ 9.8 พบว่าวิธีที่ดีที่สุดอันดับ 1 2 3 คือ RPBIL-DE DEMO และ RPBIL ตามลำดับ

จากผลการทดสอบนี้จะได้อิสรูปคล้ายๆการศึกษาที่ผ่านมาคือ เมื่อเพิ่ม DE เข้าไปทำให้กลายเป็น RPBIL-DE วิธีการนี้จะมีสมรรถนะในการหาผลเฉลยดีกว่าวิธีดั้งเดิมคือ RPBIL อย่างไรก็ตามวิธี RPBIL กลับไม่ใช่วิธีที่แย่ที่สุดในการประยุกต์ใช้กับการออกแบบโครงถัก และเป็นวิธีที่ดีอันดับที่สาม ดังนั้นการทดสอบผ่านปัญหาทดสอบมาตรฐานไม่สามารถสรุปสมรรถนะของ MOEAs สำหรับปัญหาการออกแบบชนิดอื่นได้ เช่น ปัญหาการออกแบบโครงถักนี้ เป็นต้น

ตารางที่ 9.7 Normalized hypervolumes of all MOEAs for all test problems

MOEAs Test Problems		BPBI L	UPS- EMO A	SPEA 2	MPS O	NSGA -II	RPBI L	RPBIL -DE	DEMO E	DEM O	MOEA/ D	Friedma n test (p-value)
Cas e I	Mea n	0.6074	0.8654	0.5069	0.255 2	0.4891	0.9831	0.988 2	0.4163	0.9193	0.4188	4.31x10 ⁻⁴⁶
	STD	0.0904	0.0742	0.1047	0.112 4	0.1127	0.0009	0.001 0	0.1085	0.0655	0.1348	
	Max.	0.7739	0.9913	0.7174	0.456 6	0.7017	0.9854	0.990 0	0.6423	1.0000	0.6369	
	Min.	0.4278	0.6618	0.3092	0.000 0	0.2863	0.9815	0.986 5	0.1859	0.7654	0.1279	
Cas e II	Mea n	0.3484	0.5846	0.3156	0.125 1	0.3805	0.4253	0.962 7	0.2132	0.6256	0.2185	7.14x10 ⁻⁴⁸
	STD	0.0777	0.0986	0.0839	0.082 0	0.0658	0.0735	0.020 8	0.0713	0.0699	0.0677	
	Max.	0.5449	0.8015	0.5826	0.279 0	0.4940	0.6582	1.000 0	0.3522	0.7447	0.3535	
	Min.	0.1925	0.4244	0.1118	0.000 0	0.2098	0.3256	0.914 8	0.0011	0.4810	0.0990	
Cas e III	Mea n	0.3818	0.5608	0.2531	0.113 3	0.3311	0.4012	0.966 3	0.2183	0.5840	0.1641	3.67x10 ⁻⁴⁷
	STD	0.0684	0.0900	0.0784	0.070 0	0.0709	0.0834	0.017 8	0.0794	0.0723	0.0688	
	Max.	0.5026	0.7374	0.4436	0.325 4	0.4648	0.6471	1.000 0	0.3487	0.7891	0.2872	
	Min.	0.2177	0.3394	0.1030	0.000 0	0.2309	0.2972	0.923 2	0.0958	0.4681	0.0014	
Cas e IV	Mea n	0.1499	0.3808	0.1635	0.041 4	0.2204	0.9381	0.980 7	0.1113	0.3118	0.1325	4.90x10 ⁻⁴⁷
	STD	0.0492	0.0532	0.0559	0.029 2	0.0548	0.0026	0.004 4	0.0308	0.1022	0.0527	
	Max.	0.2577	0.5062	0.2898	0.097 5	0.3578	0.9444	1.000 0	0.1828	0.5866	0.2388	
	Min.	0.0721	0.2783	0.0617	0.000 0	0.1220	0.9332	0.974 2	0.0508	0.1599	0.0143	
Cas e V	Mea n	0.5617	0.6781	0.5083	0.425 6	0.5532	0.5091	0.755 8	0.4785	0.4771	0.4158	3.82x10 ⁻³⁷
	STD	0.0360	0.0436	0.0249	0.035 9	0.0231	0.1364	0.080 5	0.0285	0.0676	0.1039	
	Max.	0.6199	0.7438	0.5755	0.490 3	0.5949	0.7857	1.000 0	0.5430	0.5828	0.5316	
	Min.	0.4939	0.5846	0.4749	0.363 9	0.4843	0.3180	0.597 1	0.4174	0.3191	0.0000	

ตารางที่ 9.8 Ranking of MOEAs for all truss design problems based on Wilcoxon rank sum test

	BPBIL	UPS-EMOA	SPEA2	MPSO	NSGA-II	RPBIL	RPBIL-DE	DEMOEA	DEMO	MOEA/D
Case I	5	4	6	10	6	2	1	8	3	8
Case II	5	2	7	10	5	4	1	8	2	8
Case III	4	2	7	10	6	4	1	7	2	9
Case IV	6	3	6	10	5	2	1	9	4	8
Case V	3	2	6	9	3	5	1	7	7	9
Σ	23	13	32	49	25	17	5	39	18	42
Rank	5	2	7	10	6	3	1	8	4	9

9.7 การผสม approximate gradient กับ RPBIL

การศึกษาก่อนหน้านี้เป็นการผสมระหว่างขั้นตอนวิธีวิวัฒนาการสองแบบที่แตกต่างกันเพื่อเพิ่มประสิทธิภาพในการหาคำตอบ ในหัวข้อนี้จะนำเสนอวิธีแบบผสมผสานระหว่างขั้นตอนวิธีวิวัฒนาการกับ local search โดยทั่วไปแล้ว ขั้นตอนวิธีวิวัฒนาการจะลู่เข้าใกล้บริเวณที่มีผลเฉลยเหมาะสมที่สุดได้เร็วมาก แต่จะมีปัญหาในการค้นหาคำตอบที่เหมาะสมที่สุด ด้วยเหตุนี้จึงได้มีแนวคิดในการใช้ local search มาช่วยเพิ่มสมรรถนะของขั้นตอนวิธีวิวัฒนาการ ในการศึกษาเลือกใช้ approximate gradient เป็น local search เพื่อช่วยเพิ่มประสิทธิภาพของ RPBIL สำหรับการหาคำตอบที่เหมาะสมที่สุดของโครงถัก

ขั้นตอนวิธีแบบผสมนี้มีกระบวนการคำนวณดังแสดงในขั้นตอนวิธีที่ 9.4 จากการศึกษาก่อนหน้านี้พบว่า RPBIL มีจุดด้อยตรงที่การลู่เข้าสู่คำตอบวงแคบ (local optimum) ก่อนเวลาอันควร ด้วยเหตุนี้ เราจะประยุกต์ใช้กระบวนการมิวเทชันของวิธี evolution strategy (ES) เพื่อเพิ่มความหลากหลายของประชากร ผลเฉลย $\mathbf{x}$ ใดๆ เมื่อผ่านการมิวเทชัน ES จะได้ผลเฉลยใหม่เป็น $\mathbf{x}'$ ซึ่งคำนวณได้จาก

$$\begin{aligned}\sigma'_i &= \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) \\ \mathbf{x}' &= \mathbf{x} + N(0, \sigma')\end{aligned}\quad (9.5)$$

เมื่อ τ' τ และ β คือ ค่าคงที่ ที่จะต้องกำหนดค่าก่อนรัน $N(0, \sigma')$ คือ เวกเตอร์การสุ่มขนาด $n \times 1$ เป็นการกระจายแบบปกติที่มีค่า mean เป็น $\mathbf{0}$ และส่วนเบี่ยงเบนมาตรฐานเป็น σ ค่า $N(0,1)$ หมายถึง ตัวที่สร้างแบบสุ่มด้วยค่าคาดหวังค่าเฉลี่ยเป็น 0 และส่วนเบี่ยงเบนมาตรฐานเป็น 1 ค่า σ คือส่วนเบี่ยงเบนมาตรฐานของประชากรที่สร้างจากเมตริกซ์ความน่าจะเป็นสามมิติ จากนั้น ทำการสุ่มเลือกผลเฉลยจาก $\mathbf{A}$ มา n_B ค่าเพื่อทำการประยุกต์ approximate gradient ดังแสดงในสมการที่ (4.8) – (4.11) ในกรณีของการหาคำตอบที่เหมาะสมที่สุดแบบหลายเป้าหมาย ค่า F ในสมการที่ (4.8) จะหาได้จากวิธีผลรวมถ่วงน้ำหนัก (weighted sum method)

$$F = w_1 f_1 + \dots + w_m f_m \quad (9.6)$$

เมื่อ w_i คือ น้ำหนักถ่วงที่ทำการสุ่มขึ้นมา ในขั้นตอนวิธีที่ 9.4 ฟังก์ชัน *ESmutate* หมายถึงการประยุกต์ใช้สมการ (9.5) ส่วน *AppGrad* หมายถึงการประยุกต์ใช้ approximate gradient ใน (4.8) – (4.11) และ (9.6)

ขั้นตอนวิธีที่ 9.4 Hybrid multiobjective RPBIL

Input: $n, n_T, m, n_I, T, max_eval, n_A, n_B$
Output: Pareto archive $\mathbf{A}$
Initialisation:
 Let $P_{ij,k} = 1/n_I; k = 1, \dots, n_T$
 Generate $\mathbf{X}$ from $P_{ij,k}$ (ขั้นตอนวิธีที่ 9.2).
 Generate a set of neighbour solutions $\mathbf{X}_N$ using a Latin hypercube sampling technique.
 Find $\mathbf{A}$ from non-dominated sorting $\mathbf{X} \cup \mathbf{X}_N$
Main procedure:
 While the total number of function evaluation is less than max_eval , do the following
 If the members in $\mathbf{A}$ are more than n_A
 Do: activate the clustering technique to remove some members from $\mathbf{A}$.
 Divide the members of $\mathbf{A}$ into n_T groups and find the centroid of each group.
 For $i = 1$ to n_T
 Use the centroid of the i -th group to update the i -th probability tray using (9.1)-(9.3).
 Next i
 Generate $\mathbf{X}$ from $P_{ij,k}$ (ขั้นตอนวิธีที่ 9.2).
 Apply ES mutation $\mathbf{X}' = ESmutate(\mathbf{X})$
 Create n_B solutions from an approximate gradient operator & binary crossover $\mathbf{X}'' = AppGrad(\mathbf{A}, \mathbf{X}_N, n_B)$
 Find new $\mathbf{A}$ from non-dominated sorting $\mathbf{X}' \cup \mathbf{A} \cup \mathbf{X}''$

วิธีหาค่าเหมาะที่สุดที่พัฒนาขึ้นมาใหม่นี้จะเขียนแทนด้วย HRPBIL ซึ่งจะมีสองเวอร์ชันคือแบบที่ใช้และไม่ใช้ approximate gradient วิธี MOEAs ต่างๆที่ใช้เปรียบเทียบสมรรถนะมีดังนี้

- BPBIL: binary multiobjective population-based incremental learning using binary codes with mutation probability and mutation shift as 0.05 and 0.2 respectively.
- UPS-EMOA: unrestricted population size evolutionary multiobjective optimisation algorithm (Aittokoski and Miettinen, 2010) using crossover probability, scaling factor, probability of choosing element from offspring in crossover, minimum population size, and burst size as 0.7, [0.6, 0.9], 0.5, 10, and 25 respectively.
- SPEA2: strength Pareto evolutionary algorithm using real codes with crossover and mutation rates of 1.0 and 0.1 respectively.
- MPSO: multiobjective particle swarm optimization using the starting inertia weight, ending inertia weight, cognitive learning factor, and social learning factor as 0.5, 0.01, 0.5 and 0.5 respectively. The best solution for computing a velocity vector is randomly selected from a Pareto archive. The adaptive grid algorithm is used as an archiving technique.
- NSGA-II: non-dominated sorting genetic algorithm using real codes with crossover mutation probabilities as 1.0 and 0.1 respectively.

- DMOEA: dynamic multiobjective evolutionary algorithm using real codes with temperature and mutation rate as 1000 and 1 respectively.
- DEMO: differential evolution for multiobjective optimization using real codes with crossover probability, scaling factor and probability of choosing element from offspring in crossover for DE operators as 0.7, [0.6, 0.9], and 0.5 respectively.
- MOEA/D: multiobjective evolutionary algorithm based on decomposition using real codes with number of neighbouring weight vectors, crossover and mutation probabilities as 6, 1.0, and 0.1 respectively. The weighted sum technique is used for decomposition.
- AMOSA: archived multiobjective simulated annealing. The algorithm performs a simple downhill operator for the first 3 generations before entering the main procedure. In this paper, the method is performed in continuous spaces where a candidate is obtained by mutating on its parent.
- RPBIL: multiobjective population-based incremental learning (ขั้นตอนวิธีที่ 9.2) using real codes with $n_I = 40$ where each probability tray produce 5 design solutions.
- HRPBIL1: RPBIL with ES mutation (ขั้นตอนวิธีที่ 9.4 without approximate gradient and binary crossover).
- HRPBIL2: RPBIL with ES mutation, approximate gradient, and binary crossover (ขั้นตอนวิธีที่ 9.2) using $n_B = 10$. Crossover and mutation probabilities as 0.7 and 1.0 respectively. The probability of choosing an element from offspring is 0.5.

ทำการเลือกปัญหาการออกแบบโครงถัก Case I-IV เพื่อใช้ในการทดสอบประสิทธิภาพของ HRPBIL ขนาดของประชากรและจำนวนลูสำหรับปัญหา Case I-III คือ 50 ตัวและ 50 รอบตามลำดับ ส่วนขนาดของประชากรและจำนวนลูสำหรับปัญหา Case IV คือ 100 ตัวและ 100 รอบตามลำดับ ทำการรันแต่ละวิธี MOEA 30 ครั้งในการหาผลเฉลยพาเรโตของแต่ละปัญหา

9.8 ผลการเปรียบเทียบ

ผลการเปรียบเทียบ MOEAs ต่างๆสำหรับกรณีศึกษาที่แสดงในตารางที่ 9.9-9.10 ในกรณีของ Case I วิธีที่ดีที่สุดอันดับ 1 และ 2 คือ HRPBIL2 และ DEMO ตามลำดับ ส่วนอันดับที่ 3 และ 4 ซึ่งมีประสิทธิภาพใกล้เคียงกัน คือ HRPBIL1 และ RPBIL ตามลำดับ ในกรณีของ Case II วิธีที่ดีที่สุดอันดับ 1 2 3 คือ HRPBIL2 HRPBIL1 และ RPBIL ตามลำดับ สำหรับ Case III ซึ่งมีจำนวนตัวแปรออกแบบมาก วิธีที่ดีที่สุดซึ่งดีกว่าวิธีอื่นอย่างเห็นได้ชัดคือ HRPBIL2 ส่วนวิธีที่ดีอันดับ 2 และ 3 คือ DEMO และ UPS-EMOA ตามลำดับ สำหรับ Case IV วิธีที่ให้ผลเฉลยที่ดีที่สุดคือ DEMO ส่วนวิธีที่ดีอันดับ 2 และ 3 คือ UPS-EMOA และ HRPBIL2 ตามลำดับ จากผลการศึกษาสามารถสรุปได้ว่า approximate gradient ช่วยเพิ่มสมรรถนะให้ RPBIL อย่างมีนัยสำคัญ ส่วนมิวเทชันของ ES ก็มีมีส่วนช่วยในการเพิ่มสมรรถนะของ HRPBIL ทั้งสองเวอร์ชัน

ตารางที่ 9.9 Comparative performance based on a hypervolume indicator

Problem		PBI L	UPS	SPE A2	MPS O	NSG A-II	DMO EA	DEM O	MOEA /D	AMOS A	RPBI L	HRPBI L1	HRPBI L2
Case I	Mean	0.4748	0.9921	0.6697	0.4123	0.8118	0.5880	0.9941	0.7705	0.6955	0.9857	0.9865	0.9992
	STD	0.1451	0.0162	0.1208	0.1929	0.1225	0.1232	0.0176	0.0594	0.1558	0.0010	0.0013	0.0002
	Max.	0.7459	1.0000	0.8824	0.6706	0.9932	0.8943	0.9999	0.8775	0.9179	0.9875	0.9892	0.9994
	Min.	0.1969	0.9336	0.4385	0.0000	0.5359	0.3808	0.9019	0.6372	0.3792	0.9845	0.9845	0.9987
Case II	Mean	0.6198	0.9728	0.5276	0.2496	0.6184	0.2895	0.9745	0.5011	0.5351	0.9589	0.9595	0.9841
	STD	0.1036	0.0161	0.1311	0.1326	0.1003	0.0971	0.0148	0.0972	0.1434	0.0011	0.0014	0.0138
	Max.	0.8137	0.9983	0.7799	0.4505	0.8044	0.4521	1.0000	0.6814	0.8160	0.9614	0.9630	0.9999
	Min.	0.4193	0.9300	0.2694	0.0000	0.4320	0.0713	0.9309	0.2375	0.2423	0.9559	0.9571	0.9604
Case III	Mean	0.4776	0.7470	0.4057	0.1105	0.5084	0.2895	0.7797	0.3637	0.1349	0.4638	0.6687	0.9954
	STD	0.0385	0.0495	0.0577	0.0628	0.0784	0.0405	0.0566	0.0470	0.0851	0.0759	0.1668	0.0022
	Max.	0.5409	0.8296	0.4907	0.2671	0.6265	0.3451	0.8906	0.4574	0.3054	0.5617	0.7997	1.0000
	Min.	0.4034	0.6455	0.2316	0.0033	0.2789	0.1874	0.6574	0.2596	0.0000	0.2499	0.2791	0.9916
Case IV	Mean	0.7882	0.9988	0.8371	0.7157	0.8977	0.7990	0.9978	0.9014	0.9107	0.8637	0.9017	0.9428
	STD	0.0759	0.0008	0.0831	0.1788	0.0530	0.0706	0.0021	0.0281	0.0417	0.1858	0.0654	0.0046
	Max.	0.9305	1.0000	0.9452	0.9263	0.9461	0.9000	0.9991	0.9406	0.9460	0.9248	0.9376	0.9466
	Min.	0.5626	0.9960	0.6405	0.2596	0.7588	0.5700	0.9877	0.8069	0.7275	0.0000	0.6183	0.9268

ตารางที่ 9.10 Overall ranking based on Wilcoxon rank sum test

	PBI L	UP S	SPE A2	MPS O	NSG A-II	DMOE A	DEM O	MOEA/ D	AMOS A	RPBI L	HRPBI L1	HRPBI L2
Case I	11	3	8	11	6	10	1	6	8	5	4	1
Case II	6	2	8	11	6	11	2	8	8	4	4	1
Case III	5	3	8	11	5	10	2	9	11	7	4	1
Case IV	11	1	9	12	4	9	2	4	4	4	4	3
Σ	33	9	33	45	21	40	7	27	31	20	16	6
Ranking	9	3	9	12	6	11	2	7	8	5	4	1

9.9 สรุปและวิจารณ์

จากการศึกษาในบทนี้แสดงให้เห็นว่า การผสมขั้นตอนวิธีวิวัฒนาการนั้นสามารถทำได้หลายรูปแบบ ซึ่งอาจจัดเป็นสองรูปแบบใหญ่ๆคือ การผสมวิธีวิวัฒนาการหลักกับตัวดำเนินการของวิธี

วิวัฒนาการอื่นเช่น RPBIL-DE ในการศึกษาี้ รูปแบบที่สองคือการผสมระหว่างขั้นตอนวิธีวิวัฒนาการหลักกับ local search ซึ่งจะช่วยให้การหาคำตอบเมื่อวิธีวิวัฒนาการเข้าใกล้จุดเหมาะสมที่สุดได้ดีขึ้น นอกจากนี้ตัวอย่างที่ยกมาในบทนี้แล้ว ยังมีตัวอย่างการศึกษาในเรื่องการเพิ่มสมรรถนะของขั้นตอนวิธีวิวัฒนาการด้วยเทคนิคการผสมผสาน ดังรายละเอียดในภาคผนวก

บทที่ 10 สรุปและวิจารณ์

งานวิจัยนี้ เป็นการศึกษาการประยุกต์ขั้นตอนวิธีวิวัฒนาการเพื่อการออกแบบ โครงถัก ใน รายละเอียดของงานสามารถแยกออกเป็นสองหมวดย่อย คือการพัฒนากระบวนการออกแบบ โครงถัก และการพัฒนาสมรรถนะของขั้นตอนวิธีวิวัฒนาการทั้งแบบเป้าหมายเดียวและหลายเป้าหมาย ใน ระหว่างทำการศึกษานี้ได้ทำการขยายการประยุกต์ใช้งานออกไปยังการออกแบบระบบทางวิศวกรรมอื่น ด้วย เพื่อให้การดำเนินการวิจัยเกิดประโยชน์สูงสุด

ได้ทำการวิจัยเพื่อพัฒนากระบวนการออกแบบอัตโนมัติของโครงข้อหมุนและโครงข้อแข็ง กระบวนการออกแบบชนิดนี้คือ การหาทอพอโลยี รูปทรงและขนาดเหมาะสมที่สุดพร้อมกันของ โครงสร้าง การดำเนินการออกแบบลักษณะนี้สามารถทำได้โดย การใช้เทคนิคเอลิเมนต์พื้นซึ่งใน งานวิจัยนี้ได้ศึกษาทั้งในส่วนของเอลิเมนต์พื้นแบบคงตัวและเอลิเมนต์พื้นแบบ adaptive ซึ่งจาก ผลการวิจัยพบว่า เอลิเมนต์พื้นแบบ adaptive จะมีประสิทธิภาพในการออกแบบที่ดีกว่า แต่ก็มีปัญหา เรื่องความยากและสลับซับซ้อนของกระบวนการดังกล่าวคืออุปสรรคที่ตามมาด้วย ประโยชน์ของ กระบวนการออกแบบนี้คือ เป็นกระบวนการที่ช่วยให้ผู้ออกแบบที่มีประสบการณ์ในการออกแบบ โครงสร้างต่ำ สามารถออกแบบโครงสร้างได้ใกล้เคียงหรืออาจจะดีกว่าผู้ออกแบบที่มีประสบการณ์สูง ทั้งนี้ผู้ออกแบบต้องมีความรู้พื้นฐานในด้านการวิเคราะห์โครงสร้างและการหาค่าเหมาะที่สุดที่เพียงพอ

ได้ทำการศึกษาการหาค่าเหมาะที่สุดแบบตรงข้ามของโครงถัก การออกแบบลักษณะนี้มี ประโยชน์คือ เมื่อปัญหาการออกแบบจริงมีพารามิเตอร์ที่ไม่สามารถระบุค่าได้อย่างชัดเจน แต่อาจ สามารถระบุช่วงของค่าของพารามิเตอร์เหล่านั้นได้ ได้ทำการพัฒนาขั้นตอนวิธีวิวัฒนาการสำหรับ ปัญหาการหาค่าเหมาะที่สุดแบบตรงข้าม จากผลการทดสอบพบว่าวิธีที่พัฒนาขึ้นมาใหม่ เป็นวิธีที่มี ประสิทธิภาพสูง สามารถใช้ในการหาค่าเหมาะที่สุดแบบตรงข้ามของโครงถักได้

ได้ทำการศึกษาการเพิ่มสมรรถนะของขั้นตอนวิธีวิวัฒนาการด้วยเทคนิคการผสมผสานสำหรับ การออกแบบโครงถักและระบบทางกลอื่นๆ ได้พัฒนาวิธี PBIL ที่ใช้เลขจำนวนจริง และผสมกับตัว ดำเนินการของวิธีวิวัฒนาการเชิงอนุพันธ์ ซึ่งจะ ได้วิธีแบบผสมที่มีสมรรถนะสูงสำหรับการหาค่าเหมาะ ที่สุดแบบหลายเป้าหมายของโครงถัก นอกจากนี้ยังได้ทำการประยุกต์ RPBIL-DE กับการออกแบบ ระบบโครงข่ายท่อน้ำประปา ซึ่งทำให้ได้ผลเฉลยที่ดีและเป็นการยืนยันสมรรถนะของวิธีผสม RPBIL-DE นอกจากนี้ สำหรับกรณีของ RPBIL ได้พัฒนา local search เพื่อใช้กับวิธี RPBIL ซึ่งสามารถ ช่วยเพิ่มสมรรถนะของ RPBIL ในการหาค่าเหมาะที่สุดแบบหลายเป้าหมายของโครงถัก ดังจะเห็นได้ จากผลการทดสอบ เมื่อเปรียบเทียบกับขั้นตอนวิธีวิวัฒนาการที่มีมาก่อน นอกจากนี้ยังได้สาธิตการ ประยุกต์ใช้ approximate gradient เพื่อเพิ่ม สมรรถนะ MOEAs อื่นๆ ซึ่งก็แสดงให้เห็นว่า

approximate gradient เป็น local search ที่มีประสิทธิภาพสูงช่วยเพิ่มสมรรถนะของ MOEAs ในการหาค่าเหมาะที่สุดแบบหลายเป้าหมายของโครงถักได้

ได้ทำการผสมผสานวิธี real code ant colony optimisation (ACOR) กับวิธี local search ซิมเพิลลิซด์ลงเขา ผลที่ได้ทำให้ ACOR มีสมรรถนะในการหาผลเฉลยการหาค่าเหมาะที่สุดแบบมีเงื่อนไขบังคับของระบบทางกลสูงขึ้น ได้ทำการผสมผสานระหว่างขั้นตอนวิธีวิวัฒนาการสองวิธีที่มีแนวคิดแตกต่างกัน กล่าวคือวิธี differential evolution และวิธี teaching-learning based optimisation เพื่อการหาค่าเหมาะที่สุดของการม้วนแผ่นโลหะซึ่งเป็นปัญหามิติสูงเพราะมีจำนวนตัวแปรออกแบบมาก ผลจากการทดสอบพบว่า วิธีที่พัฒนาขึ้นมามีสมรรถนะสูงกว่าวิธีวิวัฒนาการอื่นที่มีมาก่อน นอกจากนี้ทีมวิจัยยังได้พัฒนาหลักสูตรเกี่ยวกับการหาค่าเหมาะที่สุดของโครงถัก เพื่อใช้เป็นการบริการวิชาการกับชุมชนต่อไป

เอกสารอ้างอิง

- [1]. Achtziger W. On simultaneous optimization of truss geometry and topology. *Structural and Multidisciplinary Optimization* 2007; 33: 285–304.
- [2]. Achtziger W, Kocvara M. On the maximization of the fundamental eigenvalue in topology optimization. *Structural and Multidisciplinary Optimization* 2007; 34:181–195.
- [3]. Achtziger W, Stolpe M. Truss topology optimization with discrete design variables—Guaranteed global optimality and benchmark examples. *Structural and Multidisciplinary Optimization* 2007; 34: 1–20.
- [4]. Achtziger W, Stolpe M. Global optimization of truss topology with discrete bar areas—Part I: theory of relaxed problems. *Comput Optim Appl* 2008; 40: 247–280.
- [5]. Achtziger W, Stolpe M. Global optimization of truss topology with discrete bar areas—Part II: Implementation and numerical results. *Comput Optim Appl* 2009; 44: 315–341.
- [6]. Aittokoski T, Miettinen K. Efficient evolutionary approach to approximate the Pareto optimal set in multiobjective optimization UPS-EMOA. *Optimization Methods and Software* 2010; 25: 841 – 858.
- [7]. Ali NBH, Sellami M, Cutting-Decelle A-F, Mangin J-C. Multi-stage production cost optimization of semi-rigid steel frames using genetic algorithms. *Engineering Structures* 2009; 31: 2766-2778.
- [8]. Alvarez F, Carrasco M. Minimization of the expected compliance as an alternative approach to multiload truss optimization. *Struct Multidisc Optim* 2005; 29: 470–476.
- [9]. Asadpoure A, Tootkaboni M, Guest JK. Robust topology optimization of structures with uncertainties in stiffness – Application to truss structures. *Computers & Structures* 2011; 89: 1131-1141.
- [10]. Ashtari P, Barzegar F. Accelerating fuzzy genetic algorithm for the optimization of steel structures. *Struct Multidisc Optim*, DOI 10.1007/s00158-011-0700-5, 2011.
- [11]. Bandyopadhyay S, Saha S, Maulik U, Deb K. A Simulated annealing-based multiobjective optimization algorithm: AMOSA. *Evolutionary Computation, IEEE Transactions on* 2008; 12: 269-283.
- [12]. Bai Y, de Klerk E, Pasechnik D, Sotirov R. Exploiting group symmetry in truss topology optimization. *Optim Eng* 2009; 10: 331–349.
- [13]. Baluja S. Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Technical Report CMU_CS_95_163, Carnegie Mellon University, 1994.
- [14]. Beekers M, Fleury C. A primal-dual approach in truss topology optimization. *Computers & Structures* 1997; 64: 77-88.
- [15]. Benage WA, Dhingra AK. Single and multiobjective structural optimization in discrete-continuous variables using simulated annealing. *International Journal of Numerical Methods in Engineering* 1994; 38: 2753-2773.
- [16]. Bureerat S. Improved population-based incremental learning in continuous spaces. *Advances in Intelligent and Soft Computing* 2011; 96/2011: 77-86.
- [17]. Bureerat S. Hybrid population-based incremental learning using real codes. *Lecture Notes in Computer Science* 2011; 6683: 379-391.
- [18]. Bureerat S, Limtragool J. Structural topology optimisation using simulated annealing with multiresolution design variables. *Finite Element in Analysis and Design* 2008; 44: 737-748.

- [19]. Bureerat S, Srisomporn S. Optimum plate-fin heat sinks by using a multi-objective evolutionary algorithm. *Engineering Optimization* 2010; 42: 305-343.
- [20]. Bureerat S, Sriworamas K. Population-based incremental learning for multi-objective optimization. *Advances in Soft Computing* 2007; 39: 223-232.
- [21]. Calafiore GC, Dabbene F. Optimization under uncertainty with applications to design of truss structures. *Struct Multidisc Optim* 2008; 35:189–200.
- [22]. Chan CM, Chui JKL. Wind-induced response and serviceability design optimization of tall steel buildings. *Engineering Structures* 2006; 28: 503-513.
- [23]. Chan CM, Huang MF, Kwok KCS. Integrated wind load analysis and stiffness optimization of tall buildings with 3D modes. *Engineering Structures* 2010; 32: 1252-1261.
- [24]. Chan CM, Wong KM. Structural topology and element sizing design optimization of tall steel frameworks using a hybrid OC–GA method, *Structural and Multidisciplinary Optimization* 2008; 35: 473-488.
- [25]. Chen XZ, Zhang XJ, Ren ZZ. The Study on the Full Stress Optimal Design of Shape Steel Truss. *Advanced Materials Research* 2011; 308-310: 490-493.
- [26]. Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGAII. *IEEE Trans. On Evolutionary Computation* 2002; 6: 182-197.
- [27]. Deb K, Pratap A, Meyarivan T. Constrained test problems for multi-objective evolutionary optimization. KanGAL Report No. 200002, Kanpur Genetic Algorithms Laboratory (KanGAL), Indian Institute of Technology, Kanpur, India.
- [28]. Dominguez A, Stiharu I, Sedaghati R. Practical design optimization of truss structures using the genetic algorithms, *Research Engineering Design* 2006; 17: 73–84.
- [29]. Fonseca CM, Fleming PJ. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In: *Proc. of the 5th Inter. Conf. on Gas*, 1993, pp. 416-423.
- [30]. Gandomi AH, Yang XS. Benchmark problems in structural optimization. *Comput. Optimization, Methods and Algorithms*, Chapter 12, SCI 356, pp. 259–281.
- [31]. Gang L, Ren-Gen Z, Lian D, Wai-Fah C. Multiobjective and multilevel optimization for steel frames. *Engineering Structures* 1999; 21: 519-529.
- [32]. Gao J, Wng J. WBMOAIS: A novel artificial immune system for multiobjective optimization. *Computers & Operations Research* 2010; 37: 50 – 61.
- [33]. Giger M., Ermanni P. Evolutionary truss topology optimization using a graph-based parameterization concept. *Structural and Multidisciplinary Optimization* 2006; 32: 313–326.
- [34]. Gil L, Andreu A. Shape and cross-section optimization of a truss structure. *Computers & Structures* 2000; 79: 681-689.
- [35]. Gomes HM. Truss optimization with dynamic constraints using a particle swarm algorithm. *Expert Systems with Application* 2011; 38: 957-968.
- [36]. González-Vidoso F, Yepes V, Alcalá J, Carrera M, Perea C, Payá-Zaforteza I. Optimization of reinforced concrete structures by simulated annealing. *Simulated Annealing*, Book edited by: Cher Ming Tan, ISBN 978-953-7619-07-7, pp. 420, February 2008, I-Tech Education and Publishing, Vienna, Austria
- [37]. Guerra A, Kiousis PD. Design optimization of reinforced concrete structures. *Computers and Concrete* 2006; 3: 313-334.
- [38]. Guest JK, Igusa T. Structural optimization under uncertain loads and nodal locations. *Comput. Methods Appl. Mech. Engrg.* 2008; 198: 116–124.

- [39]. Hagishita T, Ohsaki M. Topology optimization of trusses by growing ground structure method. *Structural and Multidisciplinary Optimization* 2009; 37: 377-393.
- [40]. Hajela P, Lee E. Genetic algorithms in truss topology optimization. *International Journal of Solids and Structures* 1995; 32: 3341-3357.
- [41]. Hajirasouliha I, Pilakoutas K, Moghaddam H. Topology optimization for the seismic design of truss-like structures. *Computers & Structures* 2011; 89: 702-711.
- [42]. Hansen N, Müller SD, Koumoutsakos P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation* 2003; 11: 1-18.
- [43]. Hasanc O, Erbatur F. Layout optimization of trusses using simulated annealing. *Advances in Engineering Software* 2002; 33: 681-696.
- [44]. Herrera F, Lozano M, Molona D. Continuous scatter search: an analysis of the integration of some combination methods and improvement strategies. *European Journal of Operational Research* 2006; 169: 450-476.
- [45]. Izui K, Nishiwaki S, Yoshimura M, Nakamura M, Renaud JE. Enhanced multiobjective particle swarm optimization in combination with adaptive weighted gradient-based searching. *Engineering Optimization* 2008; 40: 789-804.
- [46]. Kalanta S, Atkociunas J, Venskus A. Discrete optimization problems of the steel bar structures. *Engineering Structures* 2009; 31: 1298-1304.
- [47]. Kanyakam S, Bureerat S. Passive vibration suppression of a walking tractor handlebar structure using multiobjective PBIL. *IEEE Congress on Evolutionary Computation (CEC 2007)*, 2007, pp. 4162-4169, Singapore.
- [48]. Karkauskas R, Norkus A. Truss optimization under stiffness, stability constraints and random loading. *Mechanics Research Communications* 2006; 33: 177-189.
- [49]. Kaveh A, Hassani B, Shojaee S, Tavakkoli SM. Structural topology optimization using ant colony methodology. *Engineering Structures* 2008; 30: 2559-2565.
- [50]. Kaveh A, Talatahari S. A hybrid particle swarm and ant colony optimization for design of truss structures. *Asian Journal of Civil Engineering (Building and Housing)* 2008; 9: 329-348.
- [51]. Kaveh A, Talatahari S. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. *Computers and Structures* 2009; 87: 267-283.
- [52]. Kaveh A, Talatahari S. Size optimization of space trusses using Big Bang-Big Crunch algorithm. *Computers and Structures* 2009; 87: 1129-1140.
- [53]. Kaveh A, Talatahari S. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. *Computers & Structures* 2009; 87: 267-283.
- [54]. Kaveh A, Talatahari S. A partial swarm ant colony optimization for truss structures with discrete variables. *Journal of Constructional Steel Research* 2009; 65: 1558-1568.
- [55]. Kaveh A, Talatahari S. An improved ant colony optimization for the design of planar steel frames. *Engineering Structures* 2010; 32: 864-873.
- [56]. Kaveh A, Talatahari S. A Novel heuristic optimization method: charged system search. *Acta Mechanica* 2010; 213: 267-289.
- [57]. Kaveh A, Talatahari S. Optimization of large-scale truss structures using modified charged system search. *Int. J. of Optimization in Civil Engineering* 2011; 1: 15-28.
- [58]. Kawamura H, Ohmori H, Kito N. Truss topology optimization by a modified genetic algorithm. *Structural and Multidisciplinary Optimization* 2002; 23: 467-472.

- [59]. Kelesoglu O. Fuzzy multiobjective optimization of truss-structures using genetic algorithm. *Advances in Engineering Software* 2007; 38: 717–721.
- [60]. Kirkpatrick S, Gelatt Jr. CD, Vecchi MP. Optimization by simulated annealing. *Science* 1983; 220: 671-680.
- [61]. Knowles JD, Corne DW. Approximating the non-dominated front using the Pareto archive evolution strategy. *Evolutionary Computation* 2000; 8: 149-172.
- [62]. Kripakaran P, Hall B, Gupta A. A genetic algorithm for design of moment-resisting steel frames. *Struct Multidisc Optim*, DOI 10.1007/s00158-011-0654-7, 2011.
- [63]. Lagarosa ND, Fragiadakis M, Papadrakakis M, Tsompanakis Y. Structural optimization: A tool for evaluating seismic design procedures. *Engineering Structures* 2006; 28: 1623–1633.
- [64]. Lamberti L. An efficient simulated annealing algorithm for design optimization of truss structures. *Computers & Structures* 2008; 86: 1936-1953.
- [65]. Lee KS, Han SW, Geem ZW. Discrete size and discrete-continuous configuration optimization methods for truss structures using the harmony search algorithm. *International Journal of Optimization in Civil Engineering* 2001; 1: 107-126.
- [66]. Lemonge ACC, Barbosa HJC, Coutinho ALGA, Borges CCH. Multiple cardinality constraints and automatic member grouping in the optimal design of steel framed structures. *Engineering Structures* 2011; 33: 433-444.
- [67]. Leps M. Multi-objective optimization of reinforced concrete frames. In: 6th World Congresses of Structural and Multidisciplinary Optimization, Rio de Janeiro, 30 May - 03 June 2005, Brazil
- [68]. Li LJ, Huang ZB, Liu F. A heuristic particle swarm optimization method for truss structures with discrete variables. *Computers and Structures* 2009; 87: 435-443.
- [69]. Li JQ, Pan QK, Gao KZ. Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. *Int J Adv Manuf Technol*, DOI 10.1007/s00170-010-3140-2, 2011.
- [70]. Li G, Zhou RG, Duan L, Chen WF. Multiobjective and multilevel optimization for steel frames. *Engineering Structures* 1999; 21: 519–529.
- [71]. Liu M, Wen YK, Burns SA. Life cycle cost oriented seismic design optimization of steel moment frame structures with risk-taking preference. *Engineering Structures* 2004; 26: 1407-1421.
- [72]. Long HV, Hung ND. Plastic optimization of 3D steel frames under fixed or repeated loading: Reduction formulation. *Engineering Structures* 2010; 32: 1092-1099.
- [73]. Liu X, Cheng G, Yan J, Jiang L. Singular optimum topology of skeletal structures with frequency constraints by AGGA. *Struct Multidisc Optim*, DOI 10.1007/s00158-011-0708-x, 2011.
- [74]. Luh GC, Lin CY. Optimal design of truss structures using ant algorithm. *Structural and Multidisciplinary Optimization* 2008; 36: 365-379.
- [75]. Makrodimopoulos A, Bhaskar A, Keane AJ. Second-order cone programming formulations for a class of problems in structural optimization. *Struct Multidisc Optim* 2010; 40: 365–380.
- [76]. Martínez P, Martí P, Querin OM. Growth method for size, topology, and geometry optimization of truss structures. *Structural and Multidisciplinary Optimization* 2007; 33: 13–26.
- [77]. Noiluplao C, Bureerat S. Simultaneous topology and shape optimisation of a 3D framework structure using simulated annealing. In: *Technology and Innovation for*

Sustainable Development Conference (TISD2008), Faculty of Engineering, Khon Kaen University, Thailand, 28-29 January 2008.

- [78]. Noilublao C, Bureerat S. Simultaneous topology shape and sizing optimisation of skeletal structures using multiobjective evolutionary algorithms. *Evolutionary Computation*, In-Tech, ISBN 978-953-307-008-7, pp. 487-580, 2009.
- [79]. Noilublao C, Bureerat S. Topology and sizing optimization of trusses with adaptive ground finite elements using multiobjective PBIL. *Advanced Materials Research* 2011; 308-310: 1116-1121.
- [80]. Noilublao N, Bureerat S. Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms. *Computers and Structures* 2011; in-press.
- [81]. Noman N, Iba H. Accelerating differential evolution using an adaptive local search. *IEEE Transaction on Evolutionary Computation* 2008; 12: 107-125.
- [82]. Ohsaki M. Simultaneous optimization of topology and geometry of a regular plane truss. *Computers & Structures* 1998; 66: 69-77.
- [83]. Ohsaki M, Katoh N. Topology optimization of trusses with stress and local constraints on nodal stability and member intersection. *Structural and Multidisciplinary Optimization* 2005; 29: 190-197.
- [84]. Oskoue AV, Sahand Sarioletlagh Fard SS, Aksogan O. Using genetic algorithm for the optimization of seismic behavior of steel planar frames with semi-rigid connections. *Struct Multidisc Optim*, DOI 10.1007/s00158-011-0697-9, 2011.
- [85]. Papadrakakis M, Lagaros ND, Plevris V. Design optimization of steel structures considering uncertainties. *Engineering Structures* 2005; 27: 1408-1418.
- [86]. Paya I, Yepes V, González-Vidos F, Hospitaler A. Multiobjective optimization of concrete frames by simulated annealing. *Computer-Aided Civil and Infrastructure Engineering* 2008; 23: 596-610.
- [87]. Pelikan M, Muhlenbein H. The bivariate marginal distribution algorithm. *Advances in Soft Computing - Engineering Design and Manufacturing* 1999: 521-535.
- [88]. Perea C, Baitsch M, González-Vidos F, Hartmann D. Optimization of reinforced concrete frame bridges by parallel genetic and memetic algorithms. *Structural Engineering, Mechanics and Computation* 3, ISBN 9789059660540, pp. 1790-1795.
- [89]. Perez RE, Behdinan K. Particle swarm approach for structural design optimization. *Computers and Structures* 2007; 85: 1579-1588.
- [90]. Plevris V, Papadrakakis M. A hybrid particle swarm-gradient algorithm for global structural optimization. *Computer-Aided Civil and Infrastructure Engineering* 2011; 26: 48-68.
- [91]. Rahami H, Kaveh A, Gholipour Y. Sizing, geometry and topology optimization of trusses via force method and genetic algorithm. *Engineering Structures* 2008; 30: 2360-2369.
- [92]. Rajeev S, Krishnamoorthy CS. Genetic algorithm-based methodology for design optimization of reinforced concrete frames. *Computer-Aided Civil and Infrastructure Engineering* 1998; 13: 63-74.
- [93]. Rasmussen MH, Stolpe M. Global optimization of discrete truss topology design problems using a parallel cut-and-branch method. *Computers & Structures* 2008; 86: 1527-1538.
- [94]. Reyes-Sierra M, Coello Coello CA. Multi-objective particle swarm optimizers: a survey of the state-of-the-art. *International Journal of Computational Intelligence Research* 2006; 2: 287-308.

- [95]. Schaffer JD. Multiobjective optimization with vector evaluated genetic algorithms. In: GAs and their Application: Proc. of 1st Inter Conf. on Gas, 1985, pp. 93-100.
- [96]. Shamie J, Schmit Jr LA. Frame optimization including frequency constraints. *Journal of the Structural Division* 1975; 101: 283-293.
- [97]. Shea K, Smith IFC. Improving full-scale transmission tower design through topology and shape optimization. *Journal of Structural Engineering* 2006; 132: pp. 781–790.
- [98]. Silih S, Premrov M, Kravanja S. Optimum design of plane timber trusses considering joint flexibility. *Engineering Structures* 2005; 27: 145-154.
- [99]. Sikiotis ES, Saouma VE. Optimum design of reinforced concrete frames using interactive computer graphics. *Engineering with Computers* 1987; 3: 101-110.
- [100]. Simon D. Biogeography-based optimization. *IEEE Transaction on Evolutionary Computation* 2008; 12: 702–713.
- [101]. Sivasubramani S, Swarup KS. Environmental/economic dispatch using multi-objective harmony search algorithm. *Electric Power Systems Research* 2011; 81: 1778–1785.
- [102]. Socha K, Dorigo M. Ant colony optimization for continuous domains. *European Journal of Operational Research* 2008; 185: 1155-1173.
- [103]. Sokol T. A 99 line code for discretized Michell truss optimization written in Mathematica. *Struct Multidisc Optim* 2011; 43:181–190.
- [104]. Sonmez M. Artificial bee colony algorithm for optimization of truss structures. *Applied Soft Computing* 2011; 11: 2406–2418.
- [105]. Srinivas N, Deb K. Multiobjective optimization using non-dominated genetic algorithms. *Evolutionary Computation* 1994; 2: 221-248.
- [106]. Stolpe M, Svanberg K. A note on stress-constrained truss topology optimization. *Structural and Multidisciplinary Optimization* 2003; 25: 62–64.
- [107]. Storn R, Price K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.* 1997; 11: 341–359.
- [108]. Su R, Wang X, Gui L, Fan Z. Multi-objective topology and sizing optimization of truss structures based on adaptive multi-island search strategy. *Structural and Multidisciplinary Optimization* 2011; 43: 275–286.
- [109]. Svanberg K, Werme M. Sequential integer programming methods for stress constrained topology optimization. *Structural and Multidisciplinary Optimization* 2007; 34: 277–299.
- [110]. Tan Y, Zhu Y. Fireworks algorithm for optimization. *Advances in Swarm Intelligence* 2010; 6145/2010: 355–364.
- [111]. Tang W, Tong L, Gu Y. Improved genetic algorithm for design optimization of truss structures with sizing, shape and topology variables. *International Journal for Numerical Methods in Engineering* 2005; 62: 1737 – 1762.
- [112]. Teh YS, Rangaiah GP. Tabu search for global optimization of continuous functions with application to phase equilibrium calculations. *Computers and Chemical Engineering* 2003; 27: 1665-1679.
- [113]. Togan V, Daloglu A. Optimization of 3d trusses with adaptive approach in genetic algorithms. *Engineering Structures* 2006; 28: 1019-1027.
- [114]. Tyas A, Gilbert M, Pritchard T. Practical plastic layout optimization of trusses incorporating stability considerations. *Computers & Structures* 2006; 84: 115-126.
- [115]. Venanzi I, Materazzi AL. Multi-objective optimization of wind-excited structures. *Engineering Structures* 2007; 29: 983-990.

- [116]. Wang FW, Chan SL. Optimization and sensitivity analysis of space frames allowing for large deflection. *Engineering Structures* 2006; 28: 1395-1406.
- [117]. Wang H, Ohmori H. Truss optimization using genetic algorithm, considering construction process. *International Journal of Space Structures* 2010; 25: 205-215.
- [118]. Wang X, Wang MY, Guo D. Structural shape and topology optimization in a level-set-based framework of region representation. *Structural and Multidisciplinary Optimization* 2004; 27: 1–19.
- [119]. Wang D, Zhang WH, Jiang JS. Truss optimization on shape and sizing with frequency constraints. *AIAA* 2004; 42: 622-630.
- [120]. Wei L, Tang T, Xie X, Shen W. Truss optimization on shape and sizing with frequency constraints based on parallel genetic algorithm. *Structural and Multidisciplinary Optimization* 2011; 43: 665–682.
- [121]. Wu CY, Tseng KY. Truss structure optimization using adaptive multi-population differential evolution. *Structural and Multidisciplinary Optimization* 2010; 42: 575–590.
- [122]. Xu B, Jiang J, Tong W, Wu K. Topology group concept for truss topology optimization with frequency constraints. *Journal of Sound and Vibration* 2003; 261: 911–925.
- [123]. Xu T, Zuo W, Xu T, Song G, Li R. An adaptive reanalysis method for genetic algorithm with application to fast truss optimization. *Acta Mech Sin* 2010; 26: 225-234.
- [124]. Yamamoto K, Inoue O. New evolutionary direction operator for genetic algorithms. *AIAA* 1995; 33: 1990-1993.
- [125]. Yonekura K, Kanno Y. Global optimization of robust truss topology via mixed integer semidefinite programming. *Optim Eng* 2010; 11: 355–379.
- [126]. Yunkang S, Xin Y. The topological optimization for truss structures with stress constraints based on the exist-null combined model. *ACTA MECHANICA SINICA (English Series)* 1998; 14: 363-370.
- [127]. Zhou M. Difficulties in truss topology optimization with stress and local buckling constraints. *Structural Optimization* 1996; 11: 134-136.
- [128]. Zhou M, Pagaldipti N, Thomas HL, Shyy YK. An integrated approach to topology, sizing, and shape optimization. *Structural and Multidisciplinary Optimization* 2004; 26: 308–317.
- [129]. Zhu J, Zhang W, Beckers P, Chen Y, Guo Z. Simultaneous design of components layout and supporting structures using coupled shape and topology optimization technique. *Structural and Multidisciplinary Optimization* 2008; 36: 29–41.
- [130]. Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation* 2000; 8: 173-195.
- [131]. Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: *Evolutionary Methods for Design, Optimization and Control*, Barcelona, Spain, 2002.
- [132]. Zitzler E, Thiele L, Laumanns M, Fonseca CM, Fonseca VG. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE trans. on Evolutionary Computation* 2003; 7: 117-132.
- [133]. Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions On Evolutionary Computation* 1999; 3: 257-271.
- [134]. Zou XK, Chan CM, Li G, Wang Q. Multiobjective optimization for performance-based design of reinforced concrete frames. *Journal of Structural Engineering* 2007; October: 1462-1474.

Output จากโครงการวิจัย

บทความที่ตีพิมพ์ในวารสารที่มี Impact factor

1. Pholdee, N., Park, W.-W., Kim, D.-K., Im, Y.-T., Bureerat, S., Kwon, H.-C., Chun, M.-S. Efficient hybrid evolutionary algorithm for optimization of a strip coiling process (2015) Engineering Optimization, 47 (4), pp. 521-532.
2. Pholdee, N., Bureerat, S. Hybrid real-code population-based incremental learning and approximate gradients for multi-objective truss design (2014) Engineering Optimization, 46 (8), pp. 1032-1051.
3. Pholdee, N., Bureerat, S. Hybrid real-code ant colony optimisation for constrained mechanical design (2014) International Journal of Systems Science, . Article in Press.
4. Pholdee, N., Bureerat, S. Comparative performance of meta-heuristic algorithms for mass minimisation of trusses with dynamic constraints (2014) Advances in Engineering Software, 75, pp. 1-13.
5. Bureerat, S., Sriworamas, K. Simultaneous topology and sizing optimization of a water distribution network using a hybrid multiobjective evolutionary algorithm (2013) Applied Soft Computing Journal, 13 (8), pp. 3693-3702.
6. Tontragunrat, P., Bureerat, S. Antioptimisation of trusses using two-level population-based incremental learning (2013) Journal of Applied Mathematics, 2013, art. no. 434636, .
7. Noilublao, N., Bureerat, S. Simultaneous topology, shape, and sizing optimisation of plane trusses with adaptive ground finite elements using MOEAs (2013) Mathematical Problems in Engineering, 2013, art. no. 838102, .
8. Pholdee, N., Bureerat, S. Hybridisation of real-code population-based incremental learning and differential evolution for multiobjective design of trusses (2013) Information Sciences, 223, pp. 136-152.
9. Pholdee, N., Bureerat, S. Performance enhancement of multiobjective evolutionary optimisers for truss design using an approximate gradient (2012) Computers and Structures, 106-107, pp. 115-124.
10. Noilublao, N., Bureerat, S. Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms (2011) Computers and Structures, 89 (23-24), pp. 2531-2538.

การนำผลงานวิจัยไปใช้ประโยชน์

- เชิงสาธารณะ ส่วนหนึ่งขององค์ความรู้จากโครงการวิจัยนี้ได้นำไปพัฒนาเป็นหลักสูตรอบรมในหัวข้อ truss optimisation ซึ่งจะดำเนินการเปิดอบรมสำหรับวิศวกรในปีการศึกษา 2558
- เชิงวิชาการ คณาจารย์บัณฑิตด้านการหาค่าเหมาะที่สุดของโครงสร้างทางวิศวกรรมโยธา 2 คน (จบการศึกษาแล้ว)

- เจริงวิชาการ เนื้อหาและองค์ความรู้จากงานวิจัยนี้ ถูกบรรจุในวิชา Optimisation ซึ่งเป็นวิชาเรียนในระดับบัณฑิตศึกษา ภาควิชาวิศวกรรมเครื่องกล มหาวิทยาลัยขอนแก่น นอกจากนี้ยังนำองค์ความรู้และเนื้อหาเหล่านี้ บรรจุในหนังสือ “การหาค่าเหมาะที่สุดของระบบทางวิศวกรรมเครื่องกล” แต่งโดยผู้วิจัย ซึ่งได้ทำการเผยแพร่ในปี 2555

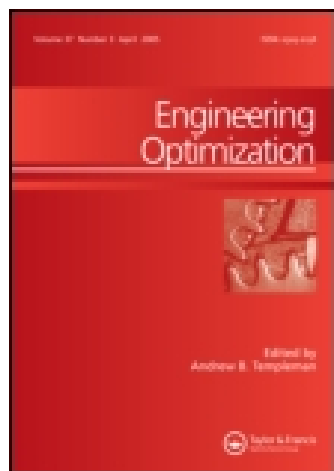
ภาคผนวก

This article was downloaded by: [Sujin Bureerat]

On: 28 April 2015, At: 04:27

Publisher: Taylor & Francis

Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



CrossMark

[Click for updates](#)

Engineering Optimization

Publication details, including instructions for authors and subscription information:

<http://www.tandfonline.com/loi/geno20>

Efficient hybrid evolutionary algorithm for optimization of a strip coiling process

Nantiwat Pholdee^a, Won-Woong Park^b, Dong-Kyu Kim^c, Yong-Taek Im^b, Sujin Bureerat^a, Hyuck-Cheol Kwon^d & Myung-Sik Chun^d

^a Sustainable Infrastructure Research and Development Centre, Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen, Thailand

^b National Research Laboratory for Computer Aided Materials Processing, Department of Mechanical Engineering, KAIST, Daejeon, Republic of Korea

^c Neutron Science Division, KAERI, Daejeon, Republic of Korea

^d Rolling & Instrumentation Research Group, Technical Research Labs, Jeonnam, Republic of Korea

Published online: 16 Apr 2014.

To cite this article: Nantiwat Pholdee, Won-Woong Park, Dong-Kyu Kim, Yong-Taek Im, Sujin Bureerat, Hyuck-Cheol Kwon & Myung-Sik Chun (2015) Efficient hybrid evolutionary algorithm for optimization of a strip coiling process, *Engineering Optimization*, 47:4, 521-532, DOI: [10.1080/0305215X.2014.905551](https://doi.org/10.1080/0305215X.2014.905551)

To link to this article: <http://dx.doi.org/10.1080/0305215X.2014.905551>

PLEASE SCROLL DOWN FOR ARTICLE

Taylor & Francis makes every effort to ensure the accuracy of all the information (the "Content") contained in the publications on our platform. However, Taylor & Francis, our agents, and our licensors make no representations or warranties whatsoever as to the accuracy, completeness, or suitability for any purpose of the Content. Any opinions and views expressed in this publication are the opinions and views of the authors, and are not the views of or endorsed by Taylor & Francis. The accuracy of the Content should not be relied upon and should be independently verified with primary sources of information. Taylor and Francis shall not be liable for any losses, actions, claims, proceedings, demands, costs, expenses, damages, and other liabilities whatsoever or howsoever caused arising directly or indirectly in connection with, in relation to or arising out of the use of the Content.

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden. Terms & Conditions of access and use can be found at <http://www.tandfonline.com/page/terms-and-conditions>

Efficient hybrid evolutionary algorithm for optimization of a strip coiling process

Nantiwat Pholdee^a, Won-Woong Park^b, Dong-Kyu Kim^c, Yong-Taek Im^{b*}, Sujin Bureerat^a,
Hyuck-Cheol Kwon^d and Myung-Sik Chun^d

^aSustainable Infrastructure Research and Development Centre, Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen, Thailand; ^bNational Research Laboratory for Computer Aided Materials Processing, Department of Mechanical Engineering, KAIST, Daejeon, Republic of Korea; ^cNeutron Science Division, KAERI, Daejeon, Republic of Korea; ^dRolling & Instrumentation Research Group, Technical Research Labs, Jeonnam, Republic of Korea

(Received 17 May 2013; accepted 14 February 2014)

This article proposes an efficient metaheuristic based on hybridization of teaching–learning-based optimization and differential evolution for optimization to improve the flatness of a strip during a strip coiling process. Differential evolution operators were integrated into the teaching–learning-based optimization with a Latin hypercube sampling technique for generation of an initial population. The objective function was introduced to reduce axial inhomogeneity of the stress distribution and the maximum compressive stress calculated by Love's elastic solution within the thin strip, which may cause an irregular surface profile of the strip during the strip coiling process. The hybrid optimizer and several well-established evolutionary algorithms (EAs) were used to solve the optimization problem. The comparative studies show that the proposed hybrid algorithm outperformed other EAs in terms of convergence rate and consistency. It was found that the proposed hybrid approach was powerful for process optimization, especially with a large-scale design problem.

Keywords: strip coiling; evolutionary optimizers; hybrid algorithm; flatness defects; spool crown

1. Introduction

In a steel mill, thin sheet metal production with high quality and low cost is a major customer requirement. In industry, it is not usually easy to achieve this goal because of equipment and operating conditions. In addition, the cold shape of the thin strip is not easy to predict because manufacturing of the hot-rolled thin strip consists of several processes (Jung and Im 1997, 1999; Kwon, Han, and Chun 2011).

Strip coiling is a process applied at the end of a rolling line in metal strip manufacturing. During the strip coiling process, flatness defects such as wavy edge or centre buckle caused by axial inhomogeneity of the stress distribution within a thin strip are usually encountered. To better understand this phenomenon, a physical model to describe coil deformation was presented by Yanagi *et al.* (1998). In this work, they found that flatness defects can be avoided by reducing coiling tension and decreasing the strip crown or increasing its exponent. The stress analysis of

*Corresponding author. Email: ytim@kaist.ac.kr

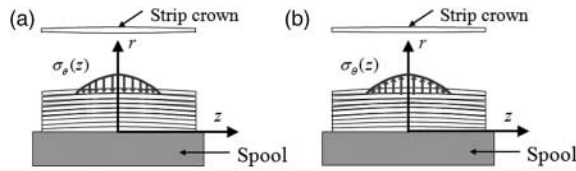


Figure 1. Circumferential stress distributions for the cases of (a) wavy edge defect and (b) centre buckle defect.

the coiling process was revisited by [Park et al. \(2014\)](#) to include the effect of the crown shape of the spool in the analysis.

In summary, manufacturing a flat strip without the crown shape is the best solution for preventing flatness defects on the metal strip during the strip coiling process. However, it is almost impossible to make flat strips in industry. Decreasing the strip crown or increasing its exponent, as shown in Figure 1, is difficult to achieve because of production constraints. Therefore, processing parameters such as spool geometry and coiling tension were selected as controlling parameters to improve the strip flatness during the strip coiling process in the present work. To determine the optimal values of these processing parameters, various optimization studies are necessary.

In general, optimization algorithms can be classified into two categories: those with a function derivative, also known as gradient-based methods; and those that do not apply a function derivative, for example evolutionary algorithms (EAs). Compared with gradient-based methods, EAs are easy to use and can be applied more generally to deal with all kinds of functions and design variables owing to the non-gradient requirement. Because of this advantage, EAs are widely used in various real-world engineering applications ([Bureerat and Limtragool 2008](#); [Yildiz 2009d, 2012](#); [Tiwari, Oduguwa, and Roy 2008](#); [Zhang and Wang 2012](#); [Gholizadeh and Barzegar 2013](#)). However, EAs also have some weaknesses due to the random search procedure. A lack of search consistency and low convergence rates are typical in EA search procedures, particularly for large-scale design problems. Therefore, hybridization of EAs to enhance their search performance has become popular ([Pholdee and Bureerat 2012, 2013](#); [Srisoporn and Bureerat 2008](#); [Kaveh and Talatahari 2009](#); [Plevris and Papadrakakis 2011](#); [Yildiz 2009b, 2009c, 2013a, 2013c](#); [Zidani et al. 2013](#); [Cagnina, Esquivel, and Coello 2011](#); [Ghiasi, Pasini, and Lessard 2011](#)).

Hybrid EAs can be classified into three categories: hybrid EAs with surrogate models ([Srisoporn and Bureerat 2008](#); [Sun et al. 2010](#)), incorporating EAs with a local search technique ([Plevris and Papadrakakis 2011](#); [Zidani et al. 2013](#); [Ghiasi, Pasini, and Lessard 2011](#); [Pholdee and Bureerat 2012](#); [Yildiz 2009a](#)) and hybridization of two EAs ([Kaveh and Talatahari 2009](#); [Pholdee and Bureerat 2013](#)). A hybrid of EAs with a surrogate model is a powerful tool for an optimization study requiring a large computation time to calculate objective function evaluations, as in the optimization of a metal forming process ([Sun et al. 2010](#)). The basic concept of this hybrid approach is to use the surrogate model as an approximate function evaluation instead of a real objective function evaluation. The results of this hybrid approach will depend on the accuracy of the surrogate model and sampling points used to construct the surrogate model. For the hybrid of EAs and local search techniques, also known as a mimetic algorithm, the basic concept is to use the EAs to provide a pseudo-global search region. Then, a local search technique is used to speed up the search procedure. However, this kind of hybrid approach includes a local search technique, which usually requires a function derivative ([Plevris and Papadrakakis 2011](#)), such that it is difficult to apply to real engineering applications. Moreover, a local search technique without applying a function derivative, such as the Nelder–Mead simplex downhill method (NM), is not efficient for problems with large-scale design variables ([Ghiasi, Pasini, and Lessard 2011](#)). In this regard, the hybrid of two EAs may be more flexible to enhance the search performance of the EAs in real engineering optimization studies. Successful implementation of this kind of hybrid

algorithm in real engineering applications can be found in the literature (Pholdee and Bureerat 2013; Kaveh and Talatahari 2009; Nwankwor, Nagar, and Reid 2013; Yildiz 2013b).

In this work, a new hybrid algorithm based on integrating differential evolution (DE) operators into the search procedure of teaching–learning-based optimization (TLBO) is developed and applied to optimization of the strip coiling process. The hybrid optimizer will be termed enhanced teaching–learning-based optimization with differential evolution (ETLBO-DE). The ETLBO-DE and several well-established EAs such as DE (Storn and Price 1997), simulated annealing (SA) (Bureerat and Limtragool 2008), artificial bee colony optimization (ABC) (Karaboga and Basturk 2007), real code ant colony optimization (ACOR) (Socha and Dorigo 2008), TLBO (Rao, Savsani, and Vakharia 2011), league championship algorithm (LCA) (Kashan 2011), charged system search (ChSS) (Kaveh and Talatahari 2010), hybrid DE and Nelder–Mead simplex method (HDENM) (Wu *et al.* 2010), hybrid DE and hill-climbing (HDEHC) (Hernandez, Leguizamón, and Mezura-Montes 2013), and hybrid particle swarm optimization and DE (HPSODE) (Liu, Cai, and Wang 2010) are employed to solve a large-scale design problem of the strip coiling process in order to improve the flatness of the strip during the process. For this purpose, the optimal processing parameters of spool geometry and coiling tension were determined by introducing the objective function to minimize axial inhomogeneity of the stress distribution and the maximum stress, which may cause the irregular surface profile of the strip during the coiling process.

2. Formulation of optimization design problem of the strip coiling process

During the strip coiling process, a wavy edge will form when the circumferential stress at the middle zone of the strip is highly compressed while the two edges are tensile or lightly compressed. On other hand, if the middle zone of the strip is highly tensile while the two edges are compressed or lightly tensile, a centre buckle will occur, according to previous studies (Yanagi *et al.* 1998; Park *et al.* 2014). Figure 1(a) and (b) shows the circumferential stress σ_θ distributions along the z direction within a thin strip that incurred a wavy edge and a centre buckle, respectively.

In the strip coiling process, when the strips are coiled, the middle zone ($z = 0$) of the strip at the inner coil will be highly compressed compared with the two edges owing to the coiling tension and strip crown. In this case, the centre buckle at the inner coil is unlikely and the wavy edge is likely to form. Therefore, the wavy edge defect at the inner coil becomes a major issue during the strip coiling process. Figure 2 shows the plot of the circumferential stress σ_θ distribution in the

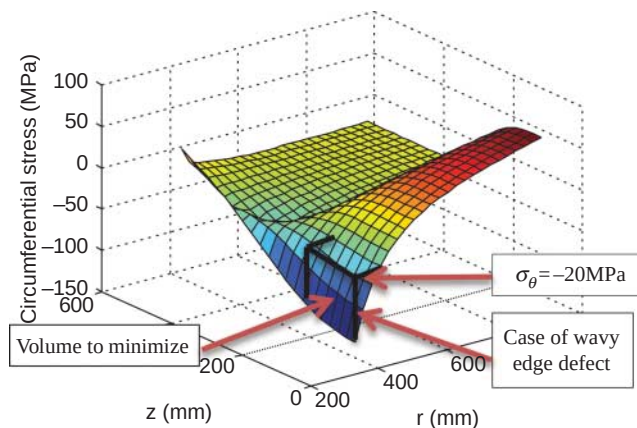


Figure 2. Circumferential stress distribution σ_θ in the coil determined by Love's elastic solution by Park *et al.* (2014).

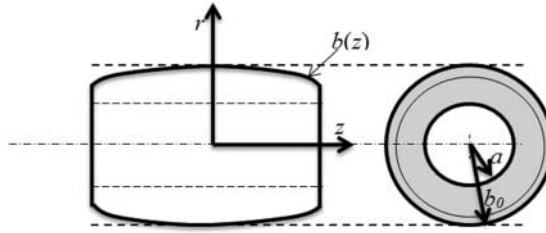


Figure 3. Spool geometry used in the present investigation.

z direction at each radius r of the coil calculated by Love's elastic solution, determined by [Park et al. \(2014\)](#), which may contribute to the formation of the wavy edge defect after finishing the coiling process.

It was found from Figure 2 that formation of the wavy edge defect can be decreased by reducing the axial inhomogeneity of the stress distribution and the maximum compressive stress in the compressive zone. In this study, optimization using EAs will be applied to find the optimal processing parameters including coiling tension σ_T and spool geometry, as described in Figure 3.

To reduce axial inhomogeneity of the stress distribution and the maximum compressive stress, the objective function was defined to minimize the maximum compressive stress and the volume of the circumferential compressive stress, as shown in Figure 2. In this figure, the volume was calculated only for the coil in which compressive stresses were higher than 20 MPa, to minimize the likeliness of the wavy edge defect.

Then, the objective function can be expressed as:

$$\begin{aligned}
 \text{Min: } f(\alpha_b, \eta_b, \sigma_{T,i}) &= \frac{V}{V_0} + \frac{\max(\sigma_{\theta c})}{\max(\sigma_{\theta c0})} \\
 \text{Subject to: } 0 &\leq \alpha_b \leq 4, \\
 0 &\leq \eta_b \leq 4, \\
 25 &\leq \sigma_{T,i} \leq 50 \text{ MPa; } i = 1 \cdots n_{\max} \\
 |\sigma_{T,i} - \sigma_{T,i-1}| &\leq 2 \text{ MPa}
 \end{aligned} \tag{1}$$

where V and $\sigma_{\theta c}$ are approximate values of the circumferential stress and the compressive circumferential stress higher than 20 MPa, respectively (see Figure 2), while V_0 and $\sigma_{\theta c0}$ are the values of the original design case. $\sigma_{T,i}$ is the coiling tension at the coil number i (generally, the coiling tension was set to be constant for all the coils). $n_{\max}$ is the maximum number of coils, which was set to be 220 in this study, and α_b , η_b are the spool crown and spool crown exponent, which were used in defining the spool geometry in Figure 3 as:

$$b(z) = b_0 - \alpha_b \left(\frac{|z|}{z_{\max}} \right)^{\eta_b} \tag{2}$$

In this equation, $b(z)$ and b_0 ($=330$ mm) are the outer radius of the spool along the z direction and the initial value of the outer radius of the spool, respectively, and $z_{\max} = 525$ mm is the width of the spool. The inner radius of the spool a in Figure 3 was set to be 300 mm. The total number of design variables considered was 222 (220 for the coiling tensions $\sigma_{T,i}$ and two for the spool geometry α_b and η_b).

3. Teaching–learning-based optimization

In the optimization algorithm based on TLBO proposed by [Rao, Savsani, and Vakharia \(2011\)](#), the search process starts with an initialized population. In this work, the population is considered as a group of learners and the best individual is a teacher. The knowledge is an objective function, which needs to be improved towards the optimum. The optimization process starts by randomly creating an initial population. Then, objective function values of the members in the population are evaluated and the best individual is identified. After that, the population is updated in two stages: the ‘teacher phase’ and the ‘learner phase’. During the teacher phase, an individual ($\mathbf{x}_i$) is updated based on the best individual ($\mathbf{x}_{\text{teacher}}$) and the mean values of the whole population ($\mathbf{x}_{\text{mean}}$), as follows:

$$\mathbf{x}_{\text{new},i} = \mathbf{x}_{\text{old},i} + r\{\mathbf{x}_{\text{teacher}} - (T_F \mathbf{x}_{\text{mean}})\} \quad (3)$$

where $r \in [0, 1]$ is a uniform random number and T_F is a teaching factor, which can be either 1 or 2.

For the learner phase, the population is updated by interactions among the individuals. Two individuals $\mathbf{x}_i$ and $\mathbf{x}_j$ are selected randomly, where $i \neq j$. The modification can be expressed as:

$$\mathbf{x}_{\text{new},i} = \begin{cases} \mathbf{x}_{\text{old},i} + r(\mathbf{x}_i - \mathbf{x}_j) & \text{if } f(\mathbf{x}_i) < f(\mathbf{x}_j) \\ \mathbf{x}_{\text{old},i} + r(\mathbf{x}_j - \mathbf{x}_i) & \text{if } f(\mathbf{x}_j) < f(\mathbf{x}_i) \end{cases} \quad (4)$$

During both the teacher and learner phases, the new solution ($\mathbf{x}_{\text{new}}$) will be accepted if it is better than the old ones; otherwise, it will be rejected. The search process is repeated until the termination criterion is satisfied. The detailed TLBO computational steps are summarized in Algorithm 1 (see below).

4. Hybrid of teaching–learning-based optimization and differential evolutionary algorithm

The DE algorithm is a simple but efficient evolutionary algorithm proposed by [Storn and Price \(1997\)](#). The DE algorithm has two main operators to update populations, *i.e.* mutation and crossover. In this work, the DE/best/2/bin strategy was used for the hybrid algorithm. The DE mutation operator can generate a new population ($\mathbf{x}_{\text{new},i}$) using the following equation:

$$\mathbf{x}_{\text{new},i} = \mathbf{x}_{\text{best}} + F(\mathbf{x}_{r,1} + \mathbf{x}_{r,2} - \mathbf{x}_{r,3} - \mathbf{x}_{r,4}) \quad (5)$$

Algorithm 1 Teaching–learning-based optimization (TLBO)

Input: Maximum iteration number *maxiter*, population size n_p .

Output: $\mathbf{x}_{\text{best}}$, f_{best}

Initialization

1. Generate an initial population randomly.
2. Evaluate objective function values.

Main algorithm

3. For $i = 1$ to *maxiter*
 - 3.1 Identify the best solution ($\mathbf{x}_{\text{teacher}}$).
(Teacher phase)
For $j = 1$ to n_p
 - 3.2 Update the population using Equation (3)
 - 3.2.1 Evaluate the objective function value $f(\mathbf{x}_{\text{new},j})$

```

3.2.2 If  $f(\mathbf{x}_{\text{new},j}) < f(\mathbf{x}_{\text{old},j})$ 
    Replace  $\mathbf{x}_{\text{old},j}$  by  $\mathbf{x}_{\text{new},j}$ 
End
End
(Learner phase)
For  $j = 1$  to  $n_p$ 
3.3 Update the population using Equation (4)
    3.3.1 Evaluate the objective function value  $f(\mathbf{x}_{\text{new},j})$ 
    3.3.2 If  $f(\mathbf{x}_{\text{new},j}) < f(\mathbf{x}_{\text{old},j})$ 
        Replace  $\mathbf{x}_{\text{old},j}$  by  $\mathbf{x}_{\text{new},j}$ 
    End
End
End

```

where F is a scaling factor and $\mathbf{x}_{r,i}$ are four randomly selected individuals. For a new individual $\mathbf{x}_{\text{new}}^T = [x_{\text{new},1}, \dots, x_{\text{new},D}]$ and an old individual $\mathbf{x}_{\text{old}}^T = [x_{\text{old},1}, \dots, x_{\text{old},D}]$, the binary crossover step can be expressed as:

$$x_{\text{new},j} = x_{\text{old},j} \text{ if } \text{rand} < \text{CR}, j = 1, \dots, D \quad (6)$$

The computational steps of the proposed ETLBO-DE hybrid algorithm are described in Algorithm 2. The hybridized TLBO and DE can be obtained in such a way that TLBO is used as a main search procedure. The process starts by randomly generating an initial population. The populations will then be updated based on the best solution from the TLBO's teacher phase, similar to Algorithm 1. However, after the teacher phase, the DE/best/2/bin strategy will be used to update the population instead of using the learner phase of the original TLBO.

DE is used as an alternative for the local search instead of classical optimization methods such as gradient-based optimizers because objective and constraint function values are evaluated through a numerical approach, which usually has some truncation errors during the computational steps. This may lead to difficulty and inaccuracy in calculating function derivatives. Moreover, direct methods such as the simplex downhill are reported to be less effective than DE in solving a large-scale optimization problem with 222 design variables.

5. Numerical experiments

To examine the search performance of the proposed ETLBO-DE hybrid algorithm, several EAs were used to compare the objective function values, which were determined by Equation (1). The EAs used in this study are as follows:

DE (Storn and Price 1997): The DE/best/2/bin strategy was used. The DE scaling factor was random, from 0.25 to 0.7, in each calculation and the crossover probability was 0.7.

Algorithm 2 Newly developed ETLBO-DE hybrid algorithm

Input: Maximum iteration number *maxiter*, population size n_p .

Output: $\mathbf{x}_{\text{best}}$, f_{best}

Initialization

1. Generate an initial population randomly.
2. Evaluate objective function values.

Main algorithm

3. For $i = 1$ to *maxiter*
 - 3.1 Identify the best solution ($\mathbf{x}_{\text{teacher}}$).

```

(Teacher phase)
For  $j = 1$  to  $n_p$ 
3.2 Update the population using equation (3)
    3.2.1 Evaluate the objective function value  $f(\mathbf{x}_{new,j})$ 
    3.2.2 If  $f(\mathbf{x}_{new,j}) < f(\mathbf{x}_{old,j})$ 
        Replace  $\mathbf{x}_{old,j}$  by  $\mathbf{x}_{new,j}$ 
    End
End
(DE operation)
For  $j = 1$  to  $n_p$ 
3.3 DE mutation using Equation (5)
For  $k = 1$  to  $D$ 
3.4 DE crossover using Equation (6)
End
3.5 Evaluate the objective function value  $f(\mathbf{x}_{new,j})$ 
3.6 If  $f(\mathbf{x}_{new,j}) < f(\mathbf{x}_{old,j})$ 
    Replace  $\mathbf{x}_{old,j}$  by  $\mathbf{x}_{new,j}$ 
End
End
End

```

SA (Bureerat and Limtragool 2008): An annealing temperature was reduced exponentially by 10 times from the value of 10 to 0.001 in the optimization searching process. On each loop $2n$ children were created by means of mutation to be compared with their parent. Here, n is the number of design variables.

ABC (Karaboga and Basturk 2007): The number of food sources was set to be $3n$. A trial counter to discard a food source was 100.

ACOR (Socha and Dorigo 2008): The parameters used for computing the weighting factor and the standard deviation in the algorithm were set to be $\xi = 1.0$ and $q = 0.2$, respectively.

TLBO (Rao, Savsani, and Vakharia 2011): There is no required parameter setting.

LCA (Kashan 2011): The default parameter settings provided by the authors were used. The lower limit for the retreat coefficient and approach coefficient was set to be 0 while the upper limit for the retreat coefficient and approach coefficient was set to be 2. The probability of success (P_c) was set to be 0.9999 and the decreasing rate of P_c was set to be 0.9995.

ChSS (Kaveh and Talatahari 2010): The number of solutions in the charge memory was $0.2N_p$. Here, N_p is the population size. The charged moving considering rate and the parameter PAR were set to be 0.75 and 0.5, respectively.

HDENM (Wu *et al.* 2010): The DE/rand/1/bin strategy was used. The DE scaling factor and crossover probability were set to be 0.5 and 0.9, respectively. The NM reflection factor, expansion factor, contraction factor and shrinkage factor were 1.0, 2.0, 0.5 and 0.5, respectively.

HDEHC (Hernandez, Leguizamon, and Mezura-Montes 2013): The DE/rand/1/bin strategy was used. The DE scaling factor and crossover probability were set to be 0.6. The hill-climbing algorithm used the same algorithm as presented in Hernandez, Leguizamon, and Mezura-Montes (2013).

HPSODE (Liu, Cai, and Wang 2010): The PSO starting inertia weight and ending inertia weight were set to be 0.5 and 0.01, respectively, while the cognitive learning factor and social learning factor were set to be 0.5. The DE/rand/1/bin, DE/current to best/1/bin and DE/rand/2/bin strategies were used. The DE scaling factor was random, from 0.25 to 0.75, in each calculation and the crossover probability was 0.8.

HTLBO-HJ: After completing the TLBO's learning phase for each generation, the pattern search (PS) method was used as a local search technique to update the current optimum solution. Thus, TLBO was used as the main search algorithm.

HTLBO-NM: After completing the TLBO's learning phase for each generation, the NM method was used as a local search technique to update the current optimum solution. Thus, TLBO was used as the main search algorithm.

ETLBO-DE1 (Algorithm 2): The DE parameter setting was used for the DE operation while an initial population was randomly generated.

ETLBO-DE2 (Algorithm 2): The ETLBO-DE1 parameter setting and Latin hypercube sampling (LHS) technique were used to generate an initial population.

Fourteen evolutionary optimizers were used to solve the proposed optimization design problem of the strip coiling process with five optimization runs. Both the population size and the number of iterations were set to be 100. For the optimizers using different population sizes, such as simulated annealing and TLBO, their search processes were terminated with the total number of function evaluations as 100×100 . The optimal results of the optimizers within a limited number of function evaluations were compared to determine the best performer. It should be noted that the optimization parameter setting used for each optimizer was obtained by selecting the best set from the test runs with several sets of parameters.

6. Results and discussion

The numerical results for the objective function values obtained from each optimizer are shown in Table 1. In this table, the mean values (Mean) are used to measure the convergence speed while the standard deviation (STD) represents the search consistency of the method. A method with a lower mean value of the objective function has better convergence speed, while a method with a lower STD has better search consistency.

For the measure of convergence speed based on the mean objective value, the best convergence method was ETLBO-DE2 and the worst was ABC. The second and third best performers were ETLBO-DE1 and DE, respectively. For the measure of search consistency based on the STD, ETLBO-DE2 had the best consistency while ABC had the worst consistency, similar to the search convergence. The second and third best consistencies were HDENM and ACOR, respectively. The minimum objective function value was obtained by ETLBO-DE2.

Table 1. Objective function values calculated.

Evolutionary algorithm	Mean	STD	Max.	Min.
DE	0.9700	0.0275	1.0096	0.9354
ABC	1.7637	0.0787	1.8800	1.6751
ACOR	1.0621	0.0070	1.0705	1.0546
ChSS	1.4026	0.0289	1.4448	1.3678
LCA	1.7116	0.0408	1.7580	1.6473
SA	1.5451	0.0645	1.6323	1.4841
TLBO	0.9915	0.0132	1.0066	0.9766
HDEHC	1.1460	0.0126	1.1286	1.1592
HDENM	1.2609	0.0050	1.2567	1.2693
HPSODE	1.2562	0.0470	1.1987	1.3130
HTLBO-HJ	1.0025	0.0922	1.1338	0.8729
HTLBO-NM	1.1146	0.1826	1.1328	0.9477
ETLBO-DE1	0.9026	0.0107	0.9134	0.8858
ETLBO-DE2	0.8850	0.0047	0.8897	0.8784

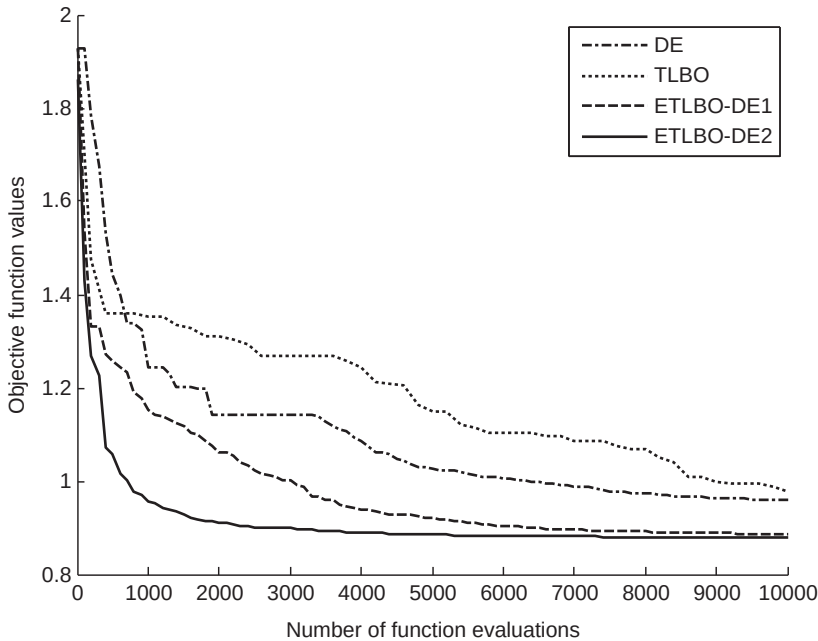


Figure 4. Search history of the original evolutionary algorithms and the hybrid algorithms.

Among the four hybrid TLBO algorithms, the results indicate that the hybrid TLBO with DE is more efficient than the hybrid TLBO using other classic local search methods, *i.e.* pattern search and the simplex downhill method. This implies that DE is a more powerful local search technique for the present optimization problem. All the hybrid versions of TLBO using DE were superior to the original TLBO. Based on the average values of the objective function, the hybrid TLBOs using PS and NM were inferior to the original TLBO, although they provided better results for some optimization runs.

Figure 4 shows a search history of the best optimization run of the original TLBO, DE and the proposed hybrid algorithms. It was found from this figure that the convergence rates of search processes of the hybrid algorithms were better than the original TLBO and DE from the beginning. From this figure, it is noticed that the original TLBO has a faster convergence rate than the original DE from the start because the TLBO has two steps for updating its population. For search processes larger than 1000 function evaluations, however, the convergence rate of DE became faster than TLBO. Accordingly, it can be said that TLBO is better when approaching to the near optimum area whereas DE is efficient for a local search of the real optimum point. Therefore, combining both optimizers can compensate for the weak points of each. The hybrid algorithms use the features of TLBO to quickly land on the area near the optimum. Then, the DE operators are used for fine searching of the optimum solution. This combination makes the hybrid algorithms more efficient than their original techniques.

Of the new optimizers, ETLBO-DE2 is superior to ETLBO-DE1. ETLBO-DE2 can converge to the optimum result after a few function evaluations, whereas ETLBO-DE1 needs more function evaluations to reach the optimum. Overall, ETLBO-DE2 was found to be the best optimization algorithm investigated in this study for solving strip coiling process optimization problems with large-scale design variables. It can be seen from the present investigation that integrating DE into the main procedure of TLBO leads to an efficient evolutionary algorithm, and the

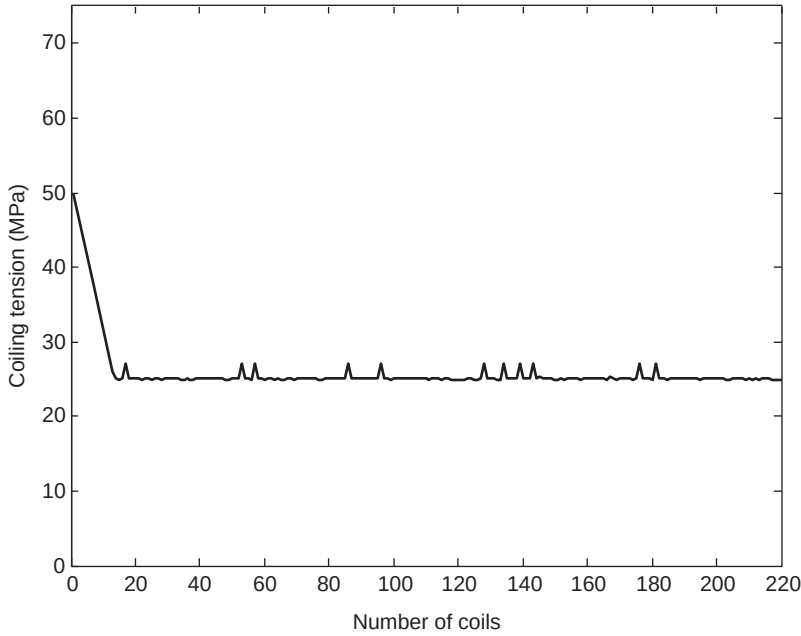


Figure 5. Coiling tension levels as a function of the number of coils.

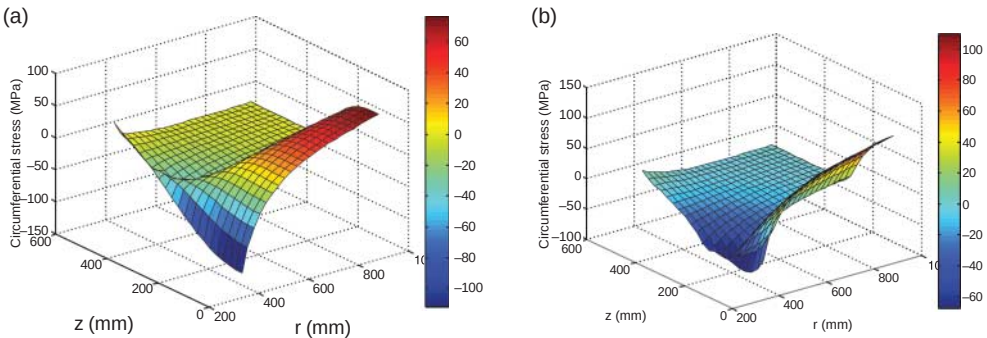


Figure 6. Comparison of circumferential stresses along the z and r directions for (a) the original design and (b) the optimal design.

search performance will be made much more efficient by introducing LHS to generate the initial population in the newly developed hybrid algorithm.

Based on the best optimization run of ETLBO-DE2, the optimal spool crown and spool crown exponent obtained were found to be 1.96 and 1.00, respectively, while the optimal coiling tensions obtained as a function of number of coils are illustrated in Figure 5. The results show that the coiling tensions started with the highest value at the first coiling and decreased when the number of coils increased. After a few series of coiling, the tension levels became almost constant, converging to the minimum value at the end of the process.

Figure 6 (a) and (b) shows the plot of the circumferential stress distributions along the z and r directions before and after the optimization, respectively. Table 2 shows a comparison of the maximum compressive stresses and the standard deviation of stresses at the inner strip between the original and optimal designs. The results indicate that the optimal processing parameters obtained by the ETLBO-DE2 algorithm can reduce the maximum compressive stress and axial

Table 2. Maximum compressive stress and standard deviation of the stresses at the inner coil.

	Original design	Optimal design
Maximum compressive stress (MPa)	111.546	67.248
Standard deviation of stresses	48.375	28.283

inhomogeneity of the stress distribution at the inner strip, which may cause wavy edge defects during the strip coiling process.

7. Conclusions

In this work, a novel hybrid algorithm of TLBO and a DE algorithm was developed for optimization of the strip coiling process. The objective function was formulated to minimize the volume of the circumferential stress and maximum compressive stress higher than 20 MPa during the strip coiling process. Ten existing EAs were used to examine the proposed hybrid algorithm and obtain the optimal processing parameters, such as the coiling tensions $\sigma_{T,i}$ and the spool geometry α_b and η_b , based on minimization of the proposed objective function with a limited number of function evaluations. In the present study, the hybrid of TLBO and DE outperformed the other EAs. The search performance of the proposed hybrid algorithm can be augmented by applying LHS to generate an initial population. The optimal processing parameters such as spool crown α_b and spool crown exponent η_b obtained from the best optimization run of the best performer were found to be 1.96 and 1.00, respectively, while the optimal coiling tensions $\sigma_{T,i}$ were obtained as a function of the number of coils. The present results show that the maximum compressive stress and axial inhomogeneity of the stress distribution at the inner coil which caused flatness defects in the strip coiling process were greatly reduced compared with the original case. The currently proposed optimization scheme is powerful in handling large-scale design problems for industrial applications. The optimum processing parameters obtained will be helpful in reducing edge wave defects during the strip coiling process to improve product quality.

Funding

The authors are grateful for the support from the Royal Golden Jubilee PhD Program [grant no. PHD/0041/2551]. Sujin acknowledges the support from the Thailand Research Fund [BRG5580017]. The research grant from POSCO was very much appreciated as well.

References

- Bureerat, S., and J. Limtragool. 2008. "Structural Topology Optimisation Using Simulated Annealing with Multiresolution Design Variables." *Finite Elements in Analysis and Design* 44: 738–747.
- Cagnina, L. C., S. C. Esquivel, and C. A. C. Coello. 2011. "Solving Constrained Optimization Problems with a Hybrid Particle Swarm Optimization Algorithm." *Engineering Optimization* 43: 843–866.
- Ghiassi, H., D. Pasini, and L. Lessard. 2011. "A Non-dominated Sorting Hybrid Algorithm for Multi-objective Optimization of Engineering Problems." *Engineering Optimization* 43 (1): 39–59.
- Gholizadeh, S., and A. Barzegar. 2013. "Shape Optimization of Structures for Frequency Constraints by Sequential Harmony Search Algorithm." *Engineering Optimization* 45 (6): 627–646.
- Hernandez, S., G. Leguizamón, and E. Mezura-Montes. 2013. "A Hybrid Version of Differential Evolution with Two Differential Mutation Operators Applied by Stages." Proceedings of 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico.
- Jung, J. Y., and Y. T. Im. 1997. "Simulation of Fuzzy Shape Control for Cold-Rolled Strip with Randomly Irregular Strip Shape." *Journal of Material Processing Technology* 63: 248–253.

- Jung, J. Y., and Y. T. Im. 1999. "Fuzzy Control Algorithm for Prediction of Tension Variations in Hot Rolling." *Journal of Material Processing Technology* 96: 163–172.
- Karaboga, D., and B. Basturk. 2007. "A Powerful and Efficient Algorithm for Numerical Function Optimisation: Artificial Bee Colony (ABC) Algorithm." *Journal of Global Optimisation* 39: 459–171.
- Kashan, A. H. 2011. "An Efficient Algorithm for Constrained Global Optimisation and Application to Mechanical Engineering Design: League Championship Algorithm (LCA)." *Computer-Aided Design* 43: 1769–1792.
- Kaveh, A., and S. Talatahari. 2009. "A Particle Swarm Ant Colony Optimization for Truss Structures with Discrete Variables." *Journal of Construction Steel Research* 65: 1558–1568.
- Kaveh, A., and S. Talatahari. 2010. "A Novel Heuristic Optimisation Method: Charged System Search." *Acta Mechanica* 213: 267–289.
- Kwon, H. C., I. S. Han, and M. S. Chun. 2011. "Examination of Thermal Behavior of Hot Rolled Coil Based on the Finite Element Modeling and Thermal Measurement, in Metal Forming." Proceedings of ICTP, Aachen, Germany, 37–40.
- Liu, H., Z. Cai, and Y. Wang. 2010. "Hybridizing Particle Swarm Optimization with Differential Evolution for Constrained Numerical and Engineering Optimization." *Applied Soft Computing* 10 (2): 629–640.
- Nwankwor, E., A. K. Nagar, and D. C. Reid. 2013. "Hybrid Differential Evolution and Particle Swarm Optimization for Optimal Well Placement." *Computational Geosciences* 17 (2): 249–268.
- Park, W. W., D. K. Kim, H. C. Kwon, Y. T. Im, and M. S. Chun. 2014. "The Effect of Processing Parameters on Elastic Deformation of the Coil During the Thin-Strip Coiling Process." in press, *Metals and Materials International*.
- Pholdee, N., and S. Bureerat. 2012. "Performance Enhancement of Multiobjective Evolutionary Optimizers for Truss Design Using an Approximate Gradient." *Computers and Structures* 106–107: 115–124.
- Pholdee, N., and S. Bureerat. 2013. "Hybridisation of Real-Code Population-Based Incremental Learning and Differential Evolution for Multiobjective Design of Trusses." *Information Sciences* 223: 136–152.
- Plevris, V., and M. Papadrakakis. 2011. "A Hybrid Particle Swarm-Gradient Algorithm for Global Structural Optimization." *Computer-Aided Civil and Infrastructure Engineering* 26: 48–68.
- Rao, R. V., V. J. Savsani, and D. P. Vakharia. 2011. "Teaching–Learning-Based Optimisation: A Novel Method for Constrained Mechanical Design Optimisation Problems." *Computer-Aided Design* 43: 303–315.
- Socha, K., and M. Dorigo. 2008. "Ant Colony Optimisation for Continuous Domains." *European Journal of Operational Research* 185: 1155–1173.
- Srisoporn, S., and S. Bureerat. 2008. "Geometrical Design of Plate-Fin Heat Sinks Using Hybridization of MOEA and RSM." *IEEE Transactions on Components and Packaging Technologies* 31: 351–360.
- Storn, R., and K. Price. 1997. "Differential Evolution—A Simple and Efficient Heuristic for Global Optimisation Over Continuous Spaces." *Journal of Global Optimisation* 11: 341–359.
- Sun, G., G. Li, Z. Gong, X. Cui, X. Yang, and Q. Li. 2010. "Multiobjective Robust Optimization Method for Drawbead Design in Sheet Metal Forming." *Materials and Design* 31: 1917–1929.
- Tiwari, A., V. R. Oduguwa, and R. Roy. 2008. "Rolling System Design Using Evolutionary Sequential Process Optimization." *IEEE Transactions on Evolutionary Computation* 12: 196–202.
- Wu, L., Y. Wang, X. Yuan, and S. Zhou. 2010. "A Hybrid Simplex Differential Evolution Algorithm." Proceedings of the Control and Decision Conference (CCDC), Xuzhou, China, 3039–3045.
- Yanagi, S., S. Hattori, Y. Maeda, H. Ibata, Y. Sugimoto, and M. Sawada. 1998. "Coil Deformation and Flatness Change through Strip Coiling Process." The 7th International Conference on Steel Rolling, Japan, 150–155.
- Yildiz, A. R. 2009a. "An Effective Hybrid Immune–Hill Climbing Optimization Approach for Solving Design and Manufacturing Optimization Problems in Industry." *Journal of Materials Processing Technology* 209 (6): 2773–2780.
- Yildiz, A. R. 2009b. "A New Design Optimization Framework Based on Immune Algorithm and Taguchi's Method." *Computers in Industry* 60 (8): 613–620.
- Yildiz, A. R. 2009c. "A Novel Hybrid Immune Algorithm for Global Optimization in Design and Manufacturing." *Robotics and Computer-Integrated Manufacturing* 25: 261–270.
- Yildiz, A. R. 2009d. "A Novel Particle Swarm Optimization Approach for Product Design and Manufacturing." *International Journal of Advanced Manufacturing Technology* 40: 617–628.
- Yildiz, A. R. 2012. "A New Hybrid Particle Swarm Optimization Approach for Structural Design Optimization in Automotive Industry." *Journal of Automobile Engineering* 226 (D10): 1340–1351.
- Yildiz, A. R. 2013a. "Hybrid Taguchi–Differential Evolution Algorithm for Optimization of Multi-pass Turning Operations." *Applied Soft Computing* 13 (3): 1433–1439.
- Yildiz, A. R. 2013b. "A New Hybrid Differential Evolution Algorithm for the Selection of Optimal Machining Parameters in Milling Operations." *Applied Soft Computing* 13 (3): 1561–1566.
- Yildiz, A. R. 2013c. "Optimization of Multi-pass Turning Operations Using Hybrid Teaching Learning-Based Approach." *International Journal of Advanced Manufacturing Technology* 66 (9–12): 1319–1326.
- Zhang, J., and Y. Wang. 2012. "Defection Recognition of Cold Rolling Strip Steel Based on ACO Algorithm with Quantum Action." *Lecture Notes in Computer Science* 7145/2012: 263–271.
- Zidani, H., E. Pagnacco, R. Sampaio, R. Ellaia, and J. E. Souza de Cursi. 2013. "Multi-objective Optimization by a New Hybridized Method: Applications to Random Mechanical Systems." *Engineering Optimization* 45 (8): 917–939.



Engineering Optimization

Publication details, including instructions for authors and
subscription information:

<http://www.tandfonline.com/loi/geno20>

Hybrid real-code population-based incremental learning and approximate gradients for multi-objective truss design

Nantiwat Pholdee^a & Suvin Bureerat^a

^a Department of Mechanical Engineering, Faculty of Engineering,
Khon Kaen University, Khon Kaen, Thailand

Published online: 19 Sep 2013.

To cite this article: Nantiwat Pholdee & Suvin Bureerat (2014) Hybrid real-code population-based
incremental learning and approximate gradients for multi-objective truss design, Engineering
Optimization, 46:8, 1032-1051, DOI: [10.1080/0305215X.2013.823194](https://doi.org/10.1080/0305215X.2013.823194)

To link to this article: <http://dx.doi.org/10.1080/0305215X.2013.823194>

PLEASE SCROLL DOWN FOR ARTICLE

Taylor & Francis makes every effort to ensure the accuracy of all the information (the
"Content") contained in the publications on our platform. However, Taylor & Francis,
our agents, and our licensors make no representations or warranties whatsoever as to
the accuracy, completeness, or suitability for any purpose of the Content. Any opinions
and views expressed in this publication are the opinions and views of the authors,
and are not the views of or endorsed by Taylor & Francis. The accuracy of the Content
should not be relied upon and should be independently verified with primary sources
of information. Taylor and Francis shall not be liable for any losses, actions, claims,
proceedings, demands, costs, expenses, damages, and other liabilities whatsoever or
howsoever caused arising directly or indirectly in connection with, in relation to or arising
out of the use of the Content.

This article may be used for research, teaching, and private study purposes. Any
substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing,
systematic supply, or distribution in any form to anyone is expressly forbidden. Terms &
Conditions of access and use can be found at [http://www.tandfonline.com/page/terms-
and-conditions](http://www.tandfonline.com/page/terms-and-conditions)



Hybrid real-code population-based incremental learning and approximate gradients for multi-objective truss design

Nantiwat Pholdee and Sujin Bureerat*

Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen, Thailand

(Received 27 September 2012; accepted 25 June 2013)

In this article, real-code population-based incremental learning (RPBIL) is extended for multi-objective optimization. The optimizer search performance is then improved by integrating a mutation operator of evolution strategies and an approximate gradient into its computational procedure. RPBIL and its variants, along with a number of established multi-objective evolutionary algorithms, are then implemented to solve four multi-objective design problems of trusses. The design problems are posted to minimize structural mass and compliance while fulfilling stress constraints. The comparative results based on a hypervolume indicator show that the proposed hybrid RPBIL is the best performer for the large-scale truss design problems.

Keywords: population-based incremental learning; hybrid evolutionary algorithms; multi-objective truss optimization; comparative performance; multi-objective evolutionary algorithms

1. Introduction

A truss is one of the most used engineering structures for several reasons, including the ease and low cost of construction. The structures can be used in a wide variety of applications such as bridges, roofs and towers. Over the past few decades, a great deal of research on design optimization of truss structures has been done (*e.g.* Kim and de Weck 2006; Hasançebi 2007; Li, Huang, and Liu 2009; Xu *et al.* 2010; Jansen and Perez 2011; Su *et al.* 2011; Chen and Huang 2013; Gholizadeh and Barzegar 2013). The design problems can be categorized as sizing (Kaveh and Zolghadr 2012; Lu *et al.* 2013; Sadollah *et al.* 2012), shape (Fadel Miguel and Fadel Miguel 2012; Gholizadeh and Barzegar 2013; Pholdee and Bureerat 2012) and topology optimization (Su *et al.* 2011; Richardson *et al.* 2012). It is also possible to operate all three types of truss optimization simultaneously, which is called automated design (Rahami, Kaveh, and Gholipour 2008; Noilublao and Bureerat 2011; Torii, Lopez, and Biondini 2012). Work involving the development and improvement of optimizers for such structural design is one of the most important issues. A typical single-objective design problem of trusses is assigned to minimize structural mass subject to safety requirements such as stress and displacement constraints. The optimizers used for tackling the design problem could be gradient-based optimizers or meta-heuristics, while the most popular method for the latter category is a genetic algorithm (Lingyun *et al.* 2005; Rahami, Kaveh, and Gholipour 2008; Gomes

*Corresponding author. Email: sujbur@kku.ac.th

2011). Other meta-heuristics or evolutionary algorithms employed to solve single-objective truss design problems include particle swarm optimization (Li, Huang, and Liu 2009; Gomes 2011), simulated annealing (Bennage and Dhingra 1994), charged system search (Kaveh and Talatahari 2010) and differential evolution (Kaveh and Talatahari 2008; Brest and Mauèec 2011; Mallipeddi *et al.* 2011).

In practice, it has been found that minimizing structural mass subject to structural safety constraints is somewhat difficult to implement. This is due to the fact that minimizing mass often results in a reduction in structural strength and consequently leads to structural reliability being decreased. As a result, multi-objective optimization is introduced to overcome such a drawback. By implementing a multi-objective design problem, a designer can optimize both structural mass and reliability (Noilublao and Bureerat 2011). Parameters that can be used to measure structural reliability are compliance minimization, structural natural frequencies (Bennage and Dhingra 1994) and frequency response function (Noilublao and Bureerat 2011).

Evolutionary algorithms (EAs) are recognized as alternative optimizers to classical gradient-based optimization methods. Using EAs is advantageous as they are simple to implement, robust and capable of searching for global optima. EAs can be used to solve almost any kind of optimization problem since they search for optimum solutions without using function derivatives. Most importantly, for multi-objective optimization, multi-objective evolutionary algorithms (MOEAs) can search for a Pareto optimal set within one optimization run. These reasons make EAs popular and they are implemented in a wide range of applications. Nevertheless, their weak points, a low convergence rate and a lack of consistency, still exist. Therefore, EA performance enhancement is always demanding. Since the first EA was invented, there have been numerous EAs presented in the literature, where some are newly invented with new concepts and others are extended from older versions. There are several ways to enhance the search performance of EAs. The hybridization of various promising EA operators leading to a new hybrid algorithm is one effective strategy (Kaveh and Talatahari 2008, 2009; Yildiz 2009a, 2009b, 2009c; Cagnina, Esquivel, and Coello 2011; Zidani *et al.* 2013). Integrating gradient-based optimizers into the main procedures of EAs is an alternative means (Cai *et al.* 2012; Niu, Zhou, and Zeng 2012). Nevertheless, this optimization strategy requires derivative computation, which means that the derivative-free feature of EAs will be missing. The success of integrating a direct search method such as a simplex downhill method and a pattern search method has been reported (Chen and Tsai 1996; Koduru *et al.* 2008; Wang, Xu, and Li 2011). Mostly, the aforementioned hybrid strategies were developed for single-objective optimization. Recently, an efficient EA operator called an approximated gradient has been employed and found to be powerful for truss design problems (Pholdee and Bureerat 2012).

This article presents an alternative hybrid evolutionary optimizer for multi-objective optimization of trusses. The multi-objective version of population-based incremental learning using real codes (RPBIL) is hybridized with the evolution strategies (ES) mutation operator and approximate gradients. The new hybrid method is derivative free. Four multi-objective design problems of trusses posed to minimize structural mass and compliance with stress constraints are used as the benchmark problems. Design variables for the four design problems are shape and sizing. Then, three variants of RPBIL and a number of established MOEAs are used to tackle the test problems. The results obtained from the various optimizers are compared based on a hypervolume and it is found that the proposed hybrid RPBIL is overall the best optimizer.

2. Real-code population-based incremental learning for single-objective optimization

The very first version of population-based incremental learning (PBIL), based on binary search spaces, was introduced by Baluja (1994). There have since been several versions of PBIL such as

multi-objective (Bureerat and Sriworamas 2007) and real-code (Bureerat 2011a, 2011b) versions. Real-code population-based incremental learning herein is the multi-objective version extended from the single-objective RPBIL presented in Bureerat (2011b). The method is classified as a simple estimation of a distribution algorithm (EDA). It is developed for solving a bound-constrained optimization in which, for the case of multiple objectives, it can be expressed as

$$\min_{\mathbf{x}} \{f_1(\mathbf{x}), \dots, f_m(\mathbf{x})\}; \quad \mathbf{L} \leq \mathbf{x} \leq \mathbf{U} \quad (1)$$

where $\mathbf{x}$ is a vector of design variables size $n \times 1$.

The concept of single-objective RPBIL is to use the so-called probability matrix representing a real-code population. Given that the probability matrix $\mathbf{P}$ is sized $n \times n_I$, it means the bounds of all design variables are divided into n_I subintervals for generating a real code population. The value of P_{ij} determines the probability of having the i th element of $\mathbf{x}$ in the j th subinterval, which is $[L_i + (j-1)d_i, L_i + jd_i]$ where $d_i = (U_i - L_i)/n_I$. Let $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_T\}$ be the real-code population with T members. The number of the i th element of $\mathbf{x}$ located in the j th subinterval can be computed as $\text{round}(TP_{ij})$, where $\text{round}(x)$ gives the nearest integer to x . The pseudo-code for generating a population $\mathbf{X}$ from the given probability matrix $\mathbf{P}$ is shown in Algorithm 1.

Algorithm 1: Generating a real-code population

```

Input:  $\mathbf{P}, T$ 
Output:  $\mathbf{X}$ 
For  $i = 1$  to  $n$ 
   $\mathbf{Y} = \{\}$ 
  For  $j = 1$  to  $n_I$ 
    Find  $n_{ij}$ 
    If  $j < n_I, n_{ij} = \text{round}(TP_{ij})$ 
    If  $j = n_I, n_{ij} = T - \sum_{k=1}^{n_I-1} n_{ik}$ 
    Randomly generate  $n_{ij}$  real number in the interval  $[L_i + (j-1)d_i, L_i + jd_i]$ 
    Set those real numbers as the members of  $\mathbf{Y}$ .
  Next  $j$ 
  Randomly shuffle the positions of the members of  $\mathbf{Y}$ .
  Set all members of  $\mathbf{Y}$  as the  $i$ th row of  $\mathbf{X}$ 
Next  $i$ 

```

For single-objective optimization, the search process starts with an initial probability matrix whose members are full of $1/n_I$. The real-code population is then generated while objective functions are evaluated. The best solution $\mathbf{x}^{\text{best}}$ based on whether maximization or minimization is then found and used to update the probability matrix. The probability matrix is updated in such a way as to increase the value P_{ir} where the i th element of $\mathbf{x}^{\text{best}}$ is located in the r th subinterval. Consequently, the values of the other elements on the i th row of $\mathbf{P}$ will be decreased. A real-code population generated from this updated probability matrix will tend to have more members around $\mathbf{x}^{\text{best}}$. However, this could cause a premature convergence for the RPBIL search. In order to avoid such a phenomenon, the updating scheme is proposed as:

$$P'_{ij} = (1 - L_R(j))P_{ij}^{\text{old}} + L_R(j) \quad (2)$$

and

$$L_R(j) = L_{R0} \exp(-(j-r)^2) \quad (3)$$

where L_{R0} is an initial learning rate to be specified (default = 0.5). The index r is the subinterval in which the i th element of the best solution is located in. Instead of increasing only the value

of P_{ir} , using Equation (3), the values of elements adjacent to P_{ir} on the i th row are also slightly increased. The updated probability matrix P_{ij} is then modified as

$$P''_{ij} = \frac{P'_{ij}}{\sum_{j=1}^{n_I} P'_{ij}} \quad (4)$$

in order to preserve the condition

$$\sum_{j=1}^{n_I} P''_{ij} = 1 \quad (5)$$

where P''_{ij} is the final update of P_{ij} . The probability matrix and the best solution are iteratively updated while the corresponding real-code population is created until a termination condition is fulfilled.

3. Real-code multi-objective population-based incremental learning

For multi-objective RPBIL, more probability matrices (say n_T) are used to produce a population so as to achieve better population diversity, which is one of the most important requirements in a MOEA process. With a number of probability matrices being used, each matrix will be called a probability tray. Each probability tray is used to generate a subpopulation while the population is the union set of all those subpopulations created using the respective probability trays. The search procedure starts with initial probability trays whose elements are full of $1/n_I$, and an initial Pareto archive. The real-code population is then generated while function evaluations are carried out. The non-dominated solutions of the union set of members in the previous Pareto archive and the members in the newly generated population are sorted and saved as the updated Pareto archive. In cases where the number of members in the archive exceeds the predefined archive size (n_A), the clustering technique (Bandyopadhyay *et al.* 2008) is activated to remove some members from the archive. Afterwards, the members in the archive are divided into n_T groups using the clustering technique. The centroid of each group is used as the best solution to update the corresponding probability tray. The pseudo-code for multi-objective RPBIL is shown in Algorithm 2.

Algorithm 2: Multi-objective real-code population-based incremental learning

Input: $n, n_T, m, n_I, T, iter_max, n_A$

Output: Pareto archive $\mathbf{A}$

Initialization:

Let $P_{ij,k} = 1/n_I; k = 1, \dots, n_T$

Set $t = 0$.

Generate $\mathbf{X}^t$ from $P_{ij,k}$ (Algorithm 1).

Find $\mathbf{A}^0$ from non-dominated sorting $\mathbf{X}^t$

Main procedure:

For $t = 1$ to $iter_max$

If the members in $\mathbf{A}^t$ are more than n_A

Do: activate the clustering technique to remove some members from $\mathbf{A}^t$.

Divide the members of $\mathbf{A}^t$ into n_T groups and find the centroid of each group.

For $i = 1$ to n_T

Use the centroid of the i th group to update the i th probability tray using (2)–(5).

Next i

Set $t = t + 1$.

Generate $\mathbf{X}^t$ from $P_{ij,k}$ (Algorithm 1).
 Find $\mathbf{A}^t$ from non-dominated sorting $\mathbf{X}^t \cup \mathbf{A}^{t-1}$
 Next t

4. Hybrid multi-objective real-code population-based incremental learning

Having implemented the multi-objective RPBIL on some simple test functions, it is found that the optimizer has acceptable performance. However, premature convergence can occur during the search because the updated probability tends to generate a population whose members are adjacent to the centroids of the grouped non-dominated solutions. In order to alleviate this shortcoming, the ES mutation is employed to modify the population before performing function evaluation. For the ES mutation, a design variable vector $\mathbf{x}$ is mutated leading to an offspring $\mathbf{x}'$ using the following strategies:

$$\sigma'_i = \sigma_i \cdot \exp(\tau' \cdot N(0, 1) + \tau \cdot N_i(0, 1)) \quad (6)$$

$$x' = x + N(0, \sigma')$$

where the constants τ' (default = $1/\sqrt{2\sqrt{n}}$) and τ (default = $1/\sqrt{2n}$) are predefined parameters. $\mathbf{N}(\mathbf{0}, \sigma')$ is a random vector sized $n \times 1$ having the mean values $\mathbf{0}$, standard deviation values σ' and rotation angles $\mathbf{0}$. $N(0,1)$ denotes the normally distributed random number having expected mean 0 and standard deviation 1. Each random number N without subscript is sampled once for one individual solution while N with subscript is sampled anew for each element of a solution. σ_i is the standard deviation of the i th element of design vectors in the population.

Moreover, as the approximate gradient is proven to be a descent direction for the multi-objective truss design (Pholdee and Bureerat 2012), it will be integrated into the main procedure of multi-objective RPBIL, leading to a new hybrid algorithm. An approximate gradient is an operation to approximate a steepest descent direction from the members of a population and their function value. Using the first order Taylor's series expansion, the function value of $\mathbf{x}_i$ which is adjacent to the current solution $\mathbf{x}$ can be predicted as:

$$f(\mathbf{x}_i) = f(\mathbf{x} + \Delta\mathbf{x}) \approx f(\mathbf{x}) + \Delta\mathbf{x}^T \nabla f(\mathbf{x}) \quad (7)$$

where $\Delta\mathbf{x} = \mathbf{x}_i - \mathbf{x}$. If there are R neighbours of $\mathbf{x}$ as $\mathbf{x}_1, \dots, \mathbf{x}_R$ where $R \geq n$, the approximate gradient of $\mathbf{x}$ can be computed as:

$$\begin{bmatrix} (\mathbf{x}_1 - \mathbf{x})^T \\ \vdots \\ (\mathbf{x}_R - \mathbf{x})^T \end{bmatrix} \nabla f(\mathbf{x}) = \mathbf{A} \nabla f(\mathbf{x}) = \begin{Bmatrix} f(\mathbf{x}_1) - f(\mathbf{x}) \\ \vdots \\ f(\mathbf{x}_R) - f(\mathbf{x}) \end{Bmatrix} = \mathbf{b} \quad (8)$$

It is recommended that the value of R should be higher than the size of design variables n in order to prevent the singularity of $\mathbf{A}$. Since $\mathbf{A}$ is not a square matrix, a pseudo-inverse operator is employed to solve Equation (8).

Having obtained approximate gradients of objective functions and constraints, the use of such gradients to update the current solution towards the optimum is based on the steepest descent direction concept. First, the exterior penalty function technique in combination with the weighted sum technique is used to modify the constrained multi-objective design problem to become an equivalent unconstrained problem with one objective. The new equivalent objective function can

be expressed as:

$$f'(\mathbf{x}) = \sum_{i=1}^m w_i f_i(\mathbf{x}) + C \sum_{i=1}^{n_g} \max(\mathbf{0}, g_i^2(\mathbf{x})) \quad (9)$$

where g_i , $i = 1, \dots, C$ is a penalty parameter, n_g are the total number of inequality constraints of the design problem and w_i ($\sum_{i=1}^m w_i = 1$) are the weighting factors (randomly generated in the procedure of hybrid RPBIL).

The update of $\mathbf{x}$, denoted by $\mathbf{x}^{\text{new}}$, is obtained as

$$\mathbf{x}^{\text{new}} = \mathbf{x} + \lambda \mathbf{s} \quad (10)$$

where λ is a step length. The direction $\mathbf{s} = -\nabla f'(\mathbf{x})$ computed using Equation (8) (by replacing f with f' in the equation) is similar to a steepest descent direction. From many previous studies on truss optimization with stress constraints, the optima are likely to be located on the constraint boundaries. The step length is therefore calculated in such a way that $\mathbf{x}^{\text{new}}$ will be located on the boundaries of the constraints and remain feasible. Using the first order Taylor's series expansion, the constraint value at $\mathbf{x}^{\text{new}}$ can be approximated as:

$$g_i(\mathbf{x} + \lambda \mathbf{s}) \approx g_i(\mathbf{x}) + \lambda \mathbf{s}^T \nabla g_i(\mathbf{x}) \quad (11)$$

Since it is expected that the solution will land at the boundary, the following relation is obtained:

$$g_i(\mathbf{x}) + \lambda \mathbf{s}^T \nabla g_i(\mathbf{x}) = 0 \quad (12)$$

Thus, the step-length λ_i for leading $\mathbf{x}^{\text{new}}$ to the boundary of g_i can be calculated as:

$$\lambda_i = \frac{-g_i(\mathbf{x})}{\mathbf{s}^T \nabla g_i(\mathbf{x})} \quad (13)$$

In cases where there are n_g constraints, the minimum value of λ_i will be chosen since $\mathbf{x}^{\text{new}}$ needs to be feasible. Figure 1 displays how the approximate gradient operator works on a two-dimensional (2D) optimization problem with two nonlinear constraints. From the current population $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_9\}$, $\mathbf{x}_1$ is updated using its nearest neighbours ($\mathbf{x}_2 - \mathbf{x}_6$) for estimating gradients. $\mathbf{x}^{\text{new,act}}$ ($\diamond$) is the updated position of $\mathbf{x}_1$ using actual gradients, whereas $\mathbf{x}^{\text{new,app}}$ (Δ) is the updated position of $\mathbf{x}_1$ using approximate gradients. It can be seen that both updated solutions are close to each other but they are infeasible owing to the use of linear approximation (first order Taylor's series expansion) for approximating nonlinear convex boundaries.

As a result, the updating scheme for an approximate gradient operator will be modified from (10) in such a way that, if $\mathbf{x}$ is feasible, $\mathbf{x}^{\text{new}}$ can be computed as:

$$\mathbf{x}^{\text{new}} = \mathbf{x} + (0.5 + 0.5 \text{rand}) \sigma \mathbf{s} \quad (14)$$

where $\text{rand} \in [0, 1]$ is a uniform random number. If $\mathbf{x}$ is infeasible, $\mathbf{x}^{\text{new}}$ is computed as:

$$\mathbf{x}^{\text{new}} = \mathbf{x} + (1 + 0.5 \text{rand}) \sigma \mathbf{s} \quad (15)$$

Using Equations (14) and (15), it is expected that the updated solutions will tend to be in the feasible region.

In the hybrid RPBIL, after the ES mutation, a few solutions (say n_B solutions) are randomly selected from the current Pareto archive to apply the approximate gradient operator leading to n_B updated solutions. These n_B updated solutions (using Equations 14 and 15) will be added to the pool with the mutated population and the members from the current Pareto archive. Nevertheless,

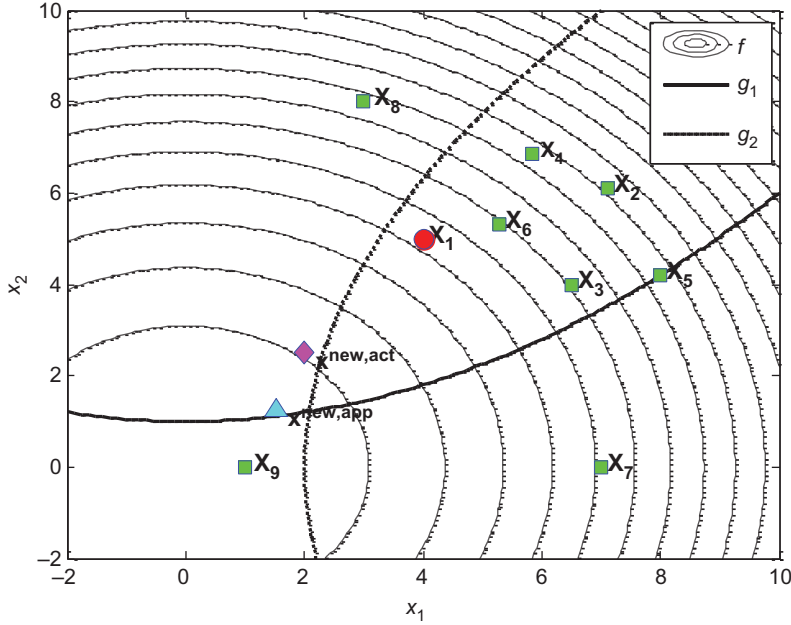


Figure 1. Step length determination for constrained optimization.

from the observation in Figure 1, using the updated solutions directly may not be beneficial as they can possibly be infeasible. Therefore, the binary crossover is applied to the n_B updated solutions mating with other n_B solutions randomly selected from the Pareto archive (this is similar to the binary crossover of differential evolution). The members in the Pareto archive are chosen to recombine with solutions from approximate gradients because they are the current best solutions being explored. The solutions obtained after the binary crossover are put in the pool for non-dominated sorting. The pseudo-code of the hybrid multi-objective RPBIL is given in Algorithm 3. Initially, a Latin hypercube sampling technique will be used to generate a number of design solutions to be used as neighbour solutions for estimating approximate gradients throughout the search procedure. The function *ESmutate* returns a mutated population based on (6), while the function *AppGrad* produces n_B solutions by means of an approximate gradient and binary crossover.

Algorithm 3: Hybrid multi-objective RPBIL

Input: $n, n_T, m, n_I, T, \max_eval, n_A, n_B$

Output: Pareto archive **A**

Initialization:

Let $P_{ij,k} = 1/n_I; k = 1, \dots, n_T$

Generate **X** from $P_{ij,k}$ (Algorithm 1).

Generate a set of neighbour solutions $\mathbf{X}_N$ using a Latin hypercube sampling technique.

Find **A** from non-dominated sorting $\mathbf{X} \cup \mathbf{X}_N$

Main procedure:

While the total number of function evaluation is less than $\max_eval$, do the following

If the members in **A** are more than n_A

Do: activate the clustering technique to remove some members from **A**.

Divide the members of **A** into n_T groups and find the centroid of each group.

For $i = 1$ to n_T

Use the centroid of the i th group to update the i th probability tray using (2)–(5).

Next i

Generate $\mathbf{X}$ from $P_{ij,k}$ (Algorithm 1).

Apply ES mutation $\mathbf{X}' = \text{ESmutate}(\mathbf{X})$

Create n_B solutions from an approximate gradient operator & binary crossover $\mathbf{X}'' = \text{AppGrad}(\mathbf{A}, \mathbf{X}_N, n_B)$

Find new $\mathbf{A}$ from non-dominated sorting $\mathbf{X}' \cup \mathbf{A} \cup \mathbf{X}''$

5. Numerical experiments

Four multi-objective optimization problems of 2D and 3D trusses are posed for MOEA performance benchmarking, and are detailed as follows.

5.1. Case I: Two-dimensional 10-bar truss

The truss structure is illustrated in Figure 2 (Xu *et al.* 2010). The structure is subject to a vertical load of -45 kN at nodes 4 and 6 and a horizontal load of 10 kN at node 6, while hinge supports are assigned at nodes 1 and 2. Design variables include the cross-sectional areas of all bar elements, and x - and y -direction positions of nodes 3 and 5. The material density and modulus of elasticity are 7833 kg/m^3 and 206.9 GPa , respectively, while the allowable stress is 65.95 MPa . The constrained multi-objective optimization problem is minimizing mass and compliance subject to stress constraints, which can be expressed as:

$$\min_{\mathbf{x}} \{f_1(\mathbf{x}), f_2(\mathbf{x})\} \quad (16)$$

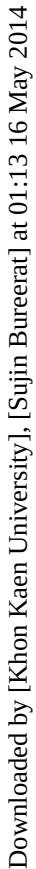
s.t.

$$\begin{aligned} |\sigma_{\max}| &\leq \sigma_y \\ 0.636 &\leq A_i \leq 132.73 \text{ cm}^2 \\ -1.0 &\leq X_i \leq 0.5 \text{ m} \quad \text{for } i = 3 \\ -0.5 &\leq X_i \leq 0 \text{ m} \quad \text{for } i = 5 \\ 0 &\leq Y_i \leq 1.0 \text{ m} \quad \text{for } i = 3 \\ 0 &\leq Y_i \leq 1.5 \text{ m} \quad \text{for } i = 5 \end{aligned}$$

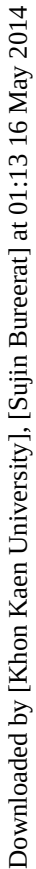
where $\mathbf{x}_{14 \times 1} = \{A_i, X_i, Y_i\}^T$ is a design vector, $f_1(\mathbf{x})$ is a function of mass, $f_2(\mathbf{x})$ is a function of compliance, $\sigma_{\max}$ is maximum stress on a structure, A_i are the elements' cross-sectional areas, X_i are the x -direction positions of nodes 3 and 5, and Y_i are the positions in the y -direction of nodes 3 and 5.

5.2. Case II: Two-dimensional simple supported 37-bar truss

The truss structure is illustrated in Figure 3 (Gomes 2011). The structure is subject to a vertical load of -10 kN at nodes 2, ..., 10 with simple supports at nodes 1 and 11, where design variables include the cross-sectional areas of all bar elements and positions in the x - and y -direction of nodes 2–5, 7–10, 12–15 and 17–20. The design variables are treated as being symmetrical with respect to the y -axis. The material density and modulus of elasticity are 7971.810 kg/m^3



Downloaded by [Khon Kaen University], [Sujin Bureerat] at 01:13 16 May 2014



Downloaded by [Khon Kaen University], [Sujin Bureerat] at 01:13 16 May 2014

Downloaded by [Khon Kaen University], [Sujin Bureerat] at 01:13 16 May 2014

Downloaded by [Khon Kaen University], [Sujin Bureerat] at 01:13 16 May 2014

Downloaded by [Khon Kaen University], [Sujin Bureerat] at 01:13 16 May 2014

Downloaded by [Khon Kaen University], [Sujin Bureerat] at 01:13 16 May 2014

Downloaded by [Khon Kaen University], [Sujin Bureerat] at 01:13 16 May 2014

Downloaded by [Khon Kaen University], [Sujin Bureerat] at 01:13 16 May 2014

Downloaded by [Khon Kaen University], [Sujin Bureerat] at 01:13 16 May 2014

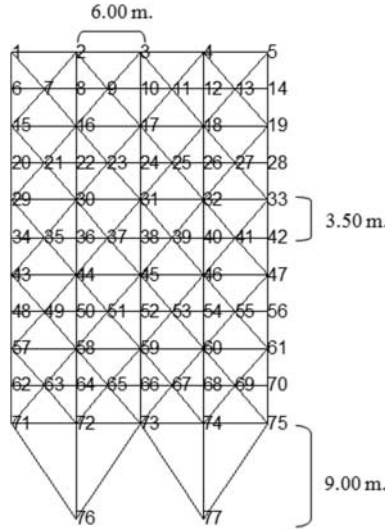


Figure 4. Two-dimensional 200-bar truss.

is expressed as:

$$\min_{\mathbf{x}} \{f_1(\mathbf{x}), f_2(\mathbf{x})\} \quad (18)$$

s.t.

$$|\sigma_{\max}| \leq \sigma_y$$

$$0.636 \leq A_i \leq 132.73 \text{ cm}^2$$

$$-1.5 \leq Y_i \leq 1.5 \text{ m} \quad \text{for } i < 71$$

$$-7 \leq Y_i \leq 1.5 \text{ m} \quad \text{for } 71 \leq i \leq 75$$

where $\mathbf{x}_{94 \times 1} = \{A_i, X_i, Y_i\}^T$ is a design vector. Positions of nodes on the same level are assigned to one design variable.

5.4. Case IV: Three-dimensional dome 120-bar truss

The truss structure is illustrated in Figures 5 and 6 (Kaveh and Talatahari 2008, 2009). The structure is subject to a vertical load of -10 kN at nodes 13–36, -30 kN at nodes 37–49 and -60 kN at node 49. The design variables include the cross-sectional areas of all elements and the z -direction positions of all nodes except for nodes 1–12. The material density and modulus of elasticity are set to be 7971.810 kg/m^3 and 210 GPa , respectively, while the stress limit is 400 MPa . The constrained multi-objective optimization problem can be expressed as:

$$\min_{\mathbf{x}} \{f_1(\mathbf{x}), f_2(\mathbf{x})\} \quad (19)$$

s.t.

$$|\sigma_{\max}| \leq \sigma_y$$

$$1.77 \leq A_i \leq 132.73 \text{ cm}^2$$

$$-1.5 \leq Z_i \leq 1.5 \text{ m}$$

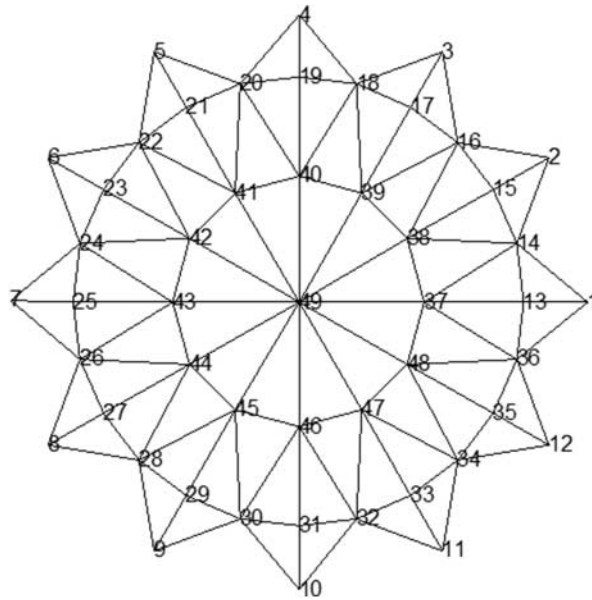


Figure 5. Top view of a three-dimensional dome 120-bar truss.

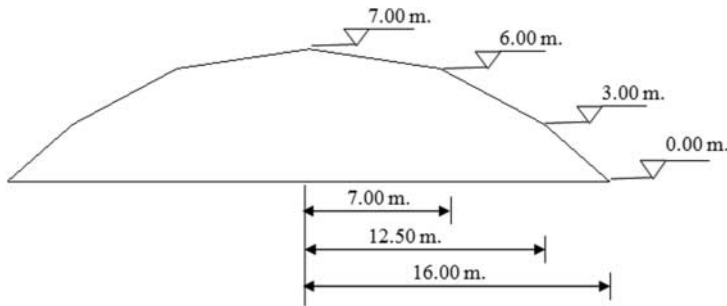


Figure 6. Side view of a three-dimensional dome 120-bar truss.

where $\mathbf{x}_{8 \times 1} = \{A_i, Z_i\}^T$ is a vector of design variables. Positions of the nodes on the same level (Figure 6) are assigned to one design variable and the sizes of the elements are set to be symmetrical.

Twelve MOEAs are employed to tackle the test problems. The optimizers and their optimization parameter settings are detailed as follows:

- BPBIL: binary multi-objective population-based incremental learning (Bureerat and Sriwornas 2007) using binary codes with mutation probability and mutation shift of 0.05 and 0.2, respectively.
- UPS-EMOA: unrestricted population size evolutionary multi-objective optimization algorithm (Aittokoski and Miettinen 2010) using crossover probability, scaling factor, probability of choosing element from offspring in crossover, minimum population size, and burst size of 0.7, [0.6, 0.9], 0.5, 10 and 25, respectively.
- SPEA2: strength Pareto evolutionary algorithm (Zitzler, Laumanns, and Thiele 2002) using real codes with crossover and mutation rates of 1.0 and 0.1, respectively.

- MPSO: multi-objective particle swarm optimization (Reyes-Sierra and Coello Coello 2006) using the starting inertia weight, ending inertia weight, cognitive learning factor, and social learning factor of 0.5, 0.01, 0.5 and 0.5, respectively. The best solution for computing a velocity vector is randomly selected from a Pareto archive. The adaptive grid algorithm is used as an archiving technique.
- NSGA-II: non-dominated sorting genetic algorithm (Deb *et al.* 2002) using real codes with crossover mutation probabilities of 1.0 and 0.1, respectively.
- DMOEA: dynamic multi-objective evolutionary algorithm (Liu *et al.* 2009) using real codes with temperature and mutation rate of 1000 and 1, respectively.
- DEMO: differential evolution for multi-objective optimization (Robiè *et al.* 2005) using real codes with crossover probability, scaling factor and probability of choosing element from offspring in crossover for differential evolution operators of 0.7, [0.6, 0.9] and 0.5, respectively.
- MOEA/D: multi-objective evolutionary algorithm based on decomposition (Zhang and Li 2007) using real codes with number of neighbouring weight vectors, crossover and mutation probabilities of 6, 1.0 and 0.1, respectively. The weighted sum technique in (9) is used for decomposition.
- AMOSA: archived multi-objective simulated annealing (Bandyopadhyay *et al.* 2008). The algorithm performs a simple downhill operator for the first three generations before entering the main procedure. In this article, the method is performed in continuous spaces where a candidate is obtained by mutating on its parent.
- RPBIL: multi-objective population-based incremental learning (Algorithm 2) using real codes with $n_I = 40$ where each probability tray produces five design solutions.
- HRPBIL1: RPBIL with ES mutation (Algorithm 3 without approximate gradient and binary crossover).
- HRPBIL2: RPBIL with ES mutation, approximate gradient and binary crossover (Algorithm 3) using $n_B = 10$, with crossover and mutation probabilities of 0.7 and 1.0, respectively. The probability of choosing an element from offspring is 0.5.

The methods using differential evolution operators, UPS-EMOA and DEMO, use a uniform random scaling factor in the range of [0.6, 0.9]. All MOEAs except for UPS-EMOA, AMOSA and HRPBIL2 use a population size of 50, and 50 generations for Cases I, II and IV, while for Case III, the population size and number of generations are set as 100 and 100, respectively. UPS-EMOA, AMOSA and HRPBIL2 use different population sizes and iteration numbers, as defined previously, but their search procedures are terminated at the same total number of function evaluations. The Pareto archive of all MOEAs is set at 500 except for NSGA-II and DEMO, which keep Pareto solutions in a population. The optimization parameter settings detailed above are obtained using several settings for each optimizer and selecting the one that gives the best results. Each design problem is solved using the various MOEAs' 30 optimization runs for performance comparison. The non-dominated sorting scheme for constrained optimization in Deb, Pratap, and Meyarivan (2001) is used for all MOEAs.

Comparison of Pareto fronts obtained from the various MOEAs is based on the performance indicator called hypervolume. The hypervolume H_V of a Pareto front A can be defined as:

$$H_V = \bigcup_{i \in A} V_i \quad (20)$$

where V_i is the hypervolume of a hypercube constructed from the i th non-dominated point and a predefined reference point, as shown in Figure 7 for a 2D case. It should be noted that the reference point must be dominated by all members of A . The hypervolume basically measures both front advancement and extension. The higher the hypervolume value, the better the Pareto front.

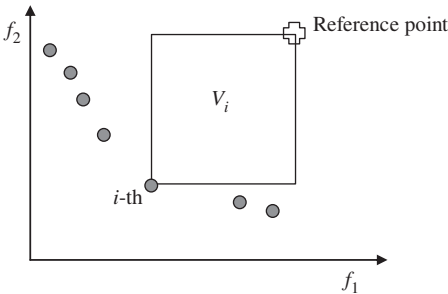


Figure 7. Two-dimensional hypervolume.

Table 1. Comparative performance based on a hypervolume indicator.

Problem	PBIL	UPS	SPEA2	MPSO	NSGA-II	DMOEa	DEMO	MOEA/D	AMOSA	RPBIL	HRPBIL1	HRPBIL2
Case I												
Mean	0.475	0.992	0.670	0.412	0.812	0.588	0.994	0.771	0.696	0.986	0.987	0.999
STD	0.145	0.016	0.121	0.193	0.123	0.123	0.018	0.059	0.156	0.001	0.001	0.000
Max.	0.746	1.000	0.882	0.671	0.993	0.894	1.000	0.878	0.918	0.988	0.989	0.999
Min.	0.197	0.934	0.439	0.000	0.536	0.381	0.902	0.637	0.379	0.985	0.985	0.999
Case II												
Mean	0.620	0.973	0.528	0.250	0.618	0.290	0.975	0.501	0.535	0.959	0.960	0.984
STD	0.104	0.016	0.131	0.133	0.100	0.097	0.015	0.097	0.143	0.001	0.001	0.014
Max.	0.814	0.998	0.780	0.451	0.804	0.452	1.000	0.681	0.816	0.961	0.963	1.000
Min	0.419	0.930	0.269	0.000	0.432	0.071	0.931	0.238	0.242	0.956	0.957	0.960
Case III												
Mean	0.478	0.747	0.406	0.111	0.508	0.290	0.780	0.364	0.135	0.464	0.669	0.995
STD	0.039	0.050	0.058	0.063	0.078	0.041	0.057	0.047	0.085	0.076	0.167	0.002
Max.	0.541	0.830	0.491	0.267	0.627	0.345	0.891	0.457	0.305	0.562	0.800	1.000
Min.	0.403	0.646	0.232	0.003	0.279	0.187	0.657	0.260	0.000	0.250	0.279	0.992
Case IV												
Mean	0.788	0.999	0.837	0.716	0.898	0.799	0.998	0.901	0.911	0.864	0.902	0.943
STD	0.076	0.001	0.083	0.179	0.053	0.071	0.002	0.028	0.042	0.186	0.065	0.005
Max.	0.931	1.000	0.945	0.926	0.946	0.900	0.999	0.941	0.946	0.925	0.938	0.947
Min.	0.563	0.996	0.641	0.260	0.759	0.570	0.988	0.807	0.728	0.000	0.618	0.927

6. Results and discussion

Having performed each optimizer to solve each test problem for 30 optimization runs with 2500 function evaluations for Cases I, II and IV, and 10,000 function evaluations for Case III, the comparative results based on the hypervolume indicator are shown in Table 1. Note that those hypervolume values are normalized to ease comparison. With 30 runs, the mean values of the front hypervolumes are used to measure the convergence rate, while the standard deviation represents the search consistency of the methods.

For Case I, the best performer (for convergence rate) is HRPBIL2, while the close second best is DEMO. The third and the very close fourth best optimizers are HRPBIL1 and RPBIL, respectively. Based on the hypervolume standard deviation, the most consistent methods are HRPBIL2 and RPBIL, followed closely by HRPBIL1. For the Case II design problem, the best optimizer based on hypervolume average is HRPBIL2, while the second best and the close third best are HRPBIL1 and RPBIL, respectively. The three most consistent methods for this case are HRPBIL1, RPBIL and HRPBIL2. For Case III, which is large scale, the noticeably best method based on convergence rate is HRPBIL2, which is head-and-shoulders above the rest. The

Table 2. Comparative hypervolumes based on Wilcoxon rank sum test for Case I.

Case I	PBIL	UPS	SPEA2	MPSO	NSGA-II	DMOEa	DEMO	MOEA/D	AMOSa	RPBIL	HRPBIL1	HRPBIL2
PBIL	0	1	1	0	1	1	1	1	1	1	1	1
UPS	0	0	0	0	0	0	0	0	0	0	0	1
SPEA	0	1	0	0	1	0	1	1	0	1	1	1
MPSO	0	1	1	0	1	1	1	1	1	1	1	1
NSGA	0	1	0	0	0	0	1	0	0	1	1	1
DMOEa	0	1	1	0	1	0	1	1	1	1	1	1
DEMO	0	0	0	0	0	0	0	0	0	0	0	0
MOEAD	0	1	0	0	0	0	1	0	0	1	1	1
AMOSa	0	1	0	0	1	0	1	1	0	1	1	1
RPBIL	0	1	0	0	0	0	1	0	0	0	1	1
HRPBIL1	0	0	0	0	0	0	1	0	0	0	0	1
HRPBIL2	0	0	0	0	0	0	0	0	0	0	0	0
Σ	0	8	3	0	5	2	9	5	3	7	8	10
Ranking	11	3	8	11	6	10	1	6	8	5	4	1

Table 3. Overall ranking based on Wilcoxon rank sum test.

	PBIL	UPS	SPEA2	MPSO	NSGA-II	DMOEa	DEMO	MOEA/D	AMOSa	RPBIL	HRPBIL1	HRPBIL2
Case I	11	3	8	11	6	10	1	6	8	5	4	1
Case II	6	2	8	11	6	11	2	8	8	4	4	1
Case III	5	3	8	11	5	10	2	9	11	7	4	1
Case IV	11	1	9	12	4	9	2	4	4	4	4	3
Σ	33	9	33	45	21	40	7	27	31	20	16	6
Ranking	9	3	9	12	6	11	2	7	8	5	4	1

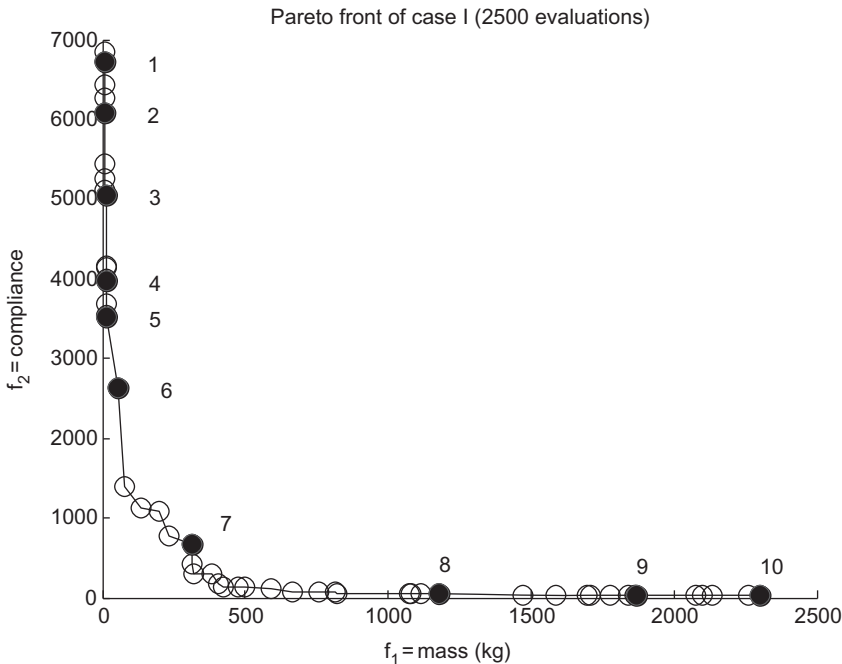


Figure 8. Pareto front of a design Case I using HRPBIL2.

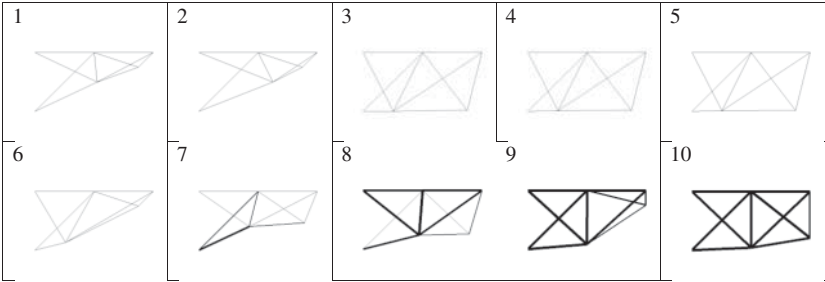


Figure 9. Truss structures according to the front in Figure 7.

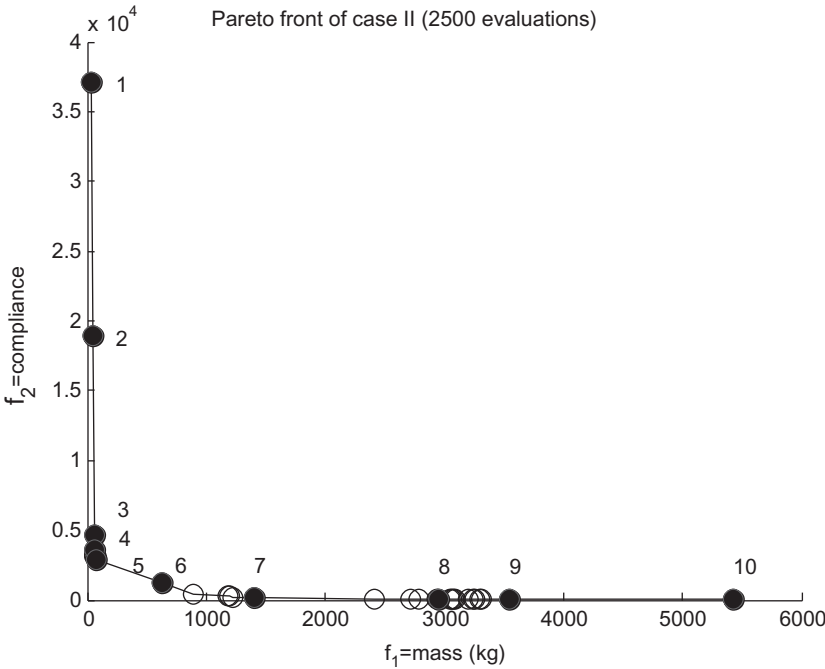


Figure 10. Pareto front of a design Case II using HRPBIL2.

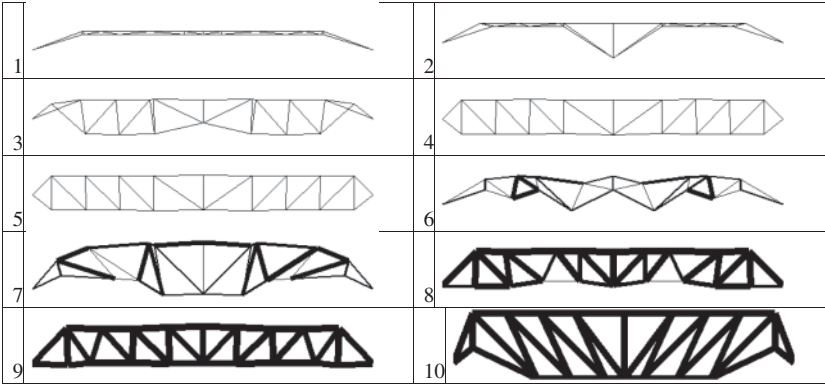


Figure 11. Truss structures according to the front in Figure 9.

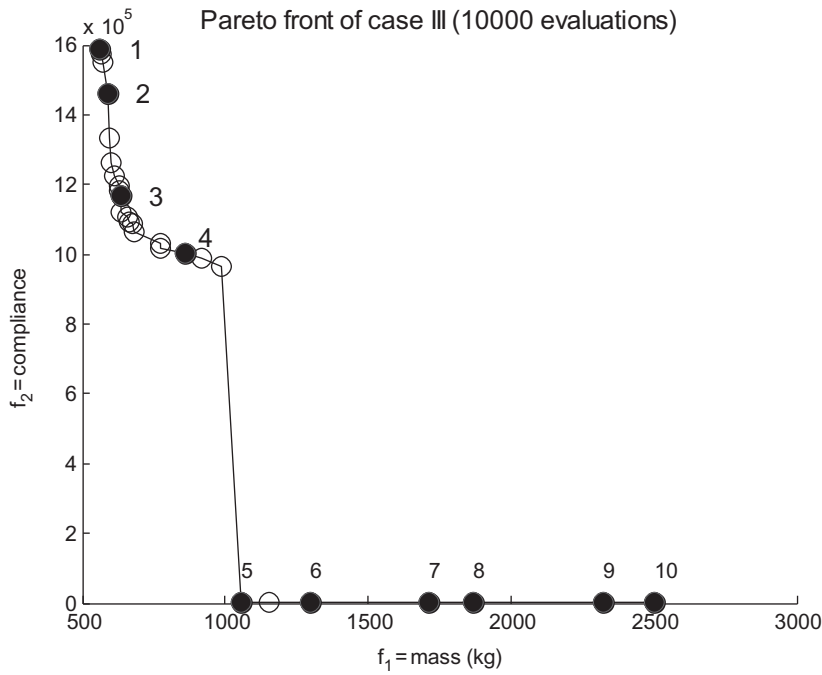


Figure 12. Pareto front of a design Case III using HRPBIL2.

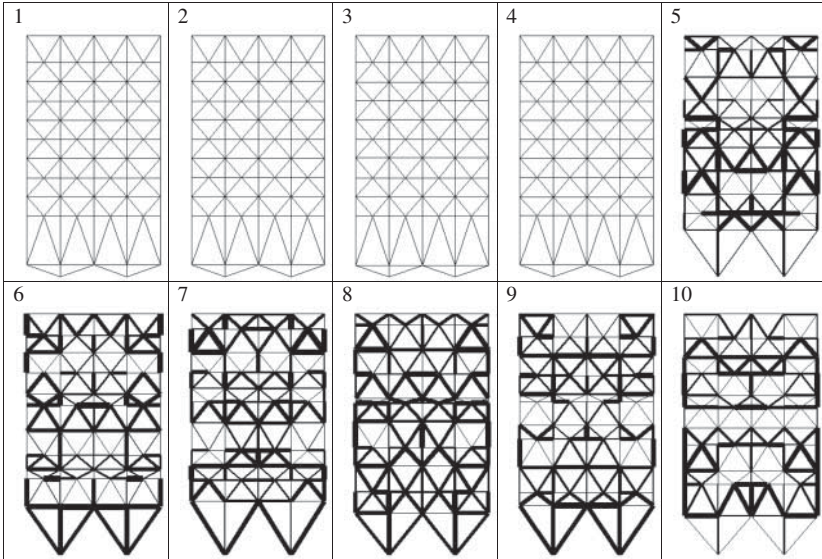


Figure 13. Truss structures according to the front in Figure 11.

second and the third best are DEMO and UPS-EMOA, respectively. The top three most consistent methods are HRPBIL2, DEMO and UPS-EMOA. For Case IV, the optimizer that gives the best convergence rate is DEMO, while the very close second and third best optimizers are UPS-EMOA and HRPBIL2, respectively. The three most consistent methods are still these three methods, with similar orders.

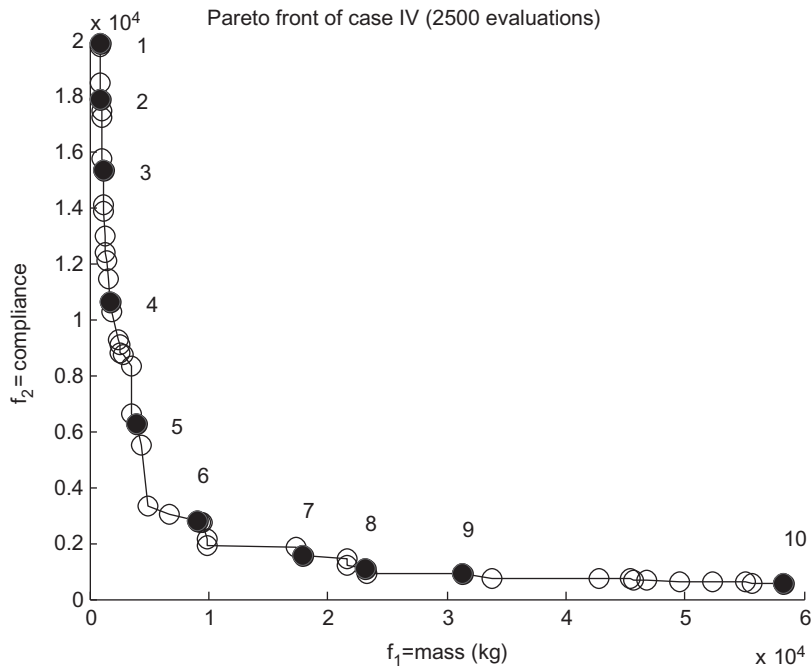


Figure 14. Pareto front of a design Case IV using HRPBIL2.

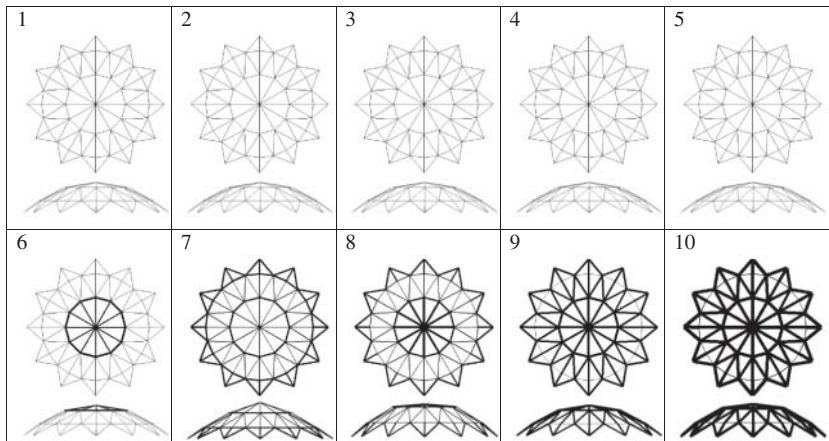


Figure 15. Truss structures according to the front in Figure 13.

Furthermore, the multi-objective evolutionary optimizers are ranked based on the statistical Wilcoxon rank sum test. Table 2 shows the ranking of MOEAs for the Case I design problem. Since 12 MOEAs are implemented, a comparison matrix sized 12×12 whose elements are full of zeros is generated. The null hypothesis for the Wilcoxon rank sum test is that, for a particular test problem, the median of 30 front hypervolumes obtained using method *I* is not different from the median of 30 front hypervolumes obtained using method *J* at a significance level of 5%. If the null hypothesis is rejected and the median from method *I* is greater, the element *IJ* of the matrix is set to be '1'. After summing up all values in matrix columns, the best optimizer is the one that has the highest score. The summation and ranking are also given Table 2. The ranking of MOEAs

when solving the four truss design problems according to the statistical test is shown in Table 3. From the results, the best performer is HRPBIL2, while the second and third best methods are DEMO and UPS-EMOA, respectively.

Overall, the best optimizer is the proposed hybrid real-code population-based incremental learning (HRPBIL2), while the second best method is DEMO. The other top performers are UPS-EMOA, HRPBIL1 and RPBIL. Among the three variants of RPBIL, HRPBIL1 is superior to RPBIL. This means that adding the ES mutation to the procedure of RPBIL can enhance its performance. The search performance is even higher when integrating the approximate gradient and binary crossover to the algorithm. The hybrid RPBIL gives the best results for both convergence rate and consistency, particularly for a large-scale problem. In the Case IV design problem, DEMO and UPS-EMOA can approach (and slightly better) HRPBIL because the total number of design variables is eight, which is not too high as it can be seen from the table that other MOEAs also have good performance for this design case.

The best Pareto fronts of the four design problems obtained using HRPBIL2 are chosen for illustration. The front of Case I is displayed in Figure 8, and some design solutions are selected and displayed as corresponding structures in Figure 9. For Case II, the front is displayed in Figure 10 and the structures are shown in Figure 11. The fronts of Cases III and IV are shown in Figures 12 and 14, respectively, while the corresponding structures are illustrated in Figures 13 and 15. From the figures, it can be seen that structures with various shapes and sizes can be obtained within one optimization run. The multi-objective design process provides a designer a number of Pareto optimum solutions with various values of structural mass and compliance for decision making.

7. Conclusions

RPBIL for single-objective optimization is extended for multi-objective optimization. The multi-objective version is then hybridized with an ES mutation operator and an approximate gradient leading to three variants of multi-objective RPBIL. The proposed algorithms along with nine established MOEAs are implemented on four multi-objective test problems of trusses. Comparative results based on the hypervolume show that the hybrid HRPBIL2 outperforms the other MOEAs, particularly for a large-scale truss design problem. Among the three variants of RPBIL, adding the ES mutation operator to the algorithm of RPBIL can improve its performance. The hybrid RPBIL is even more efficient when integrating an approximate gradient and a binary crossover to the search procedure. This shows that the proper mix of several evolutionary operators from different EAs can result in an efficient optimizer.

Acknowledgements

The authors are grateful for the support of the Royal Golden Jubilee PhD Program (RGJ), the Thailand Research Fund (TRF) (grant no. BRG5580017) and the Sustainable Infrastructure Research and Development Centre (SIRDC), Khon Kaen University, Thailand.

References

- Aittokoski, T., and K. Miettinen. 2010. "Efficient Evolutionary Approach to Approximate the Pareto-optimal Set in Multiobjective Optimization. UPS-EMOA." *Optimization Method and Software* 25 (6): 841–858.
- Baluja, S., 1994. "Population-based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning." *Technical Report CMU-CS-95_163*, Carnegie Mellon University.
- Bandyopadhyay, S., S. Saha, U. Maulik, and K. Deb. 2008. "A Simulated Annealing-based Multiobjective Optimization Algorithm: AMOSA." *IEEE Transactions on Evolutionary Computation* 12 (3): 269–283.

- Bennage, W. A., and A. K. Dhingra. 1994. "Single and Multiobjective Structural Optimization in Discrete-continuous Variables Using Simulated Annealing." *International Journal of Numerical Methods in Engineering* 38 (16): 2753–2773.
- Brest, J., and M. Maučec. 2011. "Self-adaptive Differential Evolution Algorithm Using Population Size Reduction and Three Strategies." *Soft Computing – A Fusion of Foundations, Methodologies and Applications* 15 (11): 2157–2174.
- Bureerat, S. 2011a. "Hybrid Population-based Incremental Learning Using Real Codes." *Lecture Notes in Computer Science* 6683: 379–391.
- Bureerat, S. 2011b. "Improved Population-based Incremental Learning in Continuous Spaces." *Advances in Intelligent and Soft Computing* 96: 77–86.
- Bureerat, S., and K. Sriworamas. 2007. "Population-based Incremental Learning for Multiobjective Optimization." *Advance in Soft Computing* 39: 223–232.
- Cagnina, L. C., S. C. Esquivel, and C. A. C. Coello. 2011. "Solving Constrained Optimization Problems with a Hybrid Particle Swarm Optimization Algorithm." *Engineering Optimization* 43: 843–866.
- Cai, J., Q. Li, L. Li, H. Peng, and Y. Yang. 2012. "A Hybrid CPSO–SQP Method for Economic Dispatch Considering the Valve-point Effects." *Energy Conservation and Management* 53: 175–181.
- Chen, T.-Y., and J.-H. Huang. 2013. "An Efficient and Practical Approach to Obtain a Better Optimum Solution for Structural Optimization." *Engineering Optimization* 45: 1005–1026.
- Chen, M. C., and D. M. Tsai. 1996. "A Simulated Annealing Approach for Optimization of Multi-pass Turning Operations." *International Journal of Production Research* 34 (10): 2803–2825.
- Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan. 2002. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGAI." *IEEE Transactions on Evolutionary Computation* 6 (2): 182–197.
- Deb, K., A. Pratap, and T. Meyarivan. 2001. "Constrained Test Problems for Multi-objective Evolutionary Optimization." *Lecture Notes in Computer Science* 1993: 284–298.
- Fadel Miguel, L. F., and L. F. Fadel Miguel. 2012. "Shape and Size Optimization of Truss Structures Considering Dynamic Constraints Through Modern Metaheuristic Algorithms." *Expert Systems with Applications* 39: 9458–9467.
- Gholizadeh, S., and A. Barzegar. 2013. "Shape Optimization of Structures for Frequency Constraints by Sequential Harmony Search Algorithm." *Engineering Optimization* 45 (6): 627–646.
- Gomes, H. M. 2011. "Truss Optimization with Dynamic Constraints Using a Particle Swarm Algorithm." *Expert Systems with Application* 38 (1): 957–968.
- Hasançebi, O. 2007. "Optimization of Truss Bridges within a Specified Design Domain Using Evolution Strategies." *Engineering Optimization* 39: 737–756.
- Jansen, P. W., and R. E. Perez. 2011. "Constrained Structural Design Optimization via a Parallel Augmented Lagrangian Particle Swarm Optimization Approach." *Computers and Structures* 89 (13–14): 1352–1366.
- Kaveh, A., and S. Talatahari. 2008. "A Hybrid Particle Swarm and Ant Colony Optimization for Design of Truss Structures." *Asian Journal of Civil Engineering (Building and Housing)* 9 (4): 329–348.
- Kaveh, A., and S. Talatahari. 2009. "Particle Swarm Optimizer, Ant Colony Strategy and Harmony Search Scheme Hybridized for Optimized of Truss Structures." *Computers and Structures* 87 (5–6): 267–283.
- Kaveh, A. & Zolghadr, A., 2012. Truss optimization with natural frequency constraints using a hybridized CSS-BBBC algorithm with trap recognition capability. *Computers and Structures*, 102–103, 14–27.
- Kaveh, A., and S. Talatahari. 2010. "A Novel Heuristic Optimization Method: Charged System Search." *Acta Mechanica* 213 (3): 267–289.
- Kim, I. Y., and O. L. de Weck. 2006. "Adaptive Weighted Sum Method for Multiobjective Optimization: A New Method for Pareto Front Generation." *Structural and Multidisciplinary Optimization* 31 (2): 105–116.
- Koduru, P., D. Zhanshan, S. Das, S. M. Welch, J. L. Roe, and E. Charbit. 2008. "A Multiobjective Evolutionary-simplex Hybrid Approach for the Optimization of Differential Equation Models of Gene Networks." *IEEE Transactions on Evolutionary Computation* 12 (5): 572–590.
- Li, L. J., Z. B. Huang, and F. Liu. 2009. "A Heuristic Particle Swarm Optimization Method for Truss Structures with Discrete Variables." *Computers and Structures* 87 (7–8): 435–443.
- Lingyun, W., Z. Mei, W. Guangming, and M. Guang. 2005. "Truss Optimization on Shape and Sizing with Frequency Constraints Based on Genetic Algorithm." *Computational Mechanics* 35 (5): 361–368.
- Liu, M., X. Zou, Y. Chen, and Z. Wu. 2009. "Performance Assessment of DMOEA-DD with CEC 2009 MOEA Competition Test Instances." In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC2009)*, May 2009, Norway, 2913–2918. Piscataway, NJ: IEEE Press.
- Lu, Y. C., J. C. Jan, S. L. Hung, and G. H. Hung. 2013. "Enhancing Particle Swarm Optimization Algorithm Using Two New Strategies for Optimizing Design of Truss Structures." *Engineering Optimization*, doi:10.1080/0305215X.2012.729054
- Mallipaddi, R., P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren. 2011. "Differential Evolution Algorithm with Ensemble of Parameters and Mutation Strategies." *Applied Soft Computing* 11 (2): 1679–1696.
- Niu, Q., Z. Zhou, and T. Zeng. 2012. "A Hybrid Quantum-inspired Particle Swarm Evolution Algorithm and SQP Method for Large-scale Economic Dispatch Problems." *Lecture Notes in Computer Science* 6840: 207–214.
- Noilublab, N., and S. Bureerat. 2011. "Simultaneous Topology, Shape and Sizing Optimization of a Three-dimensional Slender Truss Tower Using Multiobjective Evolutionary Algorithms." *Computers and Structures* 89 (23–24): 2531–2538.
- Pholdee, N., and S. Bureerat. 2012. "Performance Enhancement of Multiobjective Evolutionary Optimizers for Truss Design Using an Approximate Gradient." *Computers and Structures* 106–107: 115–124.

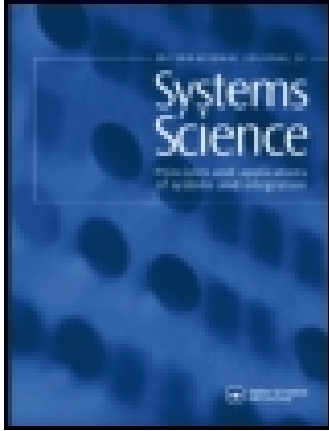
- Rahami, H., A. Kaveh, and Y. Gholipour. 2008. "Sizing, Geometry and Topology Optimization of Trusses via Force Method and Genetic Algorithm." *Engineering Structures* 30 (9): 2360–2369.
- Reyes-Sierra, M., and C. A. Coello Coello. 2006. "Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art." *International Journal of Computational Intelligence Research* 2 (3): 287–308.
- Richardson, J. N., S. Adriaenssens, P. Bouillard, and R. F. Coelho. 2012. "Multiobjective Topology Optimization of Truss Structures with Kinematic Stability Repair." *Structural Multidisciplinary Optimization* 46: 513–532.
- Robič, T., B. Filipič, C. A. Coello Coello, A. A. Hernández, and E. Zitzler. 2005. "DEMO: Differential Evolution for Multiobjective Optimization." *Lecture Notes in Computer Science* 3410: 520–533.
- Sadollah, A., A. Bahreininejad, H. Eskandar, and M. Hamdi. 2012. "Mine Blast Algorithm for Optimization of Truss Structures with Discrete Variables." *Computers and Structures* 102–103: 49–63.
- Su, R., X. Wang, L. Gui, and Z. Fan. 2011. "Multi-objective Topology and Sizing Optimization of Truss Structures Based on Adaptive Multi-island Search Strategy." *Structural and Multidisciplinary Optimization* 43 (2): 275–286.
- Torii, A. J., R. H. Lopez, and F. Biondini. 2012. "An Approach to Reliability-based Shape and Topology Optimization of Truss Structures." *Engineering Optimization* 44: 37–53.
- Wang, L., Y. Xu, and L. Li. 2011. "Parameter Identification of Chaotic Systems by Hybrid Nelder-Mead Simplex Search and Differential Evolution Algorithm." *Expert Systems with Applications* 38 (4): 3238–3245.
- Xu, T., W. Zuo, T. Xu, G. Song, and R. Li. 2010. "An Adaptive Reanalysis Method for Genetic Algorithm with Application to Fast Truss Optimization." *Acta Mechanica Sinica* 26 (2): 225–234.
- Yildiz, A. R. 2009a. "A New Design Optimization Framework Based on Immune Algorithm and Taguchi's Method." *Computers in Industry* 60 (8): 613–620.
- Yildiz, A. R. 2009b. "A Novel Hybrid Immune Algorithm for Global Optimization in Design and Manufacturing." *Robotics and Computer-Integrated Manufacturing* 25 (2): 261–270.
- Yıldız, A. R. 2009c. "A Novel Particle Swarm Optimization Approach for Product Design and Manufacturing." *International Journal of Advanced Manufacturing Technology* 40 (5): 617–628.
- Zhang, Q., and H. Li. 2007. "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition." *IEEE Transactions on Evolutionary Computation* 11 (6): 712–731.
- Zidani, H., E. Pagnacco, R. Sampaio, R. Ellaia, and J. E. Souza de Cursi. 2013. "Multi-objective Optimization by a New Hybridized Method: Applications to Random Mechanical Systems." *Engineering Optimization* 45: 917–939.
- Zitzler, E., M. Laumanns, and L. Thiele. 2002. "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization." In *Evolutionary Methods for Design, Optimization and Control*, Barcelona, Spain, 2002.

This article was downloaded by: [S. Bureerat]

On: 25 November 2014, At: 00:58

Publisher: Taylor & Francis

Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



International Journal of Systems Science

Publication details, including instructions for authors and subscription information:

<http://www.tandfonline.com/loi/tsys20>

Hybrid real-code ant colony optimisation for constrained mechanical design

Nantiwat Pholdee^a & Sujin Bureerat^a

^a Sustainable Infrastructure Research and Development Centre, Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen, Thailand

Published online: 28 Feb 2014.



[Click for updates](#)

To cite this article: Nantiwat Pholdee & Sujin Bureerat (2014): Hybrid real-code ant colony optimisation for constrained mechanical design, International Journal of Systems Science, DOI: [10.1080/00207721.2014.891664](https://doi.org/10.1080/00207721.2014.891664)

To link to this article: <http://dx.doi.org/10.1080/00207721.2014.891664>

PLEASE SCROLL DOWN FOR ARTICLE

Taylor & Francis makes every effort to ensure the accuracy of all the information (the "Content") contained in the publications on our platform. However, Taylor & Francis, our agents, and our licensors make no representations or warranties whatsoever as to the accuracy, completeness, or suitability for any purpose of the Content. Any opinions and views expressed in this publication are the opinions and views of the authors, and are not the views of or endorsed by Taylor & Francis. The accuracy of the Content should not be relied upon and should be independently verified with primary sources of information. Taylor and Francis shall not be liable for any losses, actions, claims, proceedings, demands, costs, expenses, damages, and other liabilities whatsoever or howsoever caused arising directly or indirectly in connection with, in relation to or arising out of the use of the Content.

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden. Terms & Conditions of access and use can be found at <http://www.tandfonline.com/page/terms-and-conditions>

Hybrid real-code ant colony optimisation for constrained mechanical design

Nantiwat Pholdee and Sujin Bureerat*

*Sustainable Infrastructure Research and Development Centre, Department of Mechanical Engineering, Faculty of Engineering,
Khon Kaen University, Khon Kaen, Thailand*

(Received 19 February 2013; accepted 7 January 2014)

This paper proposes a hybrid meta-heuristic based on integrating a local search simplex downhill (SDH) method into the search procedure of real-code ant colony optimisation (ACOR). This hybridisation leads to five hybrid algorithms where a Monte Carlo technique, a Latin hypercube sampling technique (LHS) and a translational propagation Latin hypercube design (TPLHD) algorithm are used to generate an initial population. Also, two numerical schemes for selecting an initial simplex are investigated. The original ACOR and its hybrid versions along with a variety of established meta-heuristics are implemented to solve 17 constrained test problems where a fuzzy set theory penalty function technique is used to handle design constraints. The comparative results show that the hybrid algorithms are the top performers. Using the TPLHD technique gives better results than the other sampling techniques. The hybrid optimisers are a powerful design tool for constrained mechanical design problems.

Keywords: real-code ant colony optimisation; simplex downhill; constrained optimisation; hybrid algorithms; Latin hypercube sampling

1. Introduction

In recent decades, meta-heuristics (MHs) and evolutionary algorithms (EAs) have arisen as alternative optimisers to classical mathematical programming. The methods are simple to use, robust, and capable of tackling global optimisation problem. They can deal with almost any kind of optimisation problem. Moreover, multiobjective versions of these types of optimisers can explore a Pareto front within one optimisation run (Bureerat & Sriwornas, 2007). Nevertheless, with such appealing advantages, they have some inevitable shortcomings, i.e. a low convergence rate and a lack of search consistency. This is because those optimisers search for optimum solutions based upon randomisation. As a result, the solutions obtained by using such optimisers are often classified as near optima or approximated Pareto fronts. This implies that performance enhancement is always an issue in the field of evolutionary computation and MHs.

Over the last few decades, a large number of MH optimisers have been developed and proposed for a wide variety of applications. The algorithms have been developed based on various concepts such as the estimation of distribution concept (Baluja, 1994; Brest et al., 2013; Bureerat, 2011a), swarm intelligence (Reyes-Sierra & Coello Coello, 2006), natural evolution of a genetic algorithm (GA) (Linfild & Penny, 1995), etc. Based on the comparative performance and its applications, one of the most power-

ful and popular optimisers is ant colony optimisation using real codes (ACOR) (Socha & Dorigo, 2008). The algorithm has been tested and found to be one of the most efficient EAs by several researchers (Bureerat, 2011a, 2011b; Socha & Dorigo, 2008). It has been successfully implemented to solve a wide variety of practical design problems, e.g. design of storm sewer networks (Afshar, 2010), neural network training (Socha & Blum, 2007), support vector machines (Zhang, Chen, & He, 2010), design of natural gas pipeline transportation (Chebouba et al., 2009), water resources (Afshar & Madadgar, 2008; Hajizadeh, Christie, & Demyanov, 2011; Madadgar & Afshar, 2009), process and control system design (Schluter et al., 2009), active vibration control (Mohamad, Tokhi, & Omar, 2011), structural health monitoring (Majumdar, Maiti, & Maity, 2012; Yu & Xu, 2011) and electrical engineering (Hazra & Sinha, 2009; Nakawiro & Erlich, 2009). Due to past successes, there have been several proposals to extend and increase the search performance of ACOR. The modifications have been made by adding evolutionary operators to ACOR (Zhu, Yang, & Ma, 2011), hybridising ACOR with other MHs (Karimi, Nobahari, & Siarry, 2010) and integrating local search into ACOR (Liao, Kuo, & Hu, 2012; Liao, Montes de Oca, Aydin, Stützle, & Dorigo, 2011). For the local search hybrid, it has been found that the integration of a simplex downhill (SDH) method (Nelder & Mead, 1965) to MHs is one of the best choices to enhance their search performance

*Corresponding author. Email: sujbur@kku.ac.th

(Chelouah & Siarry, 2005; Wang, Xu, & Li, 2011; Zahara & Kao, 2009). Nevertheless, such a local search optimiser has not been hybridised with this powerful ACOR yet.

This paper presents an alternative choice for ACOR performance enhancement. We have integrated the SDH method (also known as the Nelder and Mead method) to ACOR for constrained optimisation of mechanical systems. The hybrid approach employs ACOR as the main optimiser, while the SDH method is used for local search. Also, the initialisation stage has been enhanced by using Latin hypercube and optimum Latin hypercube sampling techniques, leading to five variants of the hybrid ACOR optimisers. The methods along with a number of established MHs were implemented on numerous constrained design problems, including automotive component design. Comparative results revealed that the proposed hybrid methods were the top performers and they can be considered as a powerful optimisation tool.

2. Related work on hybrid meta-heuristics

There have been several proposals to improve the search performance of existing or newly developed MHs. For design problems with expensive function evaluations, the use of MHs in combination with surrogate models has been an effective means (Kanyakam & Bureerat, 2012; Srisoporn & Bureerat, 2008; Tenne, Izui, & Nishiwaki, 2012). Success in hybridising MHs with some direct search optimisers such as an approximate gradient technique (Bureerat, 2011b; Pholdee & Bureerat, 2012), a pattern search method (Mariani & dos Santos Coelho, 2011) and the SDH method (Chelouah & Siarry, 2005; Wang et al., 2011; Zahara & Kao, 2009) has been reported. It is also possible to hybridise some existing EAs or MHs together to enhance their performance (Hsu, 2012; Kaveh & Talatahari, 2008; Kaveh & Talatahari, 2009; Lion, Hsieh, & Chen, 2013; Pholdee & Bureerat, 2013; Shelokar, Siarry, Jayaraman, & Kulkarni, 2007; Yildiz, 2009b, 2009c, 2012, 2013).

Hybrid ant colony optimisation for combinatorial problems on a discrete design domain has been proposed by some researchers (Huang & Liao, 2008; Kabir, Shahjahan, & Murase, 2012; Lee, Lee, Lin, & Ying, 2010; Panahi & Tavakkoli-Moghaddam, 2011; Wang, Osagie, Thulasiraman, & Thulasiram, 2009). For ACOR, it has been integrated with tabu search (Karimi et al., 2010), population-based incremental and differential evolution (Xiao & Li, 2011), and a real-code genetic algorithm (Ciornei & Kyriakides, 2012). For the use of local search, ACOR has been hybridised with gradient-based optimisers (Liao et al., 2011) and the Hooke and Jeeves local search method, also known as pattern search (Liao et al., 2012). To our knowledge, the hybrid of the SDH method and ACOR has not been investigated yet. Also, the implementation on practical constraints optimisation problems has not been adequately studied yet. Moreover, even with thousands of articles de-

scribing hybrid MHs or EAs, the optimisers still rely on randomisation and they are not efficient for all types of design problems. As a result, there is always a need to enhance the convergence rate and search consistency of MHs.

Improvement of the initialisation phase of evolutionary optimisation has been somewhat overlooked. Usually, a randomisation or Monte Carlo technique is used to generate an initial population. Yildiz (2009a) and Karen, Yildiz, Kaya, Öztürk, and Öztürk (2006) have applied the Taguchi method to improve the quality of an initial population before entering the MH main search procedure. This can result in improved performance of the optimisers. It has been found by the authors that, rather than using the Taguchi method, which is time-consuming for high-dimensional problems, an alternative idea to create a high-quality initial population is to make the population members evenly distributed throughout the design domain as much as possible. With such a concept, a Latin hypercube sampling technique can be used instead of the Monte Carlo method. Moreover, an optimum Latin hypercube sampling technique (Viana, Venter, & Balabanov, 2010) that is not time-consuming can also be useful for the initialisation phase.

3. Real-code ant colony optimisation and a simplex downhill method

Ant colony optimisation was first introduced by Dorigo (Dorigo, 1992) for solving combinatorial optimisation. Since then, a few researchers have extended the method for optimisation in continuous spaces (Guo & Zhu, 2012; Kong & Tian, 2006) (hereinafter it will be referred to as real-code ant colony optimization [ACOR]). One of the best versions of ACOR was proposed by Socha and Dorigo in 2008 (Socha & Dorigo, 2008). Unlike the very first version, which was based upon a discrete probability distribution, ACOR uses a continuous probability distribution. This method will be used to solve minimisation problems with bound constraints expressed as

$$\min f(\mathbf{x}); \mathbf{L} \leq \mathbf{x} \leq \mathbf{U}, \quad (1)$$

where f is an objective function, $\mathbf{x}$ is a design vector sized $n \times 1$, and $\mathbf{L}$ and $\mathbf{U}$ are the lower and upper bounds of $\mathbf{x}$, respectively. The computational steps for ACOR are given in Algorithm 1. The search procedure starts by generating an initial population or a solution archive, $P = \{\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^m\}$. Then, m offspring $Q = \{\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^m\}$ are created. To create the element y_i , the weighting factors, w_i , are computed:

$$w_i = \frac{1}{qm\sqrt{2\pi}} \exp\left(-\frac{N_i - 1}{2q^2m^2}\right), \quad (2)$$

where m is a population size, q is a parameter to be defined and N_i is the order of the solution $\mathbf{x}^i$, sorted based on the

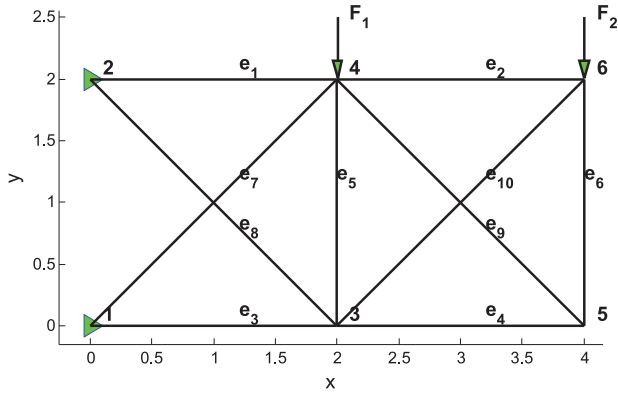


Figure 1. 10-bar truss.

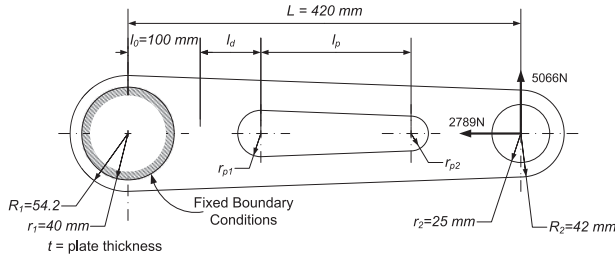


Figure 2. Torque arm structure.

objective function values from the smallest to the largest. The probability of the l th Gaussian function being selected is computed as

$$p_l = \frac{w_l}{\sum_{j=1}^m w_j}. \quad (3)$$

In this paper, the Roulette wheel is used for selecting the Gaussian function. The mean value μ_i^l and the standard deviation σ_i^l used for generating the i th element of a new solution y are set as

$$\mu_i^l = x_i^l \quad (4)$$

and

$$\sigma_i^l = \frac{\xi}{m-1} \sum_{e=1}^m |x_i^e - x_i^l|, \quad (5)$$

Table 1. Design variables and bounds.

Design variables	Bounds (mm)
$x_1 = l_d$	$0 \leq l_d \leq 60$
$x_2 = l_p$	$100 \leq l_p \leq 200$
$x_3 = r_{p1}$	$15 \leq r_{p1} \leq 35$
$x_4 = r_{p2}$	$15 \leq r_{p2} \leq 30$
$x_5 = t$	$3 \leq t \leq 4.5$

respectively, where ξ is a parameter to be defined. Then, the i th element of a new solution is generated from a normally distributed random number having mean μ_i^l and standard deviation σ_i^l , which is denoted by $N(\mu_i^l, \sigma_i^l)$. This process is repeated until m new solutions are created. The best m solutions between the old and new populations are chosen as the next generation. These steps are repeated until termination conditions are met. More details can be found in Socha and Dorigo (2008).

Algorithm 1: real-code ant colony optimisation

Input: Maximum iteration number t_{max} , population size m

Output: $\mathbf{x}_{best}, f_{best}$

Initialisation

1. Randomly generate an initial population $P(t)$, where $t = 0$ is an iteration number.
2. Set the parameters q and ξ .
3. For $t = 1$ to t_{max} , do the following:

Ant-based solution construction process: $Q(t)$

4. Find the order of each solution and find w_i using (2).
5. Compute p_i using (3).
6. For $j = 1$ to m , do

For $i = 1$ to n , do
 Use a Roulette wheel operator to choose the l th Gaussian function.
 Find μ_i^l and σ_i^l .
 Generate the i th element of a new solution j as $y_i^j = N(\mu_i^l, \sigma_i^l)$.
 Keep y_i^j in the range $[L_i, U_i]$.
 Next i
 Next j
7. Set the best m solution from $P(t-1) \cup Q(t)$ as the next-generation $P(t)$.

Next t

The SDH method proposed by Nelder and Mead (1965) is a direct search method for unconstrained optimisation. It is considered one of the most efficient local search techniques without using function derivatives. The computational steps of the SDH method are detailed in Algorithm 2. The process starts with an initial simplex consisting of $n + 1$ solutions for an n -dimensional space. The simplex is then evolved by means of reflection, expansion, contraction and simplex reduction until a termination condition is fulfilled. Note that every time a new solution is created, it will be treated as if it is inside the bounds.

Table 2. Comparison based on objective function mean values (F1–F16).

	ABC	BBBC	BPBIL	CKS	CMAES	CPBIL	CSS	ChSS	DE	EP	ES	FWA	GA
F1	0.996081	0.996018	0.996019	1.003872	0.996019	0.996018	0.996019	0.996019	0.996019	0.996019	0.996019	0.996019	0.996019
F2	0.996021	0.996021	0.996021	1.329694	0.996021	0.996021	0.996021	0.996021	0.996021	0.996021	0.996021	0.996021	0.996021
F3	0.996018	0.996018	0.996018	0.999792	0.996018	0.996018	0.996018	0.996018	0.996018	0.996018	0.996018	0.996018	0.996018
F4	0.99602	0.99602	0.99602	0.996021	0.99602	0.99602	0.99602	0.99602	0.99602	0.99602	0.99602	0.99602	0.99602
F5	1.392556	1.165416	1.022103	1.256331	1.266801	1.334517	2.151983	1.315088	1.956799	2.071599	2.122127	1.972297	2.033688
F6	0.996037	0.996036	0.996038	1.000648	0.996037	0.996036	0.996038	0.996038	0.996037	0.996036	0.996037	0.996037	0.996038
F7	0.996016	0.996016	0.996016	0.996016	0.996016	0.996016	1.712873	0.996016	1.712873	1.712873	1.712873	1.712873	1.712873
F8	0.999583	0.999762	0.999668	0.999981	0.999141	0.999272	0.999859	0.999671	0.999743	0.999125	0.999176	0.999599	0.999658
F9	0.996023	0.996021	1.002581	1.466025	0.99602	0.996021	2.073381	0.996023	2.13299	2.194912	2.290253	2.411786	2.054547
F10	0.996016	0.996016	0.996016	0.996016	0.996016	0.996016	1.391933	0.996016	1.329517	1.183683	1.2671	1.3296	1.1836
F11	0.998826	0.998821	0.998855	0.999163	0.998825	1.040493	0.998854	0.998887	0.998821	0.99882	0.998825	0.998824	0.998843
F12	0.998188	0.998194	0.998206	1.094267	0.998189	0.9982	1.002792	1	0.998188	0.998185	0.998185	0.998195	0.998207
F13	0.996922	0.996947	0.997346	2.08264	0.996866	0.996895	0.996949	0.997065	0.997064	0.996875	0.99687	1.28844	0.99723
F14	0.999065	0.999015	0.999133	1.381869	0.999022	0.999107	0.999137	0.999177	0.998967	0.998952	0.998952	1.221069	0.999097
F15	1.440414	1.21821	2.588469	0.996011	3.384673	2.255173	5.995554	4.415923	5.995542	5.995552	5.995549	5.995548	5.99555
F16	2.274425	2.163061	1.775531	2.275829	2.66231	1.997625	1.275981	2.044424	1.663883	2.107564	2.329561	0.998395	1.886511
Σ	18.06821	17.50759	18.35404	19.87418	20.274	18.58945	24.62108	20.73841	24.7645	25.23826	25.68958	24.90674	24.83992

Table 3. Comparison based on objective function mean values.

	GSA	HS	LCA	PSO	SA	TLBO	ACOR	ACORSDDH1	ACORSDDH2	ACORSDDH3	ACORSDDH4	ACORSDDH5
F1	0.996019	0.996019	0.996019	0.996019	0.996019	0.996019	0.996018	0.996018	0.996018	0.996018	0.996018	0.996018
F2	0.996021	1.204354	0.996021	0.996021	0.996021	0.996021	0.996021	0.996021	0.996021	0.996021	0.996021	0.996021
F3	0.996018	0.996019	0.996018	0.996018	0.996018	0.996018	0.996018	0.996018	0.996018	0.996018	0.996018	0.996018
F4	0.996021	0.99602	0.99602	0.99602	0.99602	0.99602	0.99602	0.99602	0.99602	0.99602	0.99602	0.99602
F5	1.108208	2.025254	1.14791	1.581598	1.412683	1.765532	1.149773	1.103622	1.078806	1.016171	1.046164	1.080042
F6	0.996038	0.99604	0.996037	0.996037	0.996037	0.996037	0.996037	0.996036	0.996036	0.996036	0.996036	0.996036
F7	0.996016	1.712873	0.996016	0.996016	0.996016	0.996016	0.996016	0.996016	0.996016	0.996016	0.996016	0.996016
F8	0.999948	0.999961	0.99969	0.999549	0.99897	0.999842	0.999365	0.999118	0.999123	0.999197	0.999076	0.999189
F9	0.996024	2.159837	0.996021	0.996021	1.043641	0.996021	0.996022	0.996021	0.99602	0.99602	0.99602	0.996021
F10	0.996016	1.20435	0.996016	0.996016	0.996016	0.996016	0.996016	0.996016	0.996016	0.996016	0.996016	0.996016
F11	0.998944	0.998873	0.99884	1.04052	0.998857	0.998842	0.998821	0.99882	0.998822	0.99882	0.998822	0.99882
F12	1.002394	0.998266	0.998221	0.998295	0.998188	0.998216	0.998185	0.998185	0.998185	0.998185	0.998185	0.998185
F13	0.997259	2.04101	0.997019	0.996934	1.246821	0.997073	0.996906	0.996873	0.996873	0.996876	0.996872	0.996878
F14	0.999423	0.999647	0.999156	1.07306	1.202469	1.054579	0.998953	0.998952	0.998952	0.998952	0.998952	0.998952
F15	2.848367	5.995561	2.242752	2.421967	0.995995	1.001128	0.995992	0.995992	0.995992	0.995992	0.995992	0.995992
F16	1.054087	0.998425	2.108698	2.107804	2.329464	2.274067	2.03747	1.885941	1.719543	1.608484	2.163196	0.998518
Σ	17.9768	25.32251	18.46045	19.1879	18.19924	18.05745	17.14363	16.94567	16.75446	16.58084	17.16543	16.03474

Algorithm 2: SDH methodInput: initial simplex $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}\}$ Output: $\mathbf{x}_{simp}$

- (1) Start with $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}$ and set $k = 0$.
- (2) Find $\mathbf{x}_h = \max(\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}\})$ and $\mathbf{x}_l = \min(\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n+1}\})$.
- (3) Reflection: find $\mathbf{x}_r$ and $f(\mathbf{x}_r)$.
- (4) If $f(\mathbf{x}_r) < f(\mathbf{x}_l)$, go to step (8).
- (5) If $f(\mathbf{x}_l) \leq f(\mathbf{x}_r) < f(\mathbf{x}_h)$, replace $\mathbf{x}_h$ with $\mathbf{x}_r$ and go to step (14).
- (6) If $f(\mathbf{x}_r) = f(\mathbf{x}_h)$, replace the second worst solution with $\mathbf{x}_r$ and go to step (14).
- (7) If $f(\mathbf{x}_r) > f(\mathbf{x}_h)$, go to step (11).
- (8) Expansion: find $\mathbf{x}_e$ and $f(\mathbf{x}_e)$.
- (9) If $f(\mathbf{x}_e) < f(\mathbf{x}_r)$, replace $\mathbf{x}_h$ with $\mathbf{x}_e$ and go to step (14).
- (10) Otherwise go to step (14)
- (11) Contraction: find $\mathbf{x}_c$ and $f(\mathbf{x}_c)$.
- (12) If $f(\mathbf{x}_c) < f(\mathbf{x}_h)$, replace $\mathbf{x}_h$ with $\mathbf{x}_c$ and go to step (14).
- (13) Otherwise, reduce the current simplex as $\mathbf{x}_i = 0.5(\mathbf{x}_i + \mathbf{x}_l)$ and go to step (14).
- (14) If the termination criterion is met, set $\mathbf{x}_{simp} = \mathbf{x}_l$ and stop; otherwise, set $k = k + 1$ and go to step (2).

4. Hybrid algorithms

The procedure of the hybrid algorithm is detailed in Algorithm 3. The hybridised ACOR and SDH can be obtained in such a way that ACOR is used as the main search procedure. When a set of offspring is created, $n + 1$ solutions are selected in some manner to be an initial simplex. Afterwards, the SDH method (Algorithm 2) is activated to find a solution $\mathbf{x}_{simp}$. The next-generation population is found by selecting the m best solutions of $P(t - 1) \cup Q(t) \cup \mathbf{x}_{simp}$. In this study, we not only propose the hybridised ACOR and SDH, but also try to improve the initialisation phase by using Latin hypercube and optimum Latin hypercube sampling techniques. We have the following five variants of the hybrid algorithm:

ACORSDH1: The hybrid algorithm as shown in Algorithm 3. The initial simplex is the best $n + 1$ solutions from $P(t - 1) \cup Q(t)$ in step (7). An initial population is created using a Monte Carlo method.

ACORSDH2: The hybrid algorithm as shown in Algorithm 3. Here a Latin hypercube sampling technique is used to generate an initial population.

ACORSDH3: The hybrid algorithm as shown in Algorithm 3. This algorithm uses a Latin hypercube sampling

technique for generating an initial population. For this variant, $N \gg n$ best solutions will be saved to an external archive in parallel with the search procedure. When the SDH method is invoked, $n + 1$ solutions for an initial simplex will be selected from the archive by using the crowding distance algorithm (Deb, Pratap, Agarwal, & Meyarivan, 2002).

ACORSDH4: The hybrid algorithm as shown in Algorithm 3. Here a translational propagation Latin hypercube algorithm is used to generate an initial population.

ACORSDH5: The hybrid algorithm is similar to ACORSDH3, but the translational propagation Latin hypercube algorithm for generating an initial population is used.

These five hybrid algorithms are employed to solve a number of test problems and compared with various established MHs.

Algorithm 3: hybrid ACOR and SDHInput: Maximum iteration number t_{max} , population size m Output: $\mathbf{x}_{best}, f_{best}$

Initialisation

1. Randomly generate an initial population $P(t)$, where $t = 0$ is an iteration number.
2. Set the parameters q and ξ .
3. For $t = 1$ to t_{max} , do the following:
 4. Create a set of offspring $Q(t)$ following steps (4)–(6) in Algorithm 1.
 5. Select the best $n + 1$ solutions from $P(t - 1) \cup Q(t)$ to generate an initial simplex.
 6. Activate Algorithm 2 to find $\mathbf{x}_{simp}$ where the procedure is terminated after I_{max} iterations.
 7. Set the best m solution from $P(t - 1) \cup Q(t) \cup \mathbf{x}_{simp}$ as the next generation $P(t)$.

Next t

The hybrid optimisers are developed to be used with constrained optimisation problems. In this work, we employ the fuzzy set theory (FST) as a penalty function technique (Cheng & Li, 1997). For a typical minimisation problem with inequality constraints:

$$\min f(\mathbf{x}) \quad (6)$$

subject to

$$g_i(\mathbf{x}) \leq 0, i = 1, \dots, M$$

$$\mathbf{L} \leq \mathbf{x} \leq \mathbf{U},$$

Table 4. Comparison of FI based on the statistical test.

	ABC	BBBC	BPBIL	CKS	CMAES	CPBIL	CSS	ChSS	DE	EP	ES	FWA	GA	GSA	HS	LCA	PSO	SA	TLBO	ACOR	ACORS DH1	ACORS DH2	ACORS DH3	ACORS DH4	ACORS DH5	
ABC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
BBBC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
BPBIL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
CKS	0	1	0	0	1	1	0	0	1	1	0	1	1	0	0	1	1	0	1	1	1	1	1	1	1	
CMAES	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
CPBIL	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
CSS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ChSS	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	
DE	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	
EP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
FWA	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
GA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
GSA	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	1	1	0	1	1	1	1	1	1	1	
HS	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	1	1	0	1	1	1	1	1	1	1	
LCA	0	1	1	0	1	1	1	1	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1	1	1	
PSO	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
SA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
TLBO	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	
ACOR	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ACORS DH1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ACORS DH2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ACORS DH3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ACORS DH4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
ACORS DH5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Σ	1	12	2	0	3	9	3	3	8	4	4	8	3	3	1	8	8	1	6	9	7	19	17	19	22	19
Ranking	22	6	21	25	16	7	16	16	9	14	14	9	16	16	22	9	9	22	13	7	2	5	2	1	2	2

Table 5. Comparison of MHs based on the *t*-test.

	ABC	BBBC	BPBIL	CKS	CMAES	CPBIL	CSS	CHSS	DE	EP	ES	FWA	GA	GSA	HS	LCA	PSO	SA	TLBO	ACOR	ACORS DH1	ACORS DH2	ACORS DH3	ACORS DH4	ACORS DH5
F1	22	6	21	25	16	7	16	16	9	14	14	9	16	16	22	9	9	22	13	7	5	2	2	1	2
F2	7	11	17	24	18	11	23	18	14	8	9	10	18	18	24	11	18	14	16	4	1	1	3	4	4
F3	15	6	16	24	13	6	18	18	13	9	9	12	21	21	24	16	18	21	11	8	1	1	1	1	1
F4	13	1	16	25	16	13	16	16	1	1	15	12	16	23	23	16	22	1	1	1	1	1	1	1	1
F5	13	10	1	10	10	13	19	13	19	19	19	19	19	6	19	8	17	13	18	8	6	4	1	1	4
F6	12	5	17	24	9	5	17	22	14	8	11	14	17	22	24	16	17	17	12	9	5	3	3	2	1
F7	10	7	10	16	10	7	19	18	19	19	19	19	19	16	19	10	10	10	1	7	1	3	3	3	3
F8	12	18	18	25	4	9	21	15	18	4	9	12	15	23	23	15	14	1	21	11	3	4	4	2	4
F9	12	8	15	18	1	1	19	12	19	19	19	19	19	15	19	8	11	15	8	12	1	1	1	1	1
F10	9	1	13	13	13	6	22	10	22	19	22	22	19	18	19	11	12	13	17	6	6	1	1	1	1
F11	10	5	17	23	10	23	17	17	5	1	10	10	14	22	17	14	23	17	14	5	2	5	2	5	2
F12	9	13	16	25	11	15	22	22	9	6	6	13	16	22	20	18	20	11	18	6	1	1	4	1	4
F13	11	11	19	24	1	9	11	15	15	6	2	22	19	19	24	15	11	22	15	10	4	7	3	7	7
F14	12	10	15	25	10	14	15	15	9	7	2	23	13	19	19	15	21	23	21	8	1	2	2	2	2
F15	10	10	15	8	15	12	21	18	19	21	21	19	21	15	21	12	14	7	9	3	3	3	1	3	3
F16	12	12	9	12	25	12	5	12	7	12	12	1	10	4	1	12	12	12	12	12	10	8	6	12	3
Σ	189	134	235	321	182	163	281	257	212	173	199	236	272	279	318	206	249	219	207	117	51	44	44	41	41

Table 6. Number of feasible optimum solutions.

ABC	BBBC	BPBIL	CKS	CMAES	CPBIL	CSS	ChSS	DE	EP	ES	FWA	GA	GSA	HS	LCA	PSO	SA	TLBO	ACOR	ACORS DH1	ACORS DH2	ACORS DH3	ACORS DH4	ACORS DH5
F1	30	30	23	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
F2	30	30	22	30	30	29	30	30	30	30	30	30	30	25	30	30	30	30	30	30	30	30	30	30
F3	30	30	27	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
F4	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
F5	18	25	29	21	19	20	1	21	5	4	2	6	4	26	6	25	15	20	10	25	26	28	27	26
F6	30	30	27	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
F7	30	30	30	30	30	0	30	0	0	0	0	0	0	0	0	30	30	30	30	30	30	30	30	30
F8	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
F9	30	30	28	6	30	30	16	30	17	16	14	12	17	30	13	30	28	30	30	30	30	30	30	30
F10	30	30	30	30	30	11	30	14	21	17	14	21	30	20	30	30	30	30	30	30	30	30	30	30
F11	30	30	30	30	29	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
F12	30	30	2	30	30	29	26	30	30	30	30	30	22	30	30	29	30	30	30	30	30	30	30	30
F13	30	30	6	30	30	30	30	30	30	30	23	30	30	5	30	30	24	30	30	30	30	30	30	30
F14	30	30	8	30	30	30	30	30	30	30	18	30	30	30	26	19	27	30	30	30	30	30	30	30
F15	24	27	10	30	11	13	0	0	0	0	0	0	0	0	5	13	30	30	30	30	30	30	30	30
F16	7	9	16	7	0	12	25	11	18	10	6	30	14	29	30	10	6	7	11	14	17	19	9	30

where g_i are inequality constraints. By using the FST, the penalty function $F(\mathbf{x})$ can be written as

$$F = f_z + K_D(1 - \mu_C), \quad (7)$$

where K_D is a penalty parameter and f_z is the objective function value in the fuzzy environment. The parameter μ_C is the total membership function value of the problem (6) computed as

$$\mu_C(\mathbf{x}) = \begin{cases} 0; & \text{if } \mathbf{x} \text{ is feasible} \\ 1; & \text{if } g_i(\mathbf{x}) > 0; \text{ for all } i = 1, \dots, M \\ \frac{1}{M} \sum_{i=1}^M \mu_i; & \text{otherwise} \end{cases} \quad (8)$$

where μ_i is the membership function value of the constraint $g_i(\mathbf{x})$ defined as

$$\mu_i(\mathbf{x}) = \begin{cases} 0; & \text{if } g_i(\mathbf{x}) \geq \bar{d}_i \\ 1 - \frac{g_i(\mathbf{x}) - \varepsilon_i}{\bar{d}_i}; & \text{if } \varepsilon_i < g_i(\mathbf{x}) < \bar{d}_i \\ 1; & \text{if } g_i(\mathbf{x}) \leq \varepsilon_i \end{cases} \quad (9)$$

where d_i and ε_i are the parameters to be specified.

In Cheng and Li (1997), the objective function is modified to the fuzzy environment as $f_z = f/(f + c)$, where c is a constant. It can be derived that the optimum of f_z is the same solution as that of the original objective function $f(\mathbf{x})$. In a classical exterior penalty function technique and some other penalty function techniques, it is difficult to define the penalty parameter since the small value of the parameter can lead the optimum to the outside of the feasible region. On the other hand, an excessively large penalty parameter may make it difficult for the optimiser to reach the real optimum. The aim of using f_z in FST is to scale down the function value in the range of $[0, 1]$ (given that $f(\mathbf{x}) \geq 0$). Also, the penalty term μ_C is scaled down to the range of $[0, 1]$. This gives FST an advantage to define the value of the penalty parameter K_D . However, it can be seen that if the range of the objective function has negative values, the value of f_z will be out of the interval $[0, 1]$ and such an advantage is gone. In this work, we modify the value of the f_z as

$$f_z = (f + c_1)/(f + c_1 + c_2), \quad (10)$$

where c_1 is a constant shift used to ensure that $f + c_1$ is always positive, and c_2 is a positive constant.

5. Test problems

Seventeen constrained test problems were used to examine the search performance of the proposed hybrid methods. Most of them are constrained mechanical design problems about which more details can be found in the literature. The

problems are detailed as follows:

- F1: Composite leaf spring design I (Rajendran & Vijayarangan, 2007), $n = 2$
- F2: Torsional bar spring design (Rajendran & Vijayarangan, 2007), $n = 2$
- F3: Composite leaf spring design II (Rajendran & Vijayarangan, 2007), $n = 3$
- F4: Coil spring design (Rajendran & Vijayarangan, 2007), $n = 3$
- F5: Design of a tension/compression spring (Rao, Savsani, & Vakharia, 2011), $n = 3$
- F6: Steel leaf spring design (Rajendran & Vijayarangan, 2007), $n = 3$
- F7: Belleville spring (Rao et al., 2011), $n = 4$
- F8: Design of a pressure vessel (Rao et al., 2011), $n = 4$
- F9: Welded beam design (Rao et al., 2011), $n = 4$
- F10: Multiple disc clutch brake design (Rao et al., 2011), $n = 5$
- F11: Weight design of a gear train (Savsani, Rao, & Vakharia, 2010), $n = 5$
- F12: Design of a gear train (Rao et al., 2011), $n = 7$
- F13: TLBO benchmark function III (Rao et al., 2011), $n = 7$
- F14: Rolling element bearing (Rao et al., 2011), $n = 10$
- F15: TLBO benchmark function I (Rao et al., 2011), $n = 13$
- F16: Design of a 10-bar truss, $n = 10$

The 10-bar truss is shown in Figure 1. The structure is subject to external forces $F_1 = 80$ N and $F_2 = 175$ N. The design problem can be written as

$$\min f(\mathbf{x}) \quad (11)$$

subject to

$$\sigma_{\max}(\mathbf{x}) - \sigma_{yt} \leq 0$$

$$P_i - P_{cr} \leq 0$$

$$x_i = d_i \in \{0.04, 0.05, 0.06, 0.07, 0.08\}, m$$

where d_i is the diameter of the i th bar element, $\sigma_{\max}$ is the maximum stress on the bar elements, P_i is the axial load acting on the i th element, while P_{cr} is the critical buckling load for the element. The structure is made up of material with Young modulus $E = 200 \times 10^9$ N/m², yield stress $\sigma_{yt} = 235 \times 10^6$ N/m² and density $\rho = 7800$ kg/m³.

- F17: Torque arm optimisation (Botkin, 1995)

The torque arm structure used as a component in an automotive suspension system is shown in Figure 2. The

Table 7. Comparative results of F17.

Optimisers	ABC	BBBC	ACOR	ACORS DH1	ACORS DH2	ACORS DH3	ACORS DH4	ACORS DH5
Mean objective values	0.922152	0.913344	0.913374	0.913344116	0.913392492	0.913305876	0.913324134	0.913347716
Ranking based on the <i>t</i> -test	8	2	2	2	2	1	2	2
Number of feasible optima	5	5	5	5	5	5	5	5

design problem can be written as

$$\min : w_g \quad (12)$$

subject to

$$\sigma_{von} - \sigma_y / S_F = 0,$$

$$1 - \lambda \leq 0,$$

$$\omega - \omega_0 \leq 0,$$

where w_g is the structural weight, ω is the first-mode structural natural frequency, λ = lateral buckling factor (the ratio of the critical load to applied load), σ_{von} is the maximum von Mises stress on the structure and $\omega_0 = 5.5$ rad/s. Furthermore, material properties are as follows: Young's modulus $E = 60 \times 10^6$ N/cm², Poisson's ratio $\nu = 0.3$, yield stress $\sigma_y = 80000$ N/cm², density $\rho = 0.00718$ kg/cm³ and safety factor $S_F = 1.5$. The design variables and their bound constraints are given in Table 1.

The optimisers used for this comparative performance study are as follows (details of notations and parameter settings can be found in the corresponding reference of each method):

ABC: Artificial bee colony algorithm (Karaboga & Basturk, 2007). The number of food sources is set to be $3n$. A trial counter to discard a food source is 100.

BBBC: Big Bang–Big Crunch algorithm (Erol & Eksin, 2006). The centre point selection method 2 is used to create a new population.

BPBIL: Population-based incremental learning (Baluja, 1994). The learning rate, mutation shift and mutation rate are set as 0.5, 0.2 and 0.05, respectively.

CKS: Cuckoo search (Yang & Deb, 2010). Discovery rate of alien eggs (p_a) is 0.25.

CMAES: Evolution strategy with covariance matrix adaptation (Hansen, Müller, & Koumoutsakos, 2003). Use the default parameter settings provided by the authors.

CPBIL: Continuous population-based incremental learning (Bureerat, 2011a). The learning rate, mutation shift and mutation rate are set as 0.5, 0.2 and 0.05, respectively. PBIL3 in the reference is employed with $T_{start} = 5$ and $T_{end} = 40$.

CSS: Continuous scatter search (Herrera, Lozano, & Molona, 2006). The BLX- α recombination method is used. The number of high-quality solutions in the reference set is $0.25 \times 6n$, and the number of diverse solutions in the reference set is $0.25 \times 6n$.

Table 8. Computational time (seconds) for solving F1–F16.

	ABC	BBBC	BPBIL	CKS	CMAES	CPBIL	CSS	ChSS	DE	EP	ES	FWA	GA
F1	0.0196	0.0157	0.0214	0.0187	0.0786	0.0198	0.0306	0.0294	0.0163	0.0157	0.0231	0.0372	0.0278
F2	0.0164	0.0112	0.0180	0.0156	0.0688	0.0217	0.0284	0.0251	0.0114	0.0116	0.0150	0.0345	0.0278
F3	0.0399	0.0306	0.0500	0.0365	0.1281	0.0426	0.0715	0.0642	0.0298	0.0306	0.0393	0.0961	0.0702
F4	0.0251	0.0151	0.0393	0.0208	0.1147	0.0305	0.0525	0.0544	0.0171	0.0193	0.0291	0.1419	0.0590
F5	0.0341	0.0260	0.0432	0.0319	0.1269	0.0363	0.0587	0.0597	0.0239	0.0273	0.0336	0.0866	0.0673
F6	0.0393	0.0315	0.0489	0.0393	0.1240	0.0466	0.0651	0.0632	0.0328	0.0312	0.0368	0.0957	0.0709
F7	0.0616	0.0540	0.1065	0.0689	0.2086	0.0741	0.1193	0.1269	0.0475	0.0589	0.0720	0.1865	0.1475
F8	0.0466	0.0354	0.0803	0.0440	0.1847	0.0579	0.1013	0.1027	0.0342	0.0414	0.0518	0.1833	0.1270
F9	0.0598	0.0427	0.0906	0.0598	0.2046	0.0715	0.1218	0.1191	0.0466	0.0536	0.0627	0.1811	0.1379
F10	0.0792	0.0547	0.1454	0.0702	0.1197	0.0977	0.1616	0.1867	0.0564	0.0682	0.0905	0.3104	0.2349
F11	0.1406	0.1097	0.1981	0.1337	0.3425	0.1468	0.2335	0.2430	0.1110	0.1208	0.1380	0.3612	0.2896
F12	0.1843	0.1311	0.3550	0.1730	0.5316	0.2134	0.4596	0.4399	0.1259	0.1431	0.1902	0.7966	0.5972
F13	0.1654	0.1109	0.3342	0.1388	0.5239	0.1929	0.4267	0.4093	0.1178	0.1252	0.1640	0.7690	0.5723
F14	0.3787	0.2682	0.9031	0.3553	1.0446	0.4588	1.0049	1.0650	0.2607	0.3036	0.3920	2.1267	1.5981
F15	0.6547	0.4963	1.9455	0.5887	1.8142	0.8948	2.4024	2.6185	0.5953	0.6743	0.9198	5.3047	4.3245
F16	2.8726	2.8080	3.4352	2.9054	3.5611	2.9704	3.8518	3.5381	2.7796	2.8159	2.9314	4.5762	4.1422
Σ	4.8178	4.2412	7.8146	4.7005	9.1767	5.3758	9.1897	9.1452	4.3061	4.5408	5.1894	15.2877	12.4942

Table 9. Computational time (seconds) for solving F1–F16.

	GSA	HS	LCA	PSO	SA	TLBO	ACOR	ACORS DH1	ACORS DH2	ACORS DH3	ACORS DH4	ACORS DH5
F1	0.0249	0.0191	0.0337	0.0158	0.0214	0.0200	0.0251	0.0291	0.0264	0.0353	0.0270	0.0337
F2	0.0184	0.0165	0.0300	0.0130	0.0142	0.0173	0.0214	0.0225	0.0232	0.0324	0.0241	0.0308
F3	0.0490	0.0435	0.0727	0.0286	0.0359	0.0409	0.0650	0.0632	0.0635	0.0816	0.0666	0.0783
F4	0.0347	0.0319	0.0541	0.0184	0.0211	0.0243	0.0543	0.0522	0.0546	0.0680	0.0535	0.0678
F5	0.0431	0.0407	0.0640	0.0242	0.0283	0.0358	0.0582	0.0565	0.0560	0.0669	0.0556	0.0685
F6	0.0482	0.0442	0.0695	0.0304	0.0345	0.0385	0.0623	0.0636	0.0652	0.0789	0.0643	0.0779
F7	0.0885	0.0788	0.1125	0.0452	0.0517	0.0646	0.1364	0.1272	0.1267	0.1598	0.1296	0.1576
F8	0.0695	0.0580	0.0996	0.0368	0.0414	0.0532	0.1128	0.1148	0.1091	0.1430	0.1180	0.1394
F9	0.0890	0.0741	0.1142	0.0448	0.0522	0.0580	0.1237	0.1222	0.1182	0.1479	0.1179	0.1517
F10	0.1338	0.1134	0.1618	0.0564	0.0632	0.0750	0.2073	0.2010	0.2015	0.2521	0.1987	0.2634
F11	0.1958	0.1585	0.2257	0.1144	0.1204	0.1428	0.2612	0.2578	0.2505	0.3065	0.2495	0.3108
F12	0.3654	0.2824	0.3509	0.1241	0.1345	0.1810	0.5431	0.5369	0.5223	0.6554	0.5251	0.6571
F13	0.3207	0.2479	0.3278	0.1045	0.1297	0.1418	0.5290	0.5161	0.5052	0.6430	0.5275	0.6582
F14	0.9399	0.7304	0.7138	0.2651	0.2781	0.3586	1.5453	1.5005	1.4958	1.8815	1.5515	1.9425
F15	2.5748	1.9538	1.4722	0.6198	0.6139	0.8814	3.3485	3.7654	3.8853	4.7136	4.2172	4.8579
F16	3.4480	3.1838	3.2285	2.7970	2.8168	2.8884	4.0477	3.9054	3.8877	4.2431	3.9473	4.2684
Σ	8.4436	7.0770	7.1309	4.3383	4.4573	5.0215	11.1414	11.3344	11.3912	13.5090	11.8733	13.7640

Table 10. Computational time (seconds) for solving F17.

Optimisers	ABC	BBBC	ACOR	ACORS DH1	ACORS DH2	ACORS DH3	ACORS DH4	ACORS DH5
Time (seconds)	2406.98	2786.641	2768.298	2781.017	2778.815	2824.503	2832.791	2847.425

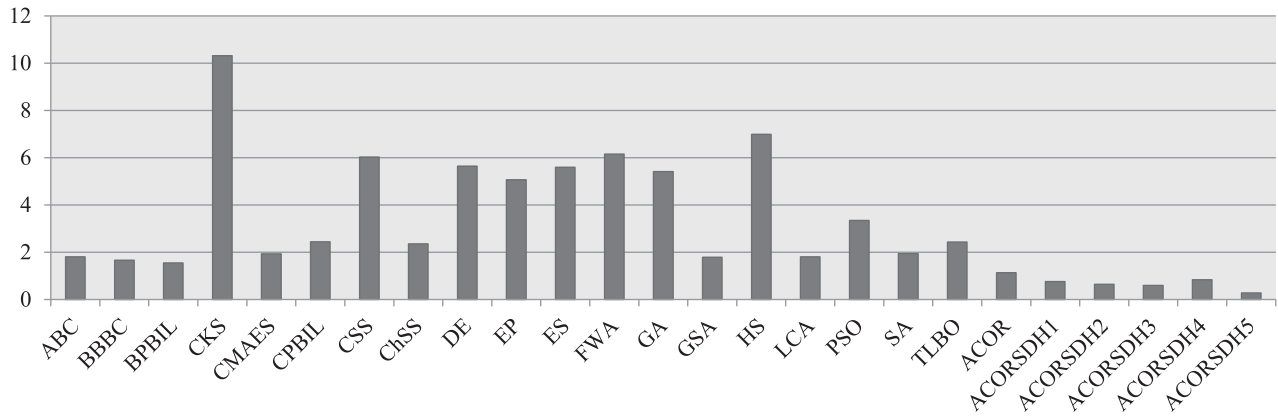


Figure 3. Summation of mean objective function values of F1–F16 from Table 2 and 3.

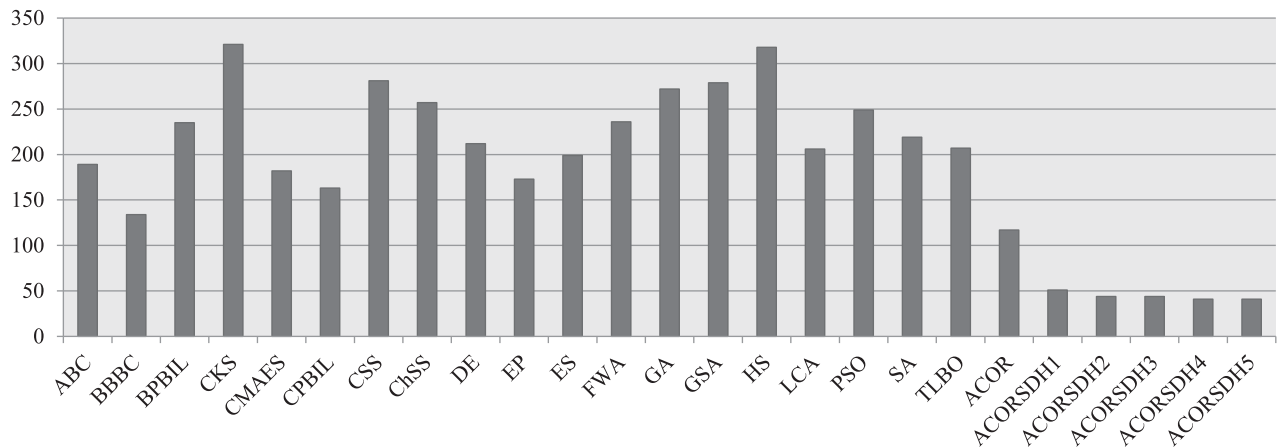


Figure 4. Summation of *t*-test ranking values from Table 5.

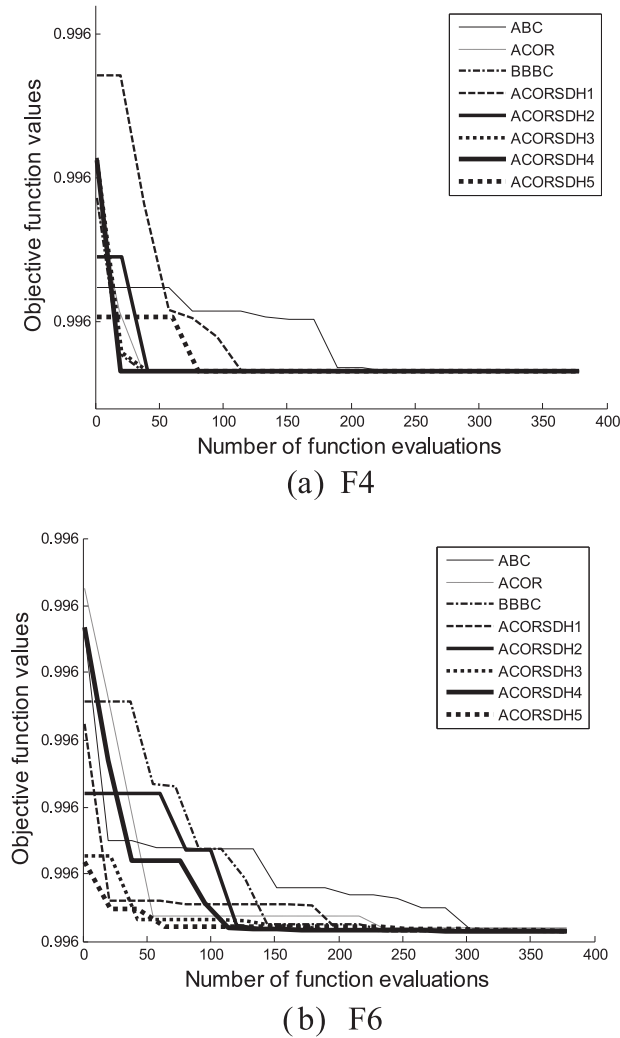


Figure 5. Search histories of (a) F4 and (b) F6.

ChSS: Charged system search (Kaveh & Talatahari, 2010). The number of solutions in the charge memory is $0.2 \times 6n$. The charged moving considering rate and the parameter PAR are set to be 0.75 and 0.5, respectively.

DE: Differential evolution (Storn & Price, 1997). Use the DE/rand/1/bin strategy. The scaling factor, crossover rate and probability of choosing offspring elements are 0.5, 0.7 and 0.8, respectively.

EP: Evolutionary programming (Bäck, 1996). Use a tournament selection operator.

ES: Evolution strategies (Bäck, 1996). Use a tournament selection operator and a simple mutation without the effect of rotation angles.

GA: Genetic algorithm (Lindfield & Penny, 1995). The crossover and mutation rates are 1.0 and 0.1, respectively.

GSA: Gravitational search algorithm (Rashedi, Nezamabadi-pour, & Saryazdi, 2009). Use the default parameter settings provided by the authors.

HS: Harmony search (Geem & Kim, 2001). The parameter settings are as follows: harmony memory size = $5n$, harmony memory considering rate ($HMCR$) = 0.8, pitch adjusting rate (PAR) $\in [0.4, 0.9]$ and bandwidth (BW) $\in [0.0001, 0.9]$.

LCA: League championship algorithm (Kashan, 2011). Use the default parameter settings provided by the authors.

PSO: Particle swarm optimisation (Reyes-Sierra & Coello Coello, 2006). The starting inertia weight, ending inertia weight, cognitive learning factor and social learning factor are assigned as 0.5, 0.01, 2.8 and 1.3, respectively.

SA: Simulated annealing (Bureerat & Limtragool, 2008). The starting and ending temperatures are 10 and 0.001, respectively. One each loop, n candidates are created by mutating on the current best solution, while another n candidates are created from mutating the current parent. The best of those $2n$ solutions are set as offspring to be compared with the parent.

TLBO: Teaching-learning-based optimisation (Rao et al., 2011).

ACOR: Real-code ant colony optimisation (Socha & Dorigo, 2008). The parameter settings are $q = 0.2$ and $\xi = 1.0$.

ACORS DH1: Use the parameter settings as ACOR. The maximum number of iterations for the SDH method is $I_{max} = 5$. An initial population is created randomly.

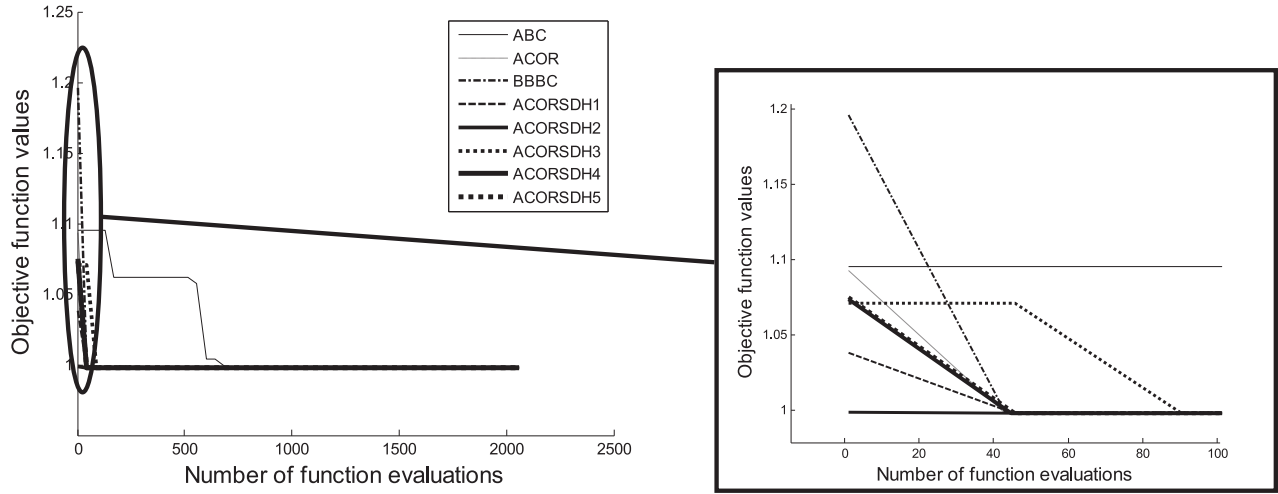
ACORS DH2: Use the parameter settings as ACORS DH1. Also, use the Latin hypercube sampling technique (LHS) for creating an initial population.

ACORS DH3: Use the parameter settings as ACORS DH2. Use the crowding distance algorithm for selecting the vertices of an initial simplex.

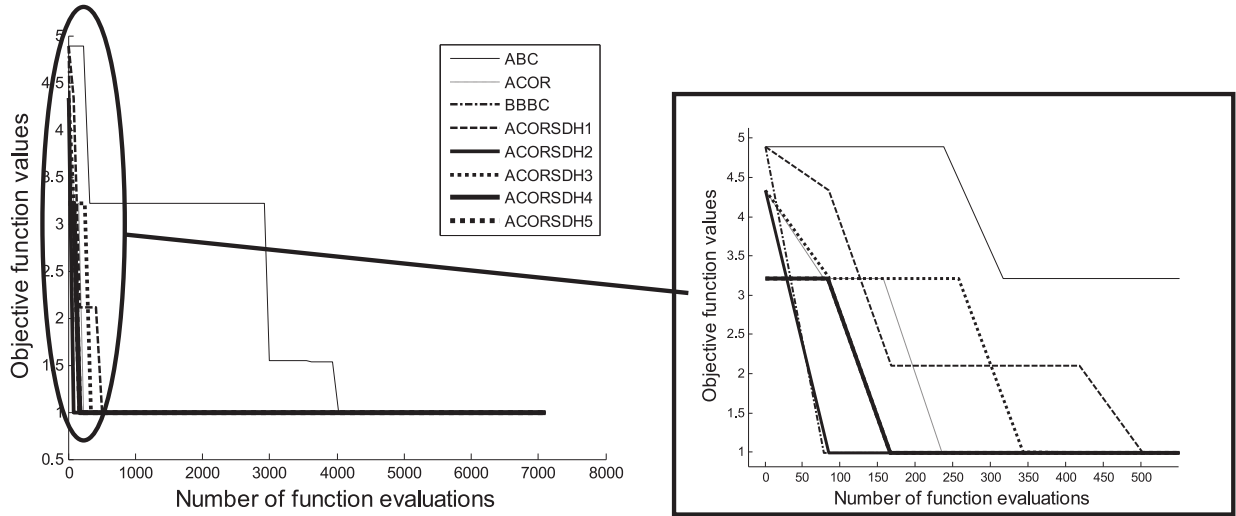
ACORS DH4: Use the parameter settings as ACORS DH1 and use translational propagation Latin hypercube design (TPLHD) for creating an initial population.

ACORS DH5: Use the parameter settings as ACORS DH4. Use the crowding distance algorithm for selecting the vertices of an initial simplex.

Each optimiser is used to tackle the optimisation test problems (F1–F16) for 30 optimisation runs. The parameters for the FST penalty function technique are set as $\varepsilon = -0.0001$, $d = 1$, $c_1 = 2500$, $c_2 = 10$ and $K_D = 5$. The best performers obtained from performance comparison based on F1–F16 will be used to solve F17 with five runs for each method since the function evaluation of this problem is somewhat time-consuming. The population size is set



(a) F12



(b) F15

Figure 6. Search histories of (a) F12 and (b) F15.

to be $6n$, whereas the number of iterations is $7n$ for all test cases. For the optimisers using a different population size or having a local search, their search processes are terminated with the total number of function evaluations as $6n \times 7n$.

6. Results

6.1. Comparative performance

The comparative performance of the various optimisers for solving the test functions F1–F16 is displayed in Table 2–10. In Table 2 and 3, each cell represents the mean value of 30 objective function values of the near optima of a test problem obtained from using one optimiser. To ease in viewing, the first eight best optimisers are highlighted in grey while

the top four methods are additionally boldfaced. Figure 3 shows the bar chart of the summation of mean objective function values for all test problems. In this figure, for each test problem, the mean objective value from the best optimiser is normalised to be 0, while that from the worst optimiser is normalised to be 1. This is in order to make a fair comparison of all test problems. When considering all test problems, it can be seen that ACORSDH5 gives the best overall results, whereas the second and third best are ACORSDH3 and ACORSDH2, respectively. The fourth and fifth bests are ACORSDH1 and ACPRSDH4, respectively. The original ACOR is the sixth best (the best for non-hybrid optimisers), while the BBBC and ABC algorithms are the seventh and eighth best optimisers, respectively. This

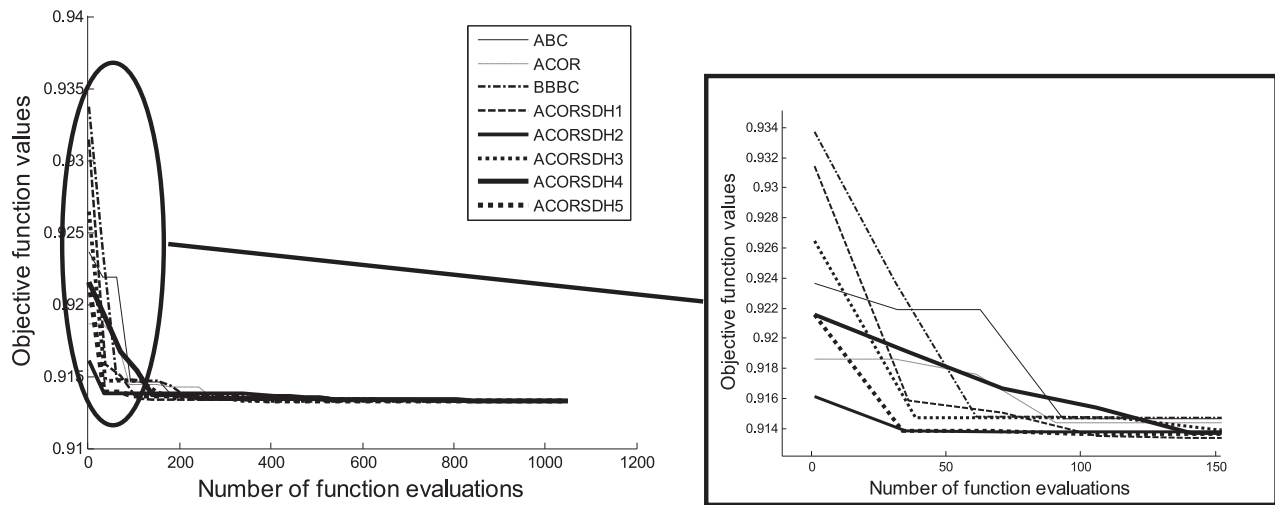


Figure 7. Search histories of the real engineering test problem of torque arm.

implies that integrating a simplex downhill local search to ACOR leads to a more powerful MH. ACORSDH4, although being ranked the fifth best for the hybrid algorithms, gives the best results (mean value = 0) for five test problems. The method only fails to converge to a near minimum of F16, 10-bar truss design.

A more reliable ranking of MHs is conducted based upon the statistical t -test. Table 4 shows how to assess the performance of MHs for solving F1. The comparison starts with a square matrix T , sized 25×25 (equal to the number of optimisers) whose elements are full of zeros. Then, the

results obtained from method i and method j are compared using the statistical t -test at 95% confidence level. In cases that the mean value from method i is significantly different from that obtained from method j , the element T_{ij} is changed to be 1, if the mean value obtained from method i is higher; otherwise, the element T_{ji} is changed to be 1. Having compared all the optimisers, the results are given in Table 4. When summing up each column, the optimiser having the highest score is said to be the best performer. In this test case, ACORSDH4 is the best method for F1. The results of ranking the optimisers for all test problems are

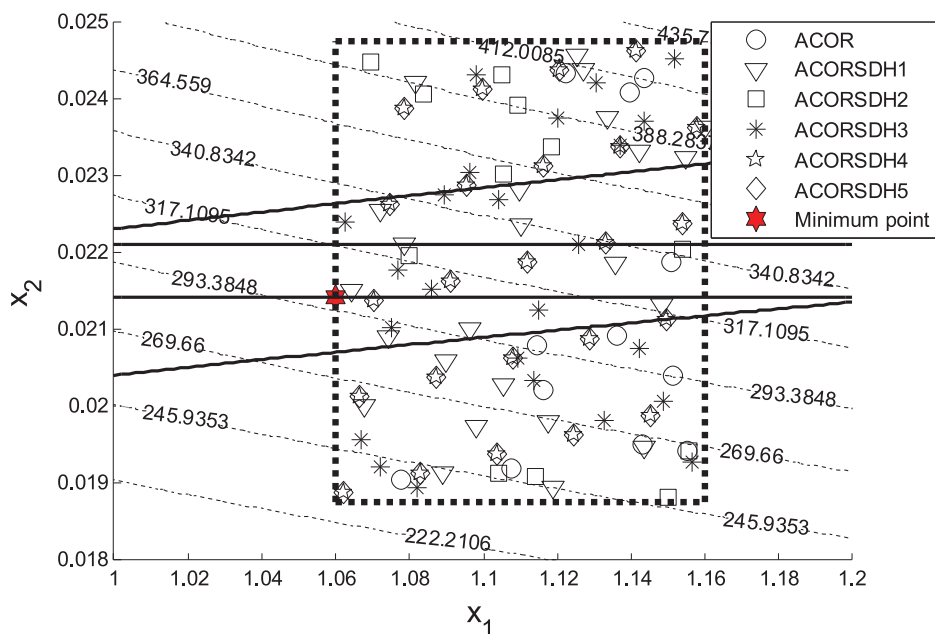


Figure 8. Design points at the initialisation step.

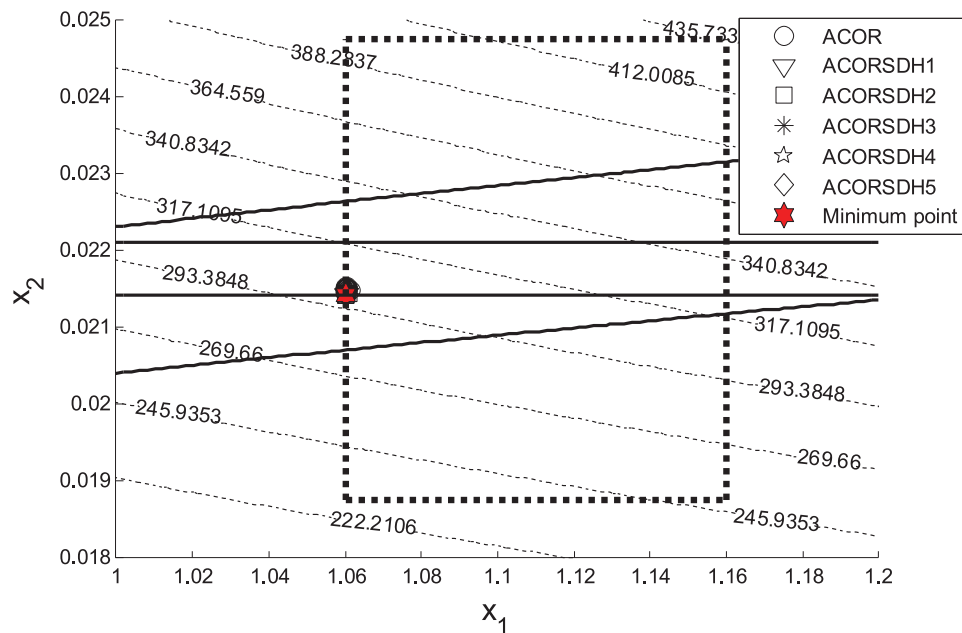


Figure 9. Design points at the final generation.

shown in Table 5. After summing up each column for all the test problems in Table 5, the optimiser having the lowest score is the best performer. Figure 4 shows the summation of the ranking values in Table 5. The overall best optimisers are ACORSDH4 and ACORSDH5, while the close third best optimisers are ACORSDH2 and ACORSDH3. ACORSDH1 is the fifth best and the worst among the proposed hybrid ACORs. The sixth best is the original ACOR. The seventh and eighth best optimisers are BBBC and CPBIL, respectively. From the two rankings, it can be seen that the integration of the SDH method to the main procedure of ACOR helps increase its search performance. When

using the Latin hypercube sampling technique to generate an initial population, the performance of the hybrid optimiser is increased. It is even further increased when the optimum Latin hypercube sampling technique (TPLHD) is employed. The two numerical schemes used to select an initial simplex gave insignificantly different optimum results, although ACORSDH3 and ACORSDH5 seem to have an edge over ACORSDH2 and ACORSDH4, respectively, based on both ranking systems.

Figures 5–7 show search histories of some test problems. The selected problems represent low-dimensional (F4 and F6), high-dimensional (F12 and F15) and

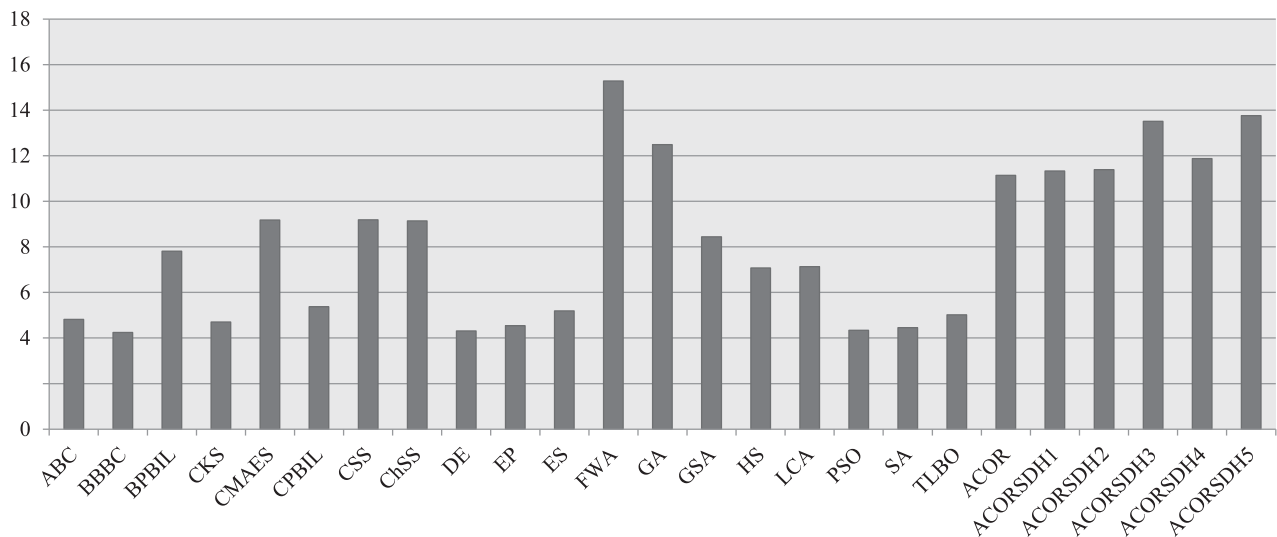


Figure 10. Summation of the computational time for all test problems from Table 8 and 9.

practical (torque arm) design problems. The selected algorithms include ABC, ACOR, BBBC, and the five proposed hybrid algorithms. The best runs of the algorithms for each problem are illustrated. For the low-dimensional problems, ACORSDH5 using TPLHD for generating initial solutions gives the best initial solutions. For the high-dimensional test problems, ACORSDH2, which uses LHS, gives the best initial solutions for F12, while ACORSDH4 using TPLHD gives the best initial solutions for F15. For the torque design problem, ACORSDH2, using LHS generates the best initial solutions. It can be seen that using LHS or TPLHD can result in better initial sets of design solutions and this enhances the convergence rate of the algorithms.

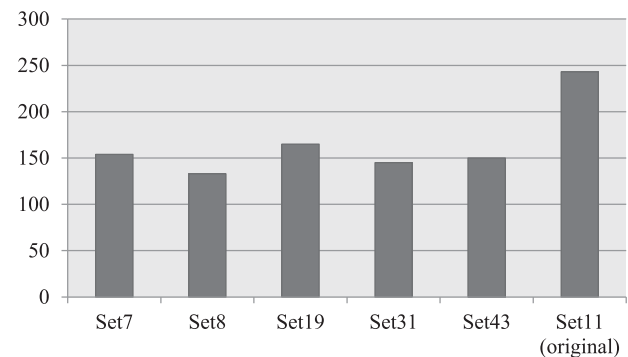
Figure 8 displays objective function contours and feasible region boundaries (dashed lines are bound constraints) of F2, whereby the initial design points by the various hybrid algorithms are plotted. It can be seen that the initial points by means of TPLHD give more evenly distributed positions. This means that an optimiser using this optimal Latin hypercube sampling technique will have better population diversity initially. Figure 9 shows the design points from the various methods at the final iteration. It is shown that all the optimisers can approach the optimum because the problem has only two design variables.

Table 6 shows the number of feasible near optima found by each optimiser solving each test problem. With the exception of F5 and F16, the five hybrid algorithms can find the feasible near optima for all 30 runs for all test problems which is superior to the other MHs. The original ACOR and its five hybrid variants along with ABC and BBBC were employed to solve the F17 test problem with five runs for each method. The comparative results are given in Table 7. All the optimisers can find feasible near optima for all five runs. Based on the t -test and the mean objective values, the best optimiser is ACORSDH3. It should be noted that these comparison results more or less rely on the parameter settings of the optimisers.

Table 8 and Table 9 show the average computational time (in seconds) spent by the optimisers for solving each test problem, while Figure 10 shows the summation of the computational time of all cases from Table 8 and 9. The results show that BBBC is the fastest in searching for optima, while the second fastest is DE. For the proposed hybrid algorithms, the computational time slightly increases from the original ACOR. The algorithms using TPLHD need a little longer computing time than those using LHS, while LHS needs a slightly longer computational time than the Monte Carlo method. In addition, the computational time of the hybrid algorithms is increased when applying the crowding distance algorithm. For the test problem F17, the fastest of the eight algorithms is ABC, while the second and third fastest are ACOR and BBBC, respectively. For the hybrid optimisers, the conclusion is similar to the cases of F1–F16.

Table 11. Comparison based on the t -test (F1–F17).

	Set7	Set8	Set19	Set31	Set43	Set11 (original)
F1	12	7	13	3	8	8
F2	2	21	6	13	9	8
F3	13	11	2	9	7	38
F4	1	1	1	1	1	1
F5	23	1	1	1	23	23
F6	16	3	7	7	15	12
F7	11	11	28	11	11	28
F8	8	17	15	23	22	1
F9	21	2	24	31	17	31
F10	1	1	1	1	1	28
F11	9	2	30	14	2	36
F12	3	13	3	2	1	3
F13	13	13	13	13	13	2
F14	9	10	3	3	7	15
F15	8	16	14	9	9	5
F16	3	3	3	3	3	3
F17	1	1	1	1	1	1
Σ	154	133	165	145	150	243

Figure 11. Summation of t -test ranking scores from Table 11.

6.2. Optimisation parameter settings

In order to investigate, the optimisation parameters of ACORSDH5, one of the top performers, q , ξ and the number of iterations of the SDH algorithms (I_{max}), were varied, leading to numerous sets of parameters. The values of the parameters q , ξ and I_{max} were assigned as $\{0.2, 0.5, 0.8, 1\}$, $\{0.2, 0.5, 0.8, 1\}$ and $\{3, 5, 10\}$, respectively. Therefore, there were $4 \times 4 \times 3 = 48$ sets of the optimisation parameters as $\{0.2, 0.5, 0.8, 1\} \times \{0.2, 0.5, 0.8, 1\} \times \{3, 5, 10\}$. Each parameter set is employed with ACORSDH5 to solve the test problems F1–F16 for 30 optimisation runs, and to solve the F17 test problem for five runs.

Having run all 48 optimisers for solving all test problems, we selected the first five best parameter sets to compare. The parameter sets were named Set 7 ($q = 0.2$, $\xi = 0.8$, $I_{max} = 3$), Set8 ($q = 0.2$, $\xi = 0.8$, $I_{max} = 5$), Set19 ($q = 0.5$, $\xi = 0.8$, $I_{max} = 3$), Set31 ($q = 0.8$, $\xi = 0.8$, $I_{max} = 3$) and Set43 ($q = 1.0$, $\xi = 0.8$, $I_{max} = 3$). The original parameter set used in the previous subsection is termed Set11 ($q = 0.2$, $\xi = 1.0$, $I_{max} = 5$). Table 11 shows the

comparison using the statistical t -test, similarly to Table 5, and Figure 11 shows the summation of the ranking scores of all test problems in Table 11. Based on the t -test ranking in Table 11 and Figure 11, the top optimisers are in the order of Set8, Set31, Set43, Set7, Set19 and Set17. The original parameter set gives the worst results. From the comparative results, the parameters that have observable impacts on the algorithm search performance are ξ and I_{max} . The former should be set at a higher value, while the latter should have a lower value.

7. Conclusions

A new hybrid MH is proposed. The method is based on the integration of a local direct search SDH method into the main search procedure of ACOR, resulting in five variants of the hybrid algorithm. The hybrid methods along with other established MHs are then implemented to solve a number of constrained test design problems. Based on two ranking methods, it can be concluded that the proposed hybrid methods give the best results. The use of the optimum Latin hypercube sampling technique for generating an initial population results in performance enhancement of the hybrid algorithms. The optimisation parameters that lead to an enhanced hybrid optimiser are ξ and I_{max} . The former should be set at a higher value, while the latter should have a lower value. The modified FST penalty function technique is capable of handling design constraints. This hybrid optimiser can be considered a powerful tool for constrained mechanical system design.

Acknowledgements

The authors are grateful for support from the Thailand Research Fund [Grant number BRG5580017].

Notes on contributors



Nantiwat Pholdee received his BEng degree (Second Class Honors) in mechanical engineering in 2008 and his PhD degree in mechanical engineering in 2013 from Khon Kaen University, Khon Kaen, Thailand. His research interests include multidisciplinary design optimisation, evolutionary computation and finite-element analysis.



Sujin Bureerat received his BEng degree in mechanical engineering from Khon Kaen University, Khon Kaen, Thailand, in 1992, and his PhD degree in mechanical engineering from Manchester University, Manchester, UK, in 2001. Currently, he is an associate professor with the Department of Mechanical Engineering, Khon Kaen University. His research interests include

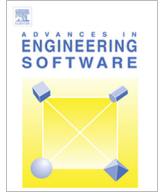
multidisciplinary design optimisation, evolutionary computation, aeroelastic design of aircraft structures, finite-element analysis, agricultural machinery and mechanical vibration.

References

- Afshar, A., & Madadgar, S. (2008). Ant colony optimization for continuous domains: Application to reservoir operation problems. In *Proceedings of the Eighth International Conference on Hybrid Intelligent Systems* (pp. 13–18). Barcelona, Spain.
- Afshar, M.H. (2010). A parameter free Continuous Ant Colony Optimization Algorithm for the optimal design of storm sewer networks: Constrained and unconstrained approach. *Advances in Engineering Software*, 41, 188–195.
- Bäck, T. (1996). *Evolutionary algorithms in theory and practice*. Oxford: Oxford University Press.
- Baluja, S. (1994). *Population-based incremental learning: A method for integrating genetic search based function optimisation and competitive learning* (Technical Report CMU-CS-95_163). Pittsburgh, PA: School of Computer Science, Carnegie Mellon University.
- Botkin, M.A. (1995). Shape optimization with buckling and stress constraints. *AIAA*, 34, 423–425.
- Brest, J., Korošec, P., Šilc, J., Zamuda, A., Bošković, B., & Maučec M.S. (2013). Differential evolution and differential ant-stigmergy on dynamic optimisation problems. *International Journal of Systems Science*, 44, 663–679.
- Bureerat, S. (2011a). Population-based incremental learning in continuous spaces. *Advances in Intelligent and Soft Computing*, 96, 77–86.
- Bureerat, S. (2011b). Hybrid population-based incremental learning using real codes. *Lecture Notes in Computer Science*, 6683, 379–391.
- Bureerat, S., & Limtragool, J. (2008). Structural topology optimisation using simulated annealing with multiresolution design variables. *Finite Elements in Analysis and Design*, 44, 738–747.
- Bureerat, S., & Sriworamas, K. (2007). Population-based incremental learning for multiobjective optimisation. *Advances in Soft Computing*, 9, 223–231.
- Chebouba, A., Yalaoui, F., Smati, A., Amodeo, L., Younsib, K., & Tairi, A. (2009). Optimization of natural gas pipeline transportation using ant colony optimization. *Computers and Operations Research*, 36, 1916–1923.
- Chelouah, R., & Siarry, P. (2005). A hybrid method combining continuous tabu search and Nelder–Mead simplex algorithms for the global optimization of multim minima functions. *European Journal of Operational Research*, 161, 636–654.
- Cheng, F.Y., & Li, D. (1997). Fuzzy set theory with genetic algorithms in constrained structural optimization. In D.M. Frangopol & F.Y. Cheng (Eds.), *Advances in structural optimization: ASCE Proceedings of the US-Japan Joint Seminar on Structural Optimization* (pp. 55–66). New York, NY: American Society of Civil Engineers.
- Ciomei, I., & Kyriakides, E. (2012). Hybrid ant colony-genetic algorithm (GAAP) for global continuous optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42, 234–245.
- Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGAII. *IEEE Transactions on Evolutionary Computation*, 6, 182–197.
- Dorigo, M. (1992). *Optimization, learning and natural algorithms* (in Italian) (Ph.D. thesis). Dipartimento di Elettronica, Politecnico di Milano, Italy.
- Erol, O.K., & Eksin, I. (2006). A new optimisation method: Big Bang–Big Crunch. *Advances in Engineering Software*, 37, 106–111.
- Geem, Z.W., & Kim, J.H. (2001). A new heuristic optimisation algorithm: Harmony search. *Simulation*, 76, 60–68.

- Guo, P., & Zhu, L. (2012). Ant colony optimization for continuous domains. In *Proceedings of the 8th International Conference on Natural Computation (ICNC 2012)* (pp. 758–762). Sichuan, China.
- Hajizadeh, Y., Christie, M., & Demyanov, V. (2011). Ant colony optimization for history matching and uncertainty quantification of reservoir models. *Journal of Petroleum Science and Engineering*, 77, 78–92.
- Hansen, N., Müller, S.D., & Koumoutsakos, P. (2003). Reducing the time complexity of the derandomised evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11, 1–18.
- Hazra, J., & Sinha, A. (2009). Application of soft computing methods for economic dispatch in power systems. *International Journal of Electrical and Electronics Engineering*, 3(9), 538–543.
- Herrera, F., Lozano, M., & Molona, D. (2006). Continuous scatter search: An analysis of the integration of some combination methods and improvement strategies. *European Journal of Operational Research*, 169, 450–476.
- Hsu, C.-M. (2012). Applying genetic programming and ant colony optimisation to improve the geometric design of a reflector. *International Journal of Systems Science*, 43, 972–986.
- Huang, K.-L., & Liao, C.-J. (2008). Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computers & Operations Research*, 35, 1030–1046.
- Kabir, M.M., Shahjahan, M., & Murase, K. (2012). A new hybrid ant colony optimization algorithm for feature selection. *Expert Systems with Applications*, 39, 3747–3763.
- Kanyakam, S., & Bureerat, S. (2012). Comparative performance of surrogate-assisted MOEAs for geometrical design of pin-fin heat sinks. *Journal of Applied Mathematics*, 2012, Article ID 534783.
- Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimisation: Artificial bee colony (ABC) algorithm. *Journal of Global Optimisation*, 39, 459–471.
- Karen, I., Yildiz, A.R., Kaya, N., Öztürk, N., & Öztürk, F. (2006). Hybrid approach for genetic algorithm and Taguchi's method based design optimization in the automotive industry. *International Journal of Production Research*, 44(22), 4897–4914.
- Karimi, A., Nobahari, H., & Siarry, P. (2010). Continuous ant colony system and tabu search algorithms hybridized for global minimization of continuous multi-minima functions. *Computational Optimization and Applications*, 45, 639–661.
- Kashan, A.H. (2011). An efficient algorithm for constrained global optimisation and application to mechanical engineering design: League championship algorithm (LCA). *Computer-Aided Design*, 43, 1769–1792.
- Kaveh, A., & Talatahari, S. (2008). A hybrid particle swarm and ant colony optimization for design of truss structures. *Asian Journal of Civil Engineering (Building and Housing)*, 9, 329–348.
- Kaveh, A., & Talatahari, S. (2009). Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. *Computers and Structures*, 87, 267–283.
- Kaveh, A., & Talatahari, S. (2010). A novel heuristic optimisation method: Charged System Search. *Acta Mechanica*, 213, 267–289.
- Kong, M., & Tian, P. (2006). A direct application of ant colony optimization to function optimization problem in continuous domain. *Lecture Notes in Computer Science*, 4150, 324–331.
- Lee, C.-Y., Lee, Z.-J., Lin, S.-W., & Ying, K.-C. (2010). An enhanced ant colony optimization (EACO) applied to capacitated vehicle routing problem. *Applied Intelligence*, 32, 88–95.
- Liao, T., Montes de Oca, M.A., Aydin, D., Stützle, T., & Dorigo, M. (2011). An incremental ant colony algorithm with local search for continuous optimization. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO'11)* (pp. 125–132). New York, NY: ACM.
- Liao, T.W., Kuo, R.J., & Hu, J.T.L. (2012). Hybrid ant colony optimization algorithms for mixed discrete–continuous optimization problems. *Applied Mathematics and Computation*, 219, 3241–3252.
- Lindfield, G., & Penny, J. (1995) *Numerical methods using MATLAB*. New York, NY: Ellis Horwood.
- Lion, C.-D., Hsieh, Y.-C., & Chen, Y.-Y. (2013). New encoding scheme-based hybrid algorithm for minimising two-machine flow-shop group scheduling problem. *International Journal of Systems Science*, 44, 77–93.
- Madadgar, S., & Afshar, A. (2009). An improved continuous ant algorithm for optimization of water resources problems. *Water Resources Management*, 23, 2119–2139.
- Majumdar, A., Maiti, D.K., & Maity, D. (2012). Damage assessment of truss structures from changes in natural frequencies using ant colony optimization. *Applied Mathematics and Computation*, 218, 9759–9772.
- Mariani, V.C., & dos Santos Coelho, L. (2011). A hybrid shuffled complex evolution approach with pattern search for unconstrained optimization. *Mathematics and Computers in Simulation*, 81, 1901–1909.
- Mohamad, M., Tokhi, M.O., & Omar, M. (2011). Continuous ant colony optimisation for active vibration control of flexible beam structures. In *Proceedings of the IEEE International Conference on Mechatronics (ICM)* (pp. 803–808). Istanbul, Turkey.
- Nakawiro, W., & Erlich, I. (2009). Optimal load shedding for voltage stability enhancement by ant colony optimization. In *Proceedings of the 15th International Conference on Intelligent System Applications to Power Systems, 2009* (pp. 1–6). Curitiba, Brazil.
- Nelder, J.A., & Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, 7, 308–313.
- Panahi, H., & Tavakkoli-Moghaddam, R. (2011). Solving a multi-objective open shop scheduling problem by a novel hybrid ant colony optimization. *Expert Systems with Applications*, 38, 2817–2822.
- Pholdee, N., & Bureerat, S. (2012). Performance enhancement of multiobjective evolutionary optimizers for truss design using an approximate gradient. *Computers and Structures*, 106–107, 115–124.
- Pholdee, N., & Bureerat, S. (2013). Hybridisation of real-code population-based incremental learning and differential evolution for multiobjective design of trusses. *Information Sciences*, 223, 136–152.
- Rajendran, I., & Vijayanagan, S. (2007). Simulated annealing approach to the optimal design of automotive suspension systems. *International Journal of Vehicle Design*, 43(1–4), 11–30.
- Rao, R.V., Savsani, V.J., & Vakharia, D.P. (2011). Teaching–learning-based optimisation: A novel method for constrained mechanical design optimisation problems. *Computer-Aided Design*, 43, 303–315.
- Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Sciences*, 179, 2232–2248.
- Reyes-Sierra, M., & Coello Coello, C.A. (2006). Multi-objective particle swarm optimisers: A survey of the state-of-the-art.

- International Journal of Computational Intelligence Research*, 2, 287–308.
- Savvani, V., Rao, R.V., & Vakharia, D.P. (2010). Optimal weight design of a gear train using particle swarm optimization and simulated annealing algorithms. *Mechanism and Machine Theory*, 45, 531–541.
- Schluter, M., Egea, J.A., Antelo, L.T., Alonso, A.A., & Banga, J.R. (2009). An extended ant colony optimization algorithm for integrated process and control system design. *Industrial & Engineering Chemistry Research*, 48, 6723–6738.
- Shelokar, P.S., Siarry, P., Jayaraman, V.K., & Kulkarni, B.D. (2007). Particle swarm and ant colony algorithms hybridized for improved continuous optimization. *Applied Mathematics and Computation*, 188, 129–142.
- Socha, K., & Blum, C. (2007). An ant colony optimization algorithm for continuous optimization: Application to feed-forward neural network training. *Neural Computing and Applications*, 16, 235–247.
- Socha, K., & Dorigo, M. (2008). Ant colony optimisation for continuous domains. *European Journal of Operational Research*, 185, 1155–1173.
- Srisoporn, S., & Bureerat, S. (2008). Geometrical design of plate-fin heat sinks using hybridization of MOEA and RSM. *IEEE Transactions on Components and Packaging Technologies*, 31, 351–360.
- Storn, R., & Price, K. (1997). Differential evolution – A simple and efficient heuristic for global optimisation over continuous spaces. *Journal of Global Optimisation*, 11, 341–359.
- Tenne, Y., Izui, K., & Nishiwaki, S. (2012). A hybrid model-classifier framework for managing prediction uncertainty in expensive optimization problems. *International Journal of Systems Science*, 43, 1305–1321.
- Viana, F.A.C., Venter, G., & Balabanov, V. (2010). An algorithm for fast optimal Latin hypercube design of experiments. *International Journal for Numerical Methods in Engineering*, 82(2), 135–156.
- Wang, J., Osagie, E., Thulasiraman, P., & Thulasiram, R.K. (2009). HOPNET: A hybrid ant colony optimization routing algorithm for mobile ad hoc network. *Ad Hoc Networks*, 7, 690–705.
- Wang, L., Xu, Y., & Li, L. (2011). Parameter identification of chaotic systems by hybrid Nelder–Mead simplex search and differential evolution algorithm. *Expert Systems with Applications*, 38, 3238–3245.
- Xiao, J., & Li, L. (2011). A hybrid ant colony optimization for continuous domains. *Expert Systems with Applications*, 38, 11072–11077.
- Yang, X.S., & Deb, S. (2010). Engineering optimisation by cuckoo search. *International Journal of Mathematical Modeling and Numerical Optimisation*, 1, 330–343.
- Yildiz, A.R. (2009a). A new design optimization framework based on immune algorithm and Taguchi's method. *Computers in Industry*, 60(8), 613–620.
- Yildiz, A.R. (2009b). A novel hybrid immune algorithm for global optimization in design and manufacturing. *Robotics and Computer-Integrated Manufacturing*, 25, 261–270.
- Yildiz, A.R. (2009c). A novel particle swarm optimization approach for product design and manufacturing. *The International Journal of Advanced Manufacturing Technology*, 40, 617–628.
- Yildiz, A.R. (2012). A new hybrid particle swarm optimization approach for structural design optimization in the automotive industry. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 226, 1340–1351.
- Yildiz, A.R. (2013). Optimization of cutting parameters in multi-pass turning using artificial bee colony-based approach. *Information Sciences*, 220, 399–407.
- Yu, L., & Xu, P. (2011). Structural health monitoring based on continuous ACO method. *Microelectronics Reliability*, 51(2), 270–278.
- Zahara, E., & Kao, Y.T. (2009). Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Systems with Applications*, 36, 3880–3886.
- Zhang, X.L., Chen, X.F., & He, Z.J. (2010). An ACO-based algorithm for parameter optimization of support vector machines. *Expert Systems with Applications*, 37, 6618–6628.
- Zhu, Q., Yang, Z., & Ma, W. (2011). A quickly convergent continuous ant colony optimization algorithm with Scout Ants. *Applied Mathematics and Computation*, 218, 1805–1819.



Comparative performance of meta-heuristic algorithms for mass minimisation of trusses with dynamic constraints



Nantiwat Pholdee, Sujin Bureerat *

Sustainable Infrastructure Research and Development Center, Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand

ARTICLE INFO

Article history:

Received 28 November 2013
Received in revised form 18 April 2014
Accepted 25 April 2014
Available online 24 May 2014

Keywords:

Meta-heuristics comparative performance
Truss minimisation
Dynamic constraints
Evolutionary algorithms
Shape and sizing optimisation
Penalty function techniques

ABSTRACT

This paper investigates the search performances of various meta-heuristics (MHs) for solving truss mass minimisation with dynamic constraints. Several established MHs were used to solve five truss optimisation problems. The results obtained from using the various MHs were statistically compared based upon convergence rate and consistency. It was found that the best optimisers for this design task are evolution strategy with covariance matrix adaptation (CMAES) and differential evolution (DE). Furthermore, the best penalty function technique was discovered while four penalty function techniques assigned with several parameter settings were used in combination with the five best optimisers to solve the truss optimisation problems.

© 2014 Elsevier Ltd. All rights reserved.

Introduction

A truss is one of the most used structures in engineering applications due to its simplicity and low cost for construction. Under working conditions, such a structure is subject to multiple types of loads, which can be categorised as being static or dynamic. The structure under dynamic excitation may cause several undesirable vibration phenomena including structural resonance. As a result, truss designers have to find ways to prevent or suppress structural vibration. The problem of truss minimisation subject to natural frequency constraints has been researched [1–6], and it has been found that predefined frequency constraints can prevent vibration resonance from certain frequency bands of dynamic loads. Since its introduction, this special design problem has been problematic because its feasible region is non-convex while the boundaries are highly non-linear [1–3]. This has led to difficulty in the use of gradient-based optimisers. Alternatively, meta-heuristics (MHs) are known to be a better choice. Many researchers in the field of structural optimisation have been interested in the investigation of solving this kind of optimisation problem using MHs. Normally, the problem has an objective in minimising structural mass with multiple frequency constraints while using design variables such as topology [4], sizing [1,7], and a combination of shape and size [2,3,5,6].

Meta-heuristic algorithms are widely used for various kinds of optimisation design problems, particularly for engineering

applications, due to their derivative-free and global optimisation capabilities. These optimisers are robust and can be used to solve almost any kind of optimisation due to the nature of soft computing. However, a lack of search consistency is unavoidable during the MH search because of procedural randomisation. Also, there is likely to be a slow convergence rate if parameter settings, which are problem-dependent, are not properly assigned. Over the last few decades, numerous MHs have been developed, improved upon, and successfully implemented on a wide variety of optimisation problems [8–14]. New MHs can be proposed as hybridisations of existing algorithms [9–10,13,14] or have been introduced as new search concepts [15–18]. Even with thousands of algorithms in the literature, it can be said that there is no single MH that can outperform other MHs for all design problems. Consequently, it is always useful to compare the performance of a number of well-established and newly developed MHs for solving a newly introduced design problem or a problem that remains difficult to solve. For truss mass minimisation with frequency constraints, there are a number of MHs that have been successfully used including a charged system search algorithm [4], a hybrid version of the charged system search algorithm [3], a genetic algorithm [6,7], a particle swarm algorithm [1,19], a harmony search [2,5] and a firefly algorithm [5]. Nevertheless, since those researchers conducted their work independently, they presented a new meta-heuristic and compared it only with the previously used optimisers in the literature. Usually, the new method was run many times and the best results were taken to show and compare. This is not an appropriate way to compare search performances of MHs since it

* Corresponding author. Tel.: +66 43 202845.

E-mail address: sujbur@kku.ac.th (S. Bureerat).

is unlikely that those optimisers can get the same results with several runs due to randomisation in the process. There are a large number of MHs that have been overlooked and not implemented, while there have been several penalty function techniques being used such as the traditional exterior penalty function technique [2], the fuzzy set theory technique [20], the Kaveh–Zolghadr technique [3], and the Murata–Kim–Sugei technique [21]. The comparison of their effectiveness would be useful for future engineers and researchers.

In this comparative study, various MHs (mostly well-established), have been employed to solve a number of traditional test problems for truss optimisation with frequency constraints. Twenty-four MHs were used to solve five test problems. The top five best performers were then used in combination with four penalty function techniques and various sets of their internal parameters to solve the test problems in order to examine the effectiveness of those penalty function techniques. Each optimiser was used to solve each test problem a number of times, while the results obtained from using the various methods were statistically compared. The best meta-heuristics and the most effective penalty function methods will be discussed. What follows in this paper is organised such that: Section ‘Test problems of truss shape and sizing optimisation’ gives details of the test problems for truss mass minimisation with frequency constraints. Section ‘Meta-heuristics’ shows the comparative performance of the 24 meta-heuristic optimisers on solving the five test problems. The top five best optimisers are then used to examine the effectiveness of the penalty function techniques while the comparative results of this study are given in section ‘Comparative study of various meta-heuristic algorithms’. Finally, concluding remarks are provided in section ‘Comparative study of various penalty function techniques’.

Test problems of truss shape and sizing optimisation

A typical truss optimisation problem with dynamic constraints can be posed as

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \omega_i \leq \omega_{i,all} \quad \text{for } i \in I \\ & \omega_j \geq \omega_{j,all} \quad \text{for } j \in J \\ & \mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U \end{aligned} \quad (1)$$

where $\mathbf{x}$ is a vector containing design variables having lower and upper bounds as $\mathbf{x}_L$ and $\mathbf{x}_U$ respectively, f is an objective function, ω_i is the i -th mode natural frequency of a truss, $\omega_{i,all}$ is an allowable frequency for the i -th mode natural frequency, I and J are the sets of mode numbers to be specified by a designer.

The objective function is structural mass whereas design variables include, in this work, sizing and shape variables. The sizing design variables include truss element cross-sectional areas while the shape design variables are nodal positions of the structure. It is commonplace that most if not all of MHs can handle optimisation problems with bound constraints. For the design problem (1), a penalty function, which will be detailed later, is adopted to handle the dynamic constraints.

Five test problems of shape and sizing optimisation of trusses with frequency constraints commonly found in the literature will be employed in this study. The design problems were assigned to minimise structural mass subject to multiple frequency constraints, and are detailed as follows:

Case I: Sizing design of a 2D 10 bar truss

The truss structure is illustrated in Fig. 1 [1,3,5,6]. The structure is subject to non-structural mass of 454 kg at each free node. The design variables include all simple bar element cross section areas. Material density and modulus of elasticity are 2770.0 kg/m³ and

6.98×10^{10} N/m² respectively. The sizing optimisation is presented to minimise structural mass subject to natural frequency constraints which can be expressed as follows:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = \text{structural mass} \\ \text{s.t.} \quad & \omega_1 \geq 7 \text{ Hz} \\ & \omega_2 \geq 15 \text{ Hz} \\ & \omega_3 \geq 20 \text{ Hz} \\ & 0.645 \leq A_i \leq 50 \text{ cm}^2 \end{aligned}$$

where $\mathbf{x}^T = \{x_1, \dots, x_{10}\} = \{A_1, \dots, A_{10}\}$ is a design vector, $f(\mathbf{x})$ is structural mass. The variables ω_1 , ω_2 and ω_3 are the natural frequencies for the first, second and third modes respectively. A_i are cross-sectional areas of all bar elements.

Case II: Shape and sizing design of a 2D 37 bar truss

The truss structure is illustrated in Fig. 2 [1,2,5,6,19]. The structure is subject to non-structural mass of 10 kg at each free node of the lower chord. The elements of the lower chord were set as bar elements with unchanged cross-sectional area of 40 cm² while the other were set as bar elements with initial cross-sectional area 1 cm². The design variables include all bar element cross-sectional areas (except the lower chord bar elements) and y -direction of nodal positions of the upper chord. The design variables are treated so to have a symmetrical structure with respect to the y axis. Material density and modulus of elasticity are 7800.0 kg/m³ and 2.1×10^{11} N/m², respectively. The shape and sizing optimisation problem is presented to minimise structural mass subject to frequency constraints which can be expressed as follows:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = \text{structural mass} \\ \text{s.t.} \quad & \omega_1 \geq 20 \text{ Hz} \\ & \omega_2 \geq 60 \text{ Hz} \\ & \omega_3 \geq 60 \text{ Hz} \\ & 1 \leq A_i \leq 10 \text{ cm}^2 \\ & 0.1 \leq Y_i \leq 3 \text{ m} \end{aligned}$$

where $\mathbf{x}^T = \{x_1, \dots, x_{19}\}$ is a design vector. Truss cross-sectional areas are assigned as the first 14 elements of the vector while the nodal positions are assigned as the 15th–19th elements. The variables ω_1 , ω_2 and ω_3 are the natural frequencies for the first, second and third modes respectively. A_i are element cross-sectional areas. Y_i are the y -direction of positions of the upper chord nodes (changes in Y_i are symmetric about y axis).

Case III: Sizing design of a 3D 72 bar space truss

The truss structure is illustrated in Fig. 3 [1,3,5,7]. Four non-structural masses of 2270 kgs are attached to the top nodes. The design variables include all bar element cross-sectional areas which were divided into 16 groups according to Table 1. Material density and modulus of elasticity are 2770.0 kg/m³ and 6.98×10^{10} N/m², respectively. The sizing design problem is assigned to minimise structural mass subject to frequency constraints, which can be expressed as follows:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = \text{structural mass} \\ \text{s.t.} \quad & \omega_1 = 4 \text{ Hz} \\ & \omega_2 \geq 6 \text{ Hz} \\ & 0.645 \leq A_i \leq 30 \text{ cm}^2 \end{aligned}$$

where $\mathbf{x}^T = \{x_1, \dots, x_{16}\}$ is a design vector. ω_1 and ω_2 are the first and second mode natural frequencies respectively. A_i are the cross-sectional areas.

Case IV: Shape and sizing design of a 3D 52 bar dome truss

The truss structure is illustrated in Fig. 4 [1–3,5,6,19]. The structure has non-structural masses of 50 kg at each free node. The

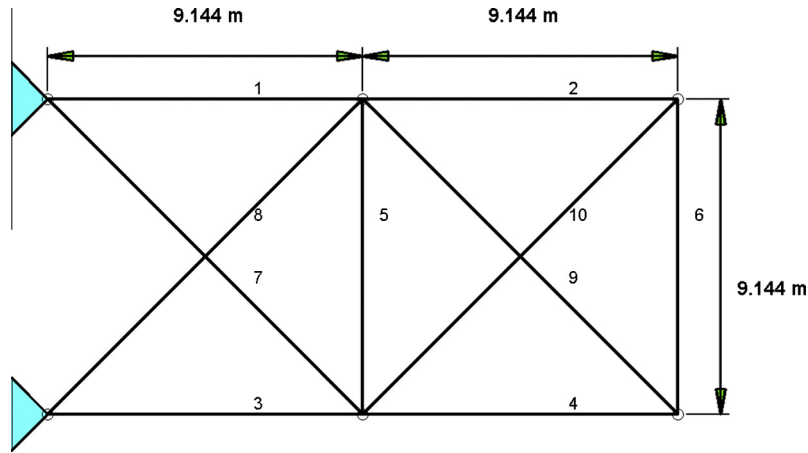


Fig. 1. 2D ten bar truss.

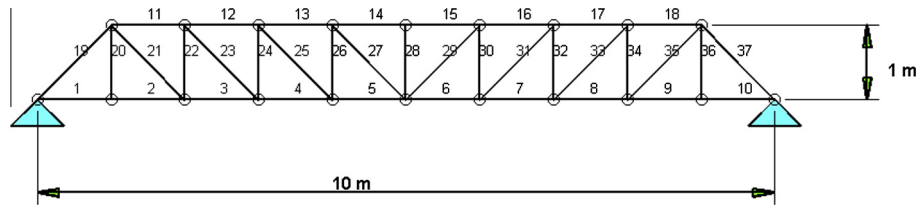


Fig. 2. 2D 37 bar truss.

Table 1
Element groups of the 72 bar space truss.

Element groups	Element number in groups
1	1–4
2	5–12
3	13–16
4	17–18
5	19–22
6	23–30
7	31–34
8	35–36
9	37–40
10	41–48
11	49–52
12	53–54
13	55–58
14	59–66
15	67–70
16	71–72

design variables include all bar element cross section areas which were divided into 8 groups according to Table 2 and all free nodes which can move ± 2 m in all directions in a symmetrical manner. Material density and modulus of elasticity are 7800.0 kg/m^3 and $2.1 \times 10^{11} \text{ N/m}^2$ respectively. The shape and sizing optimisation problem is presented to minimise structural mass subject to frequency constraints which can be expressed as follows:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = \text{structural mass} \\ \text{s.t.} \quad & \omega_1 \leq 15.916 \text{ Hz} \\ & \omega_2 \geq 28.648 \text{ Hz} \\ & 1 \leq A_i \leq 10 \text{ cm}^2 \end{aligned}$$

where $\mathbf{x}^T = \{x_1, \dots, x_{13}\}$ is a design vector (eight sizing variables and five shape variables). ω_1 and ω_2 are the first and second mode natural frequencies respectively. A_i are cross-sectional areas.

Case V: Sizing design of a 3D 120 bar dome truss

The truss structure is illustrated in Fig. 5 [3,19]. The structure has non-structural masses at free nodes as: 3000 kg at node 1, 500 kg at nodes 2–13, and 100 kg at the rest. The design variables include all bar element cross section areas which were divided into 7 groups as show in Fig. 5a. Material density and modulus of elasticity are 7971.810 kg/m^3 and $2.1 \times 10^{11} \text{ N/m}^2$ respectively. The sizing optimisation problem can be expressed as follows:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = \text{structural mass} \\ \text{s.t.} \quad & \omega_1 \geq 9 \text{ Hz} \\ & \omega_2 \geq 11 \text{ Hz} \\ & 1 \leq A_i \leq 129.3 \text{ cm}^2 \end{aligned}$$

where $\mathbf{x}^T = \{x_1, \dots, x_7\}$ is a design vector. ω_1 and ω_2 are the first and second mode natural frequencies respectively. A_i are cross-sectional areas.

Meta-heuristics

A meta-heuristic, often regarded as a probabilistic optimisation method, is an optimiser developed as an alternative to a classical gradient-based optimiser. Most of the meta-heuristic algorithms search for optima by using a group of design solutions traditionally called a population. The optimisation procedure starts with an initial population usually created at random. The population of design solutions are then updated or improved by some specific computing operations which are usually related to randomisation. The operators are created and used to achieve intensification and diversification in searching. The population is improved iteratively until a termination criterion is met. As this type of optimiser does not employ function derivatives, the procedure is stopped when no further improvement of a population for several iterations is found or the maximum number of function evaluations is reached. The method can be used to solve almost any kind of optimisation

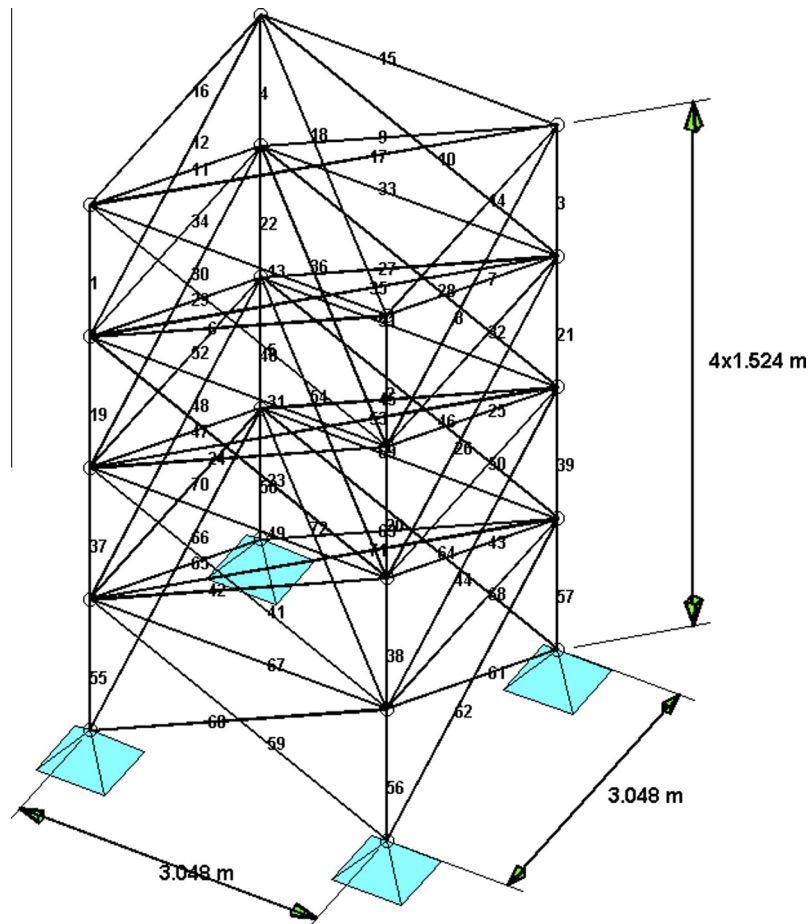


Fig. 3. 3D 72 bar space truss.

Table 2
Element groups of the 3D 52 bar dome truss.

Element group	Element number in group
1	1–4
2	5–8
3	9–16
4	17–20
5	21–28
6	29–36
7	37–44
8	45–52

problem. Over the last few decades, there have been a large number of MHs developed for solving various types of optimisation problems. The methods can be classified into many groups such as bio-inspired optimisers, swarm intelligence optimisers, and evolutionary algorithms. MHs are becoming more popular compared to their derivative-based counterparts since they are flexible to modify, improve and use. Furthermore, this research field is still wide open for further investigation and development. For this comparative performance study, 24 MHs are employed and they are briefly detailed as:

– DE: Differential evolution [22]. DE is a simple but efficient evolutionary algorithm proposed by Storn and Price (1997). The search processes start with an initial population. Then, a set of candidate solutions is generated by two main operators, mutation and crossover. The best between a particular candidate and its parent will be selected for the next generation. DE, since it was proposed, have a number of optimisation strategies which are usually termed DE/x/

y/z where x is a solution vector to be mutated, y is a number of solution pairs used for computing vector differentiation in mutation, and z is a crossover scheme.

– ABC: Artificial bee colony algorithm [15]. The search procedure of ABC imitates bee behaviour in finding food sources. In the ABC algorithm, after randomly creating a population (viewed as food source positions for employed bees), the amounts of nectar (objective function values) are computed. Then, candidate food sources are created in some manners for employed and onlooker bees. It is also useful to use some scout bees to explore new food sources if appropriate. The food source positions are updated iteratively until the procedure is stopped.

– ACOR: Real-code ant colony optimisation [23]. The very first version of ant colony optimisation was proposed by Dorigo [24] for solving combinatorial optimisation. Since then, it has become one of the most popular optimisers for such optimisation problem with numerous variants having been proposed. The real-code ant colony optimisation [22] was later developed by the same author and it is said to be among the top real-code MHs [25]. Starting with a randomly generated initial population, offspring are then created by means of an ant-based solution construction process involving roulette wheel selection and a Gaussian function [23]. The best solutions from combining the offspring and the parents are selected as the next generation. The procedure is repeated until meeting a termination criterion.

– ChSS: Charged system search [16]. The search process starts with an initial population regarded as a group of charged particles while charged memories are generated. Then, the position of each charged particle will be updated based on the Coulomb law for electrostatics and the Newton laws of motion. The new charged

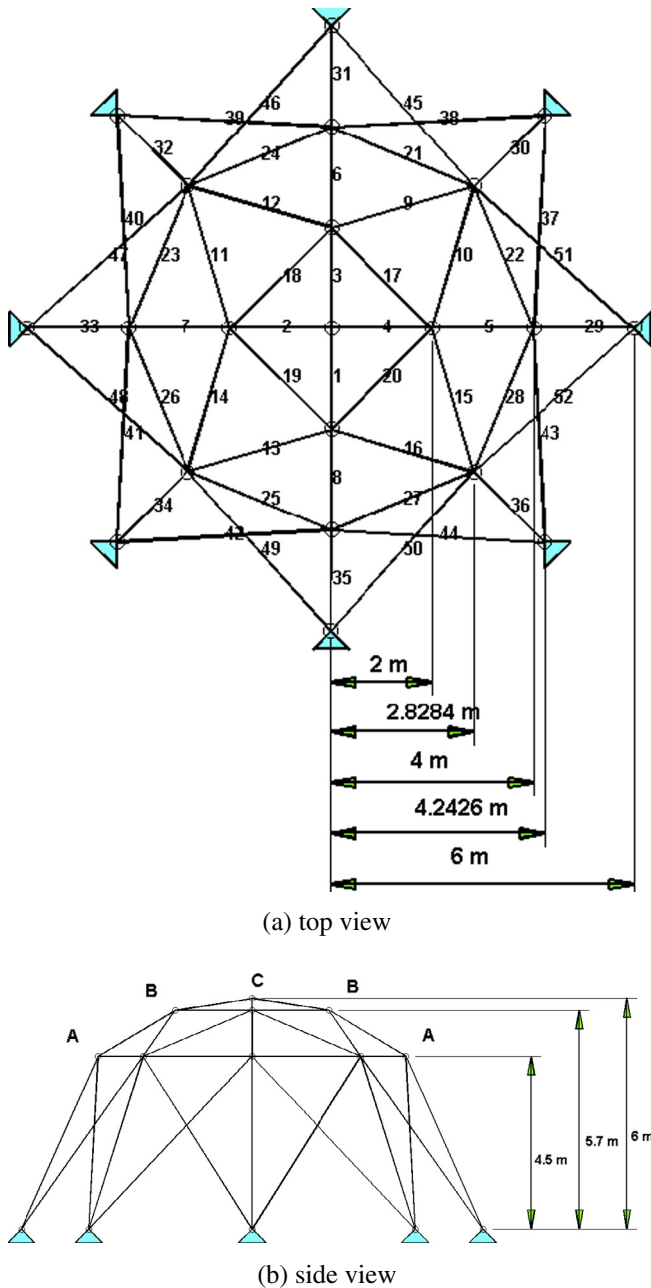


Fig. 4. 3D 52 bar dome truss (initial shape).

particles will be compared with the old ones, and the better ones will be stored in the charged memories. The procedure is repeated until the termination criterion is reached.

– LCA: League championship algorithm [17]. LCA imitates a league environment in some kind of sport, such as football. Initially a league schedule is generated for the whole season (one optimisation run). A member in a population is viewed as an individual team formation while its objective function is team strength. For each week (iteration), those teams participate in the competition. For the next week, a team may change its team formation based on match analysis and comparing playing strength to other teams. The formations of all teams are updated weekly until the season ends.

– SA: Simulated annealing [8]. SA was first proposed by Kirkpatrick et al. [26]. The method is one of the state-of-the-art MHs with numerous variants being developed and used in various kinds of applications. For this work, the search procedure of SA starts with

Table 3

Comparative performance for Case I ($\omega_{i,av}$ = the average natural frequency of the i -th mode).

MHs	Mean	STD	$\omega_{1,av} (\geq 7)$	$\omega_{2,av} (\geq 15)$	$\omega_{3,av} (\geq 20)$
DE	528.7879	3.9720	7.0000	16.8193	20.0011
ABC	548.6847	7.5164	7.0008	16.8277	20.0191
ACOR	529.4166	3.1231	7.0001	16.6574	20.0015
ChSS	574.8534	16.7275	7.0281	17.8409	20.5732
LCA	533.4568	2.8652	7.0033	17.1052	20.0596
SA	551.7187	12.1626	7.0005	16.0340	20.0040
TLBO	649.2924	18.6085	7.0164	16.9324	20.6934
BBBC	531.5318	3.3843	7.0019	16.6575	20.0196
BPBIL	561.5219	13.3590	7.0205	17.0942	20.2233
CKS	567.8387	9.7329	7.0158	16.9264	20.2256
CMAES	527.9831	3.0571	7.0000	16.9228	20.0003
CPBIL	536.6292	3.6374	7.0059	17.3533	20.0723
CSS	454.9008	0.0000	4.3780	12.1619	15.9012
ECTS	540.8210	9.4883	7.0016	17.1797	20.0272
EP	454.9008	0.0000	4.3780	12.1619	15.9012
ES	454.9008	0.0000	4.3780	12.1619	15.9012
FWA	454.9008	0.0000	4.3780	12.1619	15.9012
GA	454.9008	0.0000	4.3780	12.1619	15.9012
GSA	550.1043	19.9290	7.0001	17.3459	20.0103
HS	454.9008	0.0000	4.3780	12.1619	15.9012
PSO	548.4953	14.0746	7.0003	16.2958	20.0246
SGA	581.1268	27.4650	7.0081	16.9898	20.3414
BAT	554.5097	20.8927	7.0000	16.8689	20.0026
FFA	540.4796	6.4824	7.0039	17.2493	20.0550

an initial solution, which will be called a parent. The parent is then mutated in some manner leading to a set of children or offspring. The best offspring is said to be a candidate to challenge its parent. For minimisation, if the candidate has a lower objective value than that of its parent, the parent is replaced by the candidate. In cases where the candidate has a higher objective function value than its parent, it still has a chance to replace the parent if accepted by a Boltzmann probability. In this version of SA, the best solution will be kept and used for creating offspring along with the parent as detailed in [8].

– TLBO: Teaching–learning-based optimisation [18]. Starting with a randomly created population with objective function values being evaluated, the best individual (teacher) will be identified. Then, the population will be updated by two stages including teacher and learner phases. For the former phase, a solution is moved towards the best solution while, for the later phase, it is moved towards a better neighbour. The population is improved iteratively until meeting a stopping criterion.

– BBBC: Big bang – big crunch algorithm [27]. The method is based on the Big Bang and Big Crunch theory. The search process starts with randomly generating an initial population which is called a big bang phase. Then, the operator called big crunch is applied to find the mass centre. Thereafter, a big bang operator is used to create neighbours of the mass centre. The process is repeated until the stopping condition is met.

– BPBIL: Population-based incremental learning [28]. BPBIL is an estimation of distribution algorithm based upon a binary searching space. Rather than using a binary population directly as with a genetic algorithm, the search process starts with an initial probability vector which has the length equal to that of a binary design solution. The binary population according to the probability vector is then created where the best solution is found. The probability vector is updated based on the best binary solution while it is also useful to apply mutation with a given (usually low) probability. The vector is updated iteratively until the procedure is terminated.

– CKS: Cuckoo search [29]. CKS is created mimicking egg-laying behaviour of cuckoos. The procedure starts with an initial set of solutions (host nets). A new solution is created by means of a Levy flight. If it is better than a randomly chosen host nest, replace the

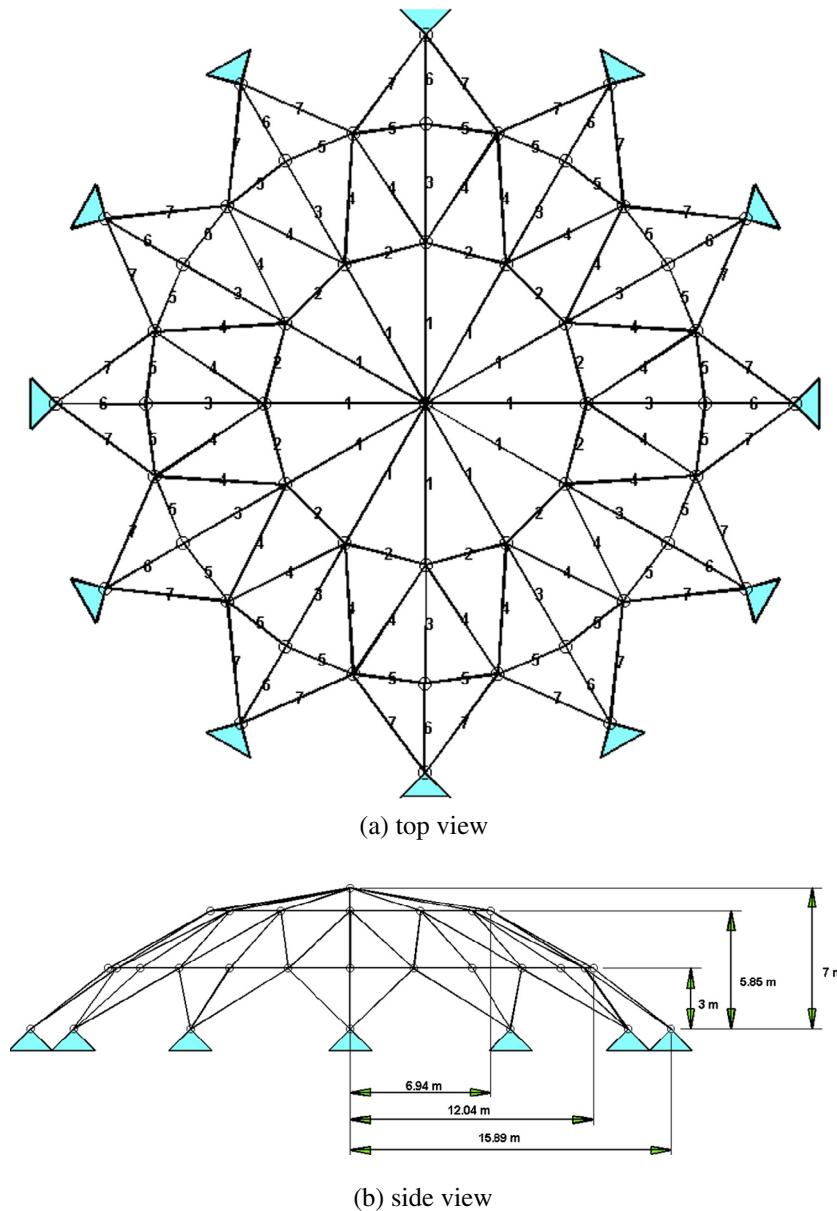


Fig. 5. 3D 120 bar dome truss.

old nest with the new one. Some of the worst nests can be discarded with a predefined probability and a new nest is created to replace them. The best nest is recorded and the procedure is repeated until a stopping condition is met.

- CMAES: Evolution strategy with covariance matrix adaptation [30]. CMAES is an extension of ES for real-code optimisation. The optimiser is said to remove randomisation of strategy parameter settings which is usually a problem in most MHs. The method starts with an initial population, a global step size, and a covariance matrix. The solutions are then updated based on the step size and normally distributed random vectors with such a covariance matrix. Having evaluated objective functions, the step size and covariance matrix are updated using an evolution path. This process is repeated until a stopping condition is met.

- CPBIL: Continuous population-based incremental learning [31]. The CPBIL algorithm is an extension of BPBIL. A probability vector is used for creating a binary population in BPBIL but, for the continuous version, a probability matrix is used. Having generated a real-code population, the probability matrix is updated according to the best real code solution. Mutation on the probability matrix is

activated with the given mutation probability. The algorithm updates the probability matrix iteratively until reaching a termination condition.

- CSS: Continuous scatter search [32]. The continuous version of scatter search, similarly to its combinatorial version, uses intensification and diversification to improve design solutions towards an optimum. The procedure starts with an initial population obtained by using a diversification generation method. Then an improvement method or local search is employed to improve the solutions. The reference set containing some best and diverse solutions is generated where new candidates are created by applying a solution recombination method on a member in the reference set. Having applied the local search the reference set is updated. The process continues until the stopping condition is fulfilled.

- ETCS: Enhanced continuous tabu search [33]. The original version of tabu search was developed by Glover [34] for combinatorial optimisation. In the ETCS algorithm, as an initial population is created, the so-called promising list containing some best solutions and the tabu list containing the rest are generated. A new solution is generated by mutation and its function evaluation is carried out

if it is not too close to the members in those lists. The tabu and promising lists are updated and the process repeats until being terminated. It should be noted that it is also useful and possible to apply local search to improve the solutions at the end of the search procedure.

– ES: Evolution strategies [35]. ES was developed in Germany during the same time period as evolutionary programming (EP) and the genetic algorithm (GA). These three optimisers were developed independently but their concepts are surprisingly similar. The first version of ES is called 1 + 1 ES where one offspring is created from mutating a parent. The later version is $\mu + \lambda$ ES standing for λ offspring are iteratively created from μ parents. Starting with a randomly generated population or a set of design vectors, a real-code recombination is applied to create a set of offspring. The offspring is then mutated based on normal distribution randomisation. The best solutions among the union set of offspring and parents are selected to the next iteration. The process is repeated until fulfilling the stopping condition. To prevent a premature convergence, a tournament selection operator is used instead of choosing best solutions in this paper.

– EP: Evolutionary programming [35]. EP is somewhat similar to ES since the reproduction process employs normally distributed randomisation for mutation. However, EP does not use a recombination operator while the selection operator is also different.

– FWA: Fireworks algorithm (FA) [36]: FWA is a simple swarm intelligence optimiser imitating the nature of firework explosion. An initial population is seen as a group of fireworks where design solutions are firework locations. The location of sparks is then simulated, which is a means to reproduce candidate solutions. Having evaluated function values, the best fireworks locations are then selected and kept to the next generation.

– GA: Genetic algorithm [37]. A genetic algorithm is probably the best known meta-heuristic which, since it was first introduced, has thousands of variants. The method, like its name, can be best thought of as imitating Darwinian natural selection in that offspring are created by mating pairs of their parents. The parents are selected by means of roulette wheel selection to a mating pool and a crossover operator is used to create offspring with a given crossover probability. Then, mutation is allowed to take place with a predefined mutation rate. The process repeats until meeting a termination criterion.

– GSA: Gravitational search algorithm [38]. GSA is another swarm intelligence optimiser which searches for optima based on the law of gravity and mass interaction. Starting with an initial population, the gravity and mass will be calculated depending on objective function values. Total forces, accelerations and velocities of the design solutions are then calculated and used to update the solution members in a population. The process is repeated until a stopping criterion is reached.

– HS: Harmony search [39]. HS is an optimisation algorithm which was developed based on improvisation of music players. The search process starts with initialised harmony memories (population). Then, the harmony memories will be updated based on aesthetic estimation which is defined by harmony memory considering rate and pitch adjusting rate. The harmony memories are updated iteratively to meet an optimum.

– PSO: Particle swarm optimisation [40]. PSO is probably the first swarm intelligent algorithm which was developed imitating the movement of a flock of birds or fishes aiming to find food. Starting with a group of birds or position vectors of particles, initial particle velocities are then defined. The position of each particle is then modified based on its velocity which is influenced by its velocity from the previous generation, the global best solution, and its best found position. The particles move iteratively until reaching an optimum.

– SGA: Stud genetic algorithm [41]. SGA is an evolution of GA. Rather than creating offspring from crossing over a pair of selected parents, the best binary design solution is used as a stud. To create an offspring, the crossover operator will be applied to the stud and an individual in a population. It is also useful to apply a mutation operator or another diversity-based operator with a predefined probability in order to prevent a premature convergence. The population is updated by mating solutions with the stud iteratively until fulfilling a termination criterion.

– BAT: Bat-inspired algorithm [42]. A swarm intelligence algorithm BAT is a meta-heuristic inspired by the echolocation of microbats. The extended version of BAT is presented in [43] with slight modification. Bats (design solutions) update their positions using velocities based on varying frequencies and loudness. A random walk is applied for diversification. The positions of the bats are updated repeatedly until the algorithm is stopped.

– FFA: Fireflies algorithm [44]. A firefly algorithm is another swarm intelligence optimiser. With an initial population of fireflies (design solutions), their objective functions are evaluated and consequently their light intensity values are computed. Each firefly will move to other fireflies based on the light intensity while the degree of attractiveness varies with a distance between a pair of fireflies. The position of the fireflies and their light intensities are updated iteratively until meeting a stopping condition. In this work, the latest version of FFA [45] will be employed.

Comparative study of various meta-heuristic algorithms

Meta-heuristic parameter settings

In this work, twenty-four established MHs were compared based on solving the five truss optimisation problems. Given that n_p is a population size (unless otherwise specified) and n is a number of design variables, the optimisation parameter settings of those MHs (details of notations can be found in the corresponding reference of each method) are detailed as:

DE: DE/best/2/bin strategy is used. A scaling factor, crossover rate and probability of choosing elements of mutant vectors are 0.5, 0.7, and 0.8 respectively.

ABC: The number of food sources is set to be n_p . A trial counter to discard a food source is 150.

ACOR: The parameter settings are $q = 0.5$, and $\xi = 0.5$. Roulette wheel selection is used for ant-based solution construction.

ChSS: The number of solutions in the charged memory is $0.2 \times n_p$. The charged moving considering rate (CMCR) and the parameter PAR are set to be 0.75 and 0.5 respectively. Charged sphere radius is $a = 0.10 \max(\|\mathbf{x}_U - \mathbf{x}_L\|)$. The parameter ε for calculating the separation distance between two charged particles is 0.000001.

LCA: The probability of success (P_c) is 0.5 and the decreasing rate to decrease P_c is $\alpha = 0.5$. The lower and upper limits for a retreat coefficient are 0 and 2 respectively. The lower and upper limits for an approach coefficient are 0 and 2 respectively. A new formation is based on best formations.

SA: Starting and ending temperatures are 10 and 0.001 respectively where the temperature is increased exponentially for every 10 loops. For each loop, n candidates are created by mutating on the current best solution while other n candidates are created from mutating the current parent. The best of those $2n$ solutions are set as an offspring to be compared with the parent.

TLBO: The teaching factor is $T_F = \text{rand}(1 \text{ or } 2)$.

BBBC: The current best solution is used as a centre point to create a new population. The search space will be restricted to half after every period of 5 iterations.

BPBIL: The learning rate, mutation shift, and mutation rate are set as 0.5, 0.7, and 0.2 respectively.

CKS: Discovery rate of alien eggs (p_a) is 0.75. The Levy coefficient is $\beta = 2/3$.

CMAES: Parameter settings are not required.

CPBIL: The learning rate, mutation shift, and mutation rate are set as 0.5, 0.2, and 0.05 respectively. RPBIL3 in [31] is employed with $T_{start} = 5$ and $T_{end} = 40$.

CSS: The BLX- α recombination method is used for reproduction. The simplex downhill method is used for local search. The number of high-quality solutions in the reference set is $0.5 \times n_p$, and the number of diverse solutions in the reference set is $0.25 \times n_p$. ETCS: The tabu list size is set to be $2 \times n_p$ while the promising list size is $0.5 \times n_p$. A linear partitioning on hyper-rectangles is used for generating neighbour solutions.

EP: For every period of 3 iterations, the number of randomly selected individuals used for computing a selection function value is increased by 1 where it is set to be 2 for the first loop. ES: Use a three-candidate tournament selection operator and a simple mutation without the effect of a covariance matrix. For each individual solution, all five ES recombination strategies are applied with equal probabilities.

FWA: The number of fireworks for each generation is $n_p/4$, the limit of sparks created with algorithm 1 is n_p , the amount of sparks created with algorithm 2 is $n_p/4$, the floor parameter for rounding the amount of sparks created with algorithm 1 is 0.004, the ceiling parameter for rounding the amount of sparks created with algorithm 1 is 0.8, and the maximum explosion amplitude is $0.5(x_{u,i} - x_{l,i})$.

GA: The crossover and mutation rates are 0.7 and 0.3 respectively. Multi-point crossover is used.

GSA: The norm in Eq. (8) in [38] is 1. The power of R_{ij} in Eq. (7) in [38] is 2. In Eq. (28) in [38], $\alpha = 20$ and $G_0 = 100$. In the last iteration, only 10 percent of agents apply force to the others.

HS: The parameter settings are: harmony memory size = n_p , harmony memory considering rate ($HMCR$) = 0.8, pitch adjusting rate (PAR) [0.4,0.9] increased linearly, and bandwidth (BW) [0.0001,0.9] decreased exponentially.

PSO: The starting inertia weight, ending inertia weight, cognitive learning factor, and social learning factor are assigned as 0.75, 0.0001, 1.5 and 2.5 respectively. The inertia weight is decreased linearly.

SGA: The so-far best solution is used as a stud whereas the probability of shuffling binary bits of a binary design solution is 0.2.

BAT: The loudness and the pulse rate are set to be 0.9 and 0.1, respectively. The minimum and maximum frequency values are 0 and 2, respectively. The maximum and minimum values of θ are 0.1 and 20, respectively.

FFA: The parameters in Equation (15) in reference [45] are set as $\alpha = 0.75$, $\Omega = 0.000001$, and $\beta_0 = 0.25$.

It should be noted that the parameters used for each optimiser are tuned to have proper values that increase the performance of the algorithm. The Case I problem is used for such parameter tuning. Each optimiser is used to solve each truss problem for 30 optimisation runs. The population size is set to be $n_p = 20$ whereas the number of iterations is 200 for all test problems. For the optimisers using different population sizes such as simulated annealing, and firework algorithm, their search processes are terminated with the total number of functions evaluations equal to 20×200 . The penalty function used in this study is the Kaveh–Zolghadr technique which has some details as:

$$f_p(\mathbf{x}) = (1 + \varepsilon_1 \cdot v)^{\varepsilon_2}$$

where f_p is a penalised function value, ε_1 and ε_2 are selected considering the exploration and exploitation rates of the search space.

$$v = \sum_{i=1}^{ng} v_i$$

where ng and v_i are the number of frequency constraints and the constraints violation respectively. At a design solution $\mathbf{x}$, if the i th constraint is not violated, v_i will be taken as zero; otherwise, v_i will be defined as:

$$v_i = \left| 1 - \frac{\omega_i}{\omega_{i,all}} \right|$$

where ω_i and $\omega_{i,all}$ are the i th mode natural frequency and the allowable i th mode natural frequency respectively.

Results and discussion

After performing 30 optimisation runs of the various optimisers on solving the truss design problems (Case I–Case V), the comparative results based on objective and constraint function values are shown in Tables 3–7. In the tables, the mean values (Mean) will be used to measure the convergence speed while the standard deviation (STD) represents the search consistency of the method. The method with a lower mean value of the objective function has better convergence speed while the method with a lower STD has better search consistency. Note that only the methods that can search for the feasible solutions will be considered for comparison.

For the measure of the search convergence based on the mean value, it is found that CMAES is the best performer for all test problems except for the Case II, which the best optimiser is DE. The second best for Case I, Case III, Case IV and Case V is DE while the CMAES was the second best for Case II. For the measure of search consistency based on the STD, the method that gives the most consistent results for Case I, Case II, Case IV and Case V is CMAES while ACOR is the most consistent for Case III. Overall, CMAES is the best performer for the assigned mass minimisation problems of trusses with frequency constraints in terms of both convergence rate and search consistency while DE is the second best optimiser.

Tables 8–12 show the comparison between the best results found so far in the literature and the best results found in this study for all design cases. It was found that, the best results found

Table 4
Comparative performance for Case II.

MHs	Mean	STD	$\omega_{1,av}$ (≥ 20)	$\omega_{2,av}$ (≥ 40)	$\omega_{3,av}$ (≥ 60)
DE	360.8391	3.0896	20.0124	40.0443	60.0664
ABC	370.1812	2.8677	20.0854	40.5136	61.3069
ACOR	362.4482	1.6613	20.0085	40.0096	60.0254
ChSS	405.1165	8.5197	20.9410	45.8991	72.7114
LCA	369.4645	2.1620	20.4770	40.6377	62.0107
SA	375.9803	5.7973	20.1168	40.0236	60.0537
TLBO	431.2922	9.3983	20.5522	42.6326	68.5651
BBBC	369.1503	6.7555	20.0988	40.2250	60.9658
BPBIL	386.4841	7.5400	20.7166	42.0721	64.4003
CKS	402.6145	7.5730	21.3250	43.3658	66.7712
CMAES	361.6511	2.5302	20.0379	40.1591	60.1419
CPBIL	369.8753	4.0229	20.3062	40.5809	61.4021
CSS	388.6446	8.3675	20.7818	41.3861	61.4275
ECTS	384.0519	9.5861	20.0638	40.1962	63.6046
EP	443.9599	0.0000	0.6359	3.6402	48.4653
ES	441.4350	13.8296	1.2814	4.8522	48.8498
FWA	443.9599	0.0000	0.6359	3.6402	48.4653
GA	384.4549	12.4004	20.3459	40.6973	61.0602
GSA	422.0086	15.7411	20.5125	44.8561	77.9204
HS	441.3328	7.8712	3.4249	8.8369	50.5985
PSO	406.2254	11.4493	20.0329	40.4639	62.9200
SGA	384.1679	11.0051	20.2395	40.7644	60.8835
BAT	402.3112	10.8363	20.1241	40.0063	63.9841
FFA	399.4114	12.0955	20.3494	41.8522	68.6302

Table 5
Comparative performance for Case III.

MHs	Mean	STD	ω_{1_av} (=4)	ω_{2_av} (≥ 6)
DE	331.6909	17.0220	4.0000	6.0003
ABC	351.3243	9.9280	4.0053	6.0614
ACOR	338.7503	7.4967	4.0000	6.0009
ChSS	469.3784	23.0313	4.0247	6.1383
LCA	393.1856	13.7117	4.0105	6.0709
SA	370.2935	11.9415	4.0032	6.1265
TLBO	38.7291	0.0826	0.9976	1.7162
BBBC	360.1404	29.8948	4.0025	6.0093
BPBIL	418.7888	34.8305	4.0197	6.1011
CKS	470.2877	34.3126	4.0213	6.3116
CMAES	331.0710	13.5805	4.0001	6.0005
CPBIL	360.4328	13.1719	4.0051	6.0294
CSS	441.4551	58.0000	3.9655	5.9895
ECTS	552.0251	64.1452	4.0019	6.0135
EP	628.0440	0.0000	3.1041	4.2057
ES	628.0440	0.0000	3.1041	4.2057
FWA	621.9087	33.6045	3.1328	4.2697
GA	406.8943	43.4109	4.0089	6.1551
GSA	590.7577	50.6341	4.0008	6.0778
HS	300.2329	234.0953	2.6447	4.3367
PSO	528.0723	83.9778	4.0000	6.0135
SGA	399.7163	41.2570	4.0077	6.0947
BAT	637.4574	71.8137	4.0000	6.0317
FFA	414.1589	39.5042	3.9932	4.0068

Table 6
Comparative performance for Case IV.

MHs	Mean	STD	ω_{1_av} (≤ 15.916)	ω_{2_av} (≥ 28.648)
DE	201.4995	14.8806	10.7823	28.6482
ABC	225.1624	14.1478	10.8390	28.6766
ACOR	221.3660	15.5975	10.4578	28.6505
ChSS	215.4694	47.3282	11.9774	26.8727
LCA	253.8265	16.1027	11.0309	28.8632
SA	256.6105	30.3086	10.0801	28.6172
TLBO	164.6637	20.2184	13.2375	20.8339
BBBC	207.5187	14.7296	10.5435	28.6516
BPBIL	284.7816	44.1585	10.7912	28.7775
CKS	352.2353	65.5756	9.5642	28.7686
CMAES	195.8252	0.6102	10.2480	28.6481
CPBIL	215.0622	8.8448	10.9074	28.6500
CSS	308.0847	51.2398	8.9476	28.9553
ECTS	253.1338	86.6098	12.4834	28.6487
EP	361.2610	236.6418	14.4327	28.7918
ES	464.8889	227.0136	15.3636	28.8545
FWA	238.8591	22.0796	8.4459	28.6735
GA	283.0605	53.8587	7.5508	28.8080
GSA	467.8169	85.2877	10.7432	29.0212
HS	176.6073	42.3522	12.9434	24.0800
PSO	402.6086	116.7340	10.9740	28.5770
SGA	263.8774	46.2667	9.4601	28.6874
BAT	303.1053	82.7459	12.3850	28.4278
FFA	255.0253	26.7056	11.3616	28.6727

in this study are better than the best results found in the literature for all cases except for Case IV.

Furthermore, the ranking of the MH optimisers was made based upon the statistical Wilcoxon rank sum test. Table 13 shows the ranking of MHs for the Case I design problem. Since there are 24 MHs implemented, a comparison matrix sized 24×24 whose elements are full of zeros is generated. The null hypothesis for the Wilcoxon rank sum test is that, for a particular test problem, the median of 30 objective function values obtained from method *I* is not different from the median of 30 objective function values obtained from using method *J* at the 5% significance level. If the null hypothesis is rejected and the median from method *J* is lower, the element *IJ* of the matrix is set to be '1'. After summing up all values in the matrix columns, the best optimiser is the one that has the highest score. The summation and the ranking are also given in the table. Note that this comparison will be made if and

Table 7
Comparative performance for Case V.

MHs	Mean	STD	ω_{1_av} (≥ 9)	ω_{2_av} (≥ 11)
DE	8708.7905	3.4456	9.0000	11.0000
ABC	7681.9709	2863.9555	8.0601	9.8982
ACOR	8798.7094	89.4076	9.0000	11.0000
ChSS	9083.1365	143.6739	9.0699	10.9922
LCA	8728.1426	9.4177	9.0032	11.0004
SA	8879.1191	144.1959	9.0074	10.9895
TLBO	501.2728	0.0000	1.9115	2.7051
BBBC	8846.3515	105.1124	9.0308	11.0039
BPBIL	10041.9281	756.9410	9.1773	11.0742
CKS	9726.8627	1862.2010	8.9422	10.8227
CMAES	8707.3853	0.2591	9.0000	11.0000
CPBIL	8848.0015	156.9981	9.0702	11.0013
CSS	9330.0753	741.1946	9.0220	11.0143
ECTS	8783.3294	68.7185	9.0201	10.9938
EP	8841.9387	401.6341	9.0174	11.0001
ES	9737.6194	2061.6896	9.0306	11.0016
FWA	9074.1572	245.1233	9.0375	11.0119
GA	11081.6603	824.1856	9.2136	11.0952
GSA	9972.4922	1458.3429	9.0934	11.0001
HS	514.4762	1.4450	2.0771	2.8668
PSO	7392.0791	4043.7420	7.3558	9.0628
SGA	10509.5583	862.7513	9.2353	11.1171
BAT	514.7687	5.8228	2.0752	2.8644
FFA	4474.0803	4623.5168	5.0815	6.3720

only if both methods *I* and *J* give feasible solutions for all runs. If such a condition does not hold, between methods *I* or *J*, the method that give more feasible solutions for all runs will be the winner. In case that both of them fail to find feasible solutions for all runs, one having lower constraint violation will win; otherwise, both of them are equal.

Table 14 shows the ranking of the 24 optimisers when solving all test cases of truss optimisation with frequency constraints based on the Wilcoxon rank sum test. The results indicated that CMAES performs the best for all test problems (except for Case I) while DE is also ranked as the best for all test problems except for Case I and Case V. For Case I, ACOR is the best performer based on the statistical test. After summing up the ranking numbers of all design cases, it was found that CMAES is best optimiser while at close second best is DE, which is similar to the performance assessment by using average objective values.

It should be noted that, for this study, the optimisers are coded in MATLAB computing language. Some of them are downloaded from the developers of those methods while some of them are created by the authors of this article based on the concepts presented in the references. The ways of coding and the optimisation parameter setting used significantly affect the performance of those optimisers. In addition, population size and a number of generations used in this study are set following Kaveh and Zolghadr [3]. These values may not be suitable for some implemented optimisers as they cannot find feasible solutions. Nevertheless, these comparative results show that some optimisers give as good and comparable results as those presented in the literature.

Comparative study of various penalty function techniques

Penalty function techniques used in this study

In order to handle frequency constraints when using MHs, the penalty function is a necessary tool which directly affects the performance of the MH search. A good penalty function technique for MHs should be able to provide information on both feasible and infeasible sub-regions around the optimum solution while its main target is to penalise solutions outside the feasible region. In this work, four penalty function techniques are investigated including:

Table 8

Best result found in this study compared with the best result found in the literature for Case I.

Size of element No. (cm ²)	Best run found in literature [3]	Best run found in this study (CMAES)
1	35.274	35.457
2	15.463	14.867
3	32.11	35.117
4	14.065	14.810
5	0.645	0.645
6	4.88	4.542
7	24.046	23.632
8	24.34	23.586
9	13.343	12.513
10	13.543	12.134
Weight (kg)	529.09	524.49
ω_1 (≥ 7 Hz)	7.000	7.000
ω_2 (≥ 15 Hz)	16.119	16.208
ω_3 (≥ 20 Hz)	20.075	20.000
No. of function evaluations	4000	4000

FP1: fuzzy set penalty function technique [20]. The penalty function can be defined as:

$$f_p(\mathbf{x}) = f_z(\mathbf{x}) + K_D(1 - \mu_c).$$

where K_D is a penalty parameter, and f_z is the objective function value in fuzzy environment expressed as:

$$f_z(\mathbf{x}) = (f(\mathbf{x}) + c_1) / (f(\mathbf{x}) + c_1 + c_2)$$

where c_1 is a constant shift used to ensure that $f + c_1$ is always positive, and c_2 is a positive constant. The parameter μ_c is the total membership function value of all constraints computed as

$$\mu_c(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \text{ is feasible} \\ L & \text{if } g_i(\mathbf{x}) \text{ is infeasible; for all } i = 1, \dots, M \\ \frac{1}{M} \sum_{i=1}^M \mu_i & \text{otherwise} \end{cases}$$

Table 9

Best result found in this study compared with the best result found in the literature for Case II.

Design variable Shape (m), size (cm ²)	Best run found in literature [2]	Best run found in this study (CMAES)
Y_{12}, Y_{20}	0.9727	0.8789
Y_{13}, Y_{19}	1.3336	1.2533
Y_{14}, Y_{18}	1.4918	1.4285
Y_{15}, Y_{17}	1.6275	1.5613
Y_{16}	1.6939	1.6293
A_{19}, A_{37}	3.4287	2.8964
A_{20}, A_{36}	1.0010	1.0124
A_{21}, A_{35}	1.0000	1.0026
A_{11}, A_{18}	2.6942	2.6028
A_{22}, A_{34}	1.0033	1.1164
A_{23}, A_{33}	1.0910	1.2548
A_{12}, A_{17}	2.6748	2.8325
A_{24}, A_{32}	1.4141	1.3682
A_{25}, A_{31}	1.5121	1.4343
A_{13}, A_{16}	2.5228	2.5155
A_{26}, A_{30}	1.2116	1.1958
A_{27}, A_{29}	1.1869	1.4160
A_{14}, A_{15}	2.5606	2.8376
A_{28}	1.0003	1.0000
Weight (kg)	360.21	359.25
ω_1 (≥ 20 Hz)	20.0019	20.0037
ω_2 (≥ 40 Hz)	40.0310	40.0661
ω_3 (≥ 60 Hz)	60.0359	60.0201
No. of function evaluations	3000	4000

Table 10

Best result found in this study compared with the best result found in the literature for Case III.

Size of element group (cm ²)	Best run found in literature [2]	Best run found in this study (DE)
1	3.660	3.534
2	8.102	7.942
3	0.665	0.646
4	0.645	0.647
5	8.577	7.927
6	8.425	8.051
7	0.645	0.645
8	0.645	0.645
9	12.366	12.774
10	7.773	7.997
11	0.645	0.645
12	0.645	0.645
13	16.681	16.973
14	7.513	7.771
15	0.649	0.645
16	0.645	0.645
Weight (kg)	324.91	324.315
ω_1 (≈ 4 Hz)	4.00005	4.00000
ω_3 (≥ 6 Hz)	6.00001	6.00001
No. of function evaluations	3000	4000

Table 11

Best result found in this study compared with the best result found in the literature for Case IV.

Design variable shape (m), Size (cm ²)	Best run found in literature [2]	Best run found in this study (DE)
Z_A	5.6065	6.1004
X_B	2.1030	2.0003
Z_B	3.7250	3.9194
X_C	3.8834	3.9999
Z_C	2.5000	2.5001
A_1	1.0000	1.0000
A_2	1.2614	1.2908
A_3	1.3139	1.3038
A_4	1.3542	1.3747
A_5	1.3864	1.4630
A_6	1.0000	1.0000
A_7	1.6160	1.4703
A_8	1.4034	1.4359
Weight (kg)	194.27	195.19
ω_1 (≤ 15.916 Hz)	11.516	9.539
ω_2 (≥ 28.648 Hz)	28.653	28.648
No. of function evaluations	3000	4000

Table 12

Best result found in this study compared with the best result found in the literature for Case V.

Size of element group (cm ²)	Best run found in literature [39]	Best run found in this study (CPBIL)
1	19.607	20.073
2	41.290	39.894
3	11.136	14.087
4	21.025	22.195
5	10.060	11.154
6	12.758	9.745
7	15.414	14.707
Weight (kg)	8890.48	8549.44
ω_1 (≥ 9 Hz)	9.0001	9.134
ω_3 (≥ 11 Hz)	11.0007	11.004
No. of function evaluations	6000	4000

Table 13

Comparative objective function values based on Wilcoxon rank sum test for Case I.

MHs	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	(21)	(22)	(23)	(24)
DE(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ABC(2)	1	0	1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1
ACOR(3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ChSS(4)	1	1	1	0	1	1	0	1	1	0	1	1	0	1	0	0	0	0	1	0	1	0	1	1
LCA(5)	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
SA(6)	1	0	1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1
TLBO(7)	1	1	1	1	1	1	0	1	1	1	1	1	0	1	0	0	0	0	1	0	1	1	1	1
BBBC(8)	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
BPBIL(9)	1	1	1	0	1	1	0	1	0	0	1	1	0	1	0	0	0	0	1	0	1	0	1	1
CKS(10)	1	1	1	0	1	1	0	1	1	0	1	1	0	1	0	0	0	0	1	0	1	0	1	1
CMAES(11)	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CPBIL(12)	1	0	1	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
CSS(13)	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	1	0	1	1	1	1
ECTS(14)	1	0	1	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
EP(15)	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	1	0	1	1	1	1
ES(16)	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	1	0	1	1	1	1
FWA(17)	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	1	0	1	1	1	1
GA(18)	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	1	0	1	1	1	1
GSA(19)	1	0	1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1
HS (20)	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	1	0	1	1	1	1
PSO(21)	1	0	1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1
SGA(22)	1	1	1	0	1	1	0	1	1	0	1	1	0	1	0	0	0	0	1	0	1	0	1	1
BAT(23)	1	0	1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1
FFA(24)	1	0	1	0	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Sum	21	11	22	7	19	11	6	20	10	7	21	17	0	16	0	0	0	0	11	0	11	7	11	16
Ranking	2	9	1	15	5	9	18	4	14	15	2	6	19	7	19	19	19	19	9	19	9	15	9	7

Table 14

Comparative objective function values based on Wilcoxon rank sum test for all design cases.

MHs	Ranking Case I	Ranking Case II	Ranking Case III	Ranking Case IV	Ranking Case V	Sum	Ranking All
DE	2	1	1	1	2	7	2
ABC	9	4	10	5	4	32	5
ACOR	1	3	3	5	6	18	3
ChSS	15	15	18	21	19	88	21
LCA	5	4	12	9	3	33	6
SA	9	8	4	20	17	58	9
TLBO	18	20	21	21	19	99	24
BBBC	4	4	9	3	7	27	4
BPBIL	14	11	17	13	11	66	15
CKS	15	15	18	21	11	80	20
CMAES	2	1	1	1	1	6	1
CPBIL	6	4	10	4	17	41	7
CSS	19	11	13	16	8	67	16
ECTS	7	11	5	5	19	47	8
EP	19	21	21	11	4	76	18
ES	19	21	21	19	8	88	21
FWA	19	21	21	8	8	77	19
GA	19	9	13	13	16	70	17
GSA	9	19	7	17	13	65	14
HS	19	21	18	21	19	98	23
PSO	9	15	5	17	13	59	10
SGA	15	9	13	11	13	61	11
BAT	9	15	8	13	19	64	13
FFA	7	14	13	10	19	63	12

Table 15

Top two optimisation strategies for FP1-FP4.

Penalty functions	Parameter sets	Best MHs	Acronyms
FP1	1. {KD, ε } = {5, 1×10^{-3} }	1.CMAES	1.CMAES-FP11
	2. {KD, ε } = {7.5, 1×10^{-5} }	2.CMAES	2.CMAES-FP12
FP2	1. { $\varepsilon_1, \varepsilon_2$ } = {4, 4}	1.CMAES	1.CMAES-FP21
	2. { $\varepsilon_1, \varepsilon_2$ } = {3, 2}	2.CMAES	2.CMAES-FP22
FP3	1. (c_p) = 1×10^{10}	1.DE	1.DE-FP31
	2. (c_p) = 1×10^{10}	2.CMAES	2.CMAES-FP32
FP4	–	1.CMAES	1.CMAES-FP41
		2.DE	2.DE-FP41

where μ_i is the membership function value of the constraint $g_i(\mathbf{x})$ defined as:

$$\mu_i(\mathbf{x}) = \begin{cases} 0; & \text{if } g_i(\mathbf{x}) \geq \bar{d}_i \\ 1 - \frac{g_i(\mathbf{x}) - \bar{d}_i}{d_i}; & \text{if } \bar{d}_i < g_i(\mathbf{x}) < \bar{d}_i + d_i \\ 1; & \text{if } g_i(\mathbf{x}) \leq \bar{d}_i \end{cases}$$

where d_i and ε_i are the parameters to be specified.

FP2: The penalty function used in Kaveh and Zolghadr [3] which is detailed in section 'Meta-heuristic parameter settings'.

FP3: Exterior penalty function technique [2]. The penalty function is defined as:

$$f_p(\mathbf{x}) = f(\mathbf{x}) + c_p \sum_{i=1}^{ng} \left[\max \left\{ 0, \frac{\omega_i}{\omega_{i,all}} - 1 \right\} \right]^2$$

where c_p is a constant parameter.

FP4: The penalty function technique presented in Kim et al. [21]:

$$f_b(\mathbf{x}) = \begin{cases} g_{\max}(\mathbf{x}); & \text{if } g_{\max}(\mathbf{x}) > 0 \\ \arctan(f(\mathbf{x})) - \frac{\pi}{2}; & \text{otherwise} \end{cases}$$

Each penalty function technique with several parameter sets is employed with the top five optimisers (CMAES, DE, ACOR, BBBC, ABC) in order to obtain the best combination of an optimiser and a penalty function with effective parameter settings. The Case I test problem will be used in this investigation. The parameters settings of each technique were defined based on several setting values, which can be detailed as:

- FP1: The values of the parameters d , c_1 , and c_2 are assigned as 0.25, 0 and 1, respectively, while the sets of {KD, ε } are varied as: $\{1 \times 10^3, 2.5\}$, $\{1 \times 10^3, 5\}$, $\{1 \times 10^5, 7.5\}$, and $\{1 \times 10^{10}, 2.5\}$.
- FP2: The values of the parameters $\{\varepsilon_1, \varepsilon_2\}$ are assigned as {2, 2}, {2, 3}, {3, 2} and {4, 4}.
- FP3: The parameter c_p is assigned as $\{1 \times 10^3, 1 \times 10^5, 1 \times 10^{10}\}$.
- FP4: The technique requires no parameter setting.

As a result, there are $5 \times (4 + 4 + 3 + 1) = 60$ optimisation strategies to be tested. Having run all of them for solving Case I for

30 trials (each strategy), the top two optimisation strategies for FP1–FP4 (based on the average objective function) which will be used for further investigation are shown in Table 15 with the new acronyms.

Results

After performing 30 optimisation runs of the 8 optimisers detailed in Table 15, the comparative results are shown in Tables 16–20. The performances of the four penalty function techniques were measured based on the average objective function and constraint violation similarly to the comparative study in section 'Results and discussion'. From Tables 16–20, the best performer of all test problems, with the exception of Case V, was CMAES-FP22 which uses the parameter of FP2 as $\varepsilon_1 = 3$ and $\varepsilon_2 = 2$. The second best for Case I, Case II, and Case IV was DE-FP31 which uses the parameter $c_p = 1 \times 10^{10}$. The second best of Case III was CMAES-FP22. For the Case V, the best performer was CMAES-FP11, while the second best was CMAES-FP22. For Case I–II, all optimisers gave slightly different results but CMAES-FP22 totally dominates the rest in Case III. CMAES-FP22 was also the obvious winner in Case IV. For Case V, they are said to be equally good except for CMAES-FP12. Overall, the best penalty function technique for handling frequency constraints of truss optimisation was FP2, the Kaveh–Zolghadr technique.

Table 16
Comparative performance of four penalty function techniques for Case I.

Optimisers	Mean objective function value	ω_{1_av} (≥ 7)	ω_{2_av} (≥ 15)	ω_{3_av} (≥ 20)
CMAES-FP11	529.1552	6.9992	16.9786	20.0010
CMAES-FP12	530.5195	7.0001	17.1779	20.0020
CMAES-FP21	527.7770	7.0000	16.8876	20.0011
CMAES-FP22	526.5098	7.0000	16.6249	20.0004
DE-FP31	527.2439	7.0002	16.4269	20.0013
CMAES-FP32	527.9652	7.0001	16.7574	20.0018
CMAES-FP41	528.7897	7.0002	16.8243	20.0008
DE-FP41	528.3497	7.0004	16.6411	20.0043

Table 17
Comparative performance of four penalty function techniques for Case II.

Optimisers	Mean objective function value	ω_{1_av} (≥ 20)	ω_{2_av} (≥ 40)	ω_{3_av} (≥ 60)
CMAES-FP11	362.8897	19.4273	40.1173	60.2017
CMAES-FP12	361.4871	18.3451	40.1264	60.1949
CMAES-FP21	363.2154	20.0423	40.1000	60.2313
CMAES-FP22	361.4538	20.0338	40.0852	60.1931
DE-FP31	362.9061	20.0357	40.0846	60.1208
CMAES-FP32	363.9191	20.0523	40.1597	60.1321
CMAES-FP41	363.2401	20.0565	40.1135	60.2101
DE-FP41	365.0338	20.0183	40.1145	60.1229

Table 18
Comparative performance of four penalty function techniques for Case III.

Optimisers	Mean objective function value	ω_{1_av} ($=4$)	ω_{3_av} (≥ 6)
CMAES-FP11	890.6069	4.0004	7.6233
CMAES-FP12	906.9221	4.0001	7.4793
CMAES-FP21	423.4423	4.0006	6.0244
CMAES-FP22	330.1653	4.0003	6.0016
DE-FP31	924.4752	4.6154	7.1908
CMAES-FP32	931.7116	4.6155	7.9549
CMAES-FP41	951.1856	4.0007	7.8526
DE-FP41	908.6447	4.0000	7.3944

Table 19
Comparative performance of four penalty function techniques for Case IV.

Optimisers	Mean objective function value	ω_{1_av} (≤ 15.916)	ω_{2_av} (≥ 28.648)
CMAES-FP11	196.9101	11.7740	27.9860
CMAES-FP12	194.5566	10.8719	27.4154
CMAES-FP21	205.1678	10.3230	28.6678
CMAES-FP22	212.4709	10.3939	28.6490
DE-FP31	208.9579	10.8067	28.6592
CMAES-FP32	245.4026	11.0400	28.6767
CMAES-FP41	240.3645	11.5884	28.6692
DE-FP41	215.7695	11.0834	28.6548

Table 20
Comparative performance of four penalty function techniques for Case V.

Optimisers	Mean objective function value	ω_{1_av} (≥ 9)	ω_{2_av} (≥ 11)
CMAES-FP11	8706.8400	8.9990	10.9991
CMAES-FP12	8437.4295	8.7638	10.7235
CMAES-FP21	8708.1012	9.0001	11.0000
CMAES-FP22	8707.4765	9.0000	11.0000
DE-FP31	8710.2675	9.0002	11.0001
CMAES-FP32	8708.1125	9.0000	11.0000
CMAES-FP41	8710.8733	9.0002	11.0001
DE-FP41	8711.5702	9.0004	11.0002

Conclusions and discussion

In this work, the performances of various established MHs were investigated for truss optimisation with frequency constraints while an efficient and effective penalty function technique used to handle the frequency constraints was also examined. After solving five optimisation problems of truss structures with frequency constraints, based on the average objective values, CMAES was the best optimiser for all test problems except Case II and gave the most consistent results except for Case III. The DE was the best according to a convergence rate while the ACOR was the most consistent for Case III. Based on the statistical Wilcoxon rank sum test, the top two optimisers were CMAES and DE with the former being slightly better. For the comparative study of various penalty function techniques, the results indicated that the Kaveh–Zolghadr technique was the most efficient and effective technique.

Acknowledgement

The second author is grateful for the support from the Thailand Research Fund (TRF), Grant no. BRG5580017.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at <http://dx.doi.org/10.1016/j.advengsoft.2014.04.005>.

References

- [1] Gomes HM. Truss optimization with dynamic constraints using a particle swarm algorithm. *Expert Syst Appl* 2011;38:957–68.
- [2] Gholizadeh S, Barzegar A. Shape optimization of structures for frequency constraints by sequential harmony search algorithm. *Eng Optimiz* 2012;45:627–46.
- [3] Kaveh A, Zolghadr A. Truss optimization with natural frequency constraints using a hybridized CSS-BBBC algorithm with trap recognition capability. *Comput Struct* 2012;102–103:14–27.
- [4] Kaveh A, Zolghadr A. Topology optimization of trusses considering static and dynamic constraints using the CSS. *Appl Soft Comput* 2013;13:2727–34.
- [5] Miguel LFF, Miguel LFF. Shape and size optimization of truss structures considering dynamic constraints through modern metaheuristic algorithms. *Expert Syst Appl* 2012;39:9458–67.

- [6] Wei L, Tang T, Xie X, Shen W. Truss optimization on shape and sizing with frequency constraints based on parallel genetic algorithm. *Struct Multidiscip Optimiz* 2011;43:665–82.
- [7] Zuo W, Xu T, Zhang H, Xu T. Fast structural optimization with frequency constraints by genetic algorithm using adaptive eigenvalue reanalysis methods. *Struct Multidiscip Optimiz* 2011;43:799–810.
- [8] Bureerat S, Limtragool J. Structural topology optimisation using simulated annealing with multiresolution design variables. *Finite Elem Anal Des* 2008;44:738–47.
- [9] Yildiz AR. A novel particle swarm optimization approach for product design and manufacturing. *Int J Adv Manuf Technol* 2009;40:617–28.
- [10] Yildiz AR. A new hybrid differential evolution algorithm for the selection of optimal machining parameters in milling operations. *Appl Soft Comput* 2013;13:1561–6.
- [11] Tiwari A, Oduguwa VR, Roy R. Rolling system design using evolutionary sequential process optimization. *IEEE Trans Evol Comput* 2008;12:196–202.
- [12] Zhang J, Wang Y. Deflection recognition of cold rolling strip steel based on ACO algorithm with quantum action. *Lect Notes Comput Sci* 2012;7145:263–71.
- [13] Pholdee N, Bureerat S. Performance enhancement of multiobjective evolutionary optimizers for truss design using an approximate gradient. *Comput Struct* 2012;106–107:115–24.
- [14] Pholdee N, Bureerat S. Hybridisation of real-code population-based incremental learning and differential evolution for multiobjective design of trusses. *Inf Sci* 2013;223:136–52.
- [15] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimisation: artificial bee colony (ABC) algorithm. *J Global Optim* 2007;39:459–71.
- [16] Kaveh A, Talatahari S. A novel heuristic optimisation method: charged system search. *Acta Mech* 2010;213:267–89.
- [17] Kashan AH. An efficient algorithm for constrained global optimisation and application to mechanical engineering design: league championship algorithm (LCA). *Comput Aided Des* 2011;43:1769–92.
- [18] Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimisation: a novel method for constrained mechanical design optimisation problems. *Comput Aided Des* 2011;43:303–15.
- [19] Kaveh A, Zolghadr A. Democratic PSO for truss layout and size optimization with frequency constraints. *Comput Struct* 2014;130:10–21.
- [20] Cheng FY, Li D. Fuzzy set theory with genetic algorithms in constrained structural optimization. In: *ASCE Proceeding of US-Japan Joint Seminar on Structural Optimization, Advances in Structural optimization*, New York, 1997. p. 55–6.
- [21] Kim TH, Maruta I, Sugie T. A simple and efficient constrained particle swarm optimization and its application to engineering design problems. *Proc Inst Mech Eng, Part C: J Mech Eng Sci* 2010;224:389–400.
- [22] Storn R, Price K. Differential evolution – a simple and efficient heuristic for global optimisation over continuous spaces. *J Global Optim* 1997;11:341–59.
- [23] Socha K, Dorigo M. Ant colony optimisation for continuous domains. *Eur J Oper Res* 2008;185:1155–73.
- [24] Dorigo M. *Optimization, Learning and Natural Algorithms* (in Italian). Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.
- [25] Pholdee N, Bureerat S. Hybrid real-code ant colony optimisation for constrained mechanical design. *Int J Syst Sci*, 2014, in press. <http://dx.doi.org/10.1080/00207172.2014.891664>.
- [26] Kirkpatrick S, Gelatt Jr CD, Vecchi MP. Optimization by simulated annealing. *Science* 1983;220(4598):671–80.
- [27] Erol OK, Eksin I. A new optimisation method: big bang-big crunch. *Adv Eng Softw* 2006;37:106–11.
- [28] Baluja S. Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimisation and Competitive Learning. Technical Report CMU-CS-95_163, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1994.
- [29] Yang XS, Deb S. Engineering optimisation by cuckoo search. *Int J Math Model Numer Op* 2010;1:330–43.
- [30] Hansen N, Müller SD, Koumoutsakos P. Reducing the time complexity of the derandomised evolution strategy with covariance matrix adaptation (CMA-ES). *Evol Comput* 2003;11:1–18.
- [31] Bureerat S. Improve population-based incremental learning in continuous spaces. *Adv Intell Soft Comput* 2011;96:77–86.
- [32] Herrera F, Lozano M, Molona D. Continuous scatter search: an analysis of the integration of some combination methods and improvement strategies. *Eur J Oper Res* 2006;169:450–76.
- [33] Teh YS, Rangaiah GP. Tabu search for global optimization of continuous functions with application to phase equilibrium calculations. *Comput Chem Eng* 2003;27:1665–79.
- [34] Glover F. Tabu search: part 1. *ORSA J Comput* 1989;1989(1):190–206.
- [35] Bäck T. *Evolutionary algorithms in theory and practice*. Oxford: Oxford University Press; 1996.
- [36] Tan Y, Zhu Y. Fireworks algorithm for optimization. In: Tan Y, Shi Y, Tan KC, editors. *ICSI 2010. LNCS*, vol. 6145. Heidelberg: Springer; 2010. p. 355–64.
- [37] Lindfield G, Penny J. *Numerical methods using MATLAB*. Ellis Horwood; 1995.
- [38] Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: a gravitational search algorithm. *Inf Sci* 2009;179:2232–48.
- [39] Geem ZW, Kim JH. A new heuristic optimisation algorithm: harmony search. *Simulation* 2001;76:60–8.
- [40] Reyes-Sierra M, Coello Coello CA. Multi-objective particle swarm optimisers: a survey of the state-of-the-art. *Int J Comput Intell Res* 2006;2:287–308.
- [41] Khatib W, Peter JF. The Stud GA: a mini revolution? *Lect Notes Comput Sc* 1998;1498:683–91.
- [42] Yang XS, Gandomi AH. Bat algorithm: a novel approach for global engineering optimization. *Eng Comput* 2012;29:464–83.
- [43] Kaveh A, Zolghadr A. Enhanced bat algorithm for optimal design of skeletal structure. *Asian J Civil Eng* 2014;15:179–212.
- [44] Yang Xin-She. *Nature-inspired metaheuristic algorithms*. 2nd ed. Luniver Press; 2010.
- [45] Baykasog A, Ozsoydan FB. An improved firefly algorithm for solving dynamic multidimensional knapsack problems. *Expert Syst Appl* 2014;41:3712–25.



Simultaneous topology and sizing optimization of a water distribution network using a hybrid multiobjective evolutionary algorithm



Sujin Bureerat^{a,*}, Krit Sriworamas^b

^a Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, 40002, Thailand

^b Department of Civil Engineering, Faculty of Engineering, Ubon Rajathaneey University, 34190, Thailand

ARTICLE INFO

Article history:

Received 18 October 2010

Received in revised form 13 July 2012

Accepted 11 April 2013

Available online 19 April 2013

Keywords:

Hybrid differential evolution

Pipe network

Population-based incremental learning

Topology optimization

Water distribution systems

ABSTRACT

This paper proposes a new direction for design optimization of a water distribution network (WDN). The new approach introduces an optimization process to the conceptual design stage of a WDN. The use of multiobjective evolutionary algorithms (MOEAs) for simultaneous topology and sizing design of piping networks is presented. The design problem includes both topological and sizing design variables while the objective functions are network cost and total head loss in pipes. The numerical technique, called a network repairing technique (NRT), is proposed to overcome difficulties in operating MOEAs for network topological design. The problem is then solved by using a number of established and newly developed MOEAs. Also, two new MOEAs namely multiobjective real code population-based incremental learning (RPBIL) and a hybrid algorithm of RPBIL with differential evolution (termed RPBIL-DE) are proposed to tackle the design problems. The optimum results obtained are illustrated and compared. It is shown that the proposed network repairing technique is an efficient and effective tool for topological design of WDNs. Based on the hypervolume indicator, the proposed RPBIL-DE is among the best MOEA performers.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

A piping or water distribution network is one of the most important engineering systems found in daily life. The network normally consists of pumps, tanks, reservoirs, valves, and other devices interconnected by pipes for providing water to the public. It can be viewed as a graph having pipes as links while nodes represent sources, consumers, pipe connections, and hydraulic control elements. Piping network design can be classified into three phases as conceptual, preliminary and detailed design. These are not completely separated design phases since we can sometimes perform several design phases simultaneously. The conceptual design is usually accomplished in such a way that a network topology (layout) is defined based on the premise that a network with loops will become more reliable and more expensive whereas one with branches will be less reliable but cheaper [1,2]. The network topology is also dependent on many restrictions such as households, roads, and other obstructions.

Having obtained a predefined network layout, network parameters e.g. pipe diameters, tank and pump sizes, are determined to meet the optimum design objectives while fulfilling network constraints. Optimal pipe sizing of WDN has been investigated

for many years. The optimization problems for pipe networks can have either single [1,3–9] or multiple [1,10–13] design objectives. A typical single-objective design problem is assigned to find a minimum network cost while nodal heads are constrained ensuring that the network is sufficiently reliable. Since Walski in 2001 [14] and other researchers illustrated that minimizing the network cost will reduce network benefits (reliability) or vice versa, it means there are multiple design objectives to be optimized for one WDN. This study leads to a new direction for the preliminary design of WDN. Network reliability can be defined as the ability of a network system to provide adequate performance under both normal and abnormal conditions [15]. Some work has been made towards network reliability models due to mechanical and hydraulic failures [4,16–20]. The network simulation by means of head-driven analysis has been proposed so as to make up for the shortcomings of the demand-driven model [15,20,21]. The criteria used to measure network reliability for use in an optimization process have been defined. The minimum surplus head index was probably the first reliability index proposed for an optimization process by Walski and Gessler in 90s [22]. Later, there have been other indices proposed for measuring the WDN reliability e.g. the total surplus head index [22], the resilience index [10], and the network resilience [2]. As for the optimizers, gradient-based linear and nonlinear optimization methods were implemented in the early years [4]. The use of evolutionary algorithms has later become more popular and attractive due to their robustness, simplicity and universality [1–3,10–13].

* Corresponding author. Tel.: +66 43 202845; fax: +66 43 202849.

E-mail address: sujbur@kku.ac.th (S. Bureerat).

Some work related to the more practical detailed design and analysis of WDNs has been made. The network connectivity/topology analysis [23] as well as the algorithm for an automatic detection of topological changes in WDN due to abnormal operating conditions [15] has been developed. As the real network system uses valves to isolate some segments from the network, the impacts of valve location and distribution on the network in both normal and abnormal conditions have been studied [24–27].

From the above-mentioned literature, considerable progress has been made towards the preliminary and detailed design stages of WDN but there has been limited work contributing to the use of optimization techniques to find optimum WDN layouts at the conceptual design stage. However, in other engineering fields, it has been found that the application of topological optimization at the conceptual design stage usually leads to better design results e.g. communication networks [28], internet topology [29], and artificial neural networks [30]. In practice, a network topology is normally formed by an experienced design engineer. If implementation of topology optimization on water distribution networks is possible, it means even inexperienced designers can perform a design task whereas practical and high performance network topology can be expected.

This paper presents a new approach for optimizing WDNs at the conceptual design stage. Multiobjective evolutionary algorithms are employed to find initial WDN layouts by means of simultaneous topological and sizing optimization. This design approach is said to combine and carry out the conceptual and preliminary design stages of WDNs within one simulation run. Design of piping networks using multiobjective evolutionary algorithms is demonstrated. The design problem includes both topological and sizing design variables while the objective functions are network cost and total head loss in pipes. The numerical technique, called a network repairing technique, is proposed to overcome difficulties in operating MOEAs for network topology optimization. The design problem is then solved by using a number of MOEAs, i.e. version two of strength Pareto evolutionary algorithm (SPEA2) using real codes [31], version two of non-dominated sorting genetic algorithm (NSGAII) using real codes [32], multiobjective particle swarm optimization (MPSO) [33], unrestricted population size-evolutionary multiobjective optimization algorithm (UPS-EMOA) [34], and multiobjective harmony search (MOHS) [35]. Apart from those state-of-the-art MOEAs, two new multiobjective metaheuristics RPBIL and RPBIL-DE are introduced. The former is based on real code population-based incremental learning [36] while the latter is a hybrid algorithm that integrates the operator of RPBIL and DE. The Pareto optimal solutions obtained from the various MOEAs are illustrated and compared. The comparative performance of MOEAs is investigated. Based on the hypervolume indicator, the proposed hybrid optimizer RPBIL-DE is among the top performers. The proposed network repairing technique is an efficient tool for topological design of water distribution networks.

The rest of the paper is organized as follows. Section 2 briefly details a simultaneous topology and sizing optimization of pipe networks. The network repairing technique used to deal with illegitimate design solution is explained in Section 3. Section 4 presents a new hybrid multiobjective evolutionary algorithm which is the combination of RPBIL and DE. Design demonstration and performance assessment of various MOEAs are setup in Section 5 whereas the comparative results are given in Section 6. Finally, the conclusions are drawn in Section 7.

2. Design problem

In practice, a pipe network is designed by taking into account economics, safety, maintenance and public health. The common

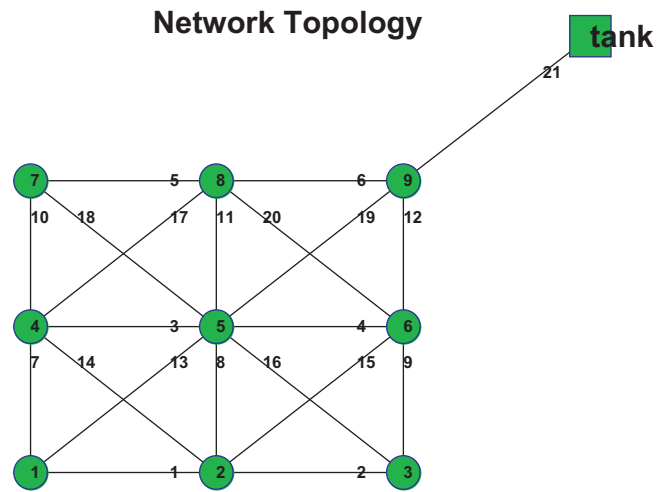


Fig. 1. Ground network.

design criteria include network cost and reliability, redundancy [21], total head loss in pipes, water quality, network infrastructure etc. The optimization process is not only applied to the design of a new network but also used in the rehabilitation of an existing network. Network cost is an unavoidable design objective for many engineering systems. For pipe network design, minimizing cost always affects network reliability; therefore, practical network design needs to simultaneously minimize cost and maximize reliability. Several indicators are proposed for measuring pipe network reliability such as network resilience [2], and total head loss in pipes.

A multiobjective optimization problem used in this work can be written as:

$$\min_{\mathbf{x}} : \{f_1, f_2\} \quad (1)$$

subject to

$$F_i(\mathbf{H}, \mathbf{D}) = 0; \quad i = 1, \dots, nn$$

$$H_i \geq H_i^{\min}; \quad i = 1, \dots, nn$$

where $\mathbf{x}$ is a design vector determining a network topology and pipe sizes, f_1 is the total network cost, f_2 is the total head loss in the network, H_i is the head at node i , and $H_i^{\min}$ is a minimum allowable nodal head of node i . The hydraulic constraints $F_i(\mathbf{H}, \mathbf{D})$ are nodal mass balance, and loop or path energy balance equations, which can be handled by using the EPANET 2 software. The objective functions can be computed as:

$$\begin{aligned} f_1 &= \sum_{i=1}^{np} \text{Cost}_i(D_i, L_i) \\ f_2 &= \sum_{i=1}^{np} L_i h_i(D_i) \end{aligned} \quad (2)$$

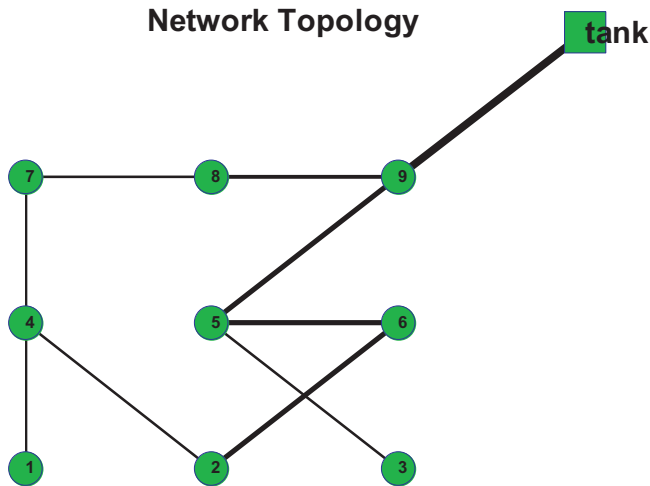
where Cost_i is the cost of pipe i , L_i are pipe lengths, h_i are head loss per unit length, and D_i are the discrete pipe sizes.

In order to achieve a simultaneous topological and sizing optimization, the predefined nodes, links, and other network conditions are assigned. Fig. 1 shows a particular 9-node network having 3×3 junctions and one tank. Then, the ground network containing all possible pipe elements connecting between nodes in the network is generated as shown in the figure. This ground network has 21 pipes where pipe 21 linking node 9 and the tank is not assigned to be a design variable. The design variables determine the 20 pipe diameters while the list of available pipe diameters, their

Table 1

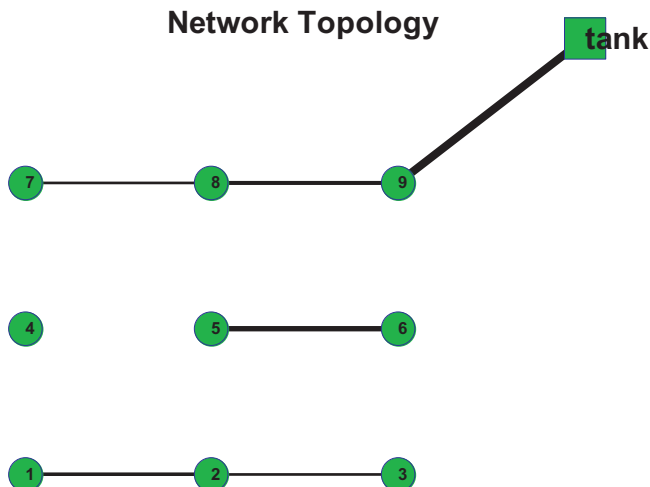
Integer variables, pipe diameters and cost (modified from [2]).

Integer variable	Pipe diameters (mm)	Cost/length (\$)
1	0	0
2	100	11
3	150	16
4	200	23

**Fig. 2.** Legitimate network topology: $\mathbf{x}_1$.

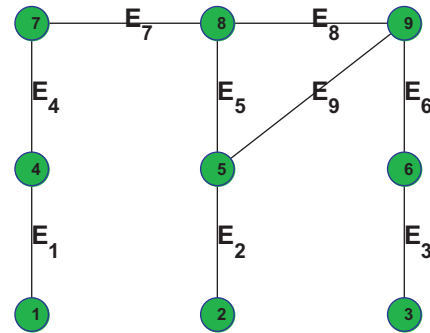
corresponding prices, and the encoded integers is given in Table 1. The diameter of the 21st pipe is fixed as 300 mm.

Figs. 2 and 3 illustrate how the proposed decoding technique works. In Fig. 2, the vector of integer design variables is $\mathbf{x}_1 = \{1, 1, 1, 4, 2, 3, 3, 1, 1, 3, 1, 1, 1, 2, 4, 2, 1, 1, 4, 1\}^T$. It can be seen that pipe elements having an integer '1' as a diameter identification number are removed from the ground network whereas the others represent the existing pipe elements with various diameters. By using this concept, a network topology is obtained. A difficulty arises when most pipe diameters become zeros as shown in Fig. 3 where $\mathbf{x}_2 = \{3, 2, 1, 4, 2, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}^T$. The obtained network topology has three disconnected pipe groups and one junction having no links to the others. Such a network is said to be unfeasible since the network analysis via EPANET cannot be accomplished. This illegitimate solution needs to be repaired so that it is usable; as

**Fig. 3.** Illegitimate network topology: $\mathbf{x}_2$.

Network Topology

tank

**Fig. 4.** List of neighbor elements.

a result, a numerical scheme called a network repairing technique is proposed.

3. Network repairing technique

According to the design solution in Fig. 3, the network needs to be repaired to enable a pipe network simulation (demand-driven analysis). The term network repairing technique is borrowed from the technique called chromosome repairing technique [37]. Although having a similar name and the same purpose of use, which is to make an unacceptable design solution become solvable, they are totally different in details and computational procedures. The process of repairing a water distribution network consists of two main stages: the first stage is used to repair nodes that have no links or isolated nodes while the second stage is used to repair disconnected pipe groups.

Initially, the ground network is created from the predefined junctions. The closest neighbor of every junction needs to be identified, which means its neighbor pipe element is obtained. Fig. 4 shows 9 junctions of the 9-node network from the previous section and their neighbor pipe elements where E_i stands for the neighbor element of the i th junction. It should be noted that all the junctions must have different neighbor elements. As an isolated node is detected, its neighbor element (10th element) with 100 mm diameter is added to the network and the input design solution is changed to be $\mathbf{x}_3 = \{3, 2, 1, 4, 2, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}^T$. The new network is shown in Fig. 5, which has 3 disconnected pipe groups.

The second repairing task is to connect all disconnected pipe groups to become one group. From Fig. 6, at the beginning, the algorithm to find the total number of pipe groups is activated. The primary pipe group is the group that contains the element which links to the tank as shown (Group 1). The secondary pipe group (Group 2) which is the closest group to the primary group is then detected and linked together by adding the pipe element that connects the closest nodes of the two groups. In this process, the node 4 of the primary group and node 5 of the secondary group are linked together by adding to the network the 3rd element with 100 mm diameter and the updated design solution becomes $\mathbf{x}_4 = \{3, 2, 2, 4, 2, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}^T$. The pipe group obtained from connecting these two groups is set as a new primary pipe group. Afterwards, a new secondary pipe group (Group 3) is found and connected to the primary pipe group by linking node 4 and node 1. The procedure is terminated when there remains only one pipe group which implies that a legitimate pipe network is obtained. The network in Fig. 3 after being repaired is shown in Fig. 6. The finally

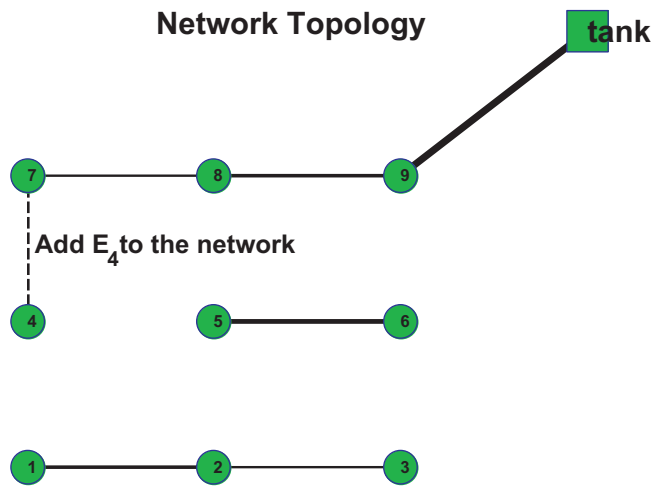


Fig. 5. Stage 1 of network repairing: $\mathbf{x}_3$.

repaired design solution is $\mathbf{x}_5 = \{3, 2, 2, 4, 2, 3, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1\}^T$.

Using this concept, the objective and constraint functions become many-to-one functions since the solutions $\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$ and $\mathbf{x}_5$ produce the same function values. In the optimization process, the junctions, ground network as well as neighbor elements of the junctions are provided initially. It is also useful to construct a matrix that is used to detect disconnected pipe groups. Function evaluation is carried out in such a way that, if the input design variables vector is legitimate, the analysis of the pipe network can then be performed. In cases that the design solution is illegitimate, the NRT is invoked to enable the network analysis. With the use of NRT, any design solution is feasible and it is always possible to analyze the network. It is possible to use evolutionary algorithms with binary, integer or real codes since we can use a round-off operator to convert design variables to integer numbers. The flowchart for NRT is given in Fig. 7.

It should be noted that selection of neighbor elements for network nodes is based on designer's decision. Obviously, with two different sets of neighbor elements, a particular solution $\mathbf{x}$ will result in different network topologies after repairing. However, optimum topologies of the design problem according to two different sets of neighbor elements will be the same and a good optimization method should be able to find it using any set of neighbor elements. In practice, there will also be some field constraints.

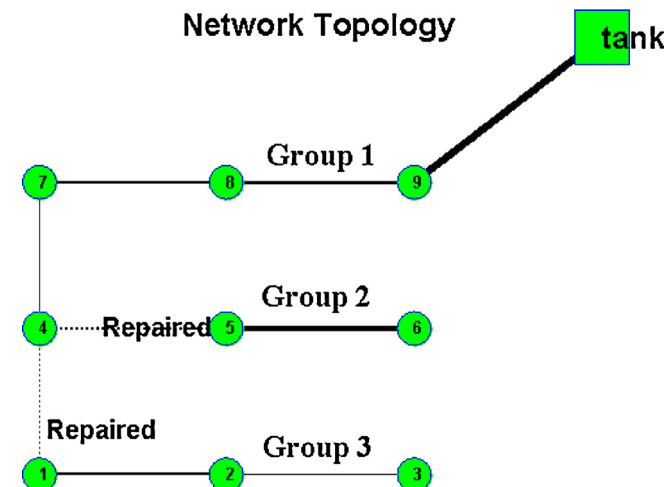


Fig. 6. Stage 2 of network repairing: $\mathbf{x}_5$.

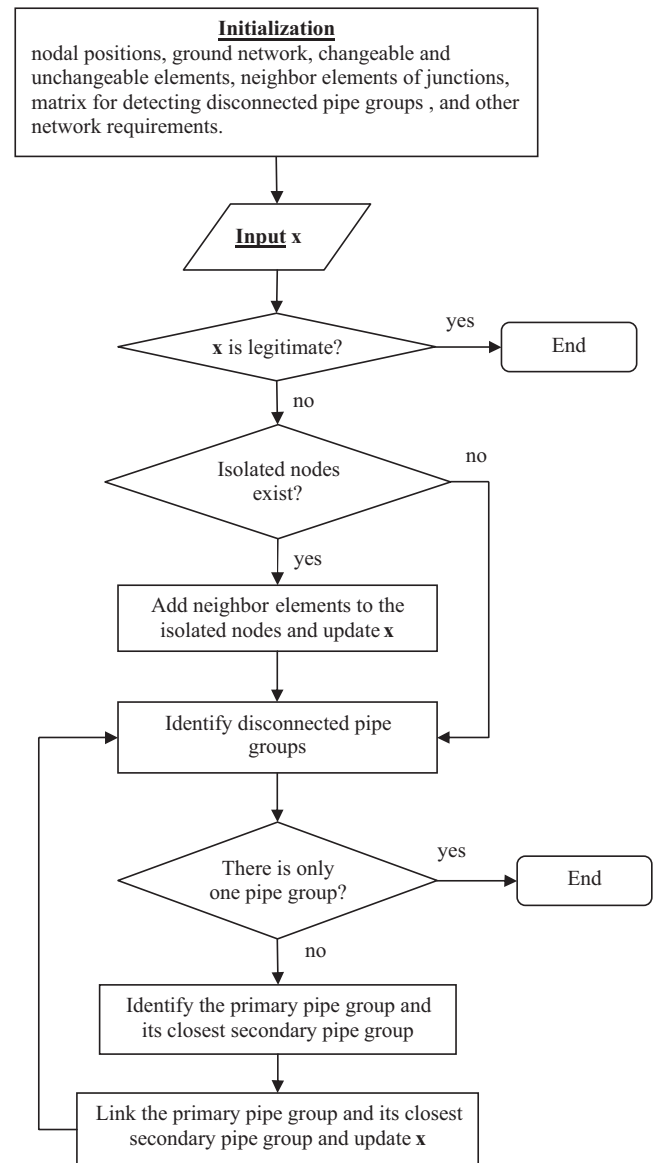


Fig. 7. Network repairing technique.

For example, diagonal pipes in a ground network must be removed due to some obstructions. Some pipe elements in the ground network are assigned to have certain diameters and thus cannot be assigned as design variables. Apart from that, nodal positions may not be aligned due to obstructions. With such constraints, the concept NRT still works as demonstrated in the OPT3 design problem in Sections 5 and 6. It is recommended to contact the corresponding author for more details of the NRT algorithm.

4. Hybrid RPBIL-DE

The first ever version of population-based incremental learning (PBIL) was introduced as a meta-heuristic that search for a single objective optimum solution based on binary codes [38]. The method was then extended for use with real codes [36]. For simplicity, RPBIL will be presented as an optimizer for a bound-constrained problem that minimize a single objective function $f(\mathbf{x})$ where $L_i \leq x_i \leq U_i$ for $i = 1, \dots, n$. L_i and U_i are lower and upper bounds of $\mathbf{x}$ respectively.

A real code population in RPBIL is represented by the so-called probability matrix $\mathbf{P}$ or P_{ij} sized $n \times m$ where m is a predefine

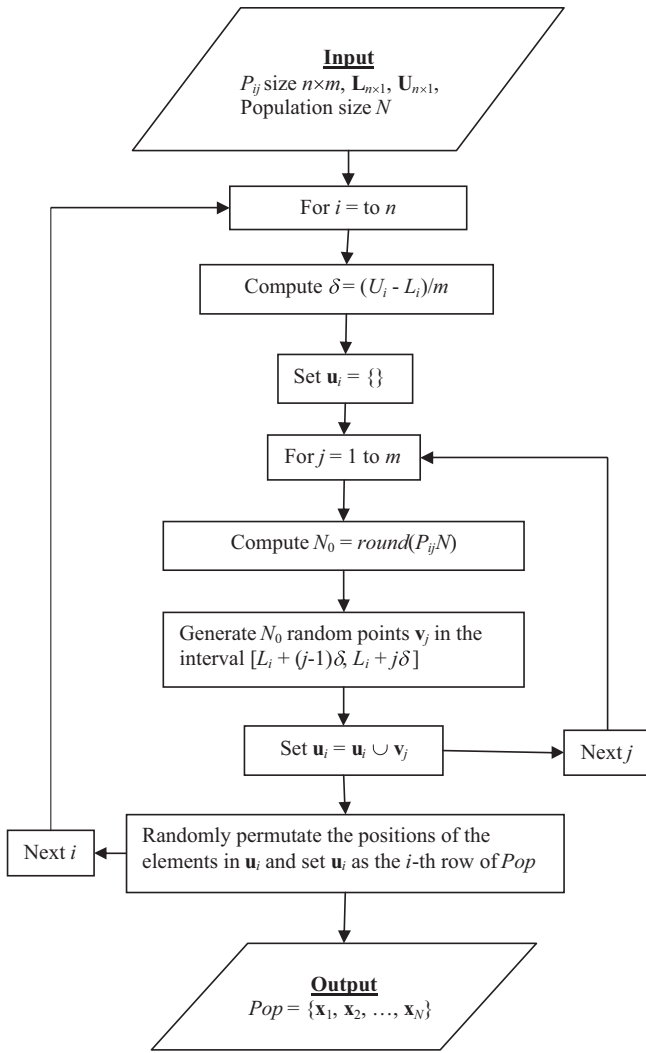


Fig. 8. Generating a population from a probability tray.

number of subsections of the interval $[L_i, U_i]$. The element P_{ij} determines the probability of which the i -th element of $\mathbf{x}$ will be located in the range $[L_i + (j-1)\delta, L_i + j\delta]$ where $\delta = (U_i - L_i)/m$. Generating a real-code population can be carried out in the similar manners as with binary PBIL where a simple procedure is given in Fig. 8. The procedure of single objective RPBIL starts with a probability matrix $\mathbf{P}$ where all elements values are assigned as $1/m$. A population according to $\mathbf{P}$ is then created with their corresponding objectives being evaluated while the best individual $\mathbf{x}_{\text{best}}$ is detected. The probability matrix is then updated based upon $\mathbf{x}_{\text{best}}$ as

$$P'_{ij} = (1 - L_R)P_{ij}^{\text{old}} + L_R \quad (3)$$

and

$$L_R = 0.5 \exp(-(j-r)^2) \quad (4)$$

The element r is determined in such a way that $\mathbf{x}_i^{\text{best}}$ is located in the range $[L_i + (r-1)\delta, L_i + r\delta]$. The learning rate L_R in Eq. (4) is set to

prevent premature convergence. In order to preserve the condition

$$\sum_{j=1}^T P_{ij} = 1, \text{ the } i\text{-th row of } \mathbf{P} \text{ is normalized as:}$$

$$P''_{ij} = \frac{1}{\sum_{j=1}^T P'_{ij}} P'_{ij} \quad (5)$$

As a result, the eventually updated P_{ij} is in Eq. (5). The probability matrix and $\mathbf{x}_{\text{best}}$ are iteratively improved until the termination condition is met.

In this paper, RPBIL is further extended to deal with a multiobjective design problem. Rather than using one probability matrix, the multiobjective version of RPBIL uses several probability matrices in order to maintain population diversity. The probability matrix will be called a probability tray to be more insightful. Given that multiobjective RPBIL uses n_T probability trays for searching, each tray will be used to produce one sub-population whereas one real code population is obtained from the combination of all n_T sub-populations. The search procedure starts with n_T of initial probability trays whose elements are full of the value $1/m$. An initial Pareto archive is also created. Then the population is generated based on the probability trays. The Pareto archive is then updated by replacing it with the non-dominated solutions sorted from the union set of the current population and the members in the previous archive. Afterwards, those non-dominated solutions are classified into n_T groups by means of a clustering technique. The centroid of each group will be used as $\mathbf{x}_{\text{best}}$ for updating a probability tray (Eqs. (3)–(5)). In cases that the number of non-dominated solutions is less than n_T , $\mathbf{x}_{\text{best}}$ will be a randomly selected solution from the archive. The procedure is repeated until a termination condition is met.

Fig. 9 shows the search procedure of multiobjective RPBIL where k is an index of probability tray number and A is an external Pareto archive. The function $\text{NonDominatedSort}(\text{Pop})$ gives non-dominated solutions obtained from sorting all members in Pop .

Multiobjective RPBIL in Fig. 9 is one efficient MOEA. However, from the performance test conducted by the authors, it is shown that the proposed multiobjective RPBIL provide good population diversity but its convergence rate is somewhat lower than expected. In the literature e.g. [39], it has been shown that hybridization of strong points of different evolutionary algorithms can lead to a better optimizer. As a result, differential evolution operators are introduced to the search procedure to explore new area of the design space. According to Fig. 9, the obtained real code population Pop is recombined with the members in the Pareto archive by using the DE operators (here it is DE/best/2/bin). The Pareto archive members are used in the recombination process so as to increase the convergence rate. This hybrid multiobjective evolutionary algorithm is termed RPBIL-DE and its computational steps are detailed in Fig. 10.

5. Numerical experiment

The numerical experiment is performed to test the proposed design strategy and the newly proposed MOEAs. Two networks with 4×4 nodes and 7×7 nodes were used for design demonstration and their ground networks are illustrated in Figs. 11 and 12 respectively. The optimization design problems for the two networks are termed OPT1 and OPT2 respectively. The third ground network (Fig. 13) is somewhat more practical than the first two ground networks as the nodal positions are not aligned, most of diagonal ground pipe elements are removed (supposedly due to some obstructions), and some pipe diameters (dashed line) are set

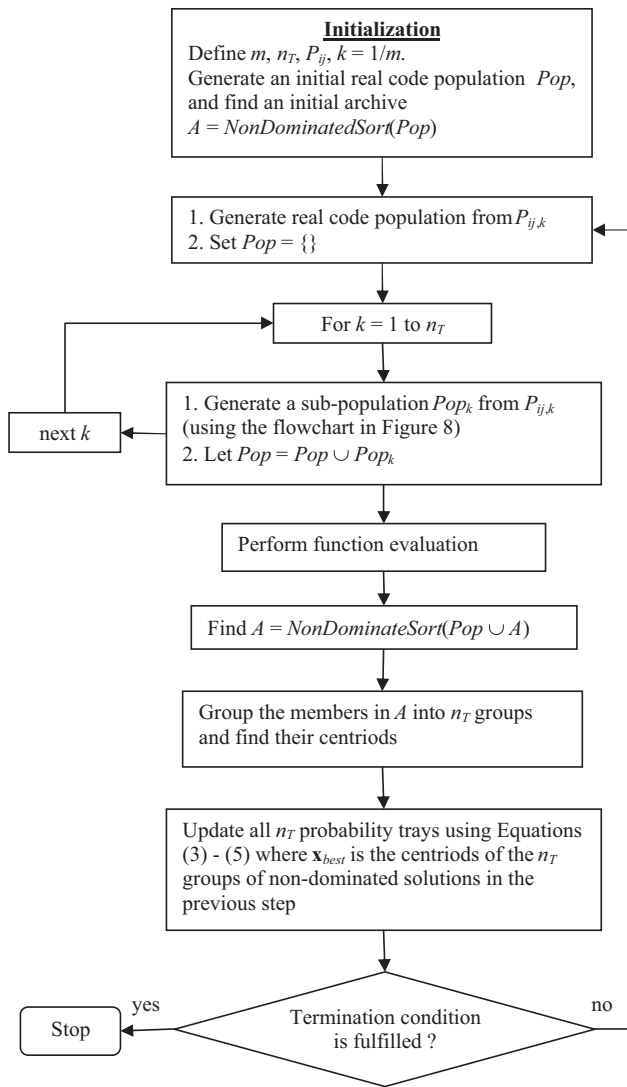


Fig. 9. Multiobjective RPIL.

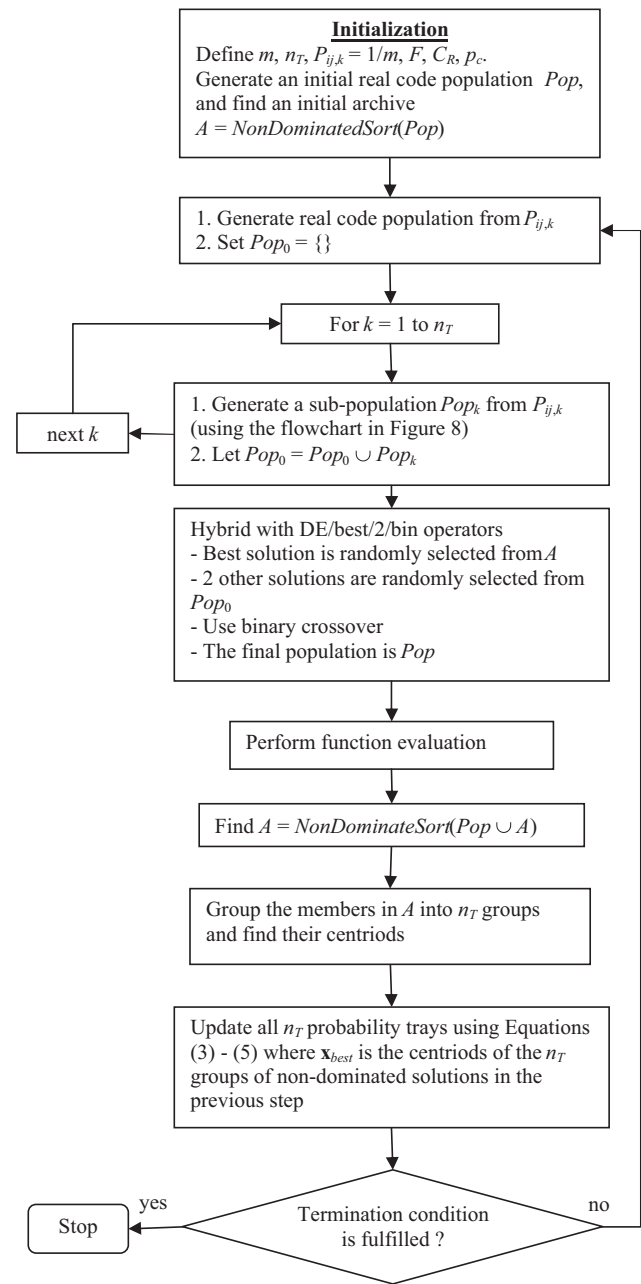


Fig. 10. Hybrid multiobjective RPIL and DE.

to be unchangeable. The design problem using this ground network is termed OPT3. The fourth network is a practical network currently used in the Chumponburee district, Surin province, Thailand. It is the network of the main pipes designed by experienced design engineers. The ground network for the Chumponburee network is illustrated in Fig. 14 and the corresponding problem is named OPT4. All the networks have one tank. The first network has 42 design variables while OPT2, OPT3 and OPT4 have 156, 64, and 20 design variables respectively. It should be noted that ground networks having diagonal pipes was created merely for design demonstration. In real world problems as with the Chumponburee network, using diagonal pipes may be impossible due to infrastructures of houses and roads. The objective and constraint functions are evaluated by using a non-commercial software EPANET version 2 [40]. Since the optimization codes were developed using MATLAB, an interface between these two programs is required.

The optimizers employed are as follows:

- MOHS using harmony memory considering rate ($HMCR$), minimum pitch adjustment rate (PAR_{min}), maximum pitch adjustment rate (PAR_{max}), minimum bandwidth rate (BW_{min}), and maximum bandwidth rate (BW_{max}) as 0.5, 0.2, 2, 0.45, and 0.9 respectively.
- UPS-EMOA using minimum population size, burst-size, crossover rate (p_c), scaling factor (F), and probability of choosing an

element from offspring in crossover (C_R) as 10, 25, 0.7, 0.8, and 0.5 respectively.

- MPSO using a starting inertia weight, an ending inertia weight, a cognitive learning factor, and a social learning factor as 0.5, 0.01, 0.5 and 0.5 respectively.
- SPEA2 using real codes where the real code crossover and mutation operators are detailed in [41]. The crossover and mutation rates are set to be 1 and 0.1 respectively.
- NSGAI1 using real codes where the real code crossover and mutation operators are detailed in [41]. The crossover and mutation rates are set to be 1 and 0.1 respectively.
- RPIL using the number of subintervals for bound constraints (m), and number of probability trays (n_T) as 30, and $N/5$ respectively.
- RPIL-DE using the number of subintervals for bound constraints, number of probability trays, crossover rate, scaling factor, and probability of choosing an element from offspring in crossover as 30, $N/5$, 0.7, 0.8, and 0.5 respectively.

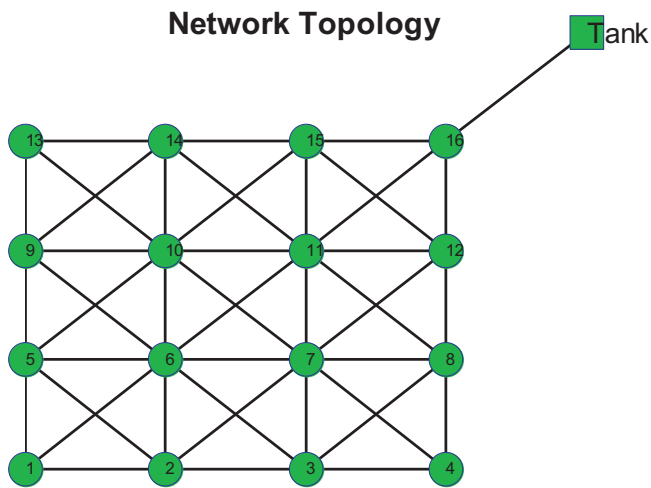
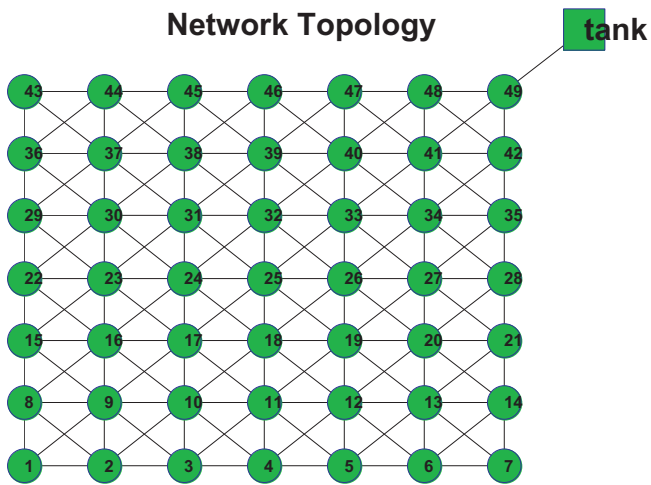
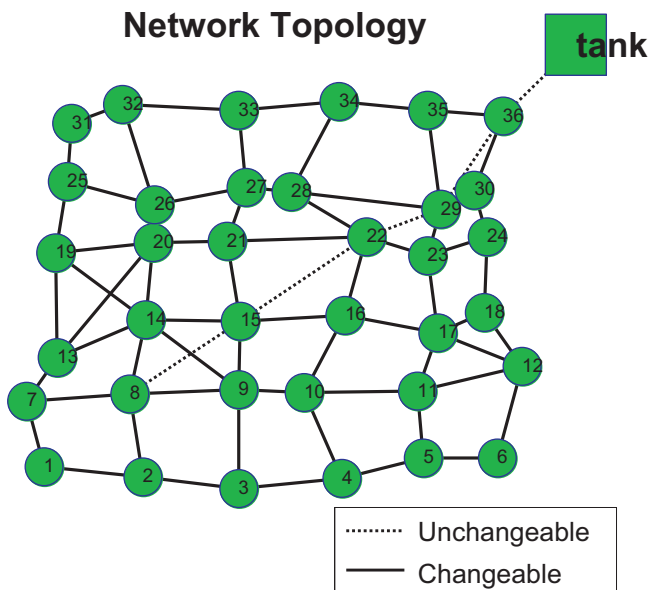
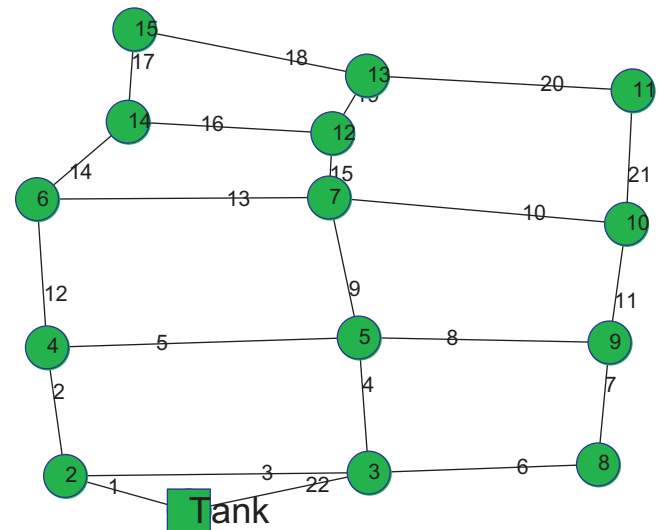
Fig. 11. Ground network of OPT1 with 4×4 nodes.Fig. 12. Ground network of OPT2 with 7×7 nodes.Fig. 13. Ground network of OPT3 with 6×6 nodes.

Fig. 14. Ground network of Chumponburee network OPT4 with 15 nodes.

All of the MOEAs employ the non-dominated sorting technique for constrained optimization presented in [42]. For the OPT1 and OPT3 design problems, the number of generations is set to be 150 loops (200 for OPT2 and 100 for OPT3) while the population size and external Pareto archive size are set to be 100 (150 for OPT2 and 80 for OPT3). Each multiobjective optimizer is used to solve the design problems for 10 runs so as to measure their convergence rate and consistency. In this work, the so-called hypervolume one of the most reliable Pareto front indicators is used for MOEAs search performance comparison.

6. Design results

Having run all the 7 multiobjective evolutionary optimizers for tackling four design problems of simultaneous topology and sizing optimization of pipe networks, $7 \times 3 \times 10$ non-dominated fronts are obtained. The performance comparison of the various MOEAs is given in Table 2. Based on the hypervolume indicator, the higher hypervolume means the better non-dominated front. The values of a hypervolume are normalized to the range of [0,1] (the best front has the hypervolume of 1 while the worst front has the hypervolume value of 0) so as to ease in comparison. From the results, the best optimizer for both convergence rate (average hypervolume) and consistency (standard deviation of hypervolume) for OPT1 is RPBIL-DE while the second best is SPEA2. For OPT2 which has higher number of design variables, the best optimizer based on convergence rate is SPEA2 while the second best is RPBIL-DE. Based on the standard deviation, the most consistent method is RPBIL-DE while the second best is SPEA2. RPBIL-DE is also the best for OPT3 while NSGAI is the close second best. The most and second most consistent methods based on the standard deviation of hypervolume are MPISO and UPS-EMOA respectively. For OPT4, the method that gives best convergence rate is RPBIL-DE while the second best is MPISO. The most consistent method is MOHS whereas the second most consistent algorithm is RPBIL-DE.

Overall, the top two performers for this performance test are RPBIL-DE and SPEA2. The former is more efficient with design problems having lower number of design variables and sufficiently high number of generations and population size. The latter is superior when dealing with a design problem with higher number of design variables and limited number of generations and population size. By adding the DE operators to RPBIL, the performance of the hybrid algorithm is greatly improved. Other MOEAs are said to

Table 2
Hypervolume comparison.

Opt	MOEAs	Average	Std	Max	Min
OPT1	MOHS	0.4905	0.0549	0.5694	0.4068
	UPS-EMOA	0.9140	0.0811	1.0000	0.7812
	MPSO	0.6973	0.0692	0.8483	0.5914
	SPEA2	0.9487	0.0279	0.9847	0.9008
	NSGAI	0.2243	0.1666	0.5189	0.0000
	RPBIL	0.5687	0.0817	0.6806	0.4111
	RPBIL-DE	0.9718	0.0148	0.9920	0.9495
OPT2	MOHS	0.1683	0.0309	0.2345	0.1353
	UPS-EMOA	0.8607	0.1424	0.9867	0.5715
	MPSO	0.5390	0.0664	0.6648	0.4680
	SPEA2	0.9544	0.0231	1.0000	0.9166
	NSGAI	0.4441	0.2604	0.8336	0.0000
	RPBIL	0.6361	0.0467	0.6871	0.5395
	RPBIL-DE	0.9162	0.0195	0.9508	0.8946
OPT3	MOHS	0.2734	0.0832	0.3871	0.1604
	UPS-EMOA	0.6700	0.0772	0.7978	0.5615
	MPSO	0.5007	0.0728	0.6357	0.3788
	SPEA2	0.6032	0.1032	0.7547	0.3735
	NSGAI	0.7704	0.1184	1.0000	0.5992
	RPBIL	0.1319	0.2005	0.4401	0.0000
	RPBIL-DE	0.7783	0.0830	0.8527	0.5666
OPT4	MOHS	0.5851	0.0512	0.6843	0.5259
	UPS-EMOA	0.8324	0.0742	0.9534	0.7238
	MPSO	0.8528	0.0633	0.9676	0.7561
	SPEA2	0.8498	0.1305	1.0000	0.6115
	NSGAI	0.7571	0.2744	0.9427	0.0000
	RPBIL	0.4801	0.0613	0.5895	0.3961
	RPBIL-DE	0.8598	0.0610	0.9511	0.7705

have acceptable performance although they are slightly inferior to the proposed RPBIL-DE.

Fig. 15 shows the best front of OPT1 obtained from running RPBIL-DE while the networks of the 12 selected design points are illustrated in Fig. 16. The non-dominated front has various network topologies. The best front of OPT2 obtained from using SPEA2 is shown in Fig. 17 where the corresponding networks of some selected points are displayed in Fig. 18. The best front of OPT3 obtained from NSGAI is displayed in Fig. 19 while the selected network topologies are shown in Fig. 20. The best front of the Chumponburee network design problem obtained from using RPBIL-DE is illustrated in Fig. 21 while some selected design points are shown as the networks in Fig. 22. The star marker in Fig. 22 is the design solution currently used and obtained from an experienced

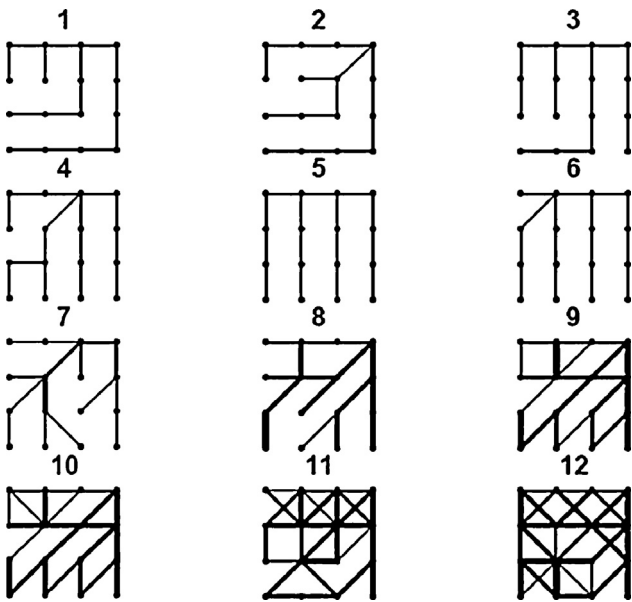


Fig. 16. Networks corresponding to the selected design point in Fig. 15.

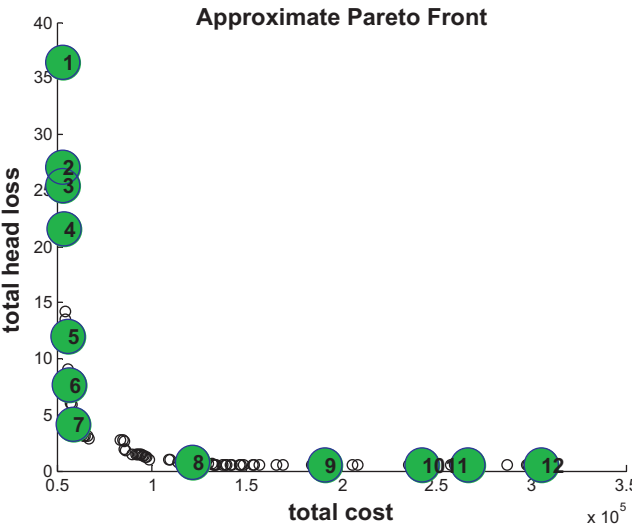


Fig. 17. Best Pareto front of OPT2 from using SPEA2.

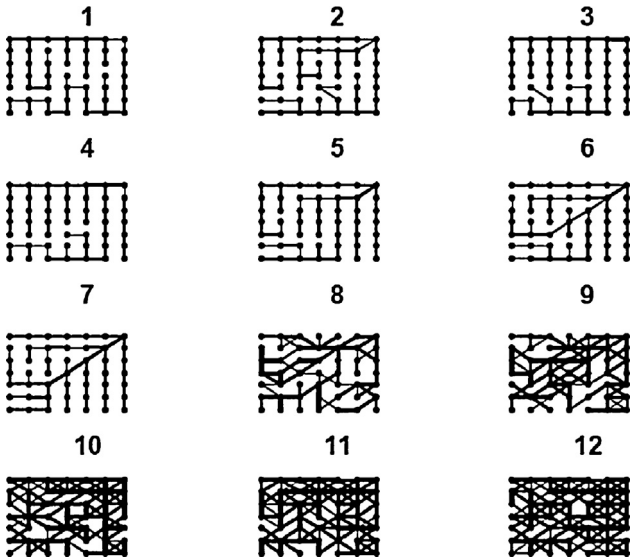


Fig. 18. Networks corresponding to the selected design points in Fig. 17.

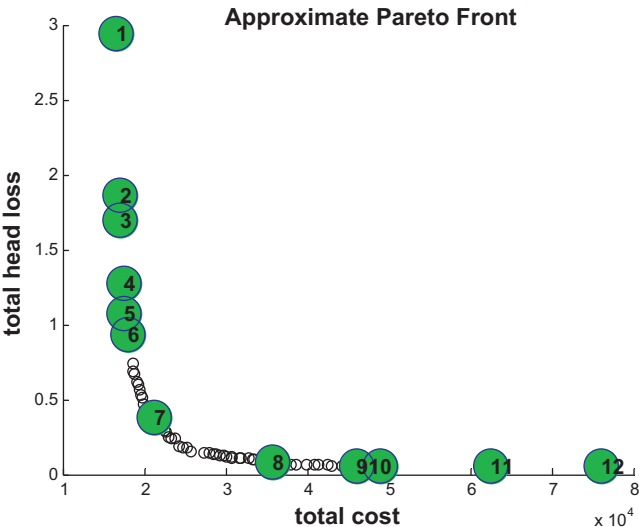


Fig. 15. Best Pareto front of OPT1 from using RPBIL-DE.

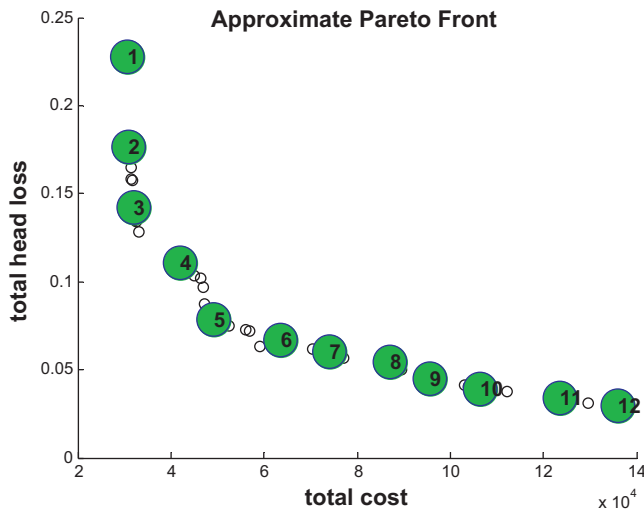


Fig. 19. Best Pareto front of OPT4 from using NSGAI1.

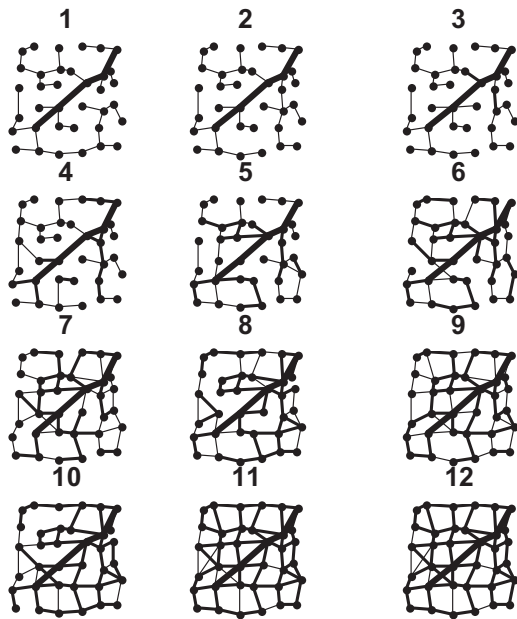


Fig. 20. Networks corresponding to the selected design points in Fig. 19.

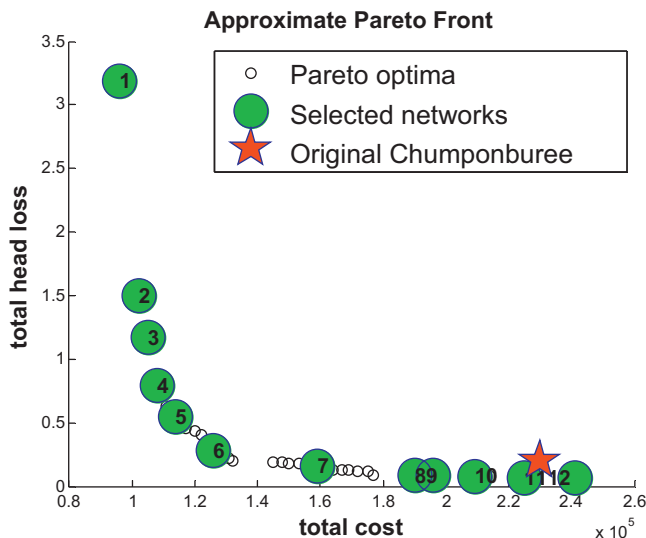


Fig. 21. Best Pareto front of OPT3 from using RPBIL-DE.

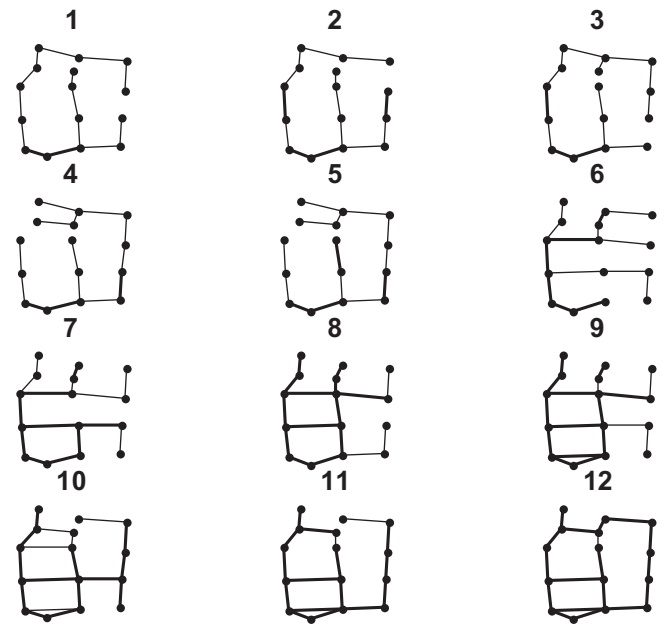


Fig. 22. Networks corresponding to the selected design points in Fig. 21.

designer. It is shown that the current solution of the Chumponburee network is dominated by a number of design solutions obtained from the proposed design process. Moreover, it can be observed from Figs. 16, 18, 20 and 22 that a network topology with less cost and higher head loss comprises branches rather than loops while, in contrast, a network having higher cost, bigger pipe diameters, and smaller head loss tends to have loops rather than branches. This result is reasonable and agrees well with common knowledge in pipe networks since a network consisting of bigger pipes and more loops will have lower total head loss.

7. Conclusions and discussion

NRT is proposed to cope with difficulties in the topological optimization process. It is an efficient numerical scheme for dealing with an illegitimate pipe network topology in the conceptual design of WDNs. NRT helps combine and achieve the conceptual and preliminary design of a water distribution network within one evolutionary optimization run. From the numerical experiment, it can be concluded that NRT can be applied for simultaneous topology and sizing optimization of water distribution networks efficiently and effectively. It can be implemented for practical networks, for which designers can define their own ground networks with realistic restrictions. The technique results in many-to-one objective and constraint functions. With the use of NRT, any multiobjective evolutionary algorithm can be employed to solve pipe network topological optimization. For the proposed hybrid RPBIL-DE for solving simultaneous topology and sizing optimization of pipe networks, it is shown that the algorithm is among the top performer MOEAs according to the hypervolume indicator. The method gives the best convergence rate for the cases of design problems with lower number of design variables. It is also the overall best multi-objective optimizer for this type of optimization problem whereas the second best is SPEA2. It should be noted that this comparative performance study is based on constant optimization parameter settings. Investigation on adaptive optimization parameters will be conducted in the future.

With the use of the proposed optimization strategy, a set of Pareto optimal solutions can be obtained and a designer can apply decision making techniques to select an appropriate network

solution. The selected network can then be refined in the detailed design stage. From this preliminary study, it is open for future studies to improve this type of design process. The comparison of NRT with a penalty function technique needs to be examined. The use of other reliability indices such as the resilience index and the network resilience as an objective function requires investigation. NRT is said to be complicated, thus a simple numerical scheme that is as effective as NRT should be invented. It is also challenging if we can perform the conceptual, preliminary, and detailed design (such as valve locations and distribution) at the same time, which means we can have a ready to work network within one optimization run.

Acknowledgement

The authors are grateful for the financial support from the Thailand Research Fund (TRF).

References

- [1] D.A. Savic, G.A. Walters, Genetic algorithms for the least-cost design of water distribution networks, *Journal of Water Resources Planning and Management* 123 (2) (1997) 67–77.
- [2] T.D. Prasad, N.S. Park, Multiobjective genetic algorithms for design of water distribution networks, *Journal of Water Resources Planning and Management* (2004) 73–82 (January–February).
- [3] A.J. Abebe, D.P. Solomatine, Application of global optimization to the design of pipe networks, in: *Proceedings of 3rd International Conference on Hydrodynamics*, Copenhagen, 1998.
- [4] B.A. Tolson, H.R. Maier, A.R. Simpson, B.J. Lence, Genetic algorithms for reliability-based optimization of water distribution systems, *Journal of Water Resources Planning and Management* 130 (1) (2004) 63–72.
- [5] A.C. Zecchin, et al., Ant colony optimization applied to water distribution system design: comparative study of five algorithms, *Journal of Water Resources Planning and Management* 133 (1) (2007) 87–92.
- [6] A. Vasan, S.P. Simonovic, Optimization of water distribution network design using differential evolution, *Journal of Water Resources Planning and Management* 136 (2) (2010) 279–287.
- [7] L. Tong, G. Han, J. Qiao, Design of water distribution network via ant colony optimization, in: *2nd International Conference on Intelligent Control and Information Processing*, 2011, pp. 366–370.
- [8] I. Montalvo, J. Izquierdo, R. Perez, M.M. Tunb, Particle swarm optimization applied to the design of water supply systems, *Computers and Mathematics with Applications* 56 (2008) 769–776.
- [9] C. Gil, R. Banos, J. Ortega, A.L. Marquez, A. Fernandez, M.G. Montoya, Ant colony optimization for water distribution network design: a comparative study, *Lecture Notes in Computer Science* 6692 (2011) 300–307.
- [10] E. Todini, Looped water distribution networks design using a resilience index based heuristic approach, *Urban Water* 2 (2) (2000) 115–122.
- [11] L.S. Vamvakiridou-Lyroudia, Fuzzy multiobjective optimization of water distribution networks, *Journal of Water Resources Planning and Management* 131 (6) (2005) 467–476.
- [12] R. Farmani, G.A. Walters, D.A. Savic, Trade-off between total cost and reliability for Anytown network, *Journal of Water Resources Planning and Management* (2005) 161–171 (May–June).
- [13] M.D. Atiquzzaman, S.Y. Liong, X. Yu, Alternative decision making in water distribution network with NSGA-II, *Journal of Water Resources Planning and Management* 132 (2006) 122–126.
- [14] T.M. Walski, The wrong paradigm – Why water distribution doesn't work? *Journal of Water Resources Planning and Management* 127 (4) (2001) 203–205.
- [15] O. Giustolisi, Z. Kapelan, D.A. Savic, An algorithm for automatic detection of topological changes in water distribution networks, *Journal of Hydraulic Engineering* 134 (4) (2008) 435–446.
- [16] S.L. Yang, N.S. Hsu, P. Louie, W.W.-G. Yeh, Water distribution network reliability: connectivity analysis, *Journal of Infrastructure Systems* 2 (2) (1996) 54–64.
- [17] C. Xu, I.C. Goulter, Probabilistic model for water distribution reliability, *Journal of Water Resources Planning and Management* 124 (4) (1998) 218–228.
- [18] C. Xu, I.C. Goulter, Reliability-based optimal design of water distribution network, *Journal of Water Resources Planning and Management* 125 (6) (1999) 352–362.
- [19] R. Gargano, D. Pianese, Reliability as tool for hydraulic network planning, *Journal of Hydraulic Engineering* 126 (5) (2000) 354–364.
- [20] T.T. Tanyimboh, M. Tabesh, R. Burrows, Appraisal of source head methods for calculating reliability of water distribution networks, *Journal of Water Resources Planning and Management* 127 (4) (2001) 206–213.
- [21] P. Kalungi, T. Tanyimboh, Redundancy model for water distribution systems, *Reliability Engineering and System Safety* 82 (3) (2003) 275–286.
- [22] T.M. Walski, J. Gessler, T.M. Walski, J. Gessler, Discussion of the paper improved design of 'Anytown' distribution network using structured messy genetic algorithms, *Urban Water* 1 (2) (1999) 265–268, 1999.
- [23] A. Ostfeld, Water distribution systems connectivity analysis, *Journal of Water Resources Planning and Management* 131 (1) (2005) 58–66.
- [24] T.M. Walski, J.S. Weiler, T. Culver, Using criticality analysis to identify impact of valve location, in: *8th Annual Water Systems Analysis Symposium*, Cincinnati, USA August 27–30, 2006, pp. 1–9.
- [25] H. Jun, G.V. Loganathan, Valve-controlled segments in water distribution systems, *Journal of Water Resources Planning and Management* 133 (2) (2007) 145–155.
- [26] H. Jun, G.V. Loganathan, A.K. Deb, W. Greyman, J. Snyder, Valve distribution and impact analysis in water distribution systems, *Journal of Environmental Engineering* 133 (8) (2007) 790–799.
- [27] O. Giustolisi, Z. Kapelan, D.A. Savic, Extended period simulation analysis considering valve shutdowns, *Journal of Water Resources Planning and Management* 134 (4) (2008) 527–537.
- [28] T. Ahonen, D.A. Sigüenza-Tortosa, H. Bin, J. Nurmi, Topology optimization for application-specific networks-on-chip, in: *Proceedings of the 2004 International Workshop on System Level Interconnect Prediction*, Paris, France, 2004, pp. 53–60.
- [29] D. Alderson, W. Willinger, L. Li, J.J. Doyle, An optimization-based approach to modelling internet topology, *Operations Research/Computer Science Interfaces Series* 33 (2006) 101–136.
- [30] M.P. Cuellar, M. Delgado, M.C. Pegalajar, Topology optimization and training of recurrent neural networks with Pareto-based multi-objective algorithms: an experimental study, *Lecture Notes in Computer Science* 4507 (2007) 359–366.
- [31] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization, in: *Evolutionary Methods for Design, Optimization and Control*, Barcelona, Spain, 2002.
- [32] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Transaction on Evolutionary Computation* 6 (2) (2002) 182–197.
- [33] M. Reyes-Sierra, C.A. Coello Coello, Multi-objective particle swarm optimizers: a survey of the state-of-the-art, *International Journal of Computational Intelligence Research* 2 (3) (2006) 287–308.
- [34] T. Aittokoski, K. Miettinen, Efficient evolutionary approach to approximate the pareto-optimal set in multiobjective optimization, *UPS-EMOA, Optimization Method and Software* 25 (2010) 841–858.
- [35] S. Sivasubramani, K.S. Swarup, Environmental/economic dispatch using multi objective harmony search algorithm, *Electric Power Systems Research* 81 (2011) 1778–1785.
- [36] S. Bureerat, Improved population-based incremental learning in continuous spaces, *Advances in Intelligent and Soft Computing* 96 (2011) 77–86.
- [37] J.A. Madeiraa, H.C. Rodriguesa, H. Pina, Multiobjective topology optimization of structures using genetic algorithms with chromosome repairing, *Structure and Multidisciplinary Optimization* 32 (2006) 31–39.
- [38] S. Baluja, Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning, *Technical Report CMU-CS.95.163*, Carnegie Mellon University, 1994.
- [39] A.R. Yildiz, Hybrid immune-simulated annealing algorithm for optimal design and manufacturing, *International Journal of Materials and Product Technology* 4 (3) (2009) 217–226.
- [40] L.A. Rossman, EPANET2 User Manual, United States Environmental Protection Agency, 2000.
- [41] S. Srisomporn, S. Bureerat, Geometrical design of plate-fin heat sinks using hybridization of MOEA and RSM, *IEEE Transaction on Components and Packaging Technologies* 31 (2008) 351–360.
- [42] K. Deb, A. Pratap, T. Meyarivan, Constrained test problems for multi-objective evolutionary optimisation, *Lecture Notes in Computer Science* 1993 (2001) 284–298.

Research Article

Antioptimisation of Trusses Using Two-Level Population-Based Incremental Learning

Phinit Tontragunrat and Sujin Bureerat

Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand

Correspondence should be addressed to Sujin Bureerat; sujbur@kku.ac.th

Received 14 December 2012; Accepted 18 March 2013

Academic Editor: Xiaojun Wang

Copyright © 2013 P. Tontragunrat and S. Bureerat. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Practical optimum design of structures often involves parameters with uncertainties. There have been several ways to deal with such optimisation problems, and one of the approaches is an antioptimisation process. The task is to find the optimum solution of common design variables while simultaneously searching for the worst case scenario of those parameters with uncertainties. This paper proposed a metaheuristic based on population-based incremental learning (PBIL) for solving antioptimisation of trusses. The new algorithm is called two-level PBIL which consists of outer and inner loops. Five antioptimisation problems are posed to test the optimiser performance. The numerical results show that the concept of using PBIL probability vectors for handling the antioptimisation of truss is powerful and effective. The two-level PBIL can be considered a powerful optimiser for antioptimisation of trusses.

1. Introduction

A truss structure is one of the most widely used engineering structures due to its simplicity and low cost for construction and design. Such a structure can be used in many engineering applications such as bridges, roofs, and towers. Throughout its history, a great deal of research work on truss design optimisation has been investigated, for example, [1–6]. The design process can be categorised as topology, shape, and sizing optimisation. The optimisers used for tackling truss design problems could be the methods with and without using function derivatives. Some of the most popular optimisers are evolutionary algorithms (EAs), which can deal with truss design having single or multiple objective functions. Using EAs, although having low convergence rate and lacking of search consistency, is popular and advantageous since they are simple to use and robust and can deal with any kind of design problems particularly truss design with discrete design variables.

Nevertheless, practical design of trusses and other structures often has inevitable difficulties when uncertainties are involved. For example, it is always difficult to define

certain applied loads on structures and material properties. This implies that the real world design problem of trusses will include parameters with uncertainties and it becomes reliability-based optimisation. The simplest way to handle this is by adding the factor of safety to structural constraints. Alternatively, multiobjective design problems which minimise cost and maximise a parameter indicating structural reliability (e.g., structural compliance and natural frequency) are posed and solved for a set of the Pareto optimum structures [6–8]. The more popular approach is the use of reliability-based optimisation techniques, which involve probabilistic design constraints. For example, the use of gradient-based optimisers [9] and evolutionary algorithms [10] for this design process has been reported.

Apart from the aforementioned approaches, a simple but effective way to deal with optimisation with uncertainties is to use the so-called antioptimisation [11]. In the antioptimisation process, there are two sets of design variables, common design variables and ones with uncertainties predefined in certain intervals. One optimisation run has two simultaneous tasks. The first task is to find the solution for the interval variables giving the worst case scenario for the structure,

Population 1				Population 2				Population 3			
0	0	1	1	1	1	1	0	0	0	0	1
1	1	1	0	0	1	0	1	1	1	0	0
0	1	0	1	1	0	1	1	0	0	0	1
1	0	0	0	0	0	0	0	0	1	0	1

Probability vectors

[0.5, 0.5, 0.5, 0.5]	[0.5, 0.5, 0.5, 0.5]	[0.25, 0.5, 0, 0.75]
----------------------	----------------------	----------------------

FIGURE 1: Probability vectors and their corresponding populations.

Initialisation $P_i = 0.5$, $k = 0$, $\mathbf{b} = \{\}$
Main procedure
(1) Generate a binary population $\mathbf{B}$ according to P_i .
(2) Perform function evaluations $f(\mathbf{B})$ and find the best solution $\mathbf{b}$ from $\mathbf{B} \cup \mathbf{b}$.
(3) Update P_i using (5).
(4) Set P_i to $[0.1, 0.9]$ in cases that it is out of the interval.
(5) If the termination condition is met, stop the procedure; otherwise, set $k = k + 1$ and go to (1).

ALGORITHM 1: Population-based incremental learning.

while the second task is finding the optimum solution for the common design variables. Optimisers for this propose have been developed, for example, two-species genetic algorithm (GA) [12], and hybrid genetic algorithm (HybridGA) [13]. From the literature, it is found that using such evolutionary algorithms for solving an antioptimisation problem is somewhat time-consuming.

This paper is concerned with developing a metaheuristic (MH) for tackling antioptimisation of trusses. The work focuses on the metaheuristic because it can deal with discrete design variables which are usually employed in real-world truss design. Population-based incremental learning (PBIL) is chosen and adapted to deal with truss antioptimisation leading to five variants of PBIL. The five methods and HybridGA are then implemented on five antioptimisation problems of 2D trusses. Optimum results show that PBIL is a powerful optimiser for truss antioptimisation and it is superior to HybridGA.

2. Antioptimisation Problem

A typical single-objective truss optimisation is mass minimisation subject to structural design constraints, which can be expressed as

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}), \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m, \end{aligned} \quad (1)$$

where $\mathbf{x}$ is a vector sized $n_1 \times 1$ of common design variables, f is structural mass, and g_i are inequality constraints for

safety requirements. When variables with uncertainties are included, the problem can be written as

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}, \mathbf{y}), \\ \text{subject to} \quad & g_i(\mathbf{x}, \mathbf{y}) \leq 0, \quad i = 1, \dots, m, \end{aligned} \quad (2)$$

where $\mathbf{y} \in [\mathbf{y}_l, \mathbf{y}_u]$ is a vector sized $n_2 \times 1$ of variables with uncertainty having lower and upper bounds as $\mathbf{y}_l$ and $\mathbf{y}_u$, respectively.

For truss antioptimisation, the problem (2) is separated into two subproblems as the design problem (1) for searching for optimum design variables $\mathbf{x}$ and the design problem for finding the worst case $\mathbf{y}$ of a structure written as

$$\max_{\mathbf{y}} F(\mathbf{y}), \quad \mathbf{y} \in [\mathbf{y}_l, \mathbf{y}_u], \quad (3)$$

where F is a function indicating the maximum violation of structural constraints. For simplicity, F can be defined as

$$F(\mathbf{y}) = \max(g_i(\mathbf{x}, \mathbf{y})) \quad (4)$$

with the design point $\mathbf{x}$ being unchanged.

A traditional strategy for solving an antioptimisation problem is to use a two-level approach. The search process starts with initial solutions for $\mathbf{x}$ and $\mathbf{y}$. In one iteration, the method searches for the optimum solution of $\mathbf{x}$ based on the problem (1) by fixing the value of $\mathbf{y}$. Then, use the updated values of $\mathbf{x}$ for finding the worst solution of $\mathbf{y}$ based on the problem (3). The process is repeated until the termination conditions are fulfilled.

3. PBIL for Antioptimisation

Over the years, there have been a significant number of metaheuristic algorithms developed for a wide variety of real-world applications, for example, [14–18]. PBIL is chosen for this work because it is a simple but powerful MH and suits well with the concept of antioptimisation. PBIL was first proposed by Baluja in 1994 [19]. The method is based upon a simple estimation of distribution algorithm and searches for an optimum using binary strings. Unlike GA which keeps a set of binary design solutions (better known as a population), PBIL uses the so-called probability vector to represent a binary population. Figure 1 shows some probability row vectors which will be used to produce binary population matrices where one row of the matrix represents a binary design solution. The element P_i of the vector determines the probability of having the string “1” on the i th column of the binary population. From the figure, it can be seen that one probability vector can produce various binary populations. As the optimisation proceeds, the probability vector is updated iteratively leading to an optimum solution. A typical computational procedure of PBIL is given in Algorithm 1. Initially, the probability vector is set as $P_i = 0.5$. Then, the corresponding population is generated where the best binary

Initialisation $P_{x,i} = 0.5$, $P_{y,i} = 0.5$, and find $\mathbf{b} = \{\mathbf{b}_x \ \mathbf{b}_y\}$ and $\mathbf{w} = \{\mathbf{w}_x \ \mathbf{w}_y\}$ from an initial population.

Main procedure

(1) For $k = 1$ to N_O (outer loop)

(2) Generate a binary population $\mathbf{B}_x$ according to $P_{x,i}$ and assign $\mathbf{w}_y$ as the second part of every solution in $\mathbf{B}_x$. The population is $\mathbf{B} = [\mathbf{B}_x \ \mathbf{w}_y]$.

(3) Perform function evaluations $f(\mathbf{B})$.

(4) Find the new best solution $\mathbf{b}$ from $\mathbf{B} \cup \mathbf{b}$ based on the objective function in (1).

(5) Update $P_{x,i}$ with $\mathbf{b}_x$ using (5).

(6) Set $P_{x,i}$ to $[0.1, 0.9]$ in cases that it is out of the interval.

(7) For $j = 1$ to N_I (inner loop)

(8) Generate a binary population $\mathbf{W}_y$ according to $P_{y,i}$ and assign $\mathbf{w}_x$ as the first part of every solution in $\mathbf{W}_y$. The population is $\mathbf{W} = [\mathbf{w}_x \ \mathbf{W}_y]$.

(9) Perform function evaluations $f(\mathbf{W})$

(10) Find the new worst solution $\mathbf{w}$ from $\mathbf{W} \cup \mathbf{w}$ based on the objective function in (3).

(11) Update $P_{y,i}$ with $\mathbf{w}_y$ using (5).

(12) Set $P_{y,i}$ to $[0.1, 0.9]$ in cases that it is out of the interval.

(13) Next j

(14) Next k

ALGORITHM 2: Population-based incremental learning for antioptimisation.

solution $\mathbf{b}$ is found. The probability vector is then updated based upon the best solution as

$$P_i^{\text{new}} = (1 - L_R) P_i^{\text{old}} + L_R b_i, \quad (5)$$

where P_i^{old} is the i th element of the probability vector from the previous iteration, P_i^{new} is the updated i th element of the probability vector, and b_i is the i th element of the best binary solution. L_R is called a learning rate which herein is set to be a uniform random number in the range of $[0.4, 0.6]$ generated anew when the probability vector update takes place. In this work, mutation is not used to modify the probability vector, but the value of P_i after being updated will be set to the range of $[0.1, 0.9]$ (if the updated value of P_i is lower than 0.1 or higher than 0.9, it will be set as 0.1 and 0.9, resp.) in order to prevent a premature convergence.

When dealing with an antioptimisation problem, the method is modified as shown in Algorithm 2. As it involves two levels of optimisation search, the developed PBIL will be named two-level population-based incremental learning. The algorithm starts with generating an initial binary population where each binary solution contains strings for design variables $\mathbf{x}$ and variables with uncertainties $\mathbf{y}$. Then, the best binary solution $\mathbf{b}$ containing two parts as $\mathbf{b}_x$ for $\mathbf{x}$ and $\mathbf{b}_y$ for $\mathbf{y}$ is found based on the design problem (1). On the other hand, the worst binary solution $\mathbf{w} = \{\mathbf{w}_x \ \mathbf{w}_y\}$ is obtained based on the problem (3). The method uses two probability vectors as $\mathbf{P}_x$ and $\mathbf{P}_y$ for representing binary populations of $\mathbf{x}$ and $\mathbf{y}$, respectively. Afterwards, a set of binary design solutions for $\mathbf{x}$ (denoted as $\mathbf{B}_x$) are created based on $\mathbf{P}_x$, while each solution in the population has its second part of a binary string for $\mathbf{y}$ as $\mathbf{w}_y$. The new best solution $\mathbf{b}$ is obtained from sorting the current population and the best solution from the previous iteration. The probability vector $\mathbf{P}_x$ is updated using the first part of the new best solution $\mathbf{b}_x$. This is the first level of the process which is said to be outer loops.

For inner loops (steps (7)–(13)) which are the search for the worst case scenario, a set of binary solutions for $\mathbf{y}$ (denoted as $\mathbf{W}_y$) is generated from the probability vector $\mathbf{P}_y$ where each solution has the first part of a binary string for $\mathbf{x}$ as $\mathbf{w}_x$ which is the first part of the current worst solution. The new worst solution $\mathbf{w}$ is then obtained based on the problem (3) by sorting the current population and the previous worst solution. The probability $\mathbf{P}_y$ is updated by using $\mathbf{w}_y$, the second part of $\mathbf{w}$. The inner loop procedure is stopped after N_I loops. Similarly, the outer loop search is terminated after N_O loops. This implies that if the population size for both inner and outer loops is N_p , the total number of function evaluations for each simulation run is $N_O(N_I + 1)N_p$. It should be noted that we use the worst solution for finding the worst case structures because it is expected to give the worst objective function values which may help accelerate PBIL to the worst case. In the case studies, we will examine the use of best and worst solutions for searching the worst case scenario.

4. Truss Design Case Studies

Five antioptimisation problems (AOPs) of three truss structures will be used to test the proposed algorithms. The first truss is called a 2-bar truss shown in Figure 2. The structure is subject to a vertical load F with uncertainty at node 3. The second structure is named a 10-bar truss [20] shown in Figure 3. The structure consists of 6 nodes and 10 bars where nodes 1 and 2 are fixed and nodes 4 and 6 are subject to external forces F_1 and F_2 , respectively. The third structure is called a 25-bar truss as displayed in Figure 3. The truss has 12 nodes and 25 bars where nodes 1 and 12 are fixed. The structure is applied by loads F_1 , F_2 , F_3 , F_2 , and F_1 at nodes 2, 4, 6, 8, and 10, respectively. The design problems can be detailed as follows.

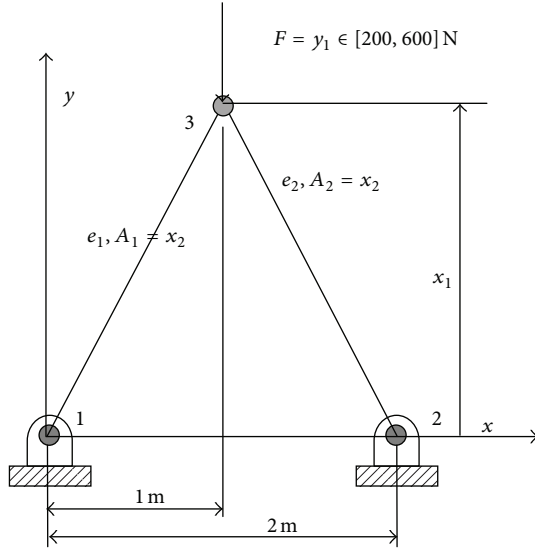


FIGURE 2: 2-bar truss.

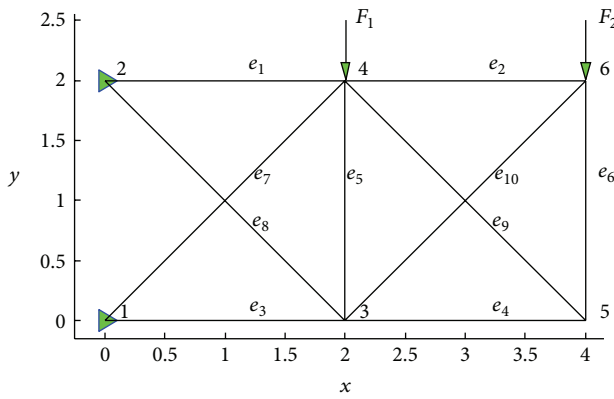


FIGURE 3: 10-bar truss.

AOP1: Optimisation of the 2-Bar Truss. The design problem can be written as:

$$\begin{aligned}
 \min_{\mathbf{x}} \quad & f(\mathbf{x}, \mathbf{y}), \quad \max_{\mathbf{y}} F(\mathbf{x}, \mathbf{y}) \\
 \text{Subject to} \quad & \sigma_{\max}(\mathbf{x}, \mathbf{y}) - \sigma_{yt} \leq 0, \text{ N/m}^2, \\
 & P_i - P_{cr} \leq 0, \text{ N}, \\
 & x_1 \in [0.5, 1.5], \text{ m}, \\
 & x_2 = A_1 = A_2 \in [0.001, 0.01], \text{ m}, \\
 & 200 \leq y_1 = F \leq 600, \text{ N},
 \end{aligned} \tag{6}$$

where A_i is the cross-sectional area of the i th bar element. $\sigma_{\max}$ is the maximum stress on truss elements. P_i is the axial load acting on the i th element, while P_{cr} is the critical buckling load for the element. The structures for all 5 test problems are made up of material with the Young modulus

$E = 200 \times 10^9 \text{ N/m}^2$, yield stress $\sigma_{yt} = 235 \times 10^6 \text{ N/m}^2$, and density $\rho = 7800 \text{ kg/m}^3$. The common design variables are shape and sizing variables. The external force F is set to be variables with uncertainties. The task is to find the external force that gives the worst case scenario for the structure while simultaneously trying to perform mass minimisation.

AOP2: Sizing Optimisation of the 10-Bar Truss. The second design problem can be written as:

$$\begin{aligned}
 \min_{\mathbf{x}} \quad & f(\mathbf{x}, \mathbf{y}), \quad \max_{\mathbf{y}} F(\mathbf{x}, \mathbf{y}), \\
 \text{subject to} \quad & \sigma_{\max}(\mathbf{x}, \mathbf{y}) - \sigma_{yt} \leq 0, \text{ N/m}^2, \\
 & P_i - P_{cr} \leq 0, \text{ N}, \\
 & U_{\max} - 0.005 \leq 0, \text{ m}, \\
 & x_i = d_i \in \{0.04, 0.05, 0.06, 0.07, 0.08, 0.09, \\
 & \quad 0.10, 0.20\}, \text{ m}, \\
 & \{75, 150\}^T \leq \mathbf{y} = \{F_1, F_2\}^T \leq \{80, 175\}^T, \text{ N},
 \end{aligned} \tag{7}$$

where d_i is the diameter of the i th bar element. $U_{\max}$ is the maximum y -direction displacement of the structure. The common design variables are set to be discrete so that it is close to a practical truss design process. The external forces F_1 and F_2 are set to be variables with uncertainties. The task is to find the external forces that give the worst case scenario for the structure while, at the same time, trying to perform mass minimisation with sizing design variables.

AOP3: Sizing Optimisation of the 25-Bar Truss. The design problem can be written as:

$$\begin{aligned}
 \min_{\mathbf{x}} \quad & f(\mathbf{x}, \mathbf{y}), \quad \max_{\mathbf{y}} F(\mathbf{x}, \mathbf{y}), \\
 \text{subject to} \quad & \sigma_{\max}(\mathbf{x}, \mathbf{y}) - \sigma_{yt} \leq 0, \text{ N/m}^2, \\
 & P_i - P_{cr} \leq 0, \text{ N}, \\
 & U_{\max} - 0.005 \leq 0, \text{ m}, \\
 & x_i \in \{0.06, 0.07, 0.08, 0.09, 0.10, 0.15, \\
 & \quad 0.20, 0.25\} \text{ m}; \quad i = 1, \dots, 13, \\
 & \{150, 150, 150, 199.00 \times 10^9\}^T \\
 & \leq \mathbf{y} = \{F_1, F_2, F_3, E\}^T \\
 & \leq \{300, 300, 300, 201.00 \times 10^9\}^T.
 \end{aligned} \tag{8}$$

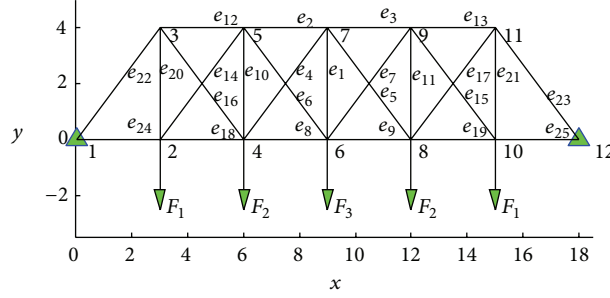


FIGURE 4: 25-bar truss.

TABLE 1: Optimum results of AOP1.

MHs	HybridGA		PBILb5		PBILw5		PBILw5m		PBILw1		PBILw10	
	Avrg	Stddev	Avrg	Stddev	Avrg	Stddev	Avrg	Stddev	Avrg	Stddev	Avrg	Stddev
x_1	0.5048	0.0242	0.5613	0.1011	0.6247	0.1176	0.6333	0.1587	0.6344	0.1558	0.5914	0.1302
x_2	0.0029	0.0001	0.0038	0.0003	0.0037	0.0004	0.0038	0.0005	0.0038	0.0004	0.0038	0.0004
y_1	200.0000	0.0000	399.1398	130.0818	370.3226	142.7784	400.8602	164.3949	405.1613	157.4620	424.0860	151.7861
$f(\mathbf{x}, \mathbf{y})$	49.9355	1.6789	67.2216	4.8174	68.2111	5.5543	70.2185	5.9409	69.9382	4.6033	69.4287	5.3346
$\max g_i$	-0.0018	0.0024	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000

In this case, there are uncertainties in applied loads and material property. The sizing design variables are set to have a symmetric structure. Thus, we have

- (i) x_1 for d_1 ,
- (ii) x_2 for d_2 and d_3 ,
- (iii) x_3 for d_4 and d_5 ,
- (iv) x_4 for d_6 and d_7 ,
- (v) x_5 for d_8 and d_9 ,
- (vi) x_6 for d_{10} and d_{11} ,
- (vii) x_7 for d_{12} and d_{13} ,
- (viii) x_8 for d_{14} and d_{15} ,
- (ix) x_9 for d_{16} and d_{17} ,
- (x) x_{10} for d_{18} and d_{19} ,
- (xi) x_{11} for d_{20} and d_{21} ,
- (xii) x_{12} for d_{22} and d_{23} ,
- (xiii) x_{13} for d_{24} and d_{25} .

AOP4: Shape and Sizing Optimisation of the 10-Bar Truss. The fourth design problem is the same problem as AOP2 with additional 4 shape design variables for $\mathbf{x}$. The shape variables determine the positions of nodes 3 and 5 in both x and y directions. The design variables are used to modify the original positions (shown in Figure 3) of nodes 3 and 5. The bound constraints are set as $-0.75 \text{ m} \leq x_i \leq 0.75 \text{ m}$ for $i = 11, \dots, 14$.

AOP5: Shape and Sizing Optimisation of the 25-Bar Truss. The fifth design problem is the same problem as AOP2 with additional 3 shape design variables for $\mathbf{x}$. The shape variables determine the y -direction positions of nodes 3, 5, 7, 9, and

11, where x_{14} is set for the positions of nodes 3 and 11, x_{15} is set for the positions of nodes 5 and 9, and x_{16} is set for the positions of node 7. The design variables are used to modify the original positions (shown in Figure 4) of those nodes. The bound constraints are set as $-3.00 \text{ m} \leq x_i \leq 3.00 \text{ m}$ for $i = 14, \dots, 16$.

By defining a few set of design parameter settings, various versions of PBIL can be obtained as follows.

- (i) PBILb5: this optimiser uses $\mathbf{b}_x$ from the best solution $\mathbf{b}$ instead of $\mathbf{w}_x$ from the worst solution $\mathbf{w}$ in step (8) of Algorithm 2. N_I is set to be 5.
- (ii) PBILw5: this is PBIL detailed in Algorithm 2, where N_I is set to be 5.
- (iii) PBILw5m: this is PBILw5 with GA mutation. Having generated binary populations in steps (2) and (8), The GA mutation is applied to the populations, where each solution has the probability of being mutated as 0.25.
- (iv) PBILw1: this is PBIL detailed in Algorithm 2, where N_I is set to be 1.
- (v) PBILw10: this is PBIL detailed in Algorithm 2, where N_I is set to be 10.

For AOP1, the population size is set to be 40, whereas the total number of function evaluations for stopping the search procedure is 2,400. For other design problems, the population size is set to be 100, whereas the total number of function evaluations for stopping the search procedure is 15,000. PBIL with various N_I values is set to examine the effect of the inner and outer loops on the search performance. The GA mutation is employed under the assumption that it may help improving PBIL convergence rate. Each optimiser is used to solve the antioptimisation problems 30 runs. Since PBIL cannot

TABLE 2: Optimum results of AOP2.

MHs	HybridGA		PBILb5		PBILw5		PBILw5m		PBILw1		PBILw10	
	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev
x_1	0.0453	0.0073	0.0500	0.0000	0.0500	0.0000	0.0500	0.0000	0.0500	0.0000	0.0503	0.0018
x_2	0.0463	0.0096	0.0493	0.0025	0.0500	0.0026	0.0497	0.0032	0.0477	0.0043	0.0543	0.0063
x_3	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0807	0.0025
x_4	0.0610	0.0040	0.0603	0.0018	0.0600	0.0000	0.0603	0.0018	0.0600	0.0000	0.0650	0.0068
x_5	0.0500	0.0087	0.0493	0.0025	0.0500	0.0026	0.0500	0.0000	0.0477	0.0043	0.0520	0.0041
x_6	0.0607	0.0025	0.0613	0.0051	0.0610	0.0040	0.0607	0.0025	0.0600	0.0000	0.0627	0.0064
x_7	0.0810	0.0031	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0803	0.0018
x_8	0.0517	0.0099	0.0503	0.0018	0.0500	0.0000	0.0500	0.0000	0.0500	0.0000	0.0517	0.0038
x_9	0.0427	0.0052	0.0493	0.0025	0.0500	0.0000	0.0500	0.0000	0.0483	0.0038	0.0493	0.0025
x_{10}	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0800	0.0000	0.0803	0.0018
y_1	74.7849	1.1783	77.5938	1.1768	80.0000	0.0000	80.0000	0.0000	80.0000	0.0000	80.0000	0.0000
y_2	171.5082	13.5158	165.3552	7.5152	175.0000	0.0000	175.0000	0.0000	174.9984	0.0089	175.0000	0.0000
$f(\mathbf{x}, \mathbf{y})$	558.7238	30.9474	567.8900	9.9080	568.7844	7.3917	568.2535	6.2552	559.2016	7.1018	594.2857	20.6724
$\max g_i$	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000

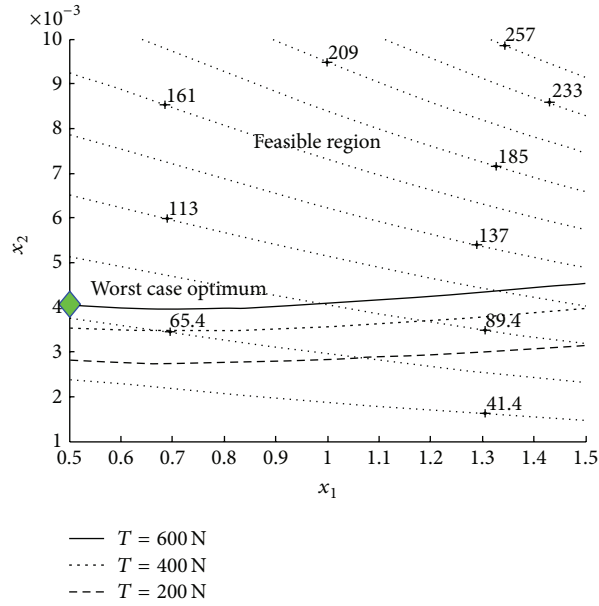


FIGURE 5: Optimum of AOP1.

deal with design constraints directly, the penalty function technique based on the fuzzy set theory [21] is employed. It should be noted that we have tested using several penalty function techniques earlier, and the fuzzy set theory gives the best results. The metaheuristics employed do not have reliable termination criterion, therefore, their search will be stopped based on the maximum number of function evaluations.

Furthermore, in order to verify the performance of the proposed algorithms, an existing evolutionary algorithm for antioptimisation called a hybrid genetic algorithm [13] will be employed to solve the test problems and compared with the various PBILs. For this performance test, HybridGA uses real codes for design variables. The method of Hooke and Jeeves is used for searching the worst case. The number of

good solutions taken for mutation is set to be 10. The roulette wheel selection method is used to select design solutions for crossover operation. The optimiser is terminated when the maximum number of function evaluations set for PBILs in the previous paragraph is reached.

5. Optimum Results

The optimum results of AOP1 are given in Table 1. Figure 5 displays the contour of the objective function and the boundaries of buckling constraints with $F = 200$ N, 400 N, and 600 N. It should be noted that the stress constraints do not affect the feasible region for this design case. From the figure, it is seen that the buckling constraint where $F = 600$ N

TABLE 4: Optimum results of AOP4.

MHs	HybridGA		PBILb5		PBILw5		PBILw5m		PBILw1		PBILw10	
	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev	Avrg	Stdev
x_1	0.0467	0.0084	0.0517	0.0053	0.0520	0.0041	0.0513	0.0043	0.0490	0.0031	0.0533	0.0084
x_2	0.0477	0.0107	0.0510	0.0040	0.0503	0.0018	0.0513	0.0035	0.0507	0.0045	0.0543	0.0082
x_3	0.0833	0.0048	0.0867	0.0055	0.0863	0.0067	0.0867	0.0048	0.0860	0.0050	0.0873	0.0069
x_4	0.0603	0.0049	0.0577	0.0082	0.0607	0.0098	0.0577	0.0068	0.0563	0.0076	0.0610	0.0099
x_5	0.0570	0.0070	0.0543	0.0068	0.0590	0.0092	0.0533	0.0071	0.0553	0.0078	0.0583	0.0105
x_6	0.0613	0.0043	0.0603	0.0072	0.0617	0.0079	0.0603	0.0093	0.0577	0.0043	0.0653	0.0107
x_7	0.0720	0.0130	0.0783	0.0046	0.0773	0.0058	0.0797	0.0056	0.0760	0.0081	0.0793	0.0087
x_8	0.0457	0.0082	0.0507	0.0037	0.0500	0.0037	0.0507	0.0037	0.0490	0.0031	0.0523	0.0082
x_9	0.0440	0.0056	0.0510	0.0031	0.0507	0.0025	0.0497	0.0032	0.0483	0.0038	0.0507	0.0045
x_{10}	0.0787	0.0035	0.0787	0.0051	0.0797	0.0041	0.0770	0.0047	0.0780	0.0048	0.0810	0.0055
x_{11}	0.0518	0.1773	0.3132	0.2705	0.1478	0.3773	0.3319	0.3448	0.2337	0.3527	0.2345	0.4409
x_{12}	0.5991	0.1766	0.5610	0.1850	0.6211	0.1079	0.5072	0.2017	0.6345	0.1444	0.5174	0.1896
x_{13}	-0.6919	0.1381	-0.5828	0.1442	-0.5369	0.2001	-0.4271	0.2176	-0.6712	0.1039	-0.4691	0.2160
x_{14}	0.6804	0.1490	0.5697	0.1723	0.5550	0.1935	0.5924	0.1489	0.6494	0.1410	0.5549	0.1754
y_1	75.3793	1.2182	78.1703	1.5065	80.0000	0.0000	80.0000	0.0000	80.0000	0.0000	80.0000	0.0000
y_2	170.9637	7.2167	166.1746	7.4693	175.0000	0.0000	175.0000	0.0000	175.0000	0.0000	175.0000	0.0000
$f(\mathbf{x}, \mathbf{y})$	479.4000	34.7275	529.8728	21.6285	535.8320	21.6881	534.5283	22.3252	501.9159	11.7035	574.2029	37.1328
$\max g_i$	-0.0047	0.0009	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000	-0.0050	0.0000

gives the most narrow feasible region and therefore leads to the worst case results. The optimum for the worst case scenario is $\mathbf{x} = \{0.5, 0.0041\}^T$ and $y = 600$ N. According to the results obtained from the various optimisers, HybridGA cannot capture the worst case for all runs, while PBILs can find the worst case for some optimisation runs. The best solutions obtained by using five versions of PBIL are the same, that is, $\mathbf{x} = \{0.5, 0.0042\}^T$ and $y = 600$ N, which can be classified as a near optimum of the design problem. Based on the worst cases being explored, PBILw10 gives the best results, while the second best is PBILw1.

The optimum results of AOP2 obtained from the various optimisers are given in Table 2. For this design problem, it is obvious that the worst possible case for the 10-bar truss is when $\{F_1, F_2\}^T = \{80, 175\}^T$ N. From the results, it can be seen that HybridGA and PBILb5 fail to explore the worst case, while other optimiser which are based on using $\mathbf{w}_x$ in step (8) of Algorithm 2 can find the worst case for almost every optimisation run. According to the mean value of objective function from 30 runs, PBILw1 gives the best results. However, it is found that PBILw1 cannot find the worst case for all 30 runs, although the values are not significant. This is due to the lower number of inner loops used to explore the worst case. In contrast, PBILw10 gives the worst results among PBILw variants because it spends less time in searching for the optimum mass. By comparing between PBILw5 and PBILw5m, the GA mutation gives insignificant improvement for PBIL.

The optimum results of AOP3 obtained from the various PBILs are given in Table 3. The worst case is $\{F_1, F_2, F_3, E\}^T = \{1000, 1000, 1000, 199.0 \times 10^9\}^T$. Similar conclusions as with the AOP2 case can be drawn except that PBILw5 is slightly

better than PBILw5m. Table 4 shows the results of AOP4. Similar to the first two cases, HybridGA and PBILb5 fail to explore the worst case scenario, while, for this case, all the versions of PBILw can capture the worst case for all runs. PBILw1 gives the best results, while the worst among PBILw versions is PBILw10. For the fifth design case as shown in Table 5, PBILw optimisers can capture the worst case scenario for almost 30 runs, while PBILb5 and HybridGA fail to find out the worst case solutions. The best optimiser is PBILw1 whereas PBILw5m is slightly better than PBILw5.

Based on the standard deviation, the variants of PBILw are rather consistent. From the constraint violation values in Tables 1–5, it can be said that the fuzzy set theory is efficient and effective for constrained truss design. Table 6 shows the total number of worst cases captured by the optimisers having solved the design problems for 30 runs. It can be seen that PBILw is effective for exploring the worst case scenarios for AOP2–5. PBILw10 fails to capture the worst case once for AOP3 and AOP5, although it spends more time to search for the worst case scenario. This can possibly be due to a premature convergence of the algorithm. For the first design problem, all the optimisers have more difficulty to search for the worst case.

The best structures obtained from running PBILw5, PBILw5m, PBILw1, and PBILw10 for AOP2, AOP3, AOP4, and AOP5 are illustrated in Figures 6, 7, 8, and 9, respectively. The structures obtained from the 4 optimisers for each design case are rather similar in size and shape to each other. In cases of the design problems AOP2–5, with 15,000 function evaluations and population size of 100, PBILw1 has a number of outer loops as $N_O = 75$ loops, while PBILw5 and PBILw10 have only 25 and 14 loops, respectively. In the test problems, the number of inner loops $N_I = 1$ is adequate to search

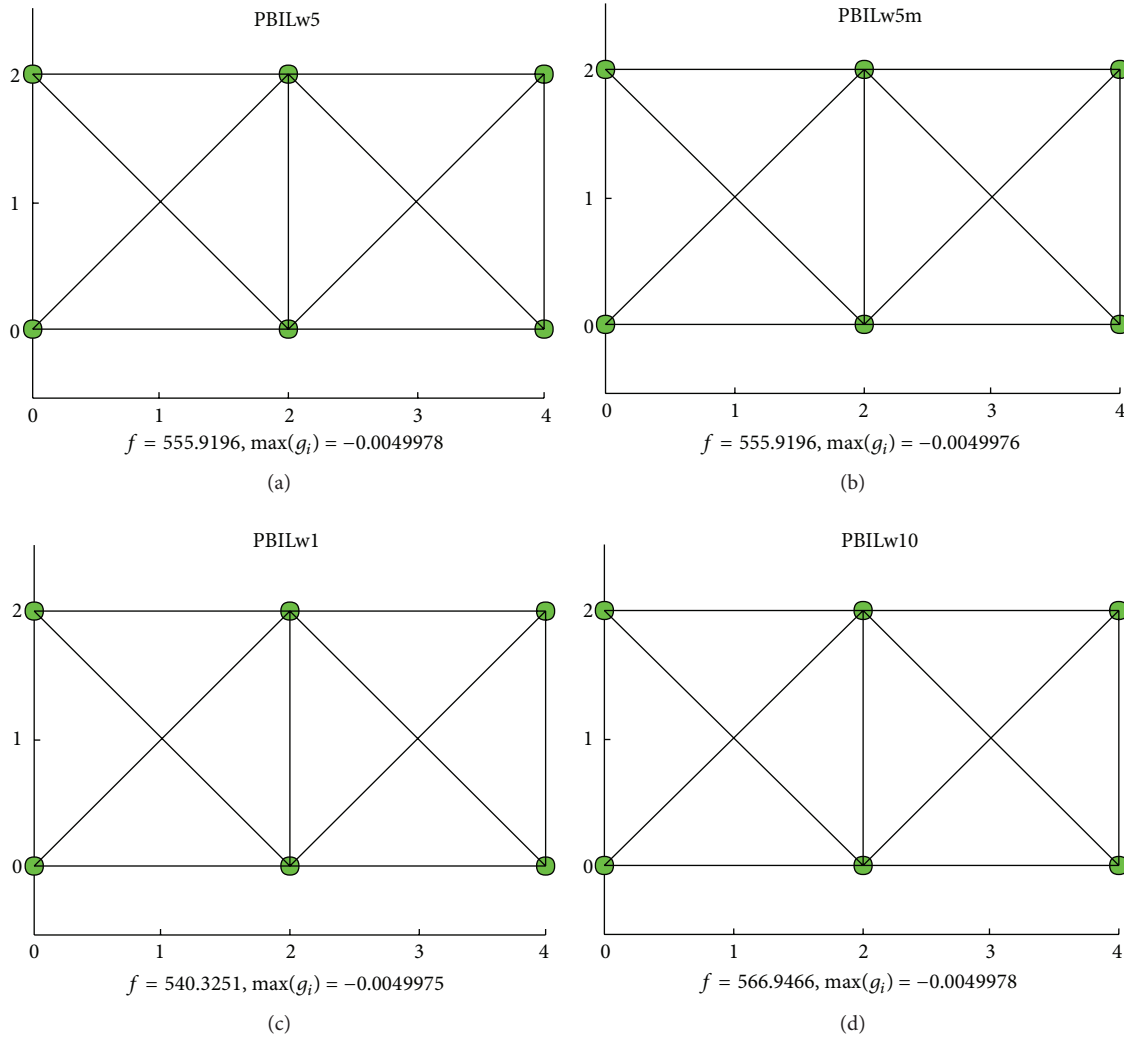


FIGURE 6: Optimum structures of AOP2 from various optimisers.

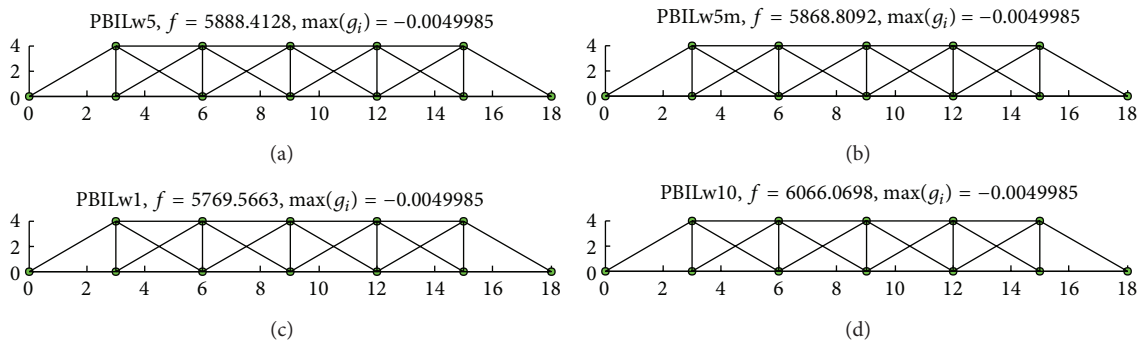


FIGURE 7: Optimum structures of AOP3 from various optimisers.

for the worst cases for all design cases; thus, PBILw1 which spends more time to search for the optimum solution of $\mathbf{x}$ gives the best results for all design cases. In order to use the proposed PBIL for antioptimisation of other engineering systems, predefining the proper value of N_I is important since

it can affect the search performance. This relies on several factors such as the size of variables $\mathbf{x}$ and $\mathbf{y}$. The hybrid genetic algorithm, on the other hand, is not efficient for the truss design problems. However, optimisation parameter settings can affect its search performance.

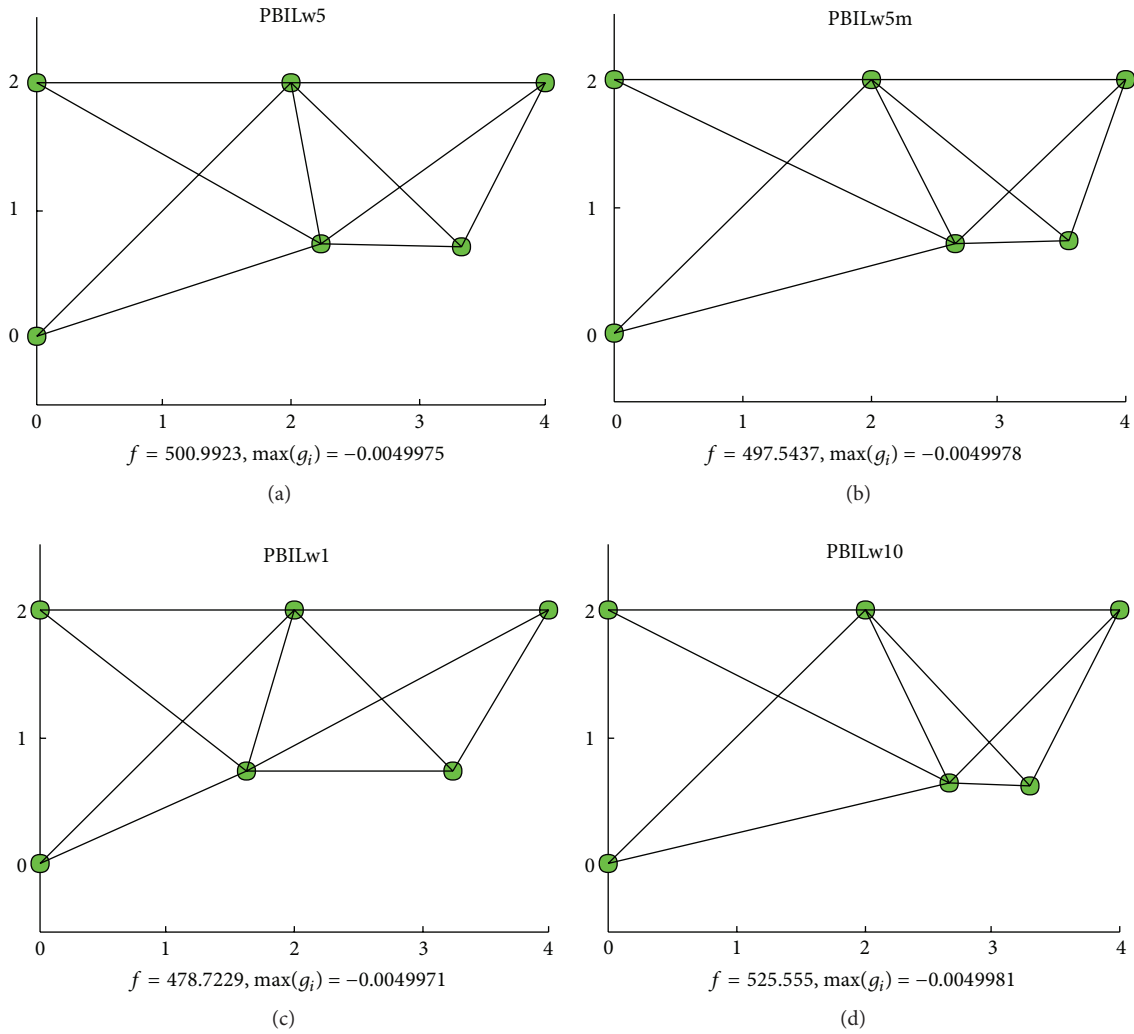


FIGURE 8: Optimum structures of AOP4 from various optimisers.

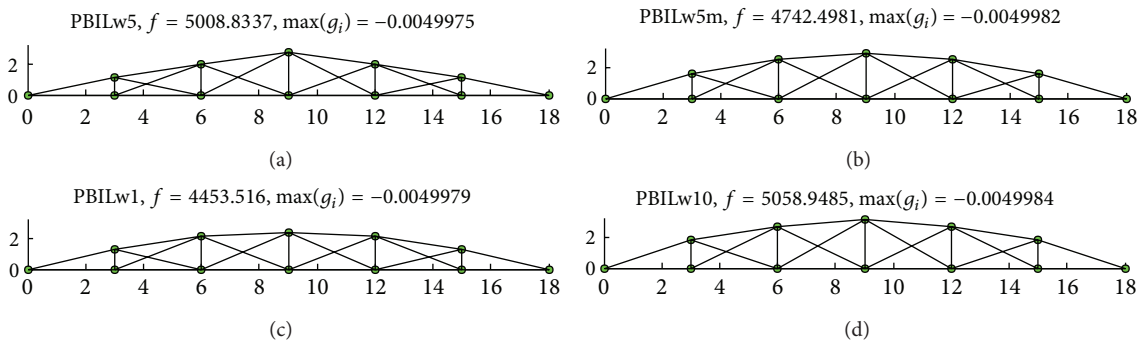


FIGURE 9: Optimum structures of AOP5 from various optimisers.

6. Conclusions and Discussion

A metaheuristic based on PBIL for antioptimisation of trusses is developed, and it is termed two-level PBIL. The method consists of two levels for two searching tasks. The first level is an outer loop used to explore the optimum of common design variables, while the inner loop is used to find the

worst case scenario of the variables with uncertainties. It is shown that when searching for the worst case, the function evaluation based on the current worst solution gives better search results than using the current best solution. The GA mutation applied to a binary population of PBIL does not have significantly impact on PBILw search performance. The results show that the proposed PBILw is an efficient optimiser

TABLE 6: Number of worst cases found by the optimisers.

MHs	AOP1	AOP2	AOP3	AOP4	AOP5
HybridGA	0	0	0	0	0
PBILb5	1	0	0	0	0
PBILw5	5	30	29	30	30
PBILw5m	8	30	27	30	30
PBILw1	8	29	24	30	30
PBILw10	9	30	29	30	29

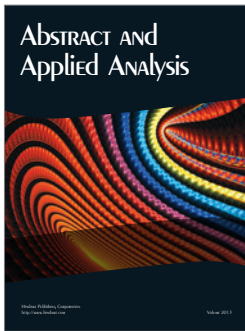
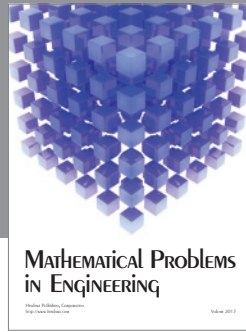
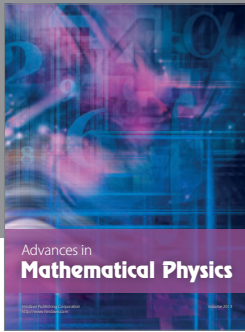
for tackling a truss antioptimisation problem. Moreover, it is superior to the hybridGA. The performance of PBILw, to a great extent, relies on the predefined number of inner loops for searching the worst case. It can be concluded that the concept of using probability vectors to represent binary populations in PBIL is powerful for dealing with antioptimisation. For future work, the hybrid approach of PBIL will be studied. Also, the applications of the two-level PBIL will be employed to tackle other antioptimisation problems.

Acknowledgments

The authors are grateful for the support from The Royal Golden Jubilee Ph. D. Program (Grant no. PHD/0145/2553) and the Thailand Research Fund (Grant no. BRG5580017).

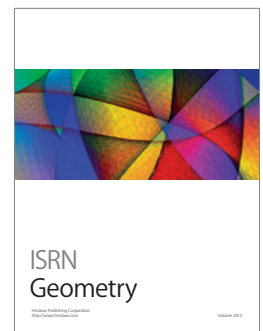
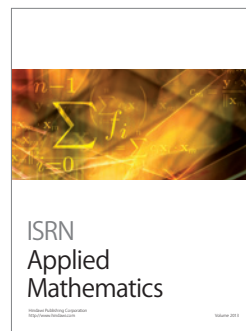
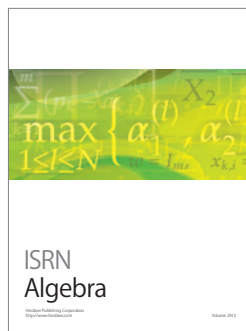
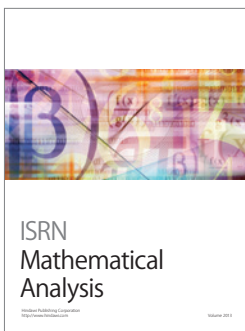
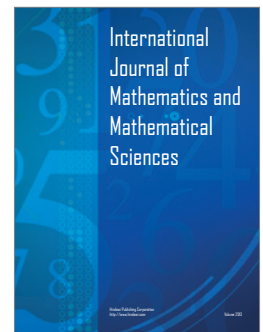
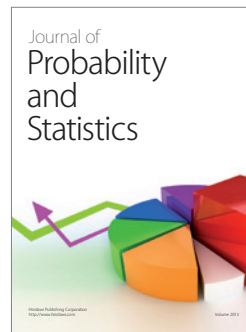
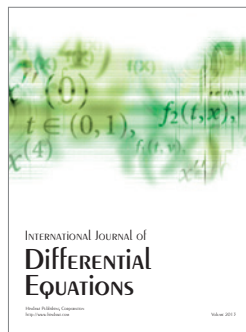
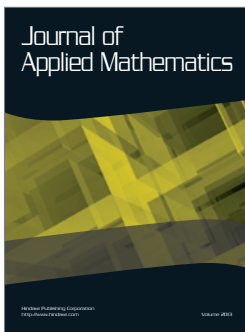
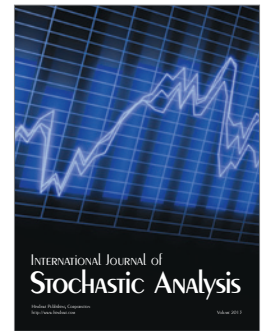
References

- [1] A. Kaveh and S. Talatahari, "Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures," *Computers and Structures*, vol. 87, no. 5-6, pp. 267-283, 2009.
- [2] P. B. Thanedar, J. S. Arora, C. H. Tseng, O. K. Lim, and G. J. Park, "Performance of some SQP algorithms on structural design problems," *International Journal for Numerical Methods in Engineering*, vol. 23, no. 12, pp. 2187-2203, 1986.
- [3] A. Kaveh and S. Talatahari, "Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures," *Computers and Structures*, vol. 87, no. 5-6, pp. 267-283, 2009.
- [4] H. M. Gomes, "Truss optimization with dynamic constraints using a particle swarm algorithm," *Expert Systems with Applications*, vol. 38, no. 1, pp. 957-968, 2011.
- [5] W. A. Bennage and A. K. Dhingra, "Single and multiobjective structural optimization in discrete-continuous variables using simulated annealing," *International Journal for Numerical Methods in Engineering*, vol. 38, no. 16, pp. 2753-2773, 1995.
- [6] N. Pholdee and S. Bureerat, "Performance enhancement of multiobjective evolutionary optimizers for truss design using an approximate gradient," *Computers and Structures*, vol. 106-107, pp. 115-124, 2012.
- [7] C. Noilublao and S. Bureerat, "Topology and sizing optimization of trusses with adaptive ground finite elements using multiobjective PBIL," *Advanced Materials Research*, vol. 308-310, pp. 1116-1121, 2011.
- [8] N. Noilublao and S. Bureerat, "Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms," *Computers and Structures*, vol. 89, pp. 2531-2538, 2011.
- [9] B. M. Adams, M. S. Eldred, and J. W. Wittwer, "Reliability-based design optimization for shape design of compliant micro-electro-mechanical systems," in *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, pp. 1042-1056, AIAA, September 2006.
- [10] K. Deb, S. Gupta, D. Daum, J. Branke, A. K. Mall, and D. Padmanabhan, "Reliability-based optimization using evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1054-1074, 2009.
- [11] M. Lombardi and R. T. Haftka, "Anti-optimization technique for structural design under load uncertainties," *Computer Methods in Applied Mechanics and Engineering*, vol. 157, no. 1-2, pp. 19-31, 1998.
- [12] G. Venter and R. T. Haftka, "Two-species genetic algorithm for design under worst case conditions," *Evolutionary Optimization*, vol. 2, no. 1, pp. 1-19, 2000.
- [13] N. Wang and Y. Yang, "Optimization of structures under load uncertainties based on hybrid genetic algorithm," in *Evolutionary Computation*, W. P. dos Santos, Ed., pp. 321-340, I-Tech, Vienna, Austria, 2009.
- [14] A. R. Yildiz, "A novel hybrid immune algorithm for global optimization in design and manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 2, pp. 261-270, 2009.
- [15] I. Durgun and A. R. Yildiz, "Structural design optimization of vehicle components using cuckoo search algorithm," *Materials Testing*, vol. 54, no. 3, pp. 185-188, 2012.
- [16] S. L. Tilahun and H. C. Ong, "Modified firefly algorithm," *Journal of Applied Mathematics*, vol. 2012, Article ID 467631, 12 pages, 2012.
- [17] X. Cai, S. Fan, and Y. Tan, "Light responsive curve selection for photosynthesis operator of APOA," *International Journal of Bio-Inspired Computation*, vol. 4, no. 6, pp. 373-379, 2012.
- [18] Z. Cui, F. Gao, Z. Cui, and J. Qu, "A second nearest-neighbor embedded atom method interatomic potential for Li-Si Alloys," *Journal of Power Sources*, vol. 207, p. 150, 2012.
- [19] S. Baluja, "Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning," Tech. Rep. CMU-CS.95.163, Carnegie Mellon University, 1994.
- [20] Z. Qiu and X. Wang, "Structural anti-optimization with interval design parameters," *Structural and Multidisciplinary Optimization*, vol. 41, no. 3, pp. 397-406, 2010.
- [21] F. Y. Cheng and D. Li, "Fuzzy set theory with genetic algorithm in constrained structural optimization," in *Proceedings of the 1st US-Japan Joint Seminar on Structural Optimization*, pp. 55-66, Advances in Structural optimization, April 1997.



Hindawi

Submit your manuscripts at
<http://www.hindawi.com>



Research Article

Simultaneous Topology, Shape, and Sizing Optimisation of Plane Trusses with Adaptive Ground Finite Elements Using MOEAs

Norapat Noilublao¹ and Sujin Bureerat²

¹ Department of Engineering Management, Faculty of Science and Technology, Rajabhat Maha Sarakham University, Maha Sarakham 44000, Thailand

² Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand

Correspondence should be addressed to Sujin Bureerat; sujbur@kku.ac.th

Received 4 October 2012; Revised 7 January 2013; Accepted 8 January 2013

Academic Editor: Sheng-yong Chen

Copyright © 2013 N. Noilublao and S. Bureerat. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper proposes a novel integrated design strategy to accomplish simultaneous topology shape and sizing optimisation of a two-dimensional (2D) truss. An optimisation problem is posed to find a structural topology, shape, and element sizes of the truss such that two objective functions, mass and compliance, are minimised. Design constraints include stress, buckling, and compliance. The procedure for an adaptive ground elements approach is proposed and its encoding/decoding process is detailed. Two sets of design variables defining truss layout, shape, and element sizes at the same time are applied. A number of multiobjective evolutionary algorithms (MOEAs) are implemented to solve the design problem. Comparative performance based on a hypervolume indicator shows that multiobjective population-based incremental learning (PBIL) is the best performer. Optimising three design variable types simultaneously is more efficient and effective.

1. Introduction

Nowadays, the use of artificial intelligence techniques in a wide variety of engineering applications has become commonplace as a means of responding to needs to deal with complicated systems which classical mathematical methods cannot solve [1, 2]. Such techniques include, for example, neural networks, fuzzy sets, and evolutionary computation [1]. Applications of evolutionary algorithms (EAs) for tackling engineering design problems have become increasingly popular due to certain advantages. The methods are simple to use, robust, and capable of dealing with almost any kind of design problem, for example, structural design [3–8], supply chain management [9], quay management [10], sensor placement [11], and manufacturing tasks [12, 13]. Design variables can be discrete, continuous, and combinatorial. They can search for global optima effectively. More attractively, the multiobjective versions of evolutionary algorithms can be used to explore a Pareto front within one optimisation run. Nevertheless, EAs have some unavoidable drawbacks as they

have a low convergence rate and a lack of consistency due to randomisation being used in a search process. As a result, there are always needs to improve EAs performance. The hybridisation of existing methods with local search techniques [14] or other EA concepts is a means to achieve such a goal [12, 15]. Also, the use of surrogate models for design problems with expensive function evaluations is an effective way to enhance EAs [16]. When using EAs to solve a new type of design problem, it is always useful to compare the search performance of a number of EAs for such a problem. This is because one particular EA may be efficient for one design problem but it is unlikely to be the best performer for other types of problems [17].

A truss structure is one of the most used structures in engineering applications. Using such a structure is said to be advantageous, since they are simple and inexpensive to construct. It can be employed for many engineering purposes, for example, a billboard structure, a factory roof structure, a bridge structure, and a wind turbine tower. In the past, the design of such a structure was usually carried out in such

a way that an initial structural configuration was formed by an experienced engineering designer for conceptual design. The classical civil engineering design approach is then applied in the preliminary and detailed design stages. In recent years, considerable research work towards design/optimisation of trusses and frames has been conducted. Thousands of research papers have been published (e.g., [18–28]). For truss optimisation, design variables can be classified as topological [23, 29–33], shape, and sizing variables [5, 7, 34–38]. In the past, a designer traditionally performs topology optimisation in the first design phase to obtain an initial structural layout. Shape and sizing optimisation are then conducted to further improve the structure. This can be called a multilevel or multistage optimisation process. Nevertheless, it has been investigated that the better design process is to perform topology, shape, and sizing optimisation simultaneously [28]. Combined topology and sizing [21, 24], shape and sizing [34–38], topology and shape [18, 20, 22], and all three variable types [19, 26, 28, 39–41] have been recently investigated.

One of the most effective design strategies for truss topology optimisation is the use of a ground element approach. The topological design process based upon ground elements can be tackled by both gradient-based [21, 31, 42] and population-based optimisers [28, 29, 33]. The gradient-based methods have advantages in that they have high convergence rate and search consistency (obtaining the same design solution when solving the same problem several times). They are powerful for a large-scale topology design problem. Nevertheless, the population-based or evolutionary approaches have other advantages, since they are more robust and do not require function derivatives for searching. Moreover, for multiobjective design cases, evolutionary algorithms have a special feature: to explore a nondominated front within one optimisation run. This, together with the capability of solving all kinds of design problems, makes the evolutionary algorithms popular among researchers and designers.

This paper presents a novel design strategy to achieve simultaneous topology shape and sizing design of a planar truss structure. A multiobjective design problem with two different sets of design variables is assigned in this work. The problem is minimising structural mass and compliance subject to stress, buckling, and compliance constraints. The first set of design variables is used to design structural topology and truss element sizing while structural shape variables are added to the second set of design variables. These two design variable sets in combination with the adaptive ground element concept result in two multiobjective design problems. MOEAs employed to tackle the problems are second version of strength Pareto evolutionary algorithm (SPEA) [43], population-based incremental learning [44], archived multiobjective simulated annealing (AMOS) [45], multiobjective particle swarm optimisation (MPSO) [46], and unrestricted population size evolutionary multiobjective optimisation algorithm (UPS-EMOA) [47]. The results obtained from using the various optimisers are compared and discussed. It is shown that the second design variables set which includes all types of design variables is superior to the first set.

The rest of the paper is organised as follows. Section 2 briefly details MOEAs. A multiobjective design problem is

expressed in Section 3. Section 4 details the decoding/encoding procedures for simultaneous topology, shape, and sizing design variables with adaptive ground elements. Design results and comparative performance are shown in Section 5 while conclusions are drawn in Section 6.

2. Multiobjective Evolutionary Algorithms

The field of multiobjective evolutionary optimisation is one of the hottest issues in evolutionary computation. The most outstanding ability of MOEAs is that they can explore a Pareto front within one simulation run. This in combination with their simplicity, robustness, and capability of dealing with all kind of variables makes this kind of optimiser more popular and attractive. MOEAs search mechanisms are based on a population of design solutions, which work in such a way that a population is evolved iteration by iteration. A matrix called an external Pareto archive is used to collect nondominated solutions iteratively. The Pareto archive is updated until a termination criterion is fulfilled. The MOEAs used for a comparative performance test in this paper are given below.

2.1. Strength Pareto Evolutionary Algorithm. SPEA was proposed by Zitzler and Thiele [48], and later its improved version SPEA2 [43]. The search procedure starts with an initial population and an external Pareto set. Fitness values are assigned to the population based upon the levels of domination and crowding. A set of solutions are then selected to a mating pool by means of a binary tournament selection operator. A new population is produced using crossover and mutation on those selected individuals. The updated external Pareto solutions are the nondominated solutions of the union set of the previous external Pareto set and the new population. In cases that the Pareto archive is full, the nearest neighbourhood technique is invoked to remove some design solutions from the archive. The Pareto archive is updated repeatedly until the termination criterion is met.

2.2. Population-Based Incremental Learning. The algorithm of multiobjective PBIL [44] starts with an external Pareto archive and initial probability matrix having all elements set to be 0.5. A set of binary design solutions are then created corresponding to the probability matrix while their function values are evaluated. The Pareto archive is updated with the nondominated solutions of the new population and the members of the previous Pareto archive. In cases that the number of nondominated solutions exceeds the predefined archive size, the normal line method is activated to remove some members from the archive. The probability matrix and the Pareto archive are iteratively updated until the termination criteria are fulfilled.

2.3. Archived Multiobjective Simulated Annealing. Simulated Annealing (SA) is one of the most popular random-directed optimisers. It has long been used in a wide variety of design applications [49, 50]. The method is based upon mimicking the random behaviour of molecules during the annealing

process, which involves slow cooling from a high temperature. For AMOSA [45], the method starts with initial temperature, a population, and a Pareto archive. A simple downhill technique is then applied to improve the population and update the archive for a few iterations. Then, a parent design solution is randomly chosen from the archive and it is used to generate a child or candidate solution by means of mutation. The parent is replaced by its offspring if the offspring dominates it. In cases that the parent is nondominated by the offspring, the parent still has a probability of being replaced. Such a probability is based on Boltzmann probability and the amount of domination between them. As the children are created, the Pareto archive is updated while the annealing temperature is reduced iteratively until a termination condition is met. An archiving technique for this algorithm is called a clustering technique.

2.4. Multiobjective Particle Swarm Optimisation. The particle swarm optimisation method uses real codes and searches for an optimum by mimicking the movement of a flock of birds, which aim to find food [46]. The search procedure herein is based on the particle swarm concepts combined with the use of an external Pareto archiving scheme. Starting with an initial set of design solutions (viewed as particles) as well as their initial velocities and objective function values, an initial Pareto archive is filled with the nondominated solutions obtained from sorting the initial population. A new population is then created by using the particle swarm updating strategy where the global best solution is randomly selected from the external Pareto archive. Afterwards, the external archive is updated by the nondominated solutions of the union set of the new population and the previous nondominated solutions. In cases that the number of nondominated solutions is too large, the adaptive grid algorithm [51] is employed to properly remove some solutions from the archive. The Pareto archive is repeatedly improved until fulfilling the termination criteria.

2.5. Unrestricted Population Size Evolutionary Multiobjective Optimisation Algorithm. The unrestricted population-size EMO algorithm was proposed by Aittokoski and Miettinen [47]. Similarly to the nondominated sorting genetic algorithm (NSGAII) [52], this optimiser uses a population to contain nondominated solutions, but its population size is unlimited. Some solutions from an initial population are randomly selected to produce some offspring by using a differential evolution operator. A new population contains nondominated solutions sorted from the combination of the offspring and the current population. The method has an unrestricted population size, therefore this somewhat provides population diversity. The optimiser is usually terminated with the maximum number of function evaluations.

3. Topological Design

Structural topology optimisation of trusses using the ground element approach can be performed in such a way that

ground finite elements, all possible combinations of predefined nodes in the given design domain, are created in the initial stage. Topological design variables determine elements' cross-sectional areas. After an optimisation run, elements with very small sizes will be removed from the structure while other elements are retained as truss members. With such a concept, an initial structural configuration is achieved. In this work, the biobjective design problem for a 2D truss with simultaneous topology shape and sizing design variables can be expressed as follows:

$$\begin{aligned} \min \quad & \{\text{mass}, c\}, \\ \text{subject to} \quad & \sigma_{\max} \leq \sigma_{\text{al}}, \\ & \lambda_i \leq 1, \\ & c \leq c_{\text{al}}, \\ & \mathbf{x} \in \Omega, \end{aligned} \tag{1}$$

$$\tag{2}$$

where mass is structural mass, c is structural compliance, c_{al} is allowable compliance, and $\sigma_{\max}$ is the maximum stress on truss elements. $\sigma_{\text{al}} = \sigma_{yt}/N_F$ is an allowable stress, λ_i is a buckling factor for each element (defined as the ratio of applied load to critical load), σ_{yt} is yield strength, N_F is the factor for safety, $\mathbf{x}$ is a vector of design variables, and Ω is a design domain of $\mathbf{x}$.

The first design objective, structural mass, is set to minimise cost while minimising the second objective, which is equivalent to structural stiffness maximisation. The stress and buckling constraints are assigned for safety requirements while the compliance constraint is imposed so as to obtain reasonable structural layouts. For one evaluation, with the input design variables, a finite element (FE) model of a truss is obtained and the FE analysis is performed. Having obtained nodal displacement and axial stresses of truss elements, the local buckling factor for each element can then be computed.

4. Adaptive Ground Elements

Two design variables sets are assigned in this paper. The first set is created for simultaneous topology and sizing optimisation. Figure 1 shows a rectangular design domain sized $W \times H$ for a 2D truss structure used in this study. The structure is subjected to point load $F = 500$ N at the top right-hand corner of the domain [33]. The width was set to be constant at 2 meters while the domain height could be varied in the range of $[H_{\min}, H_{\max}] = [0.25, 1]$ meter. Fixed boundary conditions were assigned at the left edge of the domain as shown.

Given that an array of predefined nodes is defined, ground elements as the combinations between the nodes are formed as shown in Figure 2. With the parameters n_x and n_y being the numbers of equispaced nodes in the x and y directions, respectively, $n_x \times n_y$ nodal positions were then generated where, in this study, $n_x \in \{3, 4, 5\}$ and $n_y \in \{2, 3, 4\}$. The parameters n_x and n_y are the numbers of rows and columns of an input node array, respectively. The total number of ground elements as $N_e = (n_x - 1)n_y + 3(n_x - 1)(n_y - 1)$ node combinations could be obtained as shown in Figure 2.

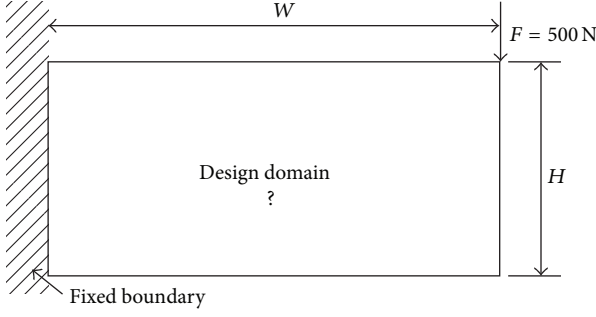


FIGURE 1: Design domain.

The design variables include the parameters H , n_x , and n_y and the diameters of the ground elements. Given that $n_{x,\max}$ and $n_{y,\max}$ are the maximum values for n_x and n_y , respectively, the possible maximum number of design variables is set as $N_{e,\max} + 3$ where $N_{e,\max} = (n_{x,\max} - 1)n_{y,\max} + 3(n_{x,\max} - 1)(n_{y,\max} - 1)$. The computational steps for design variables decoding were as follows.

Input. $\mathbf{x}$, sized $(N_{e,\max} + 3) \times 1$; $x_i \in [0, 1]$.

- (1) Set $H = H_{\min} + x_1(H_{\max} - H_{\min})$, $n_x = \text{round}(n_{x,\min} + x_2(n_{x,\max} - n_{x,\min}))$, and $n_y = \text{round}(n_{y,\min} + x_3(n_{y,\max} - n_{y,\min}))$.
- (2) Compute $N_e = (n_x - 1)n_y + 3(n_x - 1)(n_y - 1)$.
- (3) Generate $n_x \times n_y$ structural nodes, and N_e ground elements.
- (4) Assign element diameters. Set $y_j = 4x_i$ for $i = 4, \dots, N_e + 3$ and $j = 1, \dots, N_e$, which means $y_j \in [0, 4]$.
- (5) For $j = 1$ to N_e

- (5.1) If $\text{round}(y_j) = 0$, set $d_j = 0.000001$ m
- (5.2) If $\text{round}(y_j) = 1$, set $d_j = 0.01$ m
- (5.3) If $\text{round}(y_j) = 2$, set $d_j = 0.02$ m
- (5.4) If $\text{round}(y_j) = 3$, set $d_j = 0.03$ m
- (5.5) If $\text{round}(y_j) = 4$, set $d_j = 0.04$ m

End

- (6) Assign fixed boundary conditions for all nodes at $x = 0$.
- (7) Apply the y -direction point load at the last node number.

Output. Nodal positions, ground element connections, boundary conditions, force vector, and elements' diameters.

The function $\text{round}(x)$ gives the nearest integer to x . Although the design vector is sized $(N_{e,\max} + 3) \times 1$, only $N_e + 3$ elements were used for each evaluation. The design variables are said to be mixed integer/continuous, which are simpler to deal with by using MOEAs. After having the nondominated solutions, elements with 0.000001 m diameter (or the case of

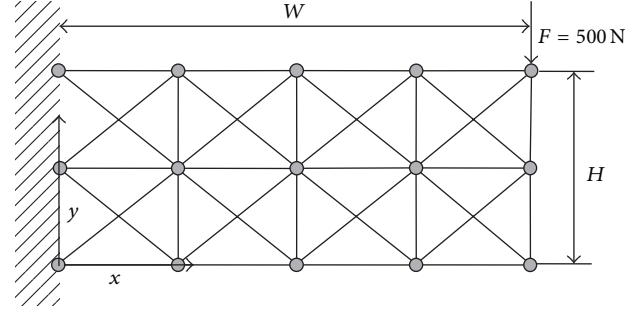


FIGURE 2: Adaptive ground elements.

round(y_j) = 0) were deleted from the ground structure to form a structural layout. A small element diameter is assigned in order to prevent singularity in a global stiffness matrix of a finite element model.

The second design variables set is an extension of the first set where parameters for nodal positions are added. For this study, initial positions of the truss joint are set as an array as the first design variables set shown in Figure 2. Additional shape design parameters determine the position changes of those nodes in x - and y -directions. All truss nodal positions except the fixed joints and the top row nodes are assigned to be varied in both x - and y -directions during an optimisation process. The top row nodal positions are allowed to be changed in the x direction in order to keep all top row nodes at the same level. As a result, the number of additional shape variables is $N_{sh} = (n_x - 1)(2n_y - 1)$. Therefore, the possible maximum number of design variables for this case is $N_{e,\max} + N_{sh,\max} + 3$ where $N_{sh,\max} = (n_{x,\max} - 1)(2n_{y,\max} - 1)$. The computational steps for design vector encoding can be given as follows.

Input. $\mathbf{x}$, sized $(N_{e,\max} + N_{sh,\max} + 3) \times 1$; $x_i \in [0, 1]$.

- (1) Set $H = H_{\min} + x_1(H_{\max} - H_{\min})$, $n_x = \text{round}(n_{x,\min} + x_2(n_{x,\max} - n_{x,\min}))$, and $n_y = \text{round}(n_{y,\min} + x_3(n_{y,\max} - n_{y,\min}))$.
- (2) Compute $N_e = (n_x - 1)n_y + 3(n_x - 1)(n_y - 1)$.
- (3) Compute $N_{sh,x} = n_y(n_x - 1)$ and $N_{sh,y} = (n_x - 1)(n_y - 1)$.
- (4) Compute the move limit for x -direction nodal position as $\Delta x_{\min} = -(W/2/n_{x,\max} - 0.01)$, and $\Delta x_{\max} = W/2/n_{x,\max} - 0.01$ meter.
- (5) Compute the move limit for y -direction nodal position as $\Delta y_{\min} = -(H_{\min}/2/n_{y,\max} - 0.01)$, and $\Delta y_{\max} = H_{\min}/2/n_{y,\max} - 0.01$ meter.
- (6) Generate $n_x \times n_y$ initial structural nodes $\mathbf{X}$, $\mathbf{Y}$, and N_e ground elements where $\mathbf{X}$ and $\mathbf{Y}$ are nodal coordinates in x - and y -directions, respectively.
- (7) Assign element diameters. Set $y_j = 4x_i$ for $i = 4, \dots, N_e + 3$ and $j = 1, \dots, N_e$.
- (8) For $j = 1$ to N_e

- (8.1) If round $(y_j) = 0$, set $d_j = 0.000001$ m
 (8.2) If round $(y_j) = 1$, set $d_j = 0.01$ m
 (8.3) If round $(y_j) = 2$, set $d_j = 0.02$ m
 (8.4) If round $(y_j) = 3$, set $d_j = 0.03$ m
 (8.5) If round $(y_j) = 4$, set $d_j = 0.04$ m
 End
 (9) Set $\Delta x_j = \Delta x_{\min} + (\Delta x_{\max} - \Delta x_{\min})x_i$ for $i = (N_e + 4), \dots, (N_e + N_{sh,x} + 3)$ and $j = 1, \dots, N_{sh,x}$. Update x -direction nodal positions $X_j = X_j + \Delta x_j$ for all node positions allowed to be changed.
 (10) Set $\Delta y_j = \Delta y_{\min} + (\Delta y_{\max} - \Delta y_{\min})x_i$ for $i = (N_e + N_{sh,x} + 4), \dots, (N_e + N_{sh,x} + N_{sh,y} + 3)$ and $j = 1, \dots, N_{sh,y}$. Update y -direction nodal positions $Y_j = Y_j + \Delta y_j$ for all node positions allowed to be changed.
 (11) Assign fixed boundary condition for all nodes at $x = 0$.
 (12) Apply the y -direction point load at the last node number.

Output. Nodal positions, ground element connections, boundary conditions, force vector, and elements' diameters.

The design vector is sized $(N_{e,\max} + N_{sh,\max} + 3) \times 1$; however, only $N_e + N_{sh} + 3$ elements were used for each finite element model. The design problem (1) using the first set of design variables is termed OPT1 whereas the problem using the second set of design variables is named OPT2. The structure is made up of a material with Young modulus, yield strength, and density of 214×10^9 Pa, 360×10^6 Pa, and 7850 kg/m^3 , respectively. The factor for safety was set to be 1.5 while the parameter c_{al} was set to be 0.04 N-m. Seven multiobjective evolutionary strategies are employed in this study where the optimisation settings are detailed as follows.

SPEA1: version two of SPEA using real codes with crossover and mutation rates of 1.0 and 0.05, respectively.

SPEA2: version two of SPEA using real codes with crossover and mutation rates of 1.0 and 0.5, respectively.

SPEA3: version two of SPEA using real codes with crossover and mutation rates of 1.0 and 0.75, respectively.

PBIL using binary codes with 0.05 mutation rate: and 0.2 mutation shift.

AMOSa: performs a simple downhill operator for the first 3 generations before entering the main procedure. In this paper, the method uses real codes where a candidate is obtained by means of mutation.

MPSO: using real codes with a starting inertia weight, an ending inertia weight, a cognitive learning factor, and a social learning factor of 0.75, 0.1, 0.75, and 0.75, respectively.

TABLE 1: Hypervolume comparison of OPT1 and OPT2.

Optimiser	OPT1		OPT2	
	Mean	Std	Mean	Std
SPEA1	0.7098	0.0998	0.8619	0.0267
SPEA2	0.8161	0.0343	0.9274	0.0184
SPEA3	0.8400	0.0564	0.9236	0.0219
PBIL	0.9330	0.0647	0.9519	0.0527
AMOSa	0.6619	0.1143	0.7997	0.0277
MPSO	0.3043	0.1371	0.6699	0.0582
UPSEMOA	0.4334	0.2808	0.6136	0.3885

UPSEMOA: using crossover probability, scaling factor, probability of choosing element from offspring in crossover, minimum population size, and burst size of 0.7, 0.8, 0.5, 10, and 25, respectively.

SPEA and PBIL are the best optimisers in the previous studies [28] while AMOSa is selected instead of the usual NSGAII. MPSO is an optimiser with a different concept from the aforementioned MOEAs while UPSEMOA is used as it is a newly developed algorithm. For OPT1, the number of generations of MOEAs is set to be 250 while both the population and archive size are set as 150. For the second test problem, the number of generations of MOEAs is set to be 300 while both the population and archive size are set as 200. Each optimiser is applied to solve each design problem for 10 optimisation runs starting with the same initial population. The last updated Pareto archive is considered a Pareto optimal set. The nondominated sorting scheme presented in [53] is used to cope with design constraints.

5. Design Results

Having performed each multiobjective evolutionary strategy solving each design problem for ten simulation runs, the performance comparison of the various MOEAs is investigated based on a hypervolume indicator. The hypervolume indicator is one of the most reliable performance indicators used to examine the quality of approximate Pareto fronts obtained from using MOEAs. The parameter determines the area for biobjective or volume for more than two objectives covered by a particular nondominated front with respect to a given reference point. Since there are ten fronts for each method and for each design problem, the average of ten hypervolume values is employed to measure the convergence rate of the optimiser whereas the standard deviation is used to indicate search consistency.

The comparative hypervolumes of the various MOEAs for solving OPT1 and OPT2 are given in Table 1. Note that the hypervolumes in the table are normalised for ease in comparing. It is shown that PBIL is the best optimiser for both design cases based upon the convergence rate while SPEA2 is the most consistent method. However, as the standard deviation of SPEA2 does not approach to zero, the convergence rate is the most important indicator for this study.

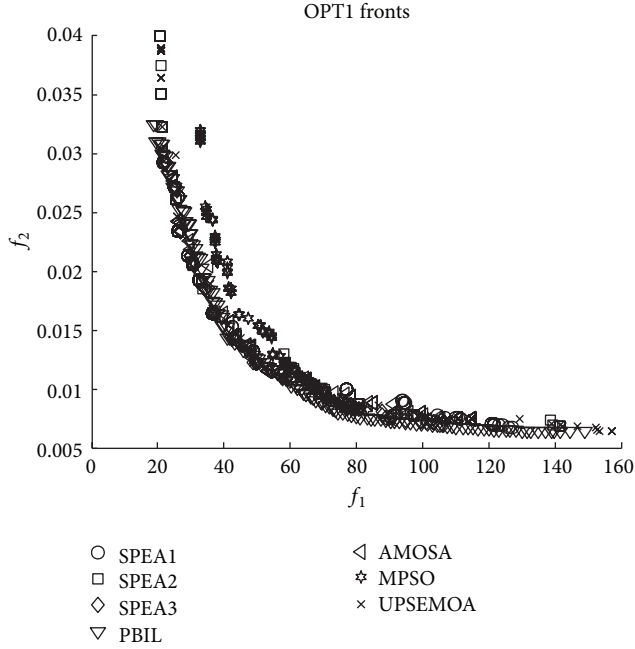


FIGURE 3: OPT1 best Pareto fronts from MOEAs.

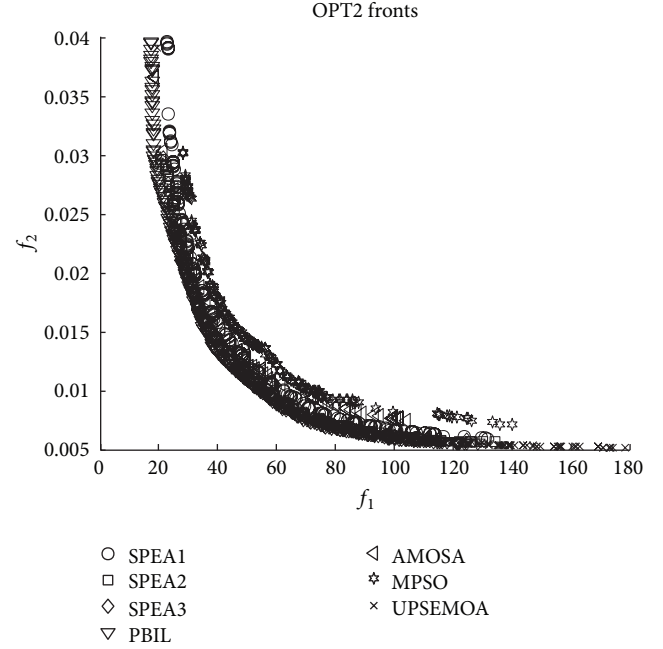


FIGURE 5: OPT2 best Pareto fronts from MOEAs.

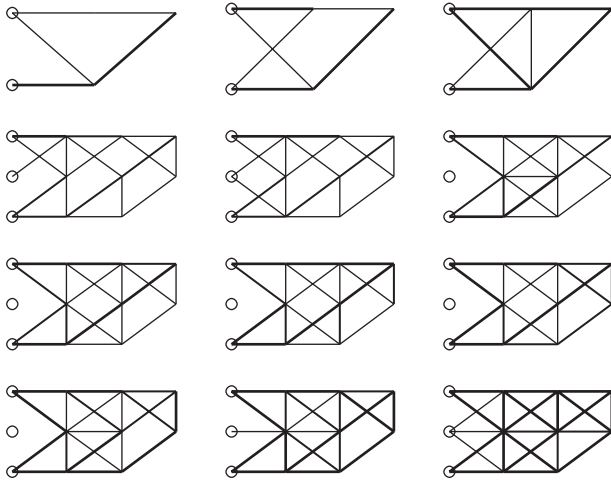


FIGURE 4: Some structures from the best front of PBIL in Figure 3.

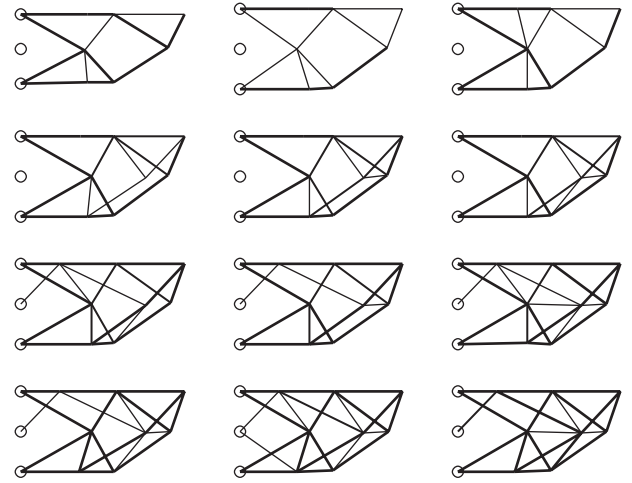


FIGURE 6: Some structures from the best front of PBIL in Figure 5.

Among the SPEA versions, it can be said that SPEA with higher mutation rate is superior to the low mutation rate version. SPEA3 is the second best optimiser for OPT1 while the worst performer for this problem is MPSO. For OPT2, the second best is SPEA2 closely followed by SPEA3 as the third best. The worst for this design case is UPSEMOA.

For the OPT1 design problem, the best fronts obtained from the best run of each multiobjective optimiser are plotted in Figure 3. It can be seen that the fronts are close together although the PBIL front has slightly more advancement and extension. This implies that MOEAs are efficient for this design problem. Some trusses selected from the PBIL front are displayed in Figure 4. The structures have a variety of structural layouts and elements' sizes; however, there are only two sets of node number, that is, 2×1 and 4×2 .

Figure 5 shows Pareto fronts obtained from the best runs of the various optimisers for the OPT2 case. Similarly to the first design case, all seven fronts are somewhat close together with the front obtained from using PBIL having slightly more advancement and extension. Some selected solutions from the PBIL front are displayed in Figure 6. The structures for this case have various topologies, and element dimensions, but tend to have fairly similar shapes. Changing nodal position has significant impact on the resulting structures. There is only one set of nodes array for the ground elements, which is 4×2 . The best fronts obtained from solving OPT1 and OPT2 are plotted together for comparison in Figure 7. It is shown that the best front of OPT2 totally dominates that of OPT1. This implies that the second set of design variables is more efficient and effective.

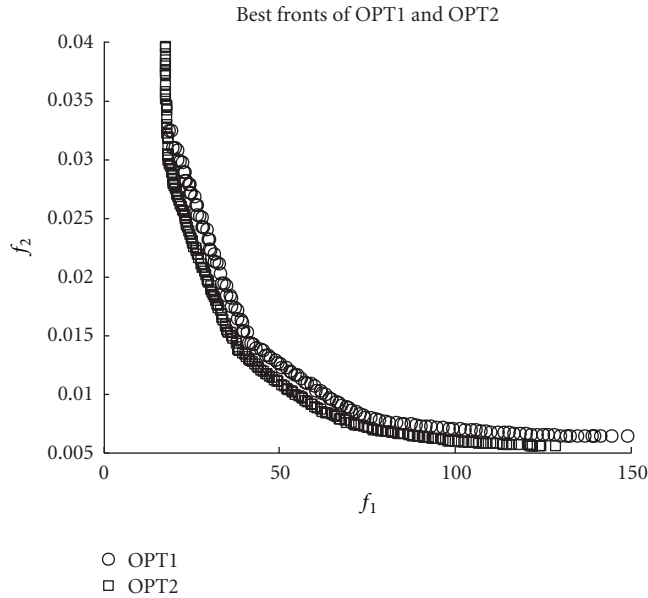


FIGURE 7: OPT1 front versus OPT2 front.

6. Conclusions and Discussion

The design results show that the proposed design strategies are efficient and effective as a variety of reasonable structural configurations could be obtained within one optimisation run. The obtained structures are said to be ready to use as they satisfy all the constraints for safety requirements. The multiobjective topological, shape, and sizing design with an adaptive ground element approach proposed herein can be tackled by an efficient multiobjective evolutionary algorithm. Among the MOEAs employed in this paper, PBIL is the most powerful method for solving both design problems based on the hypervolume indicator. For SPEA as the second best optimiser, using a higher mutation rate is more efficient. The second set of design variables which combines all types of design variables at the same time is superior to the first set which uses only topology and sizing variables. The proposed design approach can be a powerful engineering design tool in the future since, with one optimisation run, various feasible truss structures are obtained for decision making.

Future work could be the application of this design concept to a large-scale truss design problem. Alternative adaptive ground element formations can be created. The application to three-dimensional practical trusses and frames is challenging. Also, performance enhancement of multiobjective optimisers for solving this design problem is to be conducted.

Acknowledgments

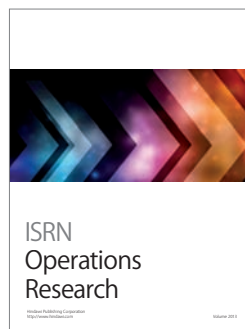
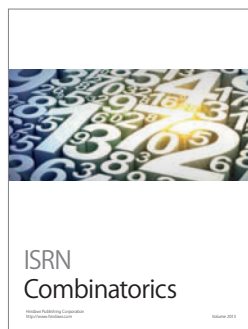
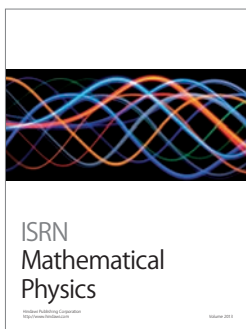
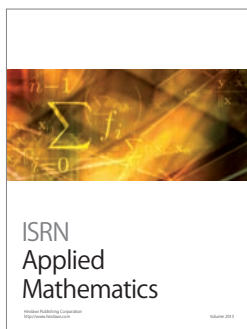
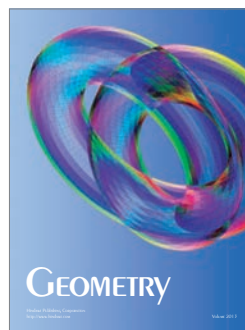
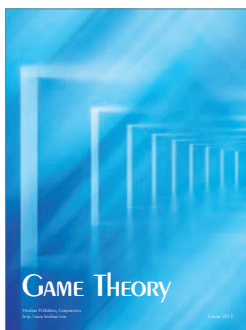
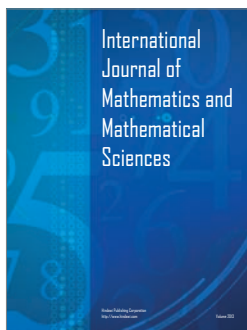
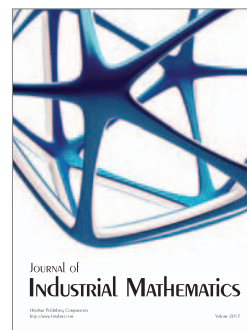
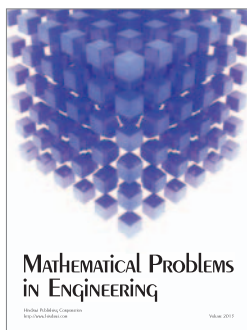
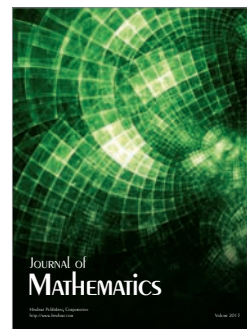
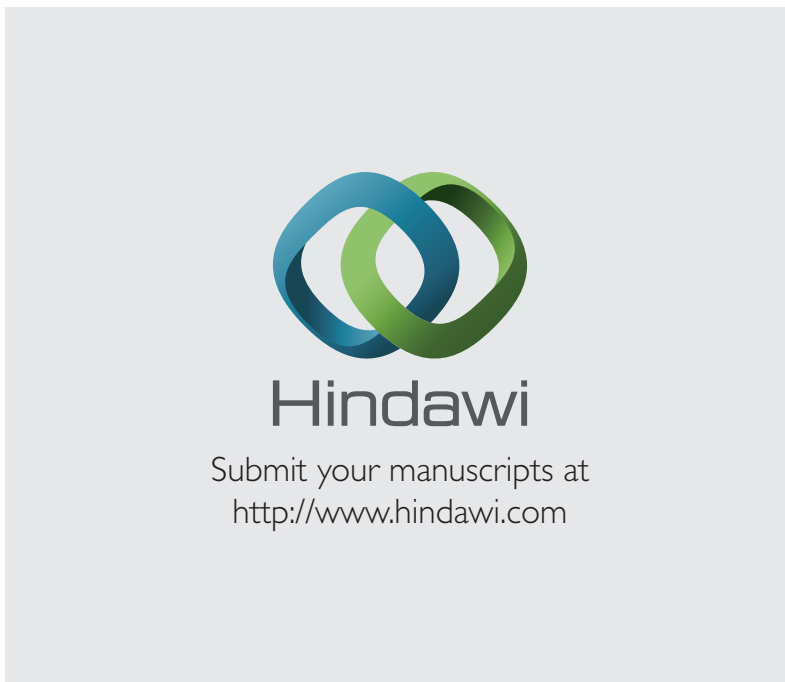
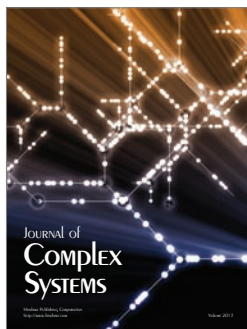
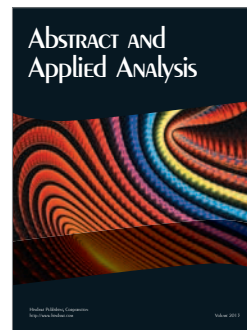
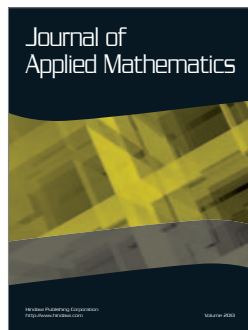
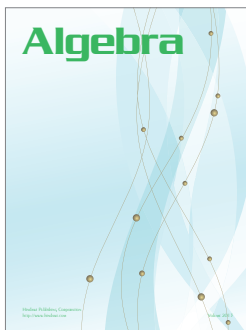
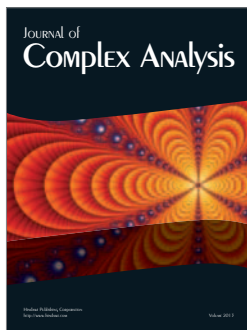
The authors are grateful for the support from the Thailand Research Fund (TRF), BRG5580017, and the Graduate School, Rajabhat Maha Sarakham University, Thailand.

References

- [1] P. Lu, S. Chen, and Y. Zheng, "Artificial intelligence in civil engineering," *Mathematical Problems in Engineering*, vol. 2012, Article ID 145974, 22 pages, 2012.
- [2] C. Cattani, S. Chen, and G. Aldashev, "Information and modeling in complexity," *Mathematical Problems in Engineering*, vol. 2012, Article ID 868413, 3 pages, 2012.
- [3] A. R. Yildiz, "Comparison of evolutionary-based optimization algorithms for structural design optimization," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 327–333, 2013.
- [4] A. R. Yildiz and K. N. Solanki, "Multi-objective optimization of vehicle crashworthiness using a new particle swarm based approach," *International Journal of Advanced Manufacturing Technology*, vol. 59, pp. 367–376, 2012.
- [5] A. F. Ulusoy and F. Erbatur, "Discrete and continuous structural optimisation using evolution strategies," in *Proceedings of the 3rd International Conference on Engineering Computational Technology (ICECT '03)*, B. H. V. Topping and Z. Bittnar, Eds., pp. 163–164, Civil-Comp press, Edinburgh, UK, 2002.
- [6] A. R. Yildiz and K. Saitou, "Topology synthesis of multicomponent structural assemblies in continuum domains," *Journal of Mechanical Design*, vol. 133, no. 1, Article ID 011008, 2011.
- [7] N. Pholdee and S. Bureerat, "Performance enhancement of multiobjective evolutionary optimizers for truss design using an approximate gradient," *Computers and Structures*, vol. 106–107, pp. 115–124, 2012.
- [8] I. Durgun and A. R. Yildiz, "Structural design optimization of vehicle components using Cuckoo search algorithm," *Materials Testing*, vol. 54, no. 3, pp. 185–188, 2012.
- [9] S. Chen, Y. Zheng, C. Cattani, and W. Wang, "Modeling of biological intelligence for SCM system optimization," *Computational and Mathematical Methods in Medicine*, vol. 2012, Article ID 769702, 10 pages, 2012.
- [10] M. Zirour, C. Y. Liong, and W. R. Ismail, "Genetic algorithm for a quay management problem," *Australian Journal of Basic and Applied Sciences*, vol. 5, no. 9, pp. 2257–2263, 2011.
- [11] S. Y. Chen and Y. F. Li, "Automatic sensor placement for model-based robot vision," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 34, no. 1, pp. 393–408, 2004.
- [12] A. R. Yildiz, "A novel hybrid immune algorithm for global optimization in design and manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 2, pp. 261–270, 2009.
- [13] A. R. Yildiz, "A new hybrid bee colony optimization approach for robust optimal design and manufacturing," *Applied Soft Computing*. In press.
- [14] V. Kelner, F. Capitanescu, O. Léonard, and L. Wehenkel, "A hybrid optimization technique coupling an evolutionary and a local search algorithm," *Journal of Computational and Applied Mathematics*, vol. 215, no. 2, pp. 448–456, 2008.
- [15] A. R. Yildiz, "A new design optimization framework based on immune algorithm and Taguchi's method," *Computers in Industry*, vol. 60, no. 8, pp. 613–620, 2009.
- [16] S. Kanyakam and S. Bureerat, "Comparative performance of surrogate-assisted MOEAs for geometrical design of pin-fin heat sinks," *Applied Mathematics*, vol. 2012, Article ID 534783, 14 pages, 2012.
- [17] T. Kunakote and S. Bureerat, "Multi-objective topology optimization using evolutionary algorithms," *Engineering Optimization*, vol. 43, no. 5, pp. 541–557, 2011.

- [18] W. Achziger, "On simultaneous optimization of truss geometry and topology," *Structural and Multidisciplinary Optimization*, vol. 33, no. 4-5, pp. 285-304, 2007.
- [19] W. Tang, L. Tong, and Y. Gu, "Improved genetic algorithm for design optimization of truss structures with sizing, shape and topology variables," *International Journal for Numerical Methods in Engineering*, vol. 62, no. 13, pp. 1737-1762, 2005.
- [20] K. Shea and I. F. C. Smith, "Improving full-scale transmission tower design through topology and shape optimization," *Journal of Structural Engineering*, vol. 132, no. 5, pp. 781-790, 2006.
- [21] C. M. Chan and K. M. Wong, "Structural topology and element sizing design optimisation of tall steel frameworks using a hybrid OC-GA method," *Structural and Multidisciplinary Optimization*, vol. 35, no. 5, pp. 473-488, 2008.
- [22] J. Zhu, W. Zhang, P. Beckers, Y. Chen, and Z. Guo, "Simultaneous design of components layout and supporting structures using coupled shape and topology optimization technique," *Structural and Multidisciplinary Optimization*, vol. 36, no. 1, pp. 29-41, 2008.
- [23] M. Giger and P. Ermanni, "Evolutionary truss topology optimization using a graph-based parameterization concept," *Structural and Multidisciplinary Optimization*, vol. 32, no. 4, pp. 313-326, 2006.
- [24] R. Su, X. Wang, L. Gui, and Z. Fan, "Multi-objective topology and sizing optimization of truss structures based on adaptive multi-island search strategy," *Structural and Multidisciplinary Optimization*, vol. 43, no. 2, pp. 275-286, 2011.
- [25] V. Togan and A. Daloglu, "Optimization of 3d trusses with adaptive approach in genetic algorithms," *Engineering Structures*, vol. 28, pp. 1019-1027, 2006.
- [26] H. Rahami, A. Kaveh, and Y. Gholipour, "Sizing, geometry and topology optimization of trusses via force method and genetic algorithm," *Engineering Structures*, vol. 30, no. 9, pp. 2360-2369, 2008.
- [27] O. Hasanc and F. Erbatur, "Layout optimization of trusses using simulated annealing," *Advances in Engineering Software*, vol. 33, pp. 681-696, 2002.
- [28] C. Noilublao and S. Bureerat, "Simultaneous topology shape and sizing optimisation of skeletal structures using multiobjective evolutionary algorithms," in *Evolutionary Computation*, pp. 487-580, 2009.
- [29] T. Hagishita and M. Ohsaki, "Topology optimization of trusses by growing ground structure method," *Structural and Multidisciplinary Optimization*, vol. 37, no. 4, pp. 377-393, 2009.
- [30] A. Asadpoure, M. Tootkaboni, and J. K. Guest, "Robust topology optimization of structures with uncertainties in stiffness—application to truss structures," *Computers and Structures*, vol. 89, no. 11-12, pp. 1131-1141, 2011.
- [31] M. Beckers and C. Fleury, "A primal-dual approach in truss topology optimization," *Computers and Structures*, vol. 64, no. 1-4, pp. 77-88, 1997.
- [32] I. Hajirasouliha, K. Pilakoutas, and H. Moghaddam, "Topology optimization for the seismic design of truss-like structures," *Computers and Structures*, vol. 89, no. 7-8, pp. 702-711, 2011.
- [33] C. Noilublao and S. Bureerat, "Topology and sizing optimization of trusses with adaptive ground finite elements using multi-objective PBIL," *Advanced Materials Research*, vol. 308-310, pp. 1116-1121, 2011.
- [34] S. Silih, M. Premrov, and S. Kravanja, "Optimum design of plane timber trusses considering joint flexibility," *Engineering Structures*, vol. 27, no. 1, pp. 145-154, 2005.
- [35] L. Gil and A. Andreu, "Shape and cross-section optimization of a truss structure," *Computers and Structures*, vol. 79, no. 7, pp. 681-689, 2001.
- [36] R. Karkauskas and A. Norkus, "Truss optimization under stiffness, stability constraints and random loading," *Mechanics Research Communications*, vol. 33, no. 2, pp. 177-189, 2006.
- [37] A. Kaveh and S. Talatahari, "Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures," *Computers and Structures*, vol. 87, no. 5-6, pp. 267-283, 2009.
- [38] A. C. C. Lemonge, H. J. C. Barbosa, A. L. G. A. Coutinho, and C. C. H. Borges, "Multiple cardinality constraints and automatic member grouping in the optimal design of steel framed structures," *Engineering Structures*, vol. 33, no. 2, pp. 433-444, 2011.
- [39] C. Noiluplao and S. Bureerat, "Simultaneous topology and shape optimisation of a 3D framework structure using simulated annealing," in *Proceedings of the Technology and Innovation for Sustainable Development Conference (TISD '08)*, Faculty of Engineering, Khon Kaen University, January 2008.
- [40] P. Martínez, P. Martí, and O. M. Querin, "Growth method for size, topology, and geometry optimization of truss structures," *Structural and Multidisciplinary Optimization*, vol. 33, no. 1, pp. 13-26, 2007.
- [41] M. Zhou, N. Pagaldipti, H. L. Thomas, and Y. K. Shyy, "An integrated approach to topology, sizing, and shape optimization," *Structural and Multidisciplinary Optimization*, vol. 26, no. 5, pp. 308-317, 2004.
- [42] T. Sokol, "A 99 line code for discretized Michell truss optimization written in Mathematica," *Structural and Multidisciplinary Optimization*, vol. 43, no. 2, pp. 181-190, 2011.
- [43] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimization and Control*, Barcelona, Spain, 2002.
- [44] S. Bureerat and S. Srisomporn, "Optimum plate-fin heat sinks by using a multi-objective evolutionary algorithm," *Engineering Optimization*, vol. 42, no. 4, pp. 305-323, 2010.
- [45] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: AMOSA," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 3, pp. 269-283, 2008.
- [46] M. Reyes-Sierra and C. A. C. Coello, "Multi-objective particle swarm optimizers: a survey of the state-of-the-art," *International Journal of Computational Intelligence Research*, vol. 2, pp. 287-308, 2006.
- [47] T. Aittokoski and K. Miettinen, "Efficient evolutionary approach to approximate the Pareto-optimal set in multiobjective optimization, UPS-EMOA," *Optimization Methods and Software*, vol. 25, no. 6, pp. 841-858, 2010.
- [48] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257-271, 1999.
- [49] W. A. Bennage and A. K. Dhinra, "Single and multiobjective structural optimization in discrete-continuous variables using simulated annealing," *International Journal for Numerical Methods in Engineering*, vol. 38, no. 16, pp. 2753-2773, 1995.
- [50] S. Bureerat and J. Limtragool, "Performance enhancement of evolutionary search for structural topology optimisation," *Finite Elements in Analysis and Design*, vol. 42, no. 6, pp. 547-566, 2006.

- [51] J. D. Knowles and D. W. Corne, "Approximating the non-dominated front using the Pareto archived evolution strategy," *Evolutionary computation*, vol. 8, no. 2, pp. 149–172, 2000.
- [52] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [53] K. Deb, A. Pratap, and T. Meyarivan, "Constrained test problems for multi-objective evolutionary optimization," KanGAL Report No. 200002, Kanpur Genetic Algorithms Laboratory (KanGAL), Indian Institute of Technology, Kanpur, India, 2000.





Hybridisation of real-code population-based incremental learning and differential evolution for multiobjective design of trusses

Nantiwat Pholdee, Sujin Bureerat *

Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand

ARTICLE INFO

Article history:

Received 5 March 2012

Received in revised form 22 August 2012

Accepted 14 October 2012

Available online 23 October 2012

Keywords:

Real-code population-based incremental learning

Differential evolution

Hybrid method

Multiobjective evolutionary algorithm

Multiobjective design of truss

ABSTRACT

This paper proposes a hybrid evolutionary algorithm for multiobjective optimisation of trusses using real-code population-based incremental learning (RPBIL) to solve multiobjective design problems. Differential evolution (DE) operators are integrated into the main procedure of RPBIL leading to a hybrid algorithm. The newly developed optimiser, along with some established multiobjective evolutionary algorithms (MOEAs) is implemented to solve a number of multiobjective design problems of trusses. Comparative performance based upon a hypervolume indicator shows that the new hybrid multiobjective evolutionary algorithm is superior to the other MOEAs particularly in cases involving large-scale truss design problems.

© 2012 Elsevier Inc. All rights reserved.

1. Introduction

The truss has been one of the most used designs throughout structural engineering history. Using such a design is advantageous in that it is simple and inexpensive to construct, modify, and maintain, especially in difficult-to-access areas. In the literature, the design optimisation of trusses has seen a resurgence of interest recently. These design problems have usually involved minimising structural weight or cost while maintaining safety. These problems may have one or more design objectives such as dynamic stiffness (or natural frequency), compliance, frequency response function, force transmissibility, and buckling factor [29].

The optimisers used in truss design problems can be categorised as gradient-based methods (or local search), and metaheuristics (MHs), more commonly known as evolutionary algorithms (EAs). Previous studies using gradient-based optimisers such as sequential linear programming [16,18], feasible direction method [42], and sequential quadratic programming [32,41], for truss design have been conducted. Some well-known EAs including genetic algorithms, have been implemented to solve structural optimisation problems [17,21,22,29,34,44,45]. Gradient-based methods have faster convergence rates and are more consistent in finding a local optimum, however, they require continuous design variables, and accurate derivative calculations of design functions. This makes them difficult to use for most cases of structural optimisation, as inaccurate estimation of function derivatives may lead their search procedures to improper solutions. The EAs, on the other hand, have emerged as strong candidates for this design task in the last few decades [10]. Compared to their gradient-based counterparts, they are easier to use, more robust, and capable of dealing with all kinds of design variables since they do not require function derivatives for searching. Moreover, their most outstanding feature is that the multiobjective versions of EAs can search for Pareto optimal sets within one optimisation run [10,15,49,55]. Nevertheless, they inevitably have slower

* Corresponding author.

E-mail address: sujbur@kku.ac.th (S. Bureerat).

convergence rates and a lack of search consistency due to the use of randomisation in their search procedure. There have been a number of EAs developed aiming for search performance enhancement [55]. Ideas to improve EA performance include new meta-heuristic concepts [2,8,12,21,26–28,51], the hybridisation of existing optimisation operators [13,19,20,30,46–48,50] and incorporating local search into the EA's main procedure [6,14,31,52].

Differential evolution (DE) operators are some of the most successful meta-heuristics used to improve EA performance. There has been some recent success of hybridising DE with several optimisers and evolutionary operators [4,9,24,30,33,38,43,52]. In addition, a number of real world problems have been posed [39] and DE was used to solve some of them e.g. in [27] and [51]. However, to our best knowledge, using such a concept for the application of multiobjective constrained optimisation particularly in truss design is rare. This paper proposes a new hybrid EA for multiobjective optimisation of trusses. The real-code PBIL for multiobjective optimisation is initially introduced, then, the incorporation of differential evolution operators into multiobjective RPBIL to improve its search performance is detailed. The hybrid optimiser is then implemented to solve several multiobjective test problems as well as multiobjective truss design problems. Comparative performance based on a hypervolume indicator shows that the proposed hybrid multiobjective evolutionary algorithm outperforms some established multiobjective evolutionary algorithms when dealing with truss design.

The rest of the paper is organised as follows. Section 2 details the basic concepts of real-code PBIL for single objective optimisation. The hybrid algorithm of RPBIL and DE is discussed in Section 3, while a search performance test is conducted in Section 4. Section 5 gives details of truss design problems. Performance comparison of the various multiobjective evolutionary optimisers on truss design is illustrated in Section 6, while Section 7 gives conclusions and some discussion about this study.

2. Real-code population-based incremental learning

The real-code version of PBIL in this paper is based on the work presented in Bureerat [7], in which a method was proposed for solving a box-constrained minimisation problem

$$\min_{\mathbf{x}} f(\mathbf{x}); \quad \mathbf{L} \leq \mathbf{x} \leq \mathbf{U}$$

where $\mathbf{x}$ is a vector of design variables sized $n \times 1$, n is the number of design variables, f is an objective function, and $\mathbf{L}$ and $\mathbf{U}$ are lower and upper limits of $\mathbf{x}$ respectively. RPBIL search is based on the concept that a real population (a set of design solutions) on each generation is generated from a current probability matrix [6,7]. This implies that a population in the RPBIL procedure is represented by the so-called probability matrix rather than carrying a matrix containing numerous design vectors throughout the search process. Let a probability matrix have size $n \times n_i$ where n_i is a predefined number of matrix columns. The parameter n_i determines the number of subintervals of the bound constraints of the problem. For simplicity, consider a 2-dimensional box-constrained problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = x_1^2 + 2x_1x_2 + 3x_2^2 \\ \text{subject to} \quad & -5 \leq x_i \leq 5; \quad i = 1, 2 \end{aligned}$$

The probability matrix for this design problem is set to have a size of 2×4 which means that the bound constraints are divided into four subintervals being $[-5, -2.5]$, $[-2.5, 0]$, $[0, 2.5]$, and $[2.5, 5]$. A sample of a probability matrix, P_{ij} , is given in Table 1 where its rows correspond to the elements of design variables, and the columns are associated with these subintervals.

The element P_{ij} determines the probability of having x_i in the subinterval j . To generate a real-code population, given that the population size (the number of design solutions in a population) is $N_p = 8$, the number of elements x_i randomly generated in the subinterval j is approximately $\text{round}(N_p P_{ij})$ where $\text{round}(\cdot)$ is a round-off operator. In this case, the number of x_1 in the first subinterval is 1, and so on. Table 2 shows real-code design variables created on the subintervals based on P_{ij} in Table 1. For better population diversity, the positions of the variables are randomly permuted as shown in the second part of the table. The population matrix, $\mathbf{R}$, containing real-code design solutions can finally be expressed as:

$$\mathbf{R} = \{\mathbf{x}_1, \dots, \mathbf{x}_8\} = \left\{ \begin{Bmatrix} -0.75 \\ 0.50 \end{Bmatrix}, \begin{Bmatrix} -3.00 \\ -2.75 \end{Bmatrix}, \begin{Bmatrix} 1.25 \\ 2.75 \end{Bmatrix}, \begin{Bmatrix} 0.20 \\ -4.00 \end{Bmatrix}, \begin{Bmatrix} 3.25 \\ 2.15 \end{Bmatrix}, \begin{Bmatrix} -2.15 \\ 4.50 \end{Bmatrix}, \begin{Bmatrix} 4.00 \\ -1.75 \end{Bmatrix}, \begin{Bmatrix} 0.05 \\ 1.00 \end{Bmatrix} \right\}$$

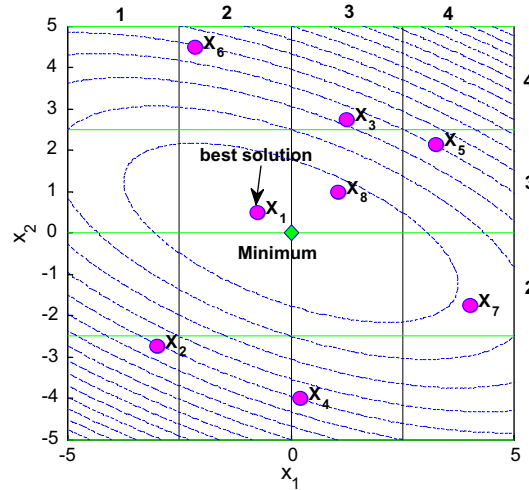
Table 1
A sample probability matrix.

Subinterval	$-5 \leq x_i < -2.5$	$-2.5 \leq x_i < 0$	$0 \leq x_i < 2.5$	$2.5 \leq x_i \leq 5$
x_1	1/8	2/8	3/8	2/8
x_2	2/8	1/8	3/8	2/8

Table 2

Generate a real-code population randomly.

Subinterval	$-5 \leq x_i < -2.5$	$-2.5 \leq x_i < 0$	$0 \leq x_i < 2.5$	$2.5 \leq x_i \leq 5$
x_1	−3.00	−2.15, −0.75	0.20, 0.05, 1.25	3.25, 4.00
x_2	−4.00, −2.75	−1.75	1.00, 0.50, 2.15	2.75, 4.50
<i>Randomly permute the positions in each row</i>				
x_1	−0.75, −3.00, 1.25, 0.20, 3.25, −2.15, 4.00, 0.05			
x_2	0.50, −2.75, 2.75, −4.00, 2.15, 4.50, −1.75, 1.00			

**Fig. 1.** Distribution of design solutions in a population.

The plot of those eight design points is illustrated in Fig. 1 where the contour plot of the objective function is shown. Based on the fitness objective function, the best solution for this population is $\mathbf{x}_1$; thus, the probability matrix will be updated towards this design solution. As the 1st element of $\mathbf{x}_1$ is in the 2nd subinterval, the value of P_{12} will be increased. Similarly, since the 2nd element of $\mathbf{x}_1$ is located in the 3rd subinterval, the value of P_{23} will be increased. The updating scheme is given in Eqs. (1) and (2) where the index r is 2 when updating P_{12} , and 3 when updating P_{23} .

$$P'_{ij} = (1 - L_R(j))P_{ij}^{old} + L_R(j) \quad (1)$$

$$L_R(j) = L_{R0} \exp(-(j - r)^2) \quad (2)$$

where L_{R0} is an initial learning rate to be specified.

The updated probability matrix P_{ij} is then modified as

$$P''_{ij} = P'_{ij} / \sum_{j=1}^{n_i} P'_{ij} \quad (3)$$

in order to preserve the condition

$$\sum_{j=1}^{n_i} P''_{ij} = 1 \quad (4)$$

where P''_{ij} is the final update of P_{ij} .

The search procedure of RPBIL for single objective optimisation is shown in Algorithm 1. Starting with an initial probability matrix $P_{ij} = 1/n_i$, a real code population is then created. The best solution among the members of the population is found and the probability matrix is updated in such a way that, in the next generation, design solutions related to the position of the current best design vector will have more chance of being generated. The process is repeated until a termination condition is met (normally when the maximum number of generations, N_G , is reached). More details of RPBIL for single objective problems can be found in [6,7].

Algorithm 1. RPBIL for single objective optimisation

Input: N_G, n_l , objective function name (fun)
Output: $\mathbf{x}^{best}, f^{best}$
Initialisation: $P_{ij} = 1/n_l, \mathbf{x}^{best} = \{\}$
1: For $i = 1$ to N_G
2: Generate a real code population $\mathbf{X}$ from P_{ij} (See Table 2) and find $\mathbf{f} = fun(\mathbf{X})$
3: Find new $\mathbf{x}^{best}$ from $\mathbf{X}$ and $\mathbf{x}^{best}$ from the previous generation
4: Update P_{ij} based on the new $\mathbf{x}^{best}$ (Eqs. (1)–(3))
5: End

3. Hybrid PBIL and DE for multiobjective design

Extension of RPBIL for multiobjective optimisation can be made in the same way as with multiobjective PBIL using binary codes [5]. In the binary version, a binary population is represented by a probability vector for single objective problems. When dealing with multiobjective problems, more probability vectors are employed to achieve population diversity. Similarly to the binary-code PBIL, the multiobjective version of RPBIL will use several probability matrices to represent a real-code population. For physical insight, we will call each probability matrix one tray. A set of probability trays is actually a three-dimensional matrix size $n \times n_l \times N_T$ where N_T is the number of trays. With such a concept, each tray will be used to generate a real-code subpopulation which has approximately N_P/N_T design solutions as its members.

The search procedure of multiobjective RPBIL is given in Algorithm 2. Starting with initial probability trays, an initial population is then created, and an initial Pareto archive is obtained. Non-dominated solutions are then selected to update the probability trays. A clustering technique detailed in [3] is activated to group the non-dominated solutions into N_T groups. Then, the centroid of each non-dominated solution group ($\mathbf{r}_G$) is used to update a probability tray in the sequence that $\mathbf{r}_G$ of the group that has the lowest value of the first objective function will be used for updating the first tray, and so on. The updating process for each tray can be carried out similarly to the updating process (1)–(3) by replacing $\mathbf{x}^{best}$ with $\mathbf{r}_G$. Afterwards, a population according to the updated trays is created. The Pareto archive is updated by replacing its members with non-dominated solutions sorted from the combination of the current population and the members in the previous archive. In cases where the number of archive members exceeds the predefined archive size, the clustering technique is activated to remove some non-dominated solutions from the archive. These steps are repeated until a stopping condition is fulfilled. Note that the clustering technique is preferred in this paper because it is a technique which does not require predefined parameters. Once the algorithm is coded, it can be used for any size of objective and solution vectors.

Algorithm 2. RPBIL for multiobjective optimisation

Input: N_G, n_l, N_T , objective function name (fun), Pareto archive size (N_A)
Output: $\mathbf{x}^{best}, f^{best}$
Initialisation: $P_{ij} = 1/n_l$ for each tray, a Pareto archive $\mathbf{A} = \{\}$
1: For $i = 1$ to N_G
2: Generate a real code population $\mathbf{X}$ from the probability trays and find $\mathbf{f} = fun(\mathbf{X})$
3: Find non-dominated solutions from $\mathbf{X} \cup \mathbf{A}$ and replace the members in $\mathbf{A}$ with these solutions
4: Group the non-dominated solutions into N_T groups using a clustering technique, and find the centroid $\mathbf{r}_G$ of each group
5: Update each tray P_{ij} based on $\mathbf{r}_G$
6: If the number of archive members is larger than N_A , remove some of the members using a clustering technique
7: End

From the predecessor [6,7], it has been found that RPBIL has very good convergence rate compared to some established EAs. Nevertheless, one weak point of the method is that, as the probability matrix relies on a current best solution, it could produce a population whose members are only the neighbours of the best solution. That means a premature convergence can possibly take place. In order to alleviate this problem, mutation and crossover of DE [9] will be incorporated into the procedure of the multiobjective RPBIL and this hybrid algorithm is defined as RPBIL-DE. The hybridisation is carried out in such a way that, for every generation, a newly generated population from the probability trays will be recombined with members in the Pareto archive by means of DE operators before performing function evaluation. More details of the procedure are given in Algorithm 3 where F is a scaling factor, p_c is a DE crossover probability, and CR is the probability of choosing an element of an offspring $\mathbf{c}$. The concept of RPBIL is a simple type of estimation of distribution algorithms (EDA). The rationale for

integrating DE operators into the procedure of RPBIL is because it has been reported that a hybrid of EDA and DE successfully improved evolutionary search performance for single objective optimisation. For example, the mixed distribution based univariate estimation of distribution algorithm (MUEDA) was employed in combination with DE in [43]. On the other hand, it has been shown that the use of a Gaussian model with diagonal covariance matrix (GM/DCM) for EDA combined with DE [38]. It has been shown that the hybrid versions were superior to their original EAs.

Algorithm 3. Hybrid RPBIL-DE for multiobjective optimisation

Input: N_G, N_p, n_l, N_T , objective function name (fun), Pareto archive size (N_A)

Output: $\mathbf{x}^{best}, f^{best}$

Initialisation: $P_{ij} = 1/n_l$ for each tray

: Generate a real code population $\mathbf{X}$ from the probability trays and find $\mathbf{f} = fun(\mathbf{X})$

: Find a Pareto archive $\mathbf{A}$

1: For $i = 1$ to N_G

2: Group the non-dominated solutions into N_T groups using a clustering technique, and find the centroid $\mathbf{r}_G$ of each group

3: Update each tray P_{ij} based on $\mathbf{r}_G$

4: Generate a real code population $\mathbf{X}$ from the probability trays

5: For $j = 1$ to N_p recombine $\mathbf{X}$ and $\mathbf{A}$ using DE operators

5.1: Select $\mathbf{p}$ from $\mathbf{A}$ randomly

5.2: Select $\mathbf{q}$ and $\mathbf{r}$ from $\mathbf{X}$ randomly, $\mathbf{q} \neq \mathbf{r}$

5.3: Calculate $\mathbf{c} = \mathbf{p} + F(\mathbf{q} - \mathbf{r})$

5.4: Set c_i into the interval $L_i \leq c_i \leq U_i$ if it is outside the bounds

5.5: If $rand < p_c$, perform crossover

5.5.1: For $k = 1$ to n

5.5.2: If $rand < CR$, $y_k = c_k$

5.5.3: Otherwise, $y_{j,k} = p_k$

5.5.4: End

6: End

7: New real-code population is $\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_j, \dots, \mathbf{y}_{Np}\}$ and find $\mathbf{f} = fun(\mathbf{Y})$

8: Find non-dominated solutions from $\mathbf{Y} \cup \mathbf{A}$ and replace the members in $\mathbf{A}$ with these solutions

9: If the number of archive members is larger than N_A , remove some of the members using a clustering technique

10: End

4. Performance test

In order to examine the search performance of the proposed algorithms, five multiobjective unconstrained test problems and seven multiobjective constrained test problems were used to illustrate the search efficiency of the proposed algorithms. The ZDT test problems presented in [56] including ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 are employed as multiobjective unconstrained test problems. Also, the multiobjective constrained test problems CF1–CF7 presented in [53] are used as multiobjective constrained test problems. It should be noted that there have recently been numerous test functions proposed for comparative studies of many types of design problems (for example, see [25,40,53]). The test functions used in this paper are some of them.

Ten multiobjective evolutionary optimisers, namely binary population-based incremental learning (BPBIL) [5], unrestricted population size evolutionary multiobjective optimisation algorithm (UPS-EMOA) [1], strength Pareto evolutionary algorithm (SPEA2) [57], multiobjective particle swarm optimisation (MPSO) [35], non-dominated sorting genetic algorithm (NSGA-II) [11], multiobjective RPBIL (Algorithm 2), dynamic multiobjective evolutionary algorithm (DMOEA) [23], differential evolution for multiobjective optimisation (DEMO) [36], multiobjective evolutionary algorithm based on decomposition (MOEA/D) [54], and RPBIL-DE (Algorithm 3) were employed to solve the test problems. BPBIL and SPEA2 have been shown to be top performers in a previous study [29]. NSGA-II has usually been regarded as the standard MOEA for testing a new method while UPS-EMOA is a newly developed evolutionary algorithm based upon DE concepts. The optimisation settings for the optimisers are:

- BPBIL using binary codes with mutation probability and mutation shift being 0.05 and 0.2 respectively.
- UPS-EMOA using crossover probability, scaling factor, probability of choosing element from offspring in crossover, minimum population size, and burst size being 0.7, 0.8, 0.5, 10, and 25 respectively.
- SPEA2 using real codes [37] with crossover and mutation rates of 1.0 and 0.1 respectively.

- MPSO using the starting inertia weight, ending inertia weight, cognitive learning factor, and social learning factor being 0.5, 0.01, 0.5 and 0.5 respectively. The best solution for computing a velocity vector is randomly selected from a Pareto archive. The adaptive grid algorithm is used as an archiving technique.
- NSGA-II using real codes with crossover mutation probabilities as 1.0 and 0.1 respectively.
- RPBIL using real codes with $N_f = 40$ where each probability tray produces five design solutions.
- RPBIL-DE using real codes with $N_f = 40$ where each probability tray produces five design solutions. Crossover probability, scaling factor and probability of choosing an element from an offspring in crossover for DE operators are set as 0.7, 0.8, and 0.5 respectively.
- DMOEA using real codes with temperature and mutation rate of 1000 and 1 respectively.
- DEMO using real codes with crossover probability, scaling factor and probability of choosing an element from an offspring in crossover for DE operators being 0.7, 0.8, and 0.5 respectively.
- MOEA/D using real codes with number of neighbouring weight vectors, crossover and mutation probabilities being 6, 1.0, and 0.1 respectively.

All MOEAs except UPS-EMOA use the population size of 100, and 250 generations for multiobjective unconstrained test problems. In addition, the population size and number of generations are 100 and 300 for multiobjective constrained test problems. UPS-EMOA uses different population size and iteration number as defined previously; however, their search procedures are terminated at the same total number of function evaluations. The Pareto archive of all MOEAs is set to be 500 except for NSGA-II and DEMO which contain Pareto solutions in a population. The optimisation parameter settings detailed above are obtained from using several settings for each optimiser and selecting the one that gives the best results.

The ten multiobjective evolutionary optimisers are employed to solve each problem over 30 runs starting with the same initial population. The comparative results based on the hypervolume indicator are shown in Tables 3 and 4. The ZDT test problems are shown in Table 3 and the constrained test problems are shown in Table 4. Note that those hypervolume values are normalised for ease in comparison. The higher hypervolume value the better the Pareto front. With 30 runs, the mean value of the front hypervolumes is used to measure the algorithm's convergence rate while the standard deviation represents the search consistency of the method. This means that a method with lower hypervolume standard deviation (STD) has better search consistency.

From Table 3, the best performer (among compared MOEAs) according to the convergence rate (average hypervolume) for ZDT1, ZDT2, ZDT3, and ZDT6 is MOEA/D while NSGA-II is the best for ZDT4. The second best of ZDT1 and ZDT2 is RPBIL while RPBIL-DE is the second best of ZDT3 and ZDT6. For the ZDT4, the second best is MOEA/D. The worst method based on the convergence rate for ZDT1, ZDT2, and ZDT3 is MPSO while the worst of ZDT4 and ZDT6 is RPBIL. For a search consistency measure which is based on the hypervolume standard deviation, the most consistent method for ZDT1, ZDT2, ZDT3 and

Table 3
Normalised hypervolumes of all MOEAs for all ZDT test problems.

Test problems	MOEAs										Friedman test (p-value)
	BPBIL	UPS-EMOA	SPEA2	MPSO	NSGA-II	RPBIL	RPBIL-DE	DMOEA	DEMO	MOEA/D	
<i>ZDT1</i>											
Mean	0.9969	0.9872	0.9867	0.3911	0.9679	0.9481	0.9868	0.8727	0.9778	0.9999	8.33×10^{-51}
STD	0.0013	0.0037	0.0017	0.1597	0.0123	0.0022	0.0010	0.0191	0.0063	0.0001	
Max.	0.9989	0.9953	0.9899	0.6368	0.9818	0.9516	0.9890	0.9178	0.9874	1.0000	
Min.	0.9939	0.9795	0.9828	0.0000	0.9350	0.9433	0.9845	0.8418	0.9597	0.9995	
<i>ZDT2</i>											
Mean	0.9977	0.9421	0.9876	0.4463	0.9782	0.9239	0.9857	0.8559	0.9839	0.9992	1.41×10^{-46}
STD	0.0012	0.0987	0.0013	0.1759	0.0072	0.0051	0.0015	0.0175	0.0044	0.0002	
Max.	1.0000	0.9975	0.9913	0.6999	0.9858	0.9353	0.9898	0.8828	0.9915	0.9995	
Min.	0.9942	0.6981	0.9853	0.0000	0.9545	0.9134	0.9828	0.8246	0.9717	0.9987	
<i>ZDT3</i>											
Mean	0.9726	0.9751	0.9861	0.2876	0.9648	0.9639	0.9890	0.8272	0.9704	0.9996	5.19×10^{-44}
STD	0.0184	0.0247	0.0018	0.1074	0.0257	0.0034	0.0012	0.0200	0.0111	0.0004	
Max.	0.9909	0.9950	0.9900	0.4660	0.9876	0.9689	0.9921	0.8632	0.9887	1.0000	
Min.	0.9013	0.8869	0.9831	0.0000	0.8653	0.9541	0.9867	0.7902	0.9488	0.9981	
<i>ZDT4</i>											
Mean	0.8801	0.5485	0.9828	0.7630	0.9941	0.4026	0.6743	0.5138	0.8463	0.9820	9.80×10^{-49}
STD	0.0453	0.1896	0.0127	0.0734	0.0038	0.1299	0.0772	0.0802	0.0550	0.0078	
Max.	0.9445	0.9053	0.9992	0.9310	1.0000	0.6253	0.7987	0.6600	0.9582	0.9957	
Min.	0.8070	0.0000	0.9564	0.6110	0.9851	0.1462	0.4743	0.3418	0.7565	0.9685	
<i>ZDT6</i>											
Mean	0.9977	0.9973	0.9951	0.9531	0.9919	0.1803	0.9988	0.2209	0.9893	0.9990	7.73×10^{-42}
STD	0.0001	0.0011	0.0045	0.1250	0.0041	0.0215	0.0010	0.1970	0.0060	0.0007	
Max.	0.9978	0.9986	0.9995	1.0000	0.9974	0.2370	0.9998	0.7714	0.9961	0.9996	
Min.	0.9976	0.9927	0.9784	0.4910	0.9800	0.1396	0.9958	0.0000	0.9688	0.9962	

Table 4

Normalised hypervolumes of all MOEAs for all constrained test problems.

Problems	MOEAs										Friedman test (p -value)
	BPBIL	UPS-EMOA	SPEA2	MPSO	NSGA-II	RPBIL	RPBIL-DE	DMOEA	DEMO	MOEA/D	
CF1											
Mean	0.8915	0.4171	0.9629	0.8978	0.8639	0.4655	0.9200	0.6463	0.3745	0.8251	5.36×10^{-42}
STD	0.0279	0.2001	0.0091	0.0825	0.1878	0.1159	0.0234	0.0241	0.0849	0.0756	
Max.	0.9364	0.9315	0.9777	1.0000	0.9744	0.7123	0.9563	0.7240	0.5545	0.9137	
Min.	0.8125	0.0000	0.9387	0.7124	0.0489	0.2330	0.8572	0.5989	0.2631	0.5911	
CF2											
Mean	0.9785	0.8765	0.9930	0.8032	0.5303	0.3676	0.9980	0.7930	0.8572	0.9677	2.29×10^{-39}
STD	0.0197	0.1240	0.0030	0.0907	0.2644	0.2866	0.0013	0.0250	0.1492	0.0260	
Max.	0.9938	0.9998	0.9990	0.9362	0.9466	0.8980	1.0000	0.8350	0.9978	0.9911	
Min.	0.9017	0.5245	0.9884	0.6257	0.0216	0.0000	0.9955	0.7320	0.6228	0.9182	
CF3											
Mean	0.6000	0.7740	0.9053	0.2238	0.9262	0.4082	0.7909	0.5361	0.8811	0.7582	1.45×10^{-42}
STD	0.0878	0.1355	0.0727	0.1210	0.0507	0.1630	0.0659	0.0366	0.0322	0.1033	
Max.	0.7579	0.9262	1.0000	0.5370	1.0000	0.6822	0.9190	0.6301	0.9324	0.9169	
Min.	0.4441	0.3729	0.7336	0.0000	0.7555	0.0939	0.6425	0.4750	0.8078	0.5397	
CF4											
Mean	0.9156	0.9008	0.9790	0.7737	0.9332	0.5008	0.9750	0.9288	0.9813	0.9187	7.35×10^{-35}
STD	0.0574	0.0737	0.0387	0.1167	0.0421	0.2968	0.0497	0.0260	0.0167	0.0597	
Max.	0.9902	0.9958	0.9994	0.9208	0.9889	0.8721	0.9982	0.9698	1.0000	0.9803	
Min.	0.8175	0.7098	0.8262	0.5072	0.8023	0.0000	0.8206	0.8524	0.9411	0.7927	
CF5											
Mean	0.5315	0.4708	0.7172	0.1020	0.7296	0.7603	0.7638	0.6296	0.7503	0.5691	2.79×10^{-30}
STD	0.1639	0.2329	0.1186	0.1215	0.1470	0.0007	0.0004	0.1397	0.0181	0.1792	
Max.	0.7744	0.8214	0.9140	0.3376	1.0000	0.7618	0.7644	0.7565	0.7646	0.8578	
Min.	0.0647	0.0000	0.4149	0.0000	0.4042	0.7592	0.7628	0.3652	0.6572	0.2209	
CF6											
Mean	0.8670	0.6565	0.9033	0.8216	0.7367	0.5429	0.8595	0.7780	0.5848	0.9056	9.26×10^{-39}
STD	0.0356	0.1646	0.0164	0.0378	0.1698	0.0465	0.1003	0.0382	0.1554	0.0636	
Max.	0.9479	0.8924	0.9403	0.8852	0.8870	0.7041	0.9685	0.8522	0.8651	1.0000	
Min.	0.7900	0.3661	0.8650	0.7429	0.0000	0.4776	0.6421	0.7105	0.2672	0.6192	
CF7											
Mean	0.7959	0.7173	0.9106	0.3208	0.8769	0.6569	0.8800	0.8180	0.9433	0.8402	1.58×10^{-31}
STD	0.0709	0.1746	0.0493	0.1631	0.0661	0.1788	0.0708	0.0397	0.0467	0.0777	
Max.	0.8829	0.9532	0.9832	0.7640	0.9768	0.8974	0.9786	0.8722	1.0000	0.9668	
Min.	0.6181	0.2648	0.7817	0.0000	0.7395	0.2056	0.7212	0.7378	0.8128	0.6437	

ZDT6 is MOEA/D while NSGA-II gives the best consistency for ZDT4. Moreover, p values for Friedman test are included in the last column of the table. The null hypothesis of the Friedman test is that the front hypervolumes obtained from the various optimisers are not different. Such a null hypothesis is rejected if p is sufficiently small (less than 0.01). From the statistical test, the hypervolumes of the 10 optimisers are different for all test problems.

For the results of multiobjective constrained test problems in Table 4, the best method according to average hypervolume for CF1 and CF5 is RPBIL-DE, while the best method for CF4 and CF7 is DEMO. For the test problems CF1, CF3, and CF6, the best methods are SPEA2, NSGA-II and MOEA/D respectively. The second best optimisers for CF1 and CF5 are RPBIL-DE and DEMO respectively, while the second best for the rest is SPEA2. The worst method for CF2, CF4, and CF6 is RPBIL, while MPSO is the worst in the cases of CF3, CF5 and CF7. For the test problem CF1, the algorithm that gives the worst results is DEMO. When considering search consistency, the most consistent method for CF2 and CF5 is RPBIL-DE, similar to the convergence rate ranking. For the test problem CF1 and CF6, the best search consistency is SPEA2 while the best for the CF3 and CF4 problems is DEMO. The Friedman test results reveal that the hypervolume values of the 10 MOEAs are different.

Further comparative studies are carried out based on the Wilcoxon rank sum test in order to make the ranking of MOEAs. The null hypothesis is that, for a particular test problem, the 30 hypervolume values obtained from method I have the same median as those obtained from method J at the 5% significant level. Table 5 shows the ranking of MOEAs for the ZDT1 problem. Firstly, the matrix sized 10×10 , whose elements are full of zeros is constructed. If method I has significantly higher hypervolume median than method J based on the Wilcoxon rank sum test, element IJ of the matrix is set to be '1'. Having compared all the methods, the best optimiser is one that gives the highest score (summed up on each column). The ranking is also given in the table. The overall rankings for both unconstrained and constrained test problems are given in Table 6. MOEA/D is the best for the unconstrained problems while the second best is BPBIL. The third best methods are SPEA2 and RPBIL-DE. For the constrained problems, the best method is SPEA2 with RPBIL-DE and NSGAII being the second and third best methods respectively.

Table 5

Ranking of MOEAs for ZTD1 based on Wilcoxon rank sum test.

	BPBIL (1)	UPSEMOA (2)	SPEA2 (3)	MPSO (4)	NSGA-II (5)	RPBIL (6)	RPBIL-DE (7)	DMOEa (8)	DEMO (9)	MOEA/D (10)
1	0	0	0	0	0	0	0	0	0	1
2	1	0	0	0	0	0	0	0	0	1
3	1	0	0	0	0	0	0	0	0	1
4	1	1	1	0	1	1	1	1	1	1
5	1	1	1	0	0	0	1	0	1	1
6	1	1	1	0	1	0	1	0	1	1
7	1	0	0	0	0	0	0	0	0	1
8	1	1	1	0	1	1	1	0	1	1
9	1	1	1	0	0	0	1	0	0	1
10	0	0	0	0	0	0	0	0	0	0
Σ	8	5	5	0	3	2	5	1	4	9
Rank	2	3	3	10	7	8	3	9	6	1

Table 6

Ranking of MOEAs for all test problems based on Wilcoxon rank sum test.

	BPBIL (1)	UPSEMOA (2)	SPEA2 (3)	MPSO (4)	NSGA-II (5)	RPBIL (6)	RPBIL-DE (7)	DMOEa (8)	DEMO (9)	MOEA/D (10)
ZDT1	2	3	3	10	7	8	3	9	6	1
ZDT2	2	6	3	10	6	8	4	9	5	1
ZDT3	4	4	3	10	4	8	2	9	4	1
ZDT4	4	8	2	6	1	10	7	8	5	2
ZDT6	3	5	6	3	7	9	1	9	7	1
Σ	15	26	17	39	25	43	17	44	27	6
Rank	2	5	3	7	4	8	3	9	6	1
CF1	5	8	1	2	2	8	2	7	10	6
CF2	3	5	2	7	9	10	1	8	5	3
CF3	7	4	1	10	1	9	4	8	3	4
CF4	4	8	1	9	4	10	1	4	1	4
CF5	8	8	3	10	1	3	1	6	5	6
CF6	3	8	1	5	6	10	3	6	9	1
CF7	7	8	2	10	4	8	2	6	1	5
Σ	37	49	11	53	27	58	14	45	34	29
Rank	6	8	1	9	3	10	2	7	5	4

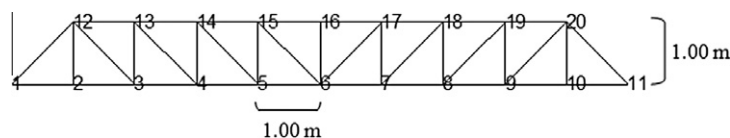
In summary, RPBIL fails to solve some multiobjective unconstrained test problems and multiobjective constrained test problems, which is similar to MPSO. However, when incorporating DE crossover and mutation into the main search procedure of RPBIL, the performance is greatly improved for both unconstrained and constrained cases. It can be said that RPBIL-DE is among the top performers based on the test problems.

5. Truss design problems

Although, many benchmark functions have been presented to verify the performance of MOEAs in the literature [25,39,40,53,56], there is no guarantee that the comparative performance results from those studies can be applied to other design problems, particularly real world applications. Therefore, when new design problems for MOEAs are introduced, comparative performance of EAs always needs to be investigated. As this work is aimed at developing a MOEA for truss design, five multiobjective design cases of 2D and 3D truss structures were proposed and used as test problems. The design problems were set to minimise structural mass and compliance, subject to stress constraints, and are as follows:

Case I: A 2D simple supported 37 bar truss

The truss structure is illustrated in Fig. 2 [17]. The structure is subject to a vertical load -10 kN at nodes 2, ..., 10 and design variables include all bar element cross-sectional areas and positions in the x and y directions of nodes 2, 3, 4, 5, 7,

**Fig. 2.** A 2D simple supported 37 bar truss.

8, 9, 10, 12, 13, 14, 15, 17, 18, 19, 20. The vertical positions of nodes 6 and 16 are also set as design variables. The design variables are treated as symmetric with respect to the y axis. Material density and modulus of elasticity are 7971.810 kg/m^3 and 210 GPa respectively. The allowable stress is set to be 400 MPa . The constrained multiobjective optimisation problem is proposed to find structural shape and element sizes while minimising mass and compliance, whereas constraints are set to avoid structural failure due to applied stresses.

$$\begin{aligned} \text{Min : } & [f_1(\mathbf{x}), f_2(\mathbf{x})] \\ \text{Subject to } & |\sigma_{\max}| \leq \sigma_y \\ & 1.77 \leq A_i \leq 113.09 \text{ cm}^2 \\ & -0.45 \leq X_i \leq 0.45 \text{ m} \\ & -0.45 \leq Y_i \leq 0.45 \text{ m} \end{aligned}$$

where $\mathbf{x} = \{x_1, \dots, x_{37}\}^T$ is a design vector, $f_1(\mathbf{x})$ is the structural mass, $f_2(\mathbf{x})$ is the structural compliance, $\sigma_{\max}$ is the maximum stress on the structure, and A_i are the cross-sectional areas of all bar elements. X_i are the x -direction positions of all nodes except for node 1 and node 11 (changes in X_i are symmetric about the y axis). Y_i are the y -direction positions of all nodes except for node 1 and node 11 (changes in Y_i are symmetric about the y axis).

Case II: Sizing design of a 2D 200 bar truss

The truss structure for Case II is illustrated in Fig. 3 [45]. The structure is subject to a vertical load of -45 kN at nodes 1, ..., 6, 8, 10, 12, 14, ..., 20, 22, 24, ..., 75, and a horizontal load of 4.5 kN at nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, and 71. Design variables include cross-sectional areas of all bar elements. The design variables are treated as being symmetric about the y axis. Material density and modulus of elasticity are 7833 kg/m^3 and 206.9 GPa respectively, while the allowable stress is 65.95 MPa . The constrained multiobjective optimisation problem is proposed so to find element sizes while minimising mass and compliance, subject to stress constraints, which can be expressed as:

$$\begin{aligned} \text{Min : } & [f_1(\mathbf{x}), f_2(\mathbf{x})] \\ \text{Subject to } & |\sigma_{\max}| \leq \sigma_y \\ & 0.636 \leq A_i \leq 132.73 \text{ cm}^2 \end{aligned}$$

where $\mathbf{x} = \{x_1, \dots, x_{84}\}^T$ is a design vector, $f_1(\mathbf{x})$ is a function of mass, $f_2(\mathbf{x})$ is a function of compliance, $\sigma_{\max}$ is the maximum stress on the structure, and A_i are elements' cross-sectional areas.

Case III: Shape and sizing design of a 2D 200 bar truss

The truss in Case II is further designed with shape and sizing design variables. Apart from the cross-sectional areas of all bar elements in Case II, the positions in the y -direction of all truss levels are set as design variables. The constrained multiobjective optimisation problem can be expressed as:

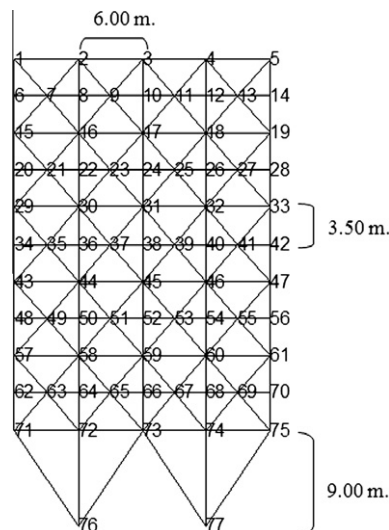


Fig. 3. A 2D 200 bar truss.

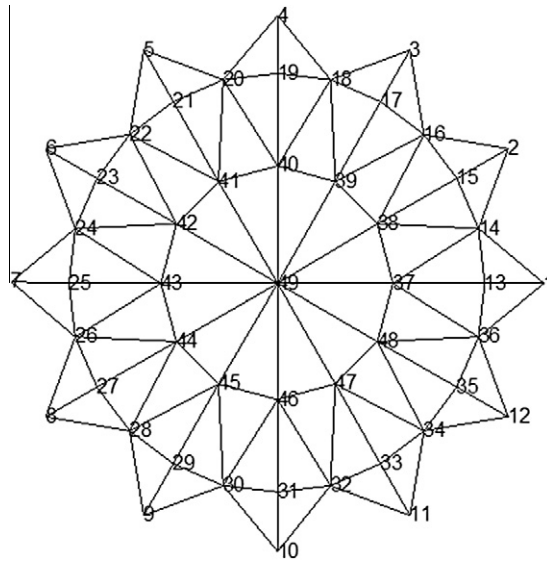


Fig. 4a. Top view of a 3D Dome 120 bar truss.

$$\begin{aligned}
 \text{Min : } & [f_1(\mathbf{x}), f_2(\mathbf{x})] \\
 \text{Subject to } & |\sigma_{\max}| \leq \sigma_y \\
 & 0.636 \leq A_i \leq 132.73 \text{ cm}^2 \\
 & -1.5 \leq Y_i \leq 1.5 \text{ m for } i > 1 \\
 & -7 \leq Y_i \leq 1.5 \text{ m for } i = 1
 \end{aligned}$$

where $\mathbf{x} = \{x_1, \dots, x_{94}\}^T$ is a design vector. Y_i is the value used to modify the height of level i from its original height as in Case II.

Case IV: Sizing design of a 3D Dome 120 bar truss

The truss structure is illustrated in Fig. 4 [19–21]. The structure is subject to a vertical load -10 kN at nodes 13, ..., 36, -30 kN at nodes 37, ..., 49, and -60 kN at node 49. The design variables include the cross-sectional areas of all elements. Material density and modulus of elasticity are set to be 7971.810 kg/m^3 and 210 GPa respectively whereas the stress limit is 400 MPa . The constrained multiobjective optimisation problem can be expressed as:

$$\begin{aligned}
 \text{Min : } & [f_1(\mathbf{x}), f_2(\mathbf{x})] \\
 \text{Subject to } & |\sigma_{\max}| \leq \sigma_y \\
 & 1.77 \leq A_i \leq 132.73 \text{ cm}^2
 \end{aligned}$$

where $\mathbf{x} = \{x_1, \dots, x_{120}\}^T$ is a design vector.

In practice, the sizes of the elements are set to be symmetric; nevertheless, structural symmetry is not considered in this work since we need a design problem with a large number of design variables for MOEA performance test.

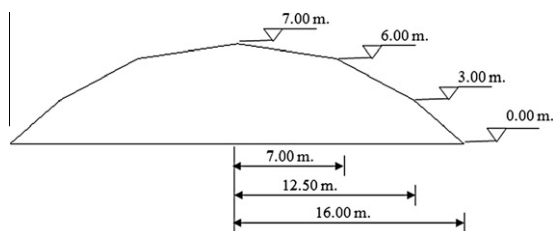


Fig. 4b. Side view of a 3D Dome 120 bar truss.

Table 7

Summary of design parameters for all truss design problems.

Parameters	No. of design variables	Population size	No. of generations	Archive size
CASE I	37	30	30	500
CASE II	84	50	50	500
CASE III	94	50	50	500
CASE IV	120	50	60	500
CASE V	123	50	60	500

Case V: Shape and sizing design of a 3D Dome 120 bar truss

This test problem is similar to Case IV with the addition of nodal positions in the z-direction as shape design variables. Additional variables determine the heights of the dome levels except for the ground level with nodes 1, ..., 12. The constrained multiobjective optimisation problem can be expressed as:

$$\begin{aligned}
 \text{Min : } & [f_1(\mathbf{x}), f_2(\mathbf{x})] \\
 \text{Subject to } & |\sigma_{\max}| \leq \sigma_y \\
 & 1.77 \leq A_i \leq 132.73 \text{ cm}^2 \\
 & -1.5 \leq Z_i \leq 1.5 \text{ m}
 \end{aligned}$$

where $\mathbf{x} = \{x_1, \dots, x_{123}\}^T$ is a design vector. Z_i is used to adjust the height of level i from its original position as shown in Fig. 4b.

Each optimisation method is used to solve each design problem over 30 optimisation runs. The Pareto archive of each algorithm is set to be 500 except for NSGA-II and DEMO where non-dominated solutions are stored in a population matrix. All MOEAs except UPS-EMOA use a population size of 30 and 30 iterations for Case I, population size of 50 and 50 iterations for Cases II and III, and population size of 50 and 60 iterations for Cases IV and V. The UPS-EMOA methods use different population sizes and numbers of iterations but their search procedures are terminated at the same total number of function evaluations for all design problems. Table 7 gives the summary of the design parameters for all truss design problems. For all design cases, all design variables are encoded in a DE vector.

Table 8

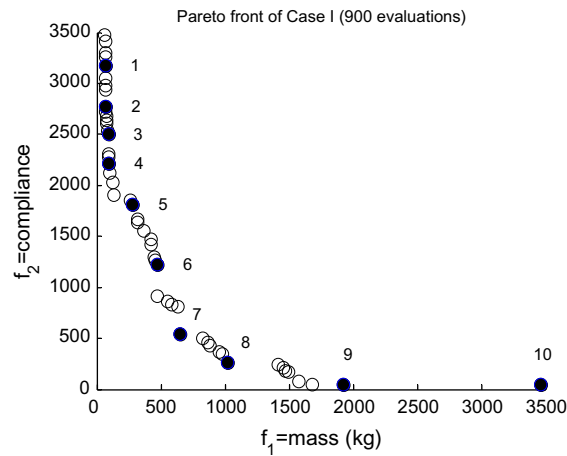
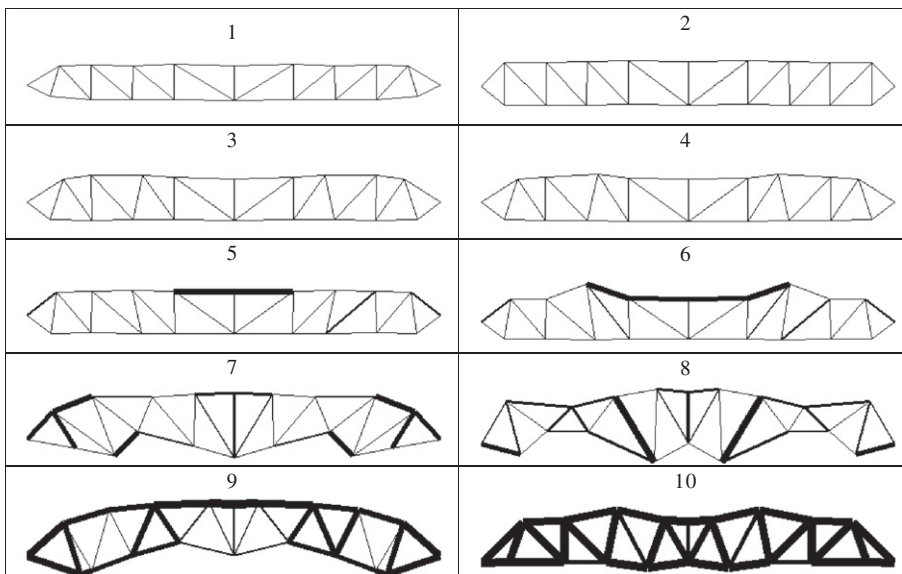
Normalised hypervolumes of all MOEAs for all truss design problems.

Test problems	MOEAs										Friedman (p -value)
	BPBIL	UPS-EMOA	SPEA2	MPSO	NSGA-II	RPBIL	RPBIL-DE	DMOEA	DEMO	MOEA/D	
<i>Case I</i>											
Mean	0.6074	0.8654	0.5069	0.2552	0.4891	0.9831	0.9882	0.4163	0.9193	0.4188	4.31×10^{-46}
STD	0.0904	0.0742	0.1047	0.1124	0.1127	0.0009	0.0010	0.1085	0.0655	0.1348	
Max.	0.7739	0.9913	0.7174	0.4566	0.7017	0.9854	0.9900	0.6423	1.0000	0.6369	
Min.	0.4278	0.6618	0.3092	0.0000	0.2863	0.9815	0.9865	0.1859	0.7654	0.1279	
<i>Case II</i>											
Mean	0.3484	0.5846	0.3156	0.1251	0.3805	0.4253	0.9627	0.2132	0.6256	0.2185	7.14×10^{-48}
STD	0.0777	0.0986	0.0839	0.0820	0.0658	0.0735	0.0208	0.0713	0.0699	0.0677	
Max.	0.5449	0.8015	0.5826	0.2790	0.4940	0.6582	1.0000	0.3522	0.7447	0.3535	
Min.	0.1925	0.4244	0.1118	0.0000	0.2098	0.3256	0.9148	0.0011	0.4810	0.0990	
<i>Case III</i>											
Mean	0.3818	0.5608	0.2531	0.1133	0.3311	0.4012	0.9663	0.2183	0.5840	0.1641	3.67×10^{-47}
STD	0.0684	0.0900	0.0784	0.0700	0.0709	0.0834	0.0178	0.0794	0.0723	0.0688	
Max.	0.5026	0.7374	0.4436	0.3254	0.4648	0.6471	1.0000	0.3487	0.7891	0.2872	
Min.	0.2177	0.3394	0.1030	0.0000	0.2309	0.2972	0.9232	0.0958	0.4681	0.0014	
<i>Case IV</i>											
Mean	0.1499	0.3808	0.1635	0.0414	0.2204	0.9381	0.9807	0.1113	0.3118	0.1325	4.90×10^{-47}
STD	0.0492	0.0532	0.0559	0.0292	0.0548	0.0026	0.0044	0.0308	0.1022	0.0527	
Max.	0.2577	0.5062	0.2898	0.0975	0.3578	0.9444	1.0000	0.1828	0.5866	0.2388	
Min.	0.0721	0.2783	0.0617	0.0000	0.1220	0.9332	0.9742	0.0508	0.1599	0.0143	
<i>Case V</i>											
Mean	0.5617	0.6781	0.5083	0.4256	0.5532	0.5091	0.7558	0.4785	0.4771	0.4158	3.82×10^{-37}
STD	0.0360	0.0436	0.0249	0.0359	0.0231	0.1364	0.0805	0.0285	0.0676	0.1039	
Max.	0.6199	0.7438	0.5755	0.4903	0.5949	0.7857	1.0000	0.5430	0.5828	0.5316	
Min.	0.4939	0.5846	0.4749	0.3639	0.4843	0.3180	0.5971	0.4174	0.3191	0.0000	

Table 9

Ranking of MOEAs for all truss design problem based on Wilcoxon rank sum test.

	BPBIL (1)	UPSEMOA (2)	SPEA2 (3)	MPSO (4)	NSGA-II (5)	RPBIL (6)	RPBIL-DE (7)	DMOE (8)	DEMO (9)	MOEA/D (10)
CASEI	5	4	6	10	6	2	1	8	3	8
CASEII	5	2	7	10	5	4	1	8	2	8
CASEIII	4	2	7	10	6	4	1	7	2	9
CASEIV	6	3	6	10	5	2	1	9	4	8
CASEV	3	2	6	9	3	5	1	7	7	9
Σ	23	13	32	49	25	17	5	39	18	42
Rank	5	2	7	10	6	3	1	8	4	9

**Fig. 5.** Pareto front of design Case I using RPBIL-DE.**Fig. 6.** Truss structures according to the front in Fig. 5.

6. Comparative results

After having performed each optimiser to solve each test problem for 30 optimisation runs, with 900 function evaluations for Case I, 2500 function evaluations for Cases II and III, and 3000 function evaluations for Cases IV and V, the comparative

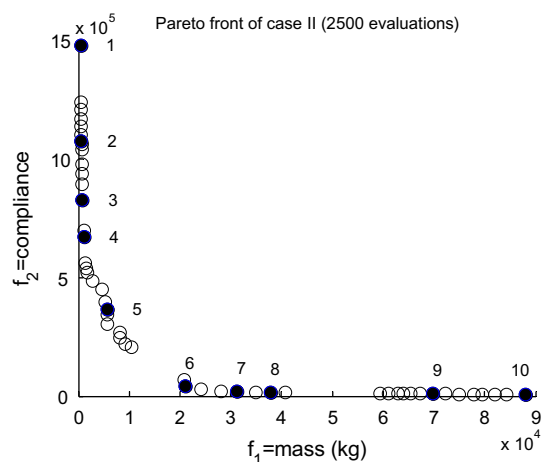


Fig. 7. Pareto front of design Case II using RPBIL-DE.

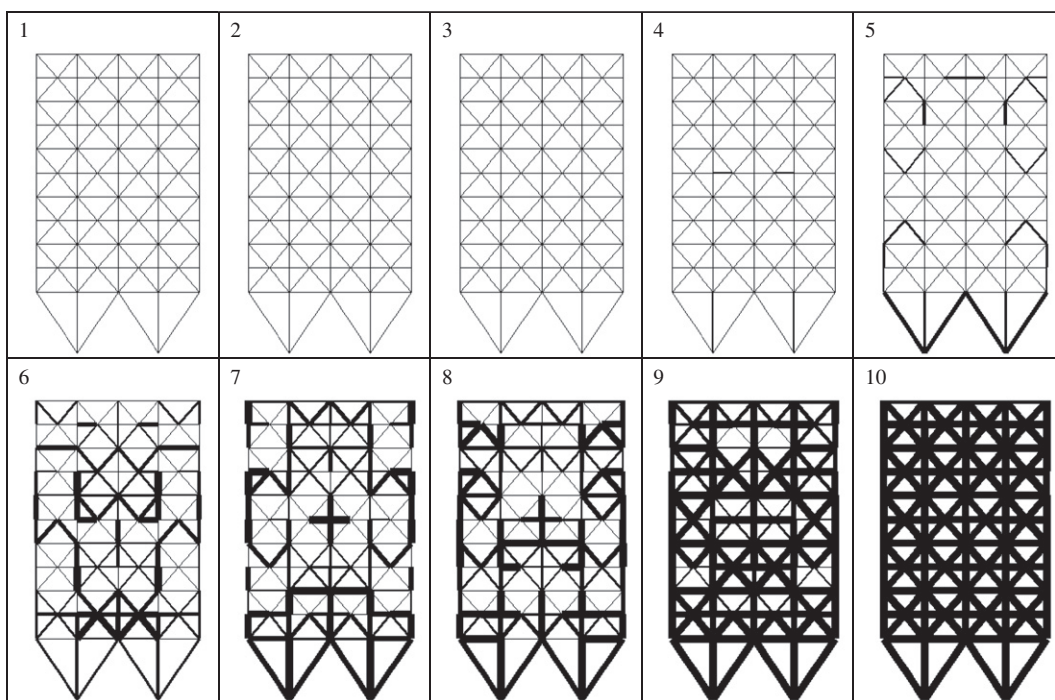


Fig. 8. Truss structures according to the front in Fig. 7.

results based on the hypervolume indicator are shown in Table 8. Similarly to the performance test in Section 4, the hypervolume values are normalised for ease in comparison. The mean value of the front hypervolumes has been used to measure the algorithm's convergence rate, while the standard deviation represents the search consistency of the methods.

For all design problems, the comparative results give similar conclusions in cases of convergence rate measurement. The clear cut best performer is RPBIL-DE whereas the worst performer is MPSO, except in design Case V, where the worst performer is MOEA/D. The second best method for the design Cases I and IV is RPBIL, while the second best for design Cases II and III is DEMO. For design Case V, UPS-EMOA has the second best convergence. For the measure of search consistency, RPBIL-DE is the most consistent method for design Cases II and III while RPBIL is the most consistent method for design Cases I and IV. For design Case V, the best consistent method is NSGA-II. The best non-dominated front for design Case I is produced by DEMO, while RPBIL-DE produces the best fronts for the other design problems. The Friedman test results confirm that the hypervolumes obtained from the various optimisers are different for all truss design problems. The ranking of MOEAs based

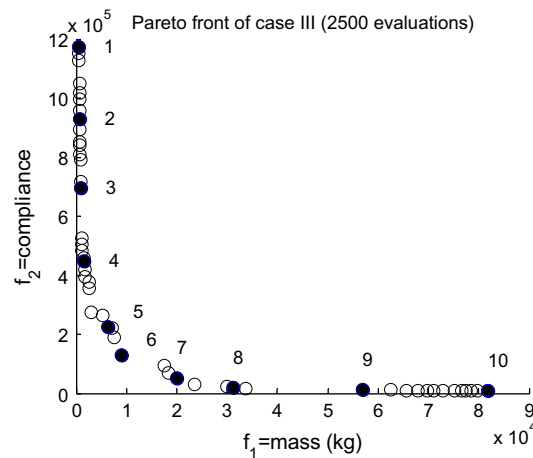


Fig. 9. Pareto front of a design Case III using RPBIL-DE.

on the Wilcoxon rank sum test is given in Table 9. The absolute best for multiobjective truss design is the proposed hybrid algorithm RPBIL-DE while the second and third best methods are UPS-EMOA and RPBIL respectively. RPBIL is said to be more efficient than the other MOEAs for some truss design problems. However, after integrating DE operators into the RPBIL procedure, performance was even better for all design problems.

Fig. 5 shows the best Pareto front of Case I obtained from using RPBIL-DE. Some solution points are selected and the corresponding truss structures are displayed in Fig. 6. The obtained structures have various shapes and element sizes. For Cases II and III, the best Pareto fronts obtained from using RPBIL-DE are shown Figs. 7 and 9 respectively. Some solution points in the non-dominated fronts are selected and their corresponding truss structures are illustrated in Figs. 8 and 10. In Figs. 11 and 13, Pareto fronts of Cases IV and V obtained from using RPBIL-DE are displayed. Some solutions are selected and the corresponding truss structures are illustrated in Figs. 12 and 14 respectively. The obtained structures are not symmetric with respect to the z axis as all bar element cross-sectional areas are assigned as design variables. However, in practice, structural symmetry has to be taken into consideration.

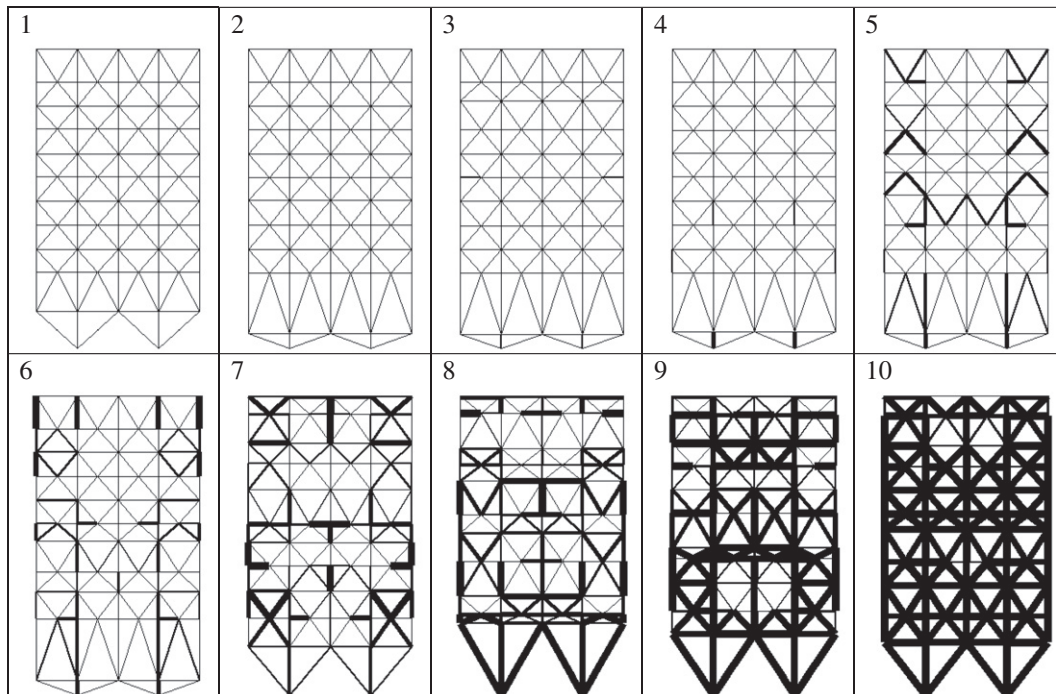


Fig. 10. Truss structures according to the front in Fig. 9.

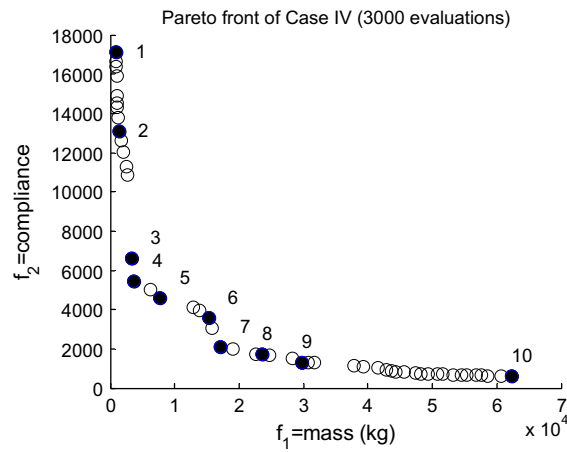


Fig. 11. Pareto front of design Case IV using RPBIL-DE.

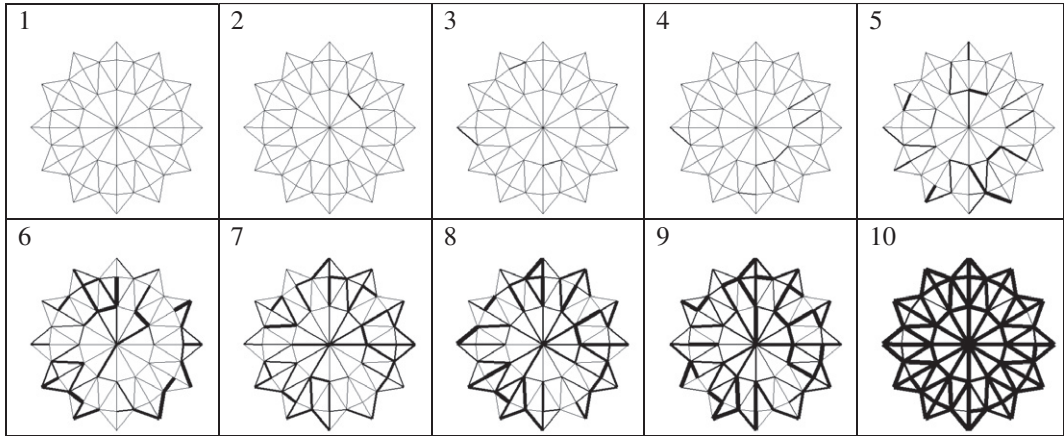


Fig. 12. Truss structures according to the front in Fig. 11.

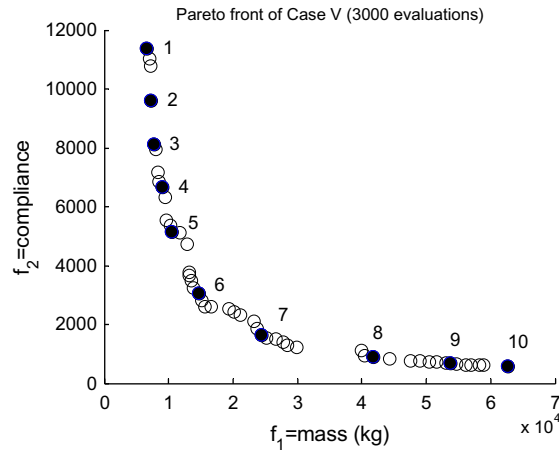


Fig. 13. Pareto front of design Case V using RPBIL-DE.

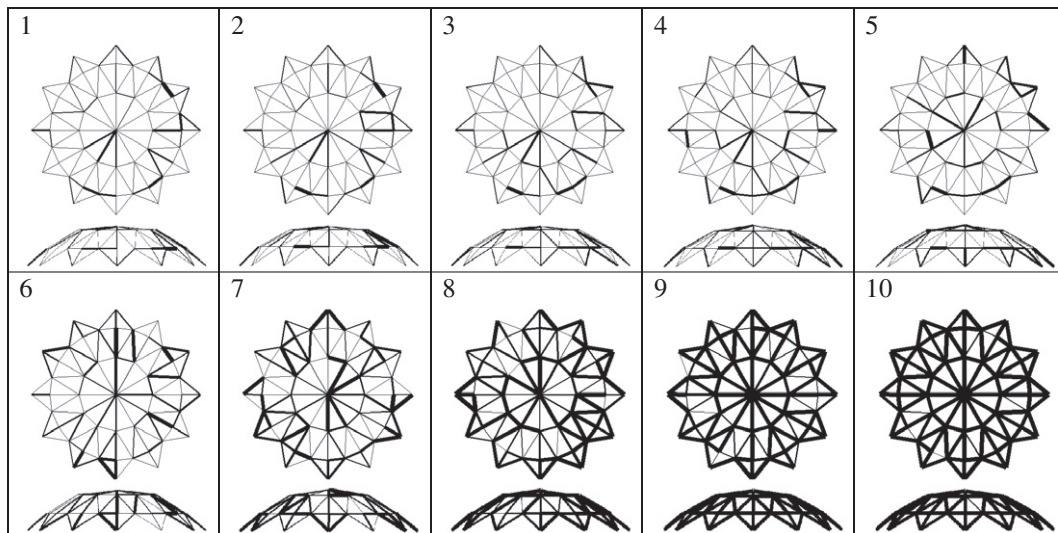


Fig. 14. Truss structures according to the front in Fig. 13.

7. Conclusions and discussion

We have proposed a new multiobjective evolutionary algorithm, whereby Real-code population-based incremental learning was extended to solve multiobjective design problems. Differential evolution operators were integrated into the main procedure of RPBIL leading to a hybrid algorithm. The performance test was conducted on multiobjective unconstrained and constrained test functions, and truss optimisation design problems. The results have shown that the RPBIL is inefficient for solving some multiobjective unconstrained and constrained test problems. However, after incorporating DE crossover and mutation into the main search procedure of RPBIL, the performance was greatly improved for both types of test problems. The hybrid RPBIL-DE was among the best MOEAs employed in this paper. For truss design test cases, RPBIL is said to be more efficient than the other MOEAs for some truss design problems. However, after integrating DE operators into RPBIL procedure, performance was greatly improved for all design problems. The proposed hybrid RPBIL-DE outperformed the other MOEAs in all design test cases. It can be said that the proposed hybrid RPBIL-DE is a powerful tool for multiobjective constrained optimisation and especially for truss optimisation design. It should be noted that the comparative studies in this work rely on optimised parameter settings. The use of self-adaptive control parameters for MOEAs may improve their performance. Our future work will be to investigate the use of RPBIL-DE in other real-world engineering design problems.

Acknowledgements

The authors are grateful for the support from The Royal Golden Jubilee PhD Program (grant no. PHD/0041/2551), and the Thailand Research Fund (grant no. BRG5580017).

References

- [1] T. Aittokoski, K. Miettinen, Efficient evolutionary approach to approximate the Pareto-optimal set in multiobjective optimization, *UPS-EMOA, Optimization Method and Software* 25 (6) (2010) 841–858.
- [2] S. Baluja, *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*, Technical Report CMU-CS-95_163, Carnegie Mellon University, 1994.
- [3] S. Bandyopadhyay, S. Saha, U. Maulik, K. Deb, A simulated annealing-based multiobjective optimization algorithm: AMOSA, *IEEE Transactions on Evolutionary Computation* 12 (3) (2008) 269–283.
- [4] J. Brest, M. Maučec, Self-adaptive differential evolution algorithm using population size reduction and three strategies, *Soft Computing – A Fusion of Foundations, Methodologies and Applications* 15 (11) (2011) 2157–2174.
- [5] S. Bureerat, K. Sriwornas, Population-based incremental learning for multiobjective optimisation, *Advances in Soft Computing* 9 (2007) 223–231.
- [6] S. Bureerat, Hybrid population-based incremental learning using real codes, *Lecture Notes in Computer Science* 6683/2011 (2011) 379–391.
- [7] S. Bureerat, Improved population-based incremental learning in continuous spaces, *Advances in Intelligent and Soft Computing* 96/2011 (2011) 77–86.
- [8] W. Chu, X. Gao, S. Sorooshian, A new evolutionary search strategy for global optimization of high-dimensional problems, *Information Sciences* 181 (22) (2011) 4909–4927.
- [9] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, *IEEE Transactions on Evolutionary Computation* 15 (1) (2011) 4–31.
- [10] K. Deb, *Multi-objective Optimization using Evolutionary Algorithms*, Wiley, 2001.
- [11] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGAII, *IEEE Transactions on Evolutionary Computation* 6 (2002) 182–197.
- [12] I. Durgun, A.R. Yildiz, Structural design optimization of vehicle components using cuckoo search algorithm, *Materials Testing* 54 (3) (2012) 185–188.
- [13] M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, Evolving cognitive, social experience in particle swarm optimization through differential evolution: a hybrid approach, *Information Sciences* 216 (2012) 50–92.

- [14] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, Y. Alizadeh, Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems, *Computer Methods in Applied Mechanics and Engineering* 197 (33–40) (2008) 3080–3091.
- [15] C.M. Fonseca, P.J. Fleming, Genetic algorithms for multiobjective optimization: formulation, discussion and generalization, in: *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, pp. 416–423.
- [16] M.B. Fuchs, A move limit strategy for the SLP approach to structural design, *International Journal of Mechanical Sciences* 23 (12) (1981) 733–740.
- [17] H.M. Gomes, Truss optimization with dynamic constraints using a particle swarm algorithm, *Expert Systems with Applications* 38 (2011) 957–968.
- [18] D.J. Gunaratnam, Adaptive stress limit approach for SLP method of structural optimization, *Computers & Structures* 21 (5) (1985) 959–968.
- [19] A. Kaveh, S. Talatahari, A hybrid particle swarm and ant colony optimization for design of truss structures, *Asian Journal of Civil Engineering (Building and Housing)* 9 (4) (2008) 329–348.
- [20] A. Kaveh, S. Talatahari, Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures, *Computers and Structures* 87 (2009) 267–283.
- [21] A. Kaveh, S. Talatahari, Size optimization of space trusses using Big Bang-Big Crunch algorithm, *Computers and Structures* 87 (2009) 1129–1140.
- [22] W. Lingyun, Z. Mei, W. Guangming, M. Guang, Truss optimization on shape and sizing with frequency constraints based on genetic algorithm, *Computational Mechanics* 35 (5) (2005) 361–368.
- [23] M. Liu, X. Zou, Y. Chen, Z. Wu, Performance assessment of DMOEA-DD with CEC 2009 MOEA competition test instances, in: *Proceeding of the IEEE Congress on Evolutionary Computation (CEC2009)*, Norway, May 2009, pp. 2913–2918.
- [24] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, *Applied Soft Computing* 11 (2) (2011) 1679–1696.
- [25] R. Mallipeddi, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2010 Competition on Constrained Real-Parameter Optimization, Technical Report, Nanyang Technological University, Singapore, 2010.
- [26] A.W. Mohamed, H.Z. Sabry, Constrained optimization based on modified differential evolution algorithm, *Information Sciences* 194 (2010) 171–208.
- [27] F. Neri, E. Mininno, Memetic compact differential evolution for cartesian robot control, *IEEE Computational Intelligence Magazine* 5 (2) (2010) 54–65.
- [28] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, *Artificial Intelligence Review* 33 (2010) 61–106.
- [29] C. Noilubla, S. Bureerat, Simultaneous topology shape and sizing optimisation of skeletal structures using multiobjective evolutionary algorithms, *Evolutionary Computation*, 2009, pp. 487–580, In-Tech, ISBN: 978-953-307-008-7 (Chapter 24).
- [30] Q.-K. Pan, L. Wang, L. Gao, W.D. Li, An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers, *Information Sciences* 181 (3) (2011) 668–685.
- [31] V. Plevris, M. Papadrakakis, A hybrid particle swarm-gradient algorithm for global structural optimization, *Computer-Aided Civil and Infrastructure Engineering* 26 (2011) 48–68.
- [32] L.X. Qian, W.X. Zhong, K.T. Cheng, Y.K. Sui, Sequential quadratic programming approach engineering structural optimisation, *Acta Mechanica Sinica* (4) (1983).
- [33] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Transactions on Evolutionary Computation* 13 (2) (2009) 398–417.
- [34] H. Rahami, A. Kaveh, Y. Gholipour, Sizing, geometry and topology optimization of trusses via force method and genetic algorithm, *Engineering Structures* 30 (9) (2008) 2360–2369.
- [35] M. Reyes-Sierra, C.A. Coello Coello, Multi-objective particle swarm optimizers: a survey of the state-of-the-art, *Inter Journal of Computational Intelligence Research* 2 (3) (2006) 287–308.
- [36] T. Robič, B. Filipič, C. Coello Coello, A. Hernández Aguirre, E. Zitzler, DEMO: differential evolution for multiobjective optimization, *Lecture Notes in Computer Science* 3410/2005 (2005) 520–533.
- [37] S. Srisomporn, S. Bureerat, Geometrical design of plate-fin heat sinks using hybridization of MOEA and RSM, *IEEE Transaction on Components and Packaging Technologies* 31 (2) (2008) 351–360.
- [38] J. Sun, Q. Zhang, E.P.K. Tsang, DE/EDA: a new evolutionary algorithm for global optimization, *Information Sciences* 169 (3–4) (2005) 249–262.
- [39] D. Swagatan, P.N. Suganthan, Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Real World Optimization Problems. Technical report, Dept. of Electronics and Telecommunication Engg., Jadavpur University, India and School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 2011.
- [40] K. Tang, Xiaodong Li, P.N. Suganthan, Z. Yang, T. Weise, Benchmark Functions for the CEC 2010 Special Session and Competition on Large Scale Global Optimization, Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2009.
- [41] P.B. Thanedar, J.S. Arora, C.H. Tseng, O.K. Lim, G.J. Park, Performance of some SQP algorithms on structural design problems, *International Journal for Numerical Methods in Engineering* 23 (12) (1986) 2187–2203.
- [42] G.N. Vanderplaats, Structural optimization by methods of feasible directions, *Computers & Structures* 3 (4) (1973) 739–755.
- [43] Y. Wang, B. Li, T. Weise, Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems, *Information Sciences* 180 (12) (2010) 2405–2420.
- [44] L. Wei, T. Tang, X. Xie, W. Shen, Truss optimization on shape and sizing with frequency constraints based on parallel genetic algorithm, *Structural and Multidisciplinary Optimization* 43 (5) (2011) 665–682.
- [45] T. Xu, W. Zuo, T. Xu, G. Song, R. Li, An adaptive reanalysis method for genetic algorithm with application to fast truss optimization, *Acta Mechanica Sinica* 26 (2010) 225–234.
- [46] A.R. Yildiz, A new design optimization framework based on immune algorithm and Taguchi's method, *Computers in Industry* 60 (8) (2009) 613–620.
- [47] A.R. Yildiz, A novel hybrid immune algorithm for global optimization in design and manufacturing, *Robotics and Computer-Integrated Manufacturing* 25 (2) (2009) 227–261.
- [48] A.R. Yildiz, A novel particle swarm optimization approach for product design and manufacturing, *The International Journal of Advanced Manufacturing Technology* 40 (5) (2009) 617–628.
- [49] A.R. Yildiz, K. Saitou, Topology synthesis of multicomponent structural assemblies in continuum domains, *Journal of Mechanical Design, Transactions of the ASME* 133 (1) (2011) 011008.1.
- [50] Q. Yuan, F. Qian, W. Du, A hybrid genetic algorithm with the Baldwin effect, *Information Sciences* 180 (5) (2010) 640–652.
- [51] A. Zamuda, J. Brest, B. Boskovic, V. Zumer, Differential evolution for parameterized procedural woody plant models reconstruction, *Applied Soft Computing* 11 (8) (2011) 4904–4912.
- [52] A. Zamuda, J. Brest, B. Boskovic, V. Zumer, Differential evolution with self-adaptation and local search for constrained multiobjective optimization, in: *Proceeding of the IEEE Congress on Evolutionary Computation (CEC2009)*, Norway, May 2009, pp. 195–202.
- [53] Q. Zhang, A. Zhou, S. Zhao, P.N. Suganthan, W. Liu, S. Tiwari, Multiobjective optimization Test Instances for the CEC 2009 Special Session and Competition (revised on 20/04/2009), Working Report, CES-887, School of Computer Science and Electrical Engineering, University of Essex, 2008.
- [54] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, *IEEE Transactions on Evolutionary Computation* 11 (6) (2007) 712–731.
- [55] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: a survey of the state of the art, *Swarm and Evolutionary Computation* 1 (1) (2011) 32–49.
- [56] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: empirical results, *Evolutionary Computation* 8 (2) (2000) 173–195.
- [57] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization, *Evolutionary Methods for Design, Optimization and Control*, Barcelona, Spain, 2002.



Performance enhancement of multiobjective evolutionary optimisers for truss design using an approximate gradient

Nantiwat Pholdee, Sujin Bureerat *

Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand

ARTICLE INFO

Article history:

Received 24 August 2011

Accepted 30 April 2012

Available online 29 May 2012

Keywords:

Hybrid algorithm

Approximate gradient

Truss structural optimisation

Multiobjective evolutionary algorithms

ABSTRACT

This paper proposes hybridisation of evolutionary algorithms (EAs) and an efficient search strategy for truss optimisation. During an optimisation process, function gradients are approximated using already explored design solutions. The approximate gradient is then employed as a local search direction. The approximate gradient operator is integrated into the main search procedure of three multiobjective evolutionary algorithms (MOEAs) leading to three hybrid optimisers. The proposed hybrid strategies along with their original MOEAs are implemented on multiobjective design of truss structures. From the comparative results, it is found that the approximate gradient operator can greatly improve the search performance of MOEAs.

© 2012 Elsevier Ltd. All rights reserved.

1. Introduction

Trusses are one of the most widely used engineering structures when requiring a lightweight and strong structure such as for bridges, buildings, domes, radio and television towers, high-voltage towers and space-based constructions. Design/optimisation of a truss structure is usually posed to find the best solutions of design variables that lead to optimised design objectives while fulfilling structural constraints. In the literature, much work on applying an optimisation method for truss structural design has been made [1–9]. Truss optimisation can be classified into three categories being shape, sizing and topology optimisation while an optimisation problem can have single and multiple design objectives. The successful development and improvement for all three design categories has been reported worldwide. In the previous studies, most optimisation problems have a single objective such as mass minimisation [2–10]. There have also been some studies related to multiobjective design cases e.g. in [1,11–13].

Recently, there have been numerous optimisation methods developed and successfully used in truss design [1–10,14]. The optimisation methods can be classified as optimisers with and without using function derivatives. The most popular and efficient strategies without using function derivatives are evolutionary algorithms. In comparison of EAs with optimisation methods using derivative (or gradient-based methods), the later are far superior in terms of convergence rate and consistency. Nevertheless, using EAs is advantageous in that: they are simple to use, robust, and capable of tackling global optimisation. The methods can deal with a wide variety of design variable types as they operate using randomisation and

need no function derivatives in searching. The multiobjective versions of EAs usually called multiobjective evolutionary algorithms [15–18] have recently gained more popularity and are attractive due to their outstanding ability to explore a non-dominated (Pareto) front within one optimisation run. However, their lack of search consistency and low convergence rate are inevitable particularly for the cases of large-scale design problems [5,6,14]. To enhance the search performance of MOEAs, many researchers have combined MOEAs with surrogate models [17,19–23]. This hybridisation approach is said to be well-established and successfully implemented on various real world applications that have expensive function evaluation. On the other hand, since the weak and strong points of MOEAs operators have been thoroughly investigated, their proper combinations can also improve MOEAs performance [6,10,24].

This paper proposes a hybrid of evolutionary algorithms and some efficient local search strategies for constrained multiobjective optimisation of trusses. The design problem is assigned to optimise mass and compliance subject to stress constraints. The multiobjective evolutionary algorithms used in this study include unrestricted population-size EMO algorithm (UPS-EMOA) [24], strength Pareto evolutionary algorithm (SPEA2) [25] and multiobjective particle swarm optimisation (MPSO) [26]. The approximate gradient operator is presented and integrated into the MOEAs procedures leading to three hybrid optimisers. The proposed hybrid strategies along with their original MOEAs are implemented on shape and sizing design of 2D and 3D truss structures. The search performance of the developed hybrid algorithms for constrained multiobjective optimisation is compared with their original strategy based upon the hypervolume indicator.

The paper is organised as follows. Section 2 gives the details of MOEAs, approximate gradient operators and the hybrid algorithm. Section 3 shows the numerical experiment and details the design

* Corresponding author.

E-mail address: sujbur@kku.ac.th (S. Bureerat).

test problems. Section 4 shows the result and discussions, and the conclusion of this study is given in Section 5.

2. Integrating an approximate gradient into a MOEA procedure

For simplicity, a bi-objective design problem with inequality and bound constraints is considered in this paper which can be expressed as:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \{f_1(\mathbf{x}), f_2(\mathbf{x})\} \\ \text{Subject to} \quad & g_j(\mathbf{x}) \leq 0 \text{ for } j = 1, \dots, m \\ & \mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u \end{aligned}$$

where $\mathbf{x}$ is a design vector sized $n \times 1$, f_1 and f_2 are design objectives, g_j are inequality constraints, and $\mathbf{x}^l$ and $\mathbf{x}^u$ are lower and upper bounds of $\mathbf{x}$ respectively.

MOEAs are said to be a well-established design technology. A great number of MOEAs have been developed and proposed [15–18,24–26] while practical engineering design problems have been solved by those optimisers. These derivative-free algorithms are popular due to some appealing advantages e.g. capability of exploring a non-dominated front within one run.

Evolutionary algorithms or population-based methods are the optimisation methods that search for optima based upon evolutionary mechanisms and randomisation. The method starts with a group of initial design solutions called a *population* and the population is then evolved, in some manner, generation to generation until reaching the optimum. The use of a population-based method for multiobjective optimisation can be achieved by combining it with the non-dominated solution technique. The procedure starts with an initial population and a bin for collecting non-dominated solutions during the search process (it is named a Pareto archive). The children are then created from performing evolutionary operations on the current population or parents. The non-dominated solutions obtained from sorting the union set of the parents, children and the non-dominated solution from the previous generation are then saved to the current Pareto archive. In case that the archive is full (which is common for multiobjective evolutionary search), some kind of Pareto archiving technique is activated to remove some unwanted non-dominated solutions. The procedure is repeated until the maximum number of iterations is reached. The general MOEA computational steps are shown as follows:

Algorithm 1 MOEA procedure.

Input: Population size, archive size, optimisation settings
Output: Non-dominated front (members in a Pareto archive)

- 1: Initialise a population and Pareto archive.
- 2: Generate offspring from the parent points using evolutionary operators.
- 3: Evaluate objective functions and constraints values.
- 4: Sort for non-dominated solutions of the combination of parent points, offspring, and the members in the Pareto archive.
- 5: Replace the members in the archive by non-dominated solutions in step 4.
- 6: If the archive is full (except for the case of UPS-EMOA, which has unrestricted archive size), an archiving technique is activated to remove some unwanted non-dominated solutions from the Pareto archive.
- 7: Update the parents for the next generation (if required).
- 8: If a termination condition is not met, go to step 2. Otherwise, stop the algorithm.

2.1. Approximate gradient

An approximate gradient is estimated from the members of a population and their corresponding objective values [27,28]. The approximate gradient is calculated exploiting the relation of a directional derivative [29]. The directional derivative of a function f in the direction of $\mathbf{s}$ at a point $\mathbf{x}$ in R^n space, denoted by df/ds , can be approximated as:

$$\mathbf{u}_s^T \nabla f = \frac{df}{ds} \approx \frac{f(\mathbf{x} + \mathbf{s}) - f(\mathbf{x})}{s} \quad (1)$$

where $\mathbf{u}_s$ is a unit vector of $\mathbf{s}$. Having the current population size N_p , $\mathbf{P} = \{\mathbf{x}_1, \dots, \mathbf{x}_{N_p}\}$, their corresponding objective values $\{f_1, \dots, f_{N_p}\}$, the current best solution $\mathbf{x}^{best}$, and its objective f^{best} , the approximate gradient of f at the point $\mathbf{x}^{best}$ can be computed based on definition (1) as [28]:

$$\begin{bmatrix} (\mathbf{x}^{best} - \mathbf{x}_1)^T \\ \vdots \\ (\mathbf{x}^{best} - \mathbf{x}_R)^T \end{bmatrix} \nabla f = \mathbf{A} \nabla f = \begin{Bmatrix} (f^{best} - f_1)^T \\ \vdots \\ (f^{best} - f_R)^T \end{Bmatrix} = \mathbf{b} \quad (2)$$

R individuals as the closest neighbours to $\mathbf{x}^{best}$ are chosen to approximate the gradient. The value R is set to be greater than n in order to prevent matrix singularity. Since the matrix $\mathbf{A}$ is not a square matrix, Eq. (2) can be solved using the pseudo-inverse operation; thus, the approximate gradient is termed a pseudo-gradient in [27]. Note that, in this paper, the system of linear Eq. (2) is solved by means of the least squares technique (operator “\” in MATLAB). For a single-objective unconstrained optimisation case, the design variables can be updated based upon the steepest descent method where the search direction is set to be $\mathbf{s} = -\nabla f$.

A solution extending from $\mathbf{x}^{best}$ along the search direction is found as follows:

$$\mathbf{x}^{new} = \mathbf{x}^{best} + \lambda \cdot \mathbf{s} \quad (3)$$

where λ is a scalar parameter which can be found by performing a line search or a one-dimensional optimiser.

Fig. 1a) shows the contour plot of a particular 2D objective function with five design points located randomly. The vectors shown in Fig. 1b) are the descent directions from using the actual gradient (the shortest vector) and approximate gradients. The approximate gradients at the point $\mathbf{x}^{best}$ are computed using different sets of its neighbours as $\{\mathbf{x}_1, \mathbf{x}_2\}$, $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$, and $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$. It can be seen that all the approximate gradients are acceptably accurate compared to the actual vector.

In cases of constrained optimisation, the evaluation of a step length λ is more complicated as there are constraint boundaries in the design problem. From the predecessor, it has been found that in most cases the optimum of the constrained optimisation of trusses is usually located at some constraint boundaries. Therefore, in this study, we will use the approximate gradient as a descent direction and the step length should lead to a new solution to be located at a constraint boundary. For a particular constraint function g , by using Taylor's series expansion, the function value of the new design solution $\mathbf{x}^{new} = \mathbf{x}^{best} + \lambda \mathbf{s}$ can be approximated as:

$$g(\mathbf{x}^{best} + \lambda \mathbf{s}) \approx g(\mathbf{x}^{best}) + \lambda \mathbf{s}^T \nabla g|_{\mathbf{x}^{best}}$$

Since it is expected that the solution will land at the boundary, we can have the relation

$$g(\mathbf{x}^{best}) + \lambda \mathbf{s}^T \nabla g|_{\mathbf{x}^{best}} = 0$$

As a result, the step length value to approximately lead the new solution to the boundary can be computed as:

$$\lambda = \frac{-g(\mathbf{x}^{best})}{\mathbf{s}^T \nabla g|_{\mathbf{x}^{best}}} \quad (4)$$

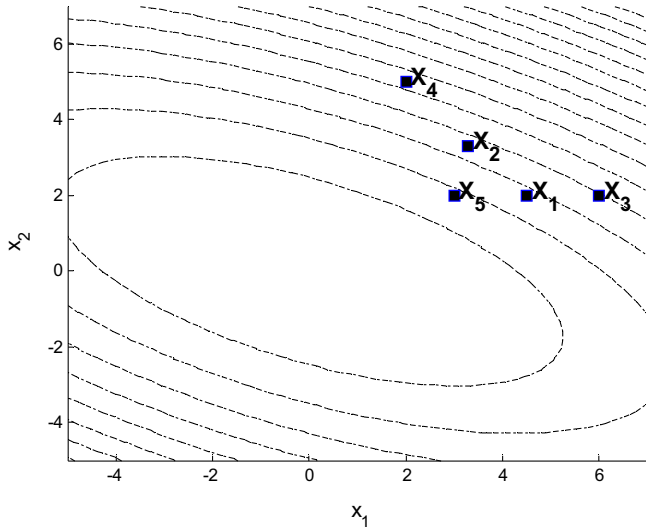


Fig. 1a. Contour plot of a 2D objective function and some individuals.

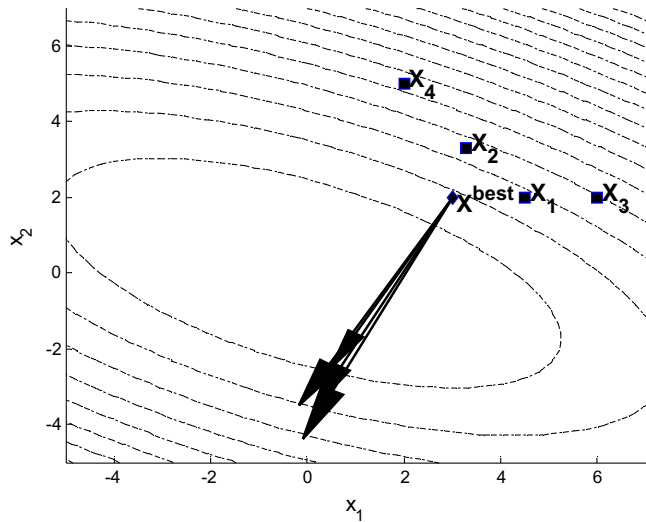


Fig. 1b. Search directions from an actual gradient and approximate gradients.

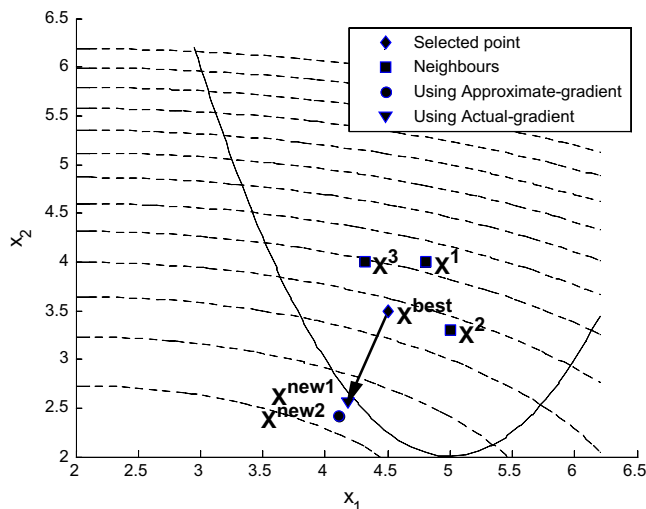


Fig. 2. Step length determination for constrained optimisation.

Fig. 2 displays how the solution updating process in Eq. (4) works, which is the case of feasible design solutions. In the figure, the contour of an objective function and a constraint boundary are illustrated. The approximate gradient is computed from using three neighbours of $\mathbf{x}^{best}$ as $\mathbf{x}^1$, $\mathbf{x}^2$ and $\mathbf{x}^3$. $\mathbf{x}^{new1}$ is an updated solution using the step length determination in (4) and the actual gradient while $\mathbf{x}^{new2}$ is a solution updated by using the approximate gradient. It can be seen that the updated solutions are located close to but not exactly on the constraint boundary. In cases that there are more design constraints which means many step lengths are obtained, the shortest step length is chosen for updating.

In cases of infeasible design solutions, the gradient of the objective function alone may not lead an updated solution to the feasible region; as a result, the objective function f is modified using an exterior penalty function technique such as:

$$f'(\mathbf{x}) = f(\mathbf{x}) + C \sum_{i=1}^m \max(0, g_i^2(\mathbf{x})) \quad (5)$$

where C is a penalty parameter to be specified. In an EA search procedure, the above updating strategy is applied to only a few best solutions in a population. For MOEA search, the best solutions are selected from the Pareto archive while the clustering technique presented in [30] is employed if there are too many members in the archive. In order to achieve more accurate gradient approximation, neighbour solutions for use in Eq. (2) should be spread throughout the design space ensuring that $\mathbf{x}^{best}$ will always have adjacent neighbours. For this requirement, a Latin hypercube sampling (LHS) technique is herein used to generate sampling points throughout the space at the initialisation stage while those design points will be used for gradient estimation for every search generation. Apart from that, the search direction also needs to be modified using the weighted sum technique. For example, in cases of bi-objective optimisation, the search direction can be obtained as

$$\mathbf{s} = -(w_1 \nabla f'_1 + w_2 \nabla f'_2) \quad (6)$$

where w_1 and w_2 are two randomly generated numbers preserving the condition $w_1 + w_2 = 1$, and f'_i is the penalised function value of f_i . The computational steps for producing some offspring from an approximate gradient are detailed in Algorithm 2 where $\mathbf{N} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ is a matrix containing the neighbour solutions for estimating gradients and $\mathbf{A} = \{\mathbf{x}_1, \dots, \mathbf{x}_{N_A}\}$ is the Pareto archive matrix. N_A is the number of solutions in the archive, and $NB \ll NP$ is the number of selected non-dominated solutions for producing offspring $\mathbf{C} = \{\mathbf{x}_1^{new}, \dots, \mathbf{x}_{NB}^{new}\}$ using an approximate gradient operator while NP is a population size.

Algorithm 2. Approximate gradient operator.

Input: neighbour solutions $\mathbf{N} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, current Pareto archive $\mathbf{A} = \{\mathbf{x}_1, \dots, \mathbf{x}_{N_A}\}$, and a parameter N_B

Output: offspring $\mathbf{C} = \{\mathbf{x}_1^{new}, \dots, \mathbf{x}_{NB}^{new}\}$

1: Apply the clustering technique to select N_B best solutions from $\mathbf{A}$, $\mathbf{B} = \{\mathbf{x}_1^{best}, \dots, \mathbf{x}_{NB}^{best}\}$

2: For $i = 1$ to N_B

3: Find the closest $R > n$ neighbours of $\mathbf{x}_i^{best}$ from the set $\mathbf{N}$.

4: Approximate $\nabla f'_1$, $\nabla f'_2$, and ∇g_j for $\mathbf{x}_i^{best}$

5: Randomly generate w_1 and w_2 , and find a search direction $\mathbf{s}$ using (6).

6: Find λ_j from g_j , ∇g_j , and $\mathbf{s}$ using Eq. (4). Then use the minimum step length $\lambda = \min(\lambda_j)$ to find $\mathbf{x}_i^{new}$ following Eq. (3).

7: If $\mathbf{x}_i^{new}$ violates bound constraints, modify it to be inside the bounds.

8: End For

2.2. Hybrid MOEAs

The hybrid optimiser is achieved by integrating Algorithm 2 into Algorithm 1. The computational procedure for hybridisation of the approximate gradient operator and the original MOEA algorithm is detailed in Algorithm 3. The process starts with an initialisation step of the original MOEA with an initial population, Pareto archive, and neighbour matrix for computing an approximate gradient. Initially, the neighbour design solutions and their function values for computing an approximate gradient are created by using LHS. Offspring are then created by using evolutionary operators. The approximate gradient operator is then activated after having non-dominated solutions. Afterwards, the new individuals produced from the evolutionary and approximate gradient operators, and the non-dominated solutions from the previous generation are put together and sorted for the non-dominated solutions. The new non-dominated solutions are used to replace the members in the archive from the previous generation. In cases that the number of non-dominated solutions exceeds the predefined archive size, an archiving technique is used to remove some solutions from the archive. The computational steps are repeated until the maximum number of function evaluations is reached.

Algorithm 3. Hybrid MOEA.

Input: Population size (N_P), archive size (N_A), neighbouring matrix size (N), parameter N_B , optimisation settings
Output: Non-dominated front (Members in a Pareto archive)

- 1: Create a neighbouring matrix $\mathbf{N} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ using LHS throughout the space $\mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u$.
- 2: Initialise a population $\mathbf{P}(t)$ and a Pareto archive $\mathbf{A}(t)$. Set a generation counter $t = 0$.
- 3: Generate offspring, $\mathbf{X}_{\text{offspring}}$, from the parent points, $\mathbf{P}(t)$, using evolutionary operators.
- 4: Evaluate objective functions and constraints values.
- 5: Given that N_F is the number of current non-dominated solutions
 - 5.1: If $N_F \leq N_B$
 - 5.1.1: Generate N_F offspring using Algorithm 2.
 - 5.2: Else
 - 5.2.1: Select N_B non-dominated solutions using the clustering technique.
 - 5.2.2: Generate N_B offspring using Algorithm 2.
- 6: Sort for non-dominated solutions of the combination of offspring from 3 and 5, and the members in the Pareto archive.
- 7: Replace the members in the archive by non-dominated solutions in step 6.
- 8: If the archive is full (except for the case of UPS-EMOA, which has unrestricted archive size), an archiving technique is activated to remove some unwanted non-dominated solutions from the Pareto archive.
- 9: Set $t = t + 1$ update $\mathbf{P}(t)$ the parents for the next generation (if required).
- 10: If a termination condition is not met, go to step 3. Otherwise, stop the algorithm.

3. Numerical experiment

In order to examine the search performance of the proposed hybrid algorithms, multiobjective optimisation problems of 2D and 3D truss structures with various design variables sizes are posed.

The test problems are set to minimise structural mass and compliance subject to stress constraints, which are detailed as follows.

3.1. Case I: A 2D simple supported 37 bar truss

The truss structure is illustrated in Fig. 3 [3]. The structure is subject to a vertical load -10 kN at nodes 2, ..., 10 where design variables include all bar element cross-sectional areas and positions in x and y directions of nodes 2, 3, 4, 5, 7, 8, 9, 10, 12, 13, 14, 15, 17, 18, 19, 20. The design variables are treated to be symmetric with respect to the x axis. Material density and modulus of elasticity are 7971.810 kg/m^3 and 210 GPa respectively. The allowable stress is set to be 400 MPa . The constrained multiobjective optimisation problem is posed to find structural shape and element sizes such that minimising mass and compliance whereas constraints are set to avoid structural failure due to applied stresses

$$\min_{\mathbf{x}} \{f_1(\mathbf{x}), f_2(\mathbf{x})\}$$

Subject to

$$\begin{aligned} |\sigma_{\max}| &\leq \sigma_y \\ 1.77 &\leq A_i \leq 113.09\text{ cm}^2 \\ -0.45 &\leq X_i \leq 0.45\text{ m} \\ -0.45 &\leq Y_i \leq 0.45\text{ m} \end{aligned}$$

where $\mathbf{x} = \{x_1, \dots, x_{37}\}^T$ is a design vector, $f_1(\mathbf{x})$ is structural mass, $f_2(\mathbf{x})$ is structural compliance, $\sigma_{\max}$ is the maximum stress on the structure, and A_i are cross-sectional areas of all bar elements. X_i are the x -direction positions of all nodes except for node 1 and node 11 (changes in X_i are symmetric about the y axis). Y_i are the y -direction positions of all nodes except for node 1 and node 11 (changes in Y_i are symmetric about the y axis).

3.2. Case II: a 2D 200 bar truss

The truss structure is illustrated in Fig. 4 [4,14]. The structure is subject to a vertical load -45 kN at nodes 1, ..., 6, 8, 10, 12, 14, ..., 20, 22, 24, ..., 75, and a horizontal load 4.5 kN at nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, and 71. Design variables include cross-sectional areas of all bar elements, and position in y -direction of all nodes except for nodes 76 and 77. The design variables are treated to be symmetric about the y axis. Material density and modulus of elasticity are 7833 kg/m^3 and 206.9 GPa respectively while the allowable stress is 65.95 MPa . The constrained multiobjective optimisation problem is posed to find structural shape and size minimising mass and compliance subject to stress constraints, which can be expressed as:

$$\min_{\mathbf{x}} \{f_1(\mathbf{x}), f_2(\mathbf{x})\}$$

Subject to

$$\begin{aligned} |\sigma_{\max}| &\leq \sigma_y \\ 0.636 &\leq A_i \leq 132.73\text{ cm}^2 \\ -1.5 &\leq Y_i \leq 1.5\text{ m for } i < 71 \\ -7 &\leq Y_i \leq 1.5\text{ m for } 71 \leq i \leq 75 \end{aligned}$$

where $\mathbf{x} = \{x_1, \dots, x_{94}\}^T$ is a design vector, $f_1(\mathbf{x})$ is a function of mass, $f_2(\mathbf{x})$ is a function of compliance, $\sigma_{\max}$ is maximum stress on a structure, A_i are elements' cross-sectional areas, and Y_i are the positions in y direction of node i . Positions of nodes on the same level are assigned to one design variable.

3.3. Case III: a 3D dome 120 bar truss

The truss structure is illustrated in Figs. 5a and 5b [7,9,14]. The structure is subject to a vertical load -10 kN at nodes 13, ..., 36,

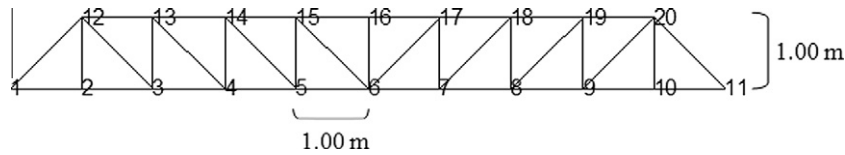


Fig. 3. A 2D simple supported 37 bar truss.

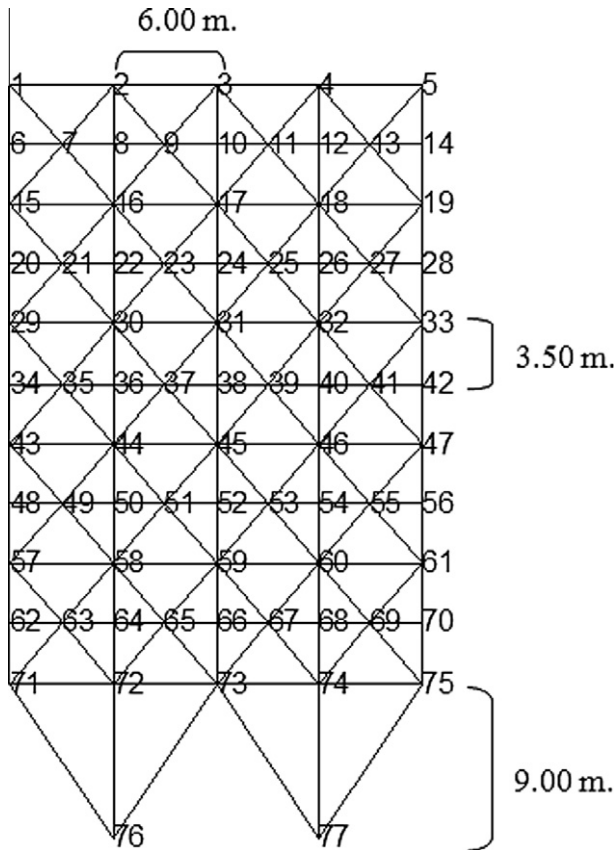


Fig. 4. A 2D 200 bar truss.

–30 kN at nodes 37, ..., 49, and –60 kN at node 49. The design variables includes the cross-sectional areas of all elements, and the z-direction positions of all nodes except for nodes 1, ..., 12. Material density and modulus of elasticity are set to be 7971.810 kg/m³ and 210 GPa respectively whereas the stress limit is 400 MPa. The constrained multiobjective optimisation problem can be expressed as:

$$\min_{\mathbf{x}} \{f_1(\mathbf{x}), f_2(\mathbf{x})\}$$

Subject to

$$|\sigma_{\max}| \leq \sigma_y$$

$$1.77 \leq A_i \leq 132.73 \text{ cm}^2$$

$$-1.5 \leq Z_i \leq 1.5 \text{ m}$$

where $\mathbf{x} = \{x_1, \dots, x_{123}\}^T$ is a vector of design variables, $f_1(\mathbf{x})$ is a function of mass, $f_2(\mathbf{x})$ is a function of compliance, $\sigma_{\max}$ is maximum stress on the structure, A_i are element cross-sectional areas, and Z_i are positions in the z direction of the selected nodes. Positions of the nodes on the same level (Fig. 4b) are assigned to one design variable. In practice, the sizes of the elements are set to be symmetric; nevertheless, structural symmetry is not considered in this work since we need a design problem with a large number of design variables as a MOEA performance test. The first design test problem is

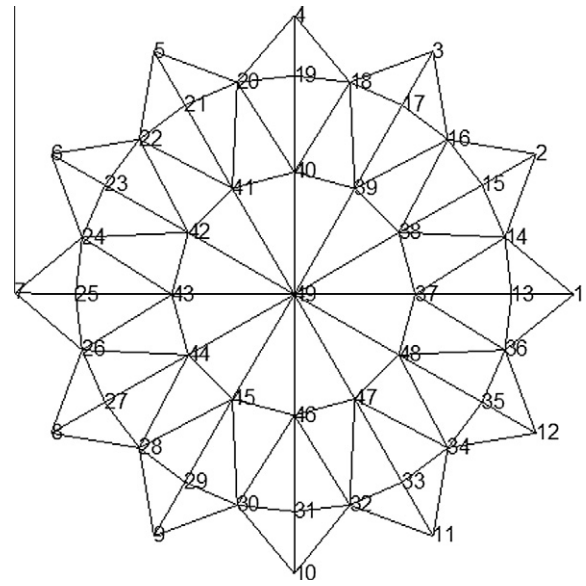


Fig. 5a. Top view of a 3D dome 120 bar truss.

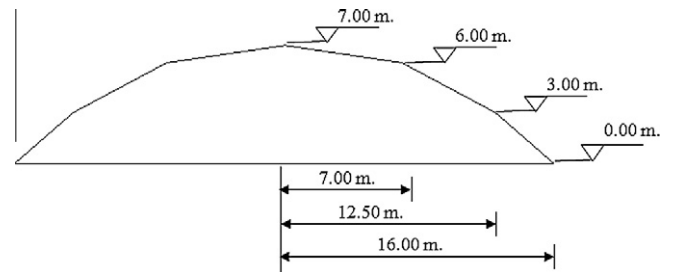


Fig. 5b. Side view of a 3D dome 120 bar truss.

said to be a small-scale problem as there are only 37 design variables while the others are considered as large-scale problems.

Three multiobjective evolutionary optimisers namely UPS-EMOA, SPEA2, and MPSO along with their hybrid counterparts (defines as Hybrid_UPS, Hybrid_SPEA2, and Hybrid_MPSO respectively) are employed to tackle the test problems. The optimisation settings for six optimisers are detailed as:

- UPS-EMOA using crossover probability, scaling factor, probability of choosing elements from offspring in crossover, minimum population size, and burst size as 0.7, 0.8, 0.5, 10, and 25 respectively.
- SPEA2 using real codes [17] with crossover and mutation rates of 1.0 and 0.1 respectively.
- MPSO using the starting inertia weight, ending inertia weight, cognitive learning factor, and social learning factor as 0.5, 0.01, 0.5 and 0.5 respectively.
- Hybrid_UPS using UPS-EMOA parameters, and $N_B = 5$. The burst size for Case II and Case III is set to be 45 so as to reduce the number of generations and consequently decrease computing time.

Table 1

Total number of function evaluations used by original and hybrid MOEAs for all design problems.

Design Problem	Original MOEAs	Hybrid versions
Case I	2250	1950
Case II	4750	4450
Case III	5500	4950

Table 2

Hypervolume comparison of all MOEAs for all test problems.

MOEAs		Original version			Hybrid version		
		UPS-EMOA	SPEA2	MPSO	UPS-EMOA	SPEA2	MPSO
Case I	Mean	0.8990	0.5711	0.2917	0.9435	0.8565	0.8049
	STD	0.0514	0.1169	0.1337	0.0250	0.1154	0.1603
	Max	1.0000	0.7827	0.5818	0.9845	0.9793	0.9279
	Min	0.8042	0.3512	0.0000	0.8808	0.4228	0.2063
Case II	Mean	0.9068	0.5067	0.2621	0.9730	0.8284	0.7193
	STD	0.0325	0.1160	0.1304	0.0136	0.0898	0.1406
	Max	0.9680	0.6932	0.5218	1.0000	0.9705	0.9345
	Min	0.8243	0.2299	0.0000	0.9467	0.5889	0.4203
Case III	Mean	0.4913	0.3334	0.1377	0.5726	0.3633	0.1855
	STD	0.0972	0.0818	0.0714	0.2380	0.0815	0.1150
	Max	0.6959	0.5120	0.2615	1.0000	0.5967	0.4345
	Min	0.2941	0.1810	0.0000	0.2218	0.2475	0.0233

Table 3

Average computing time of all MOEAs for all test problems.

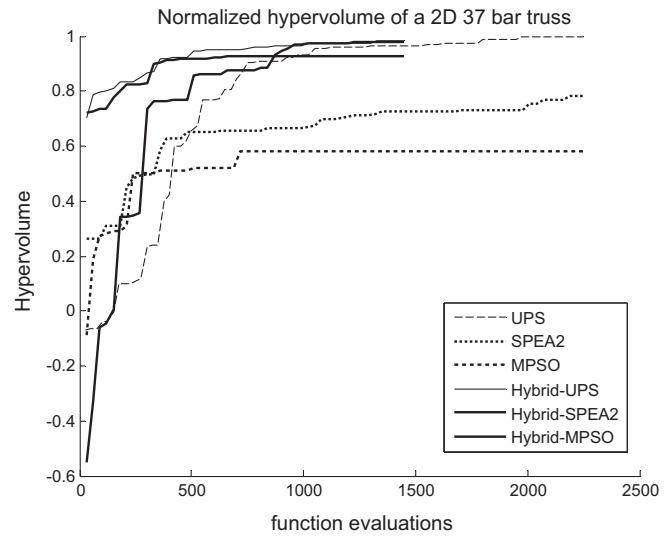
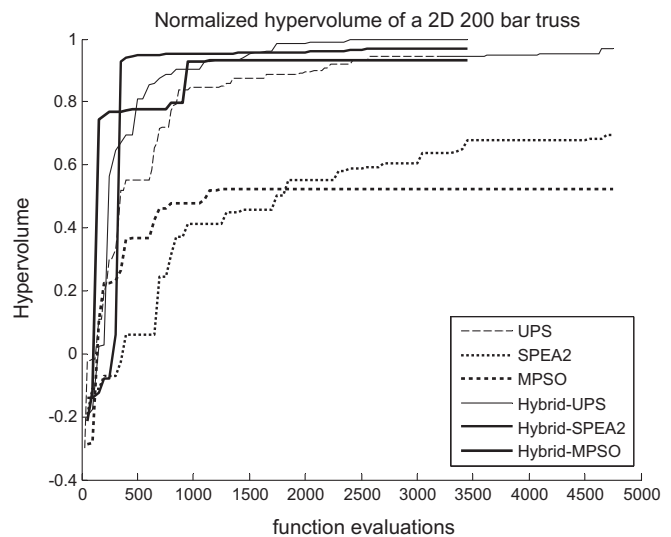
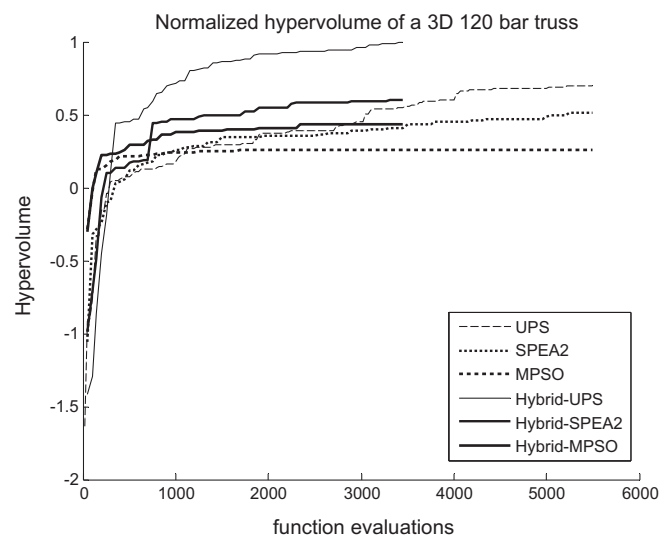
MOEAs	Case I (s)	Case II (s)	Case III (s)
UPS-EMOA	7.8720	171.4055	83.5322
SPEA2	7.8746	154.6294	59.3258
MPSO	8.9534	161.6332	72.2322
Hybrid_UPS	9.0340	151.4845	59.2062
Hybrid_SPEA2	8.5570	152.2235	58.1426
Hybrid_MPSO	8.6232	154.6392	60.7828

- Hybrid_SPEA2 using SPEA2 parameters, and $N_B = 5$.
- Hybrid_MPSO using MPSO parameters, and $N_B = 5$.

The allowable Pareto archive size of SPEA2, MPSO, and their hybrid versions are set to be 500. For all hybrid algorithms, the initial neighbouring matrix size is set to be 500 for Case I, 1000 for Case II, and 1500 for Case III. Each optimisation method is used to solve each design problem over 30 optimisation runs. SPEA2 and MPSO use: population size of 30 and 75 iterations for Case I, population size of 50 and 95 iterations for Case II, and population size of 50 and 110 iterations for Case III. UPS-EMOA uses different population size and number of iterations but their search procedures are terminated at the same maximum number of function evaluations as SPEA2 and MPSO. The penalty parameter C in Eq. (5) is set to be 100. The hybrid versions are set to use lower total numbers of function evaluations as shown in Table 1 in order that all the optimisers will take approximately equal running time for all design cases. The total numbers of function evaluations for the three test problems are limited so that we can see significant differences between MOEAs with and without using the approximate gradients.

4. Results and discussion

Having performed each optimiser to solve each test problem for 30 optimisation runs for all design cases, the comparative results based on the hypervolume indicator are shown in Table 2. Note

**Fig. 6.** Search history for Case I.**Fig. 7.** Search history for Case II.**Fig. 8.** Search history for Case III.

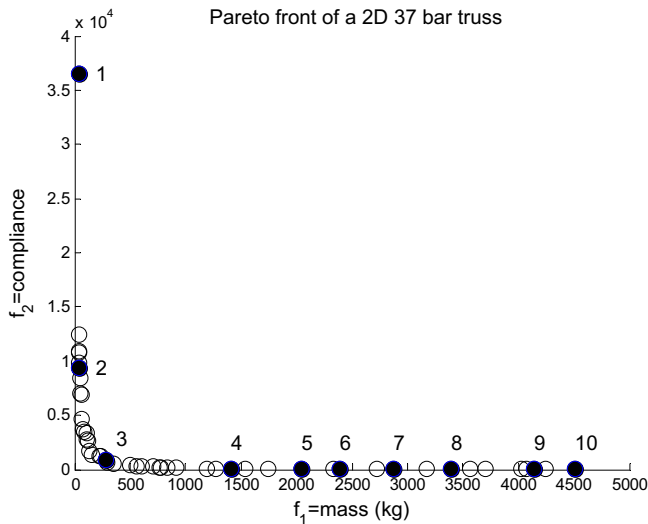


Fig. 9. Pareto front of a 2D 37 bar truss structure using Hybrid_UPS.

that those hypervolume values are normalised to ease comparison. The higher hypervolume value the better the Pareto front. With 30 runs, the mean value of the front hypervolumes will be used to measure the algorithms convergence rate while the standard deviation represents the search consistency of the methods. This means a method with lower hypervolume standard deviation (STD) has better search consistency.

The comparative results show that all hybrid optimisers outperform their original versions for all design cases. Based on the

statistical t -test at 90% confident level, all the hybrid optimisers are significantly superior to their original versions for all design cases with the exception of SPEA2 for Case III in which the hybrid SPEA2 is slightly better. The best original MOEA is UPS-EMOA while the worst is MPSO. Similar conclusions can be made for the hybrid versions using approximate gradients. For the measure of search consistency based on the standard deviation of the front hypervolume, it is somewhat inconclusive to indicate the most consistent method although the hybrid UPS-EMOA has the lowest hypervolume STD for the first two design problems. For Case I, the best non-dominated front is produced by the original UPS-EMOA while the hybrid UPS-EMOA produces the best fronts for the other design problems. The worst fronts for all design test problems are obtained from using the original MPSO.

It can be said that adding the approximate gradient operator (Algorithm 3) to the main search procedures of the original MOEAs help enhance the search performance of the original versions. The hybrid UPS-EMOA is overall the best optimiser for all design test problems. Among the original MOEAs, UPS-EMOA is the most powerful method in dealing with this type of design problem as the results obtained from this method are far superior to the other original MOEAs and it is only second to its hybrid counterpart. The worst MOEA is MPSO; however, after integrating the approximate gradient operator to the algorithm, its search performance is greatly improved for all design test problems.

The average computing time (every optimisation run is performed using the same computer) of each method when solving each design problem is given in Table 3. Note that the CPU time of all hybrid methods includes the time of performing LHS. The average computing times of the various optimisers are said to be slightly different except for the computing time of UPS-EMOA for Cases II and III. This is because the method has unrestricted Pareto

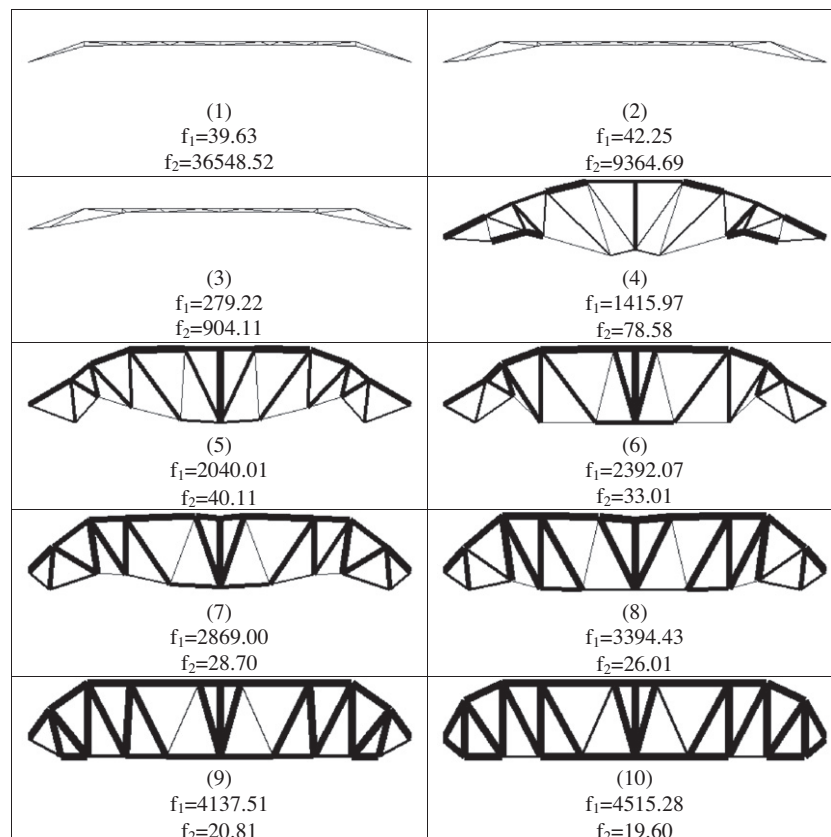


Fig. 10. Truss structures according to the front in Fig. 9.

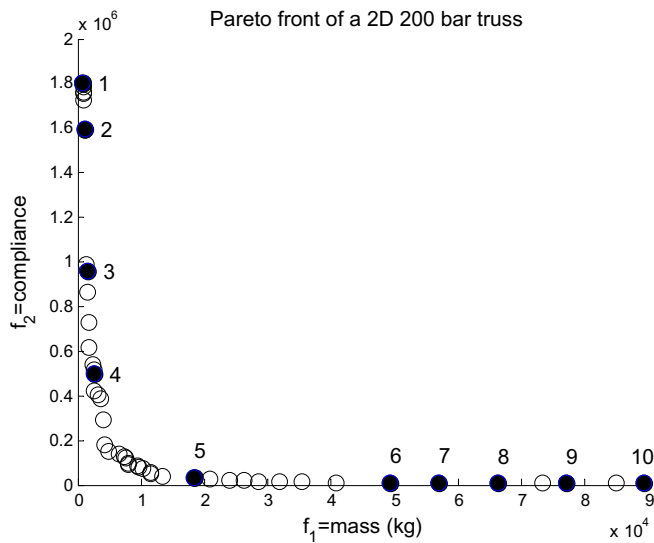


Fig. 11. Pareto front of a 2D 200 bar truss structure using Hybrid_UPS.

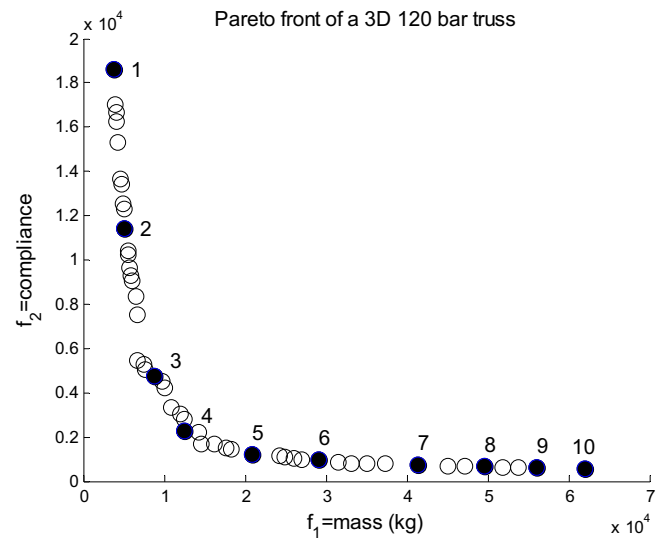


Fig. 13. Pareto front of 3D 120 bar truss structures using Hybrid_UPS.

archive size which takes more memory and time when the archive size is exceedingly large.

Fig. 6 displays the search history of the best run of the six algorithms for Case I. The plots of hypervolumes of non-dominated solutions against numbers of function evaluations are illustrated. Fig. 7 shows the search history of the best run of the six algorithms for Case II. Fig. 8 shows the search history of the best run of the six algorithms for Case III. From the figures, it can be seen that the hybrid versions of all MOEAs converge to the Pareto fronts from the early iterations. For the design Case I and Case II, the search processes of the hybrid MOEAs show better convergence rate than all original MOEAs from the beginning of the optimisation run. As the searching process continues, the UPS-EMOA is the only

original MOEA that still has non-dominated front improvement similarly to the hybrid MOEAs. For the design Case III, the original MOEAs and their hybrid versions speedily converge to the Pareto fronts while the hybrid versions have better convergence rate. As the procedure goes on for a while, the hybrid UPS-EMOA greatly improves its non-dominated front and it is better than the originals. The hybrid UPS-EMOA gives the best results for all design cases.

Fig. 9 shows the best Pareto front of Case I obtained from using Hybrid_UPS. Some solution points are selected and the corresponding truss structures are displayed in Fig. 10. The obtained structures have various shapes and element sizes. Fig. 11 shows the best Pareto front of Case II obtained from using Hybrid_UPS. Some

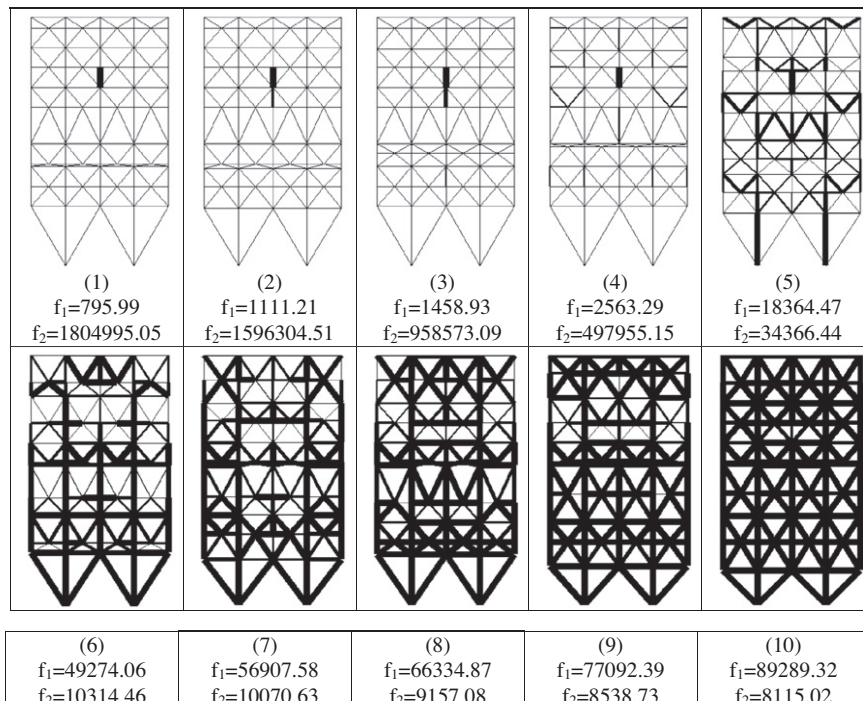


Fig. 12. Truss structures according to the front in Fig. 11.

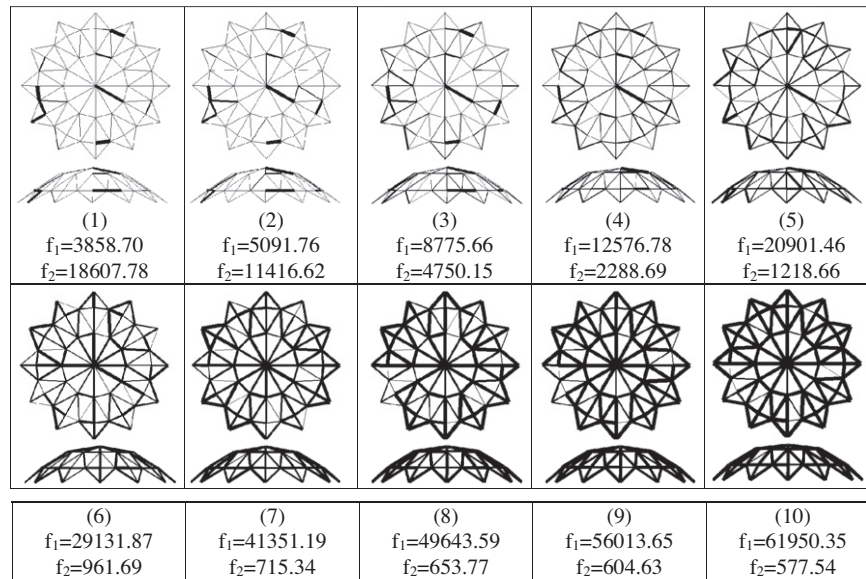


Fig. 14. Truss structures according to the front in Fig. 13.

solution points are selected and their corresponding truss structures are illustrated in Fig. 12. Similarly to the first case, the structures have various shapes and sizes. The best Pareto front of Case III obtained from using Hybrid_UPS is plotted in Fig. 13. Some solution points are selected and the corresponding truss structures are displayed in Fig. 14. The obtained structures are not symmetric with respect to the z axis as all bar element cross-sectional areas are assigned as design variables. However, in practice, structural symmetry has to be taken into consideration.

5. Conclusions

A study of enhancing search performance of MOEAs was carried out. During the search process of MOEAs, gradient vectors can be estimated from already explored individuals without requiring direct function derivative evaluations. The so-called approximate gradient operator which exploits the search direction concept of gradient-based optimisers is integrated into the main procedure of the three MOEAs. The comparative results show that hybrid MOEAs with the use of the approximate gradient operator outperform their original algorithms. UPS-EMOA is the best method among the original MOEAs whereas MPSO is the worst performer. However, when incorporating the approximate gradient into the search procedures, its search performance can be compared to the other methods. For the hybrid versions, the Hybrid of UPS-EMOA is overall the best optimiser. The concept of approximate gradients can not only be applied to multiobjective evolutionary optimisers but it can also be implemented with single objective evolutionary algorithms. The implementation of this concept on other real world optimisation problems will be studied as future work.

Acknowledgments

The authors are grateful for support from The Royal Golden Jubilee PhD Program (RGJ), and the Thailand Research Fund (TRF).

References

- [1] Su R, Wang X, Gui L, Fan Z. Multi-objective topology and sizing optimization of truss structures based on adaptive multi-island search strategy. *Struct Multidiscip Optimiz* 2011;43:275–86.
- [2] Jansen PW, Perez RE. Constrained structural design optimization via a parallel augmented Lagrangian particle swarm optimization approach. *Comput Struct* 2011;89:1352–66.
- [3] Gomes HM. Truss optimization with dynamic constraints using a particle swarm algorithm. *Expert Syst Appl* 2011;38:957–68.
- [4] Xu T, Zuo W, Xu T, Song G, Li R. An adaptive reanalysis method for genetic algorithm with application to fast truss optimization. *Acta Mech Sin* 2010;26:225–34.
- [5] Li LJ, Huang ZB, Liu F. A heuristic particle swarm optimization method for truss structures with discrete variables. *Comput Struct* 2009;87:435–43.
- [6] Kaveh A, Talatahari S. A particle swarm ant colony optimization for truss structures with discrete variables. *J Constr Steel Res* 2009;65:1558–68.
- [7] Kaveh A, Talatahari S. Size optimization of space trusses using Big Bang–Big Crunch algorithm. *Comput Struct* 2009;87:1129–40.
- [8] Luh GC, Lin CY. Optimal design of truss structures using ant algorithm. *Struct Multidiscip Optimiz* 2008;36:365–79.
- [9] Kaveh A, Talatahari S. A hybrid particle swarm and ant colony optimization for design of truss structures. *Asian J Civil Eng (Build Housing)* 2008;9:329–48.
- [10] Plevris V, Papadrakakis M. A hybrid particle swarm–gradient algorithm for global structural optimization. *Comput-Aided Civil Infrastruct Eng* 2011;26:48–68.
- [11] Liu GP, Han X, Jiang C. A novel multi-objective optimization method based on an approximation model management technique. *Comput Methods Appl Mech Eng* 2008;197:2719–31.
- [12] Sanchis J, Martínez M, Blasco X, Salcedo J. A new perspective on multiobjective optimization by enhanced normalized normal constraint method. *Struct Multidiscip Optimiz* 2008;36:537–46.
- [13] Kim I, de Weck O. Adaptive weighted sum method for multiobjective optimization: a new method for Pareto front generation. *Struct Multidiscip Optimiz* 2006;31:105–16.
- [14] Kaveh A, Talatahari S. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. *Comput Struct* 2009;87:267–83.
- [15] Bureerat S, Sriwornamas K. Population-based incremental learning for multiobjective optimisation. *Adv Soft Comput* 2007;9:223–31.
- [16] Pholdee N, Bureerat S. Passive vibration control of an automotive component using evolutionary optimisation. In: *The first TSME international conference on mechanical engineering*, Ubon Ratchathani, Thailand; 2010.
- [17] Srisomporn S, Bureerat S. Geometrical design of plate-fin heat sinks using hybridization of MOEA and RSM. *IEEE Trans Compon Packag Technol* 2008;31:351–60.
- [18] Kunakote T, Bureerat S. Multiobjective topology optimisation using evolutionary algorithms. *Eng Optimiz* 2011;43:541–57.
- [19] Song XG, Hoon J, Son HJ, Park JH, Lee KH, Park YC. Metamodel-based optimization of a control arm considering strength and durability performance. *Comput Math Appl* 2010;60:976–80.
- [20] Park HS, Dang XP. Structural optimisation based on CAD-CAM integration and metamodeling techniques. *Comput Aided Des* 2010;2010.
- [21] Kaya N, Karen I, Öztürk F. Re-design of a failed clutch fork using topology and shape optimisation by the response surface method. *Mater Des* 2010;31:3008–14.
- [22] Oudjene M, Ben-Ayed L, Delameziere A, Batoz J-L. Shape optimization of clineching tools using the response surface methodology with moving least-square. *J Mater Process Technol* 2009;209:289–96.

- [23] Goel T, Vaidyanathan R, Haftka RT, Shyy W, Queipo NV, Tucker K. Response surface approximation of Pareto optimal front in multi-objective optimization. *Comput Methods Appl Mech Eng* 2007;196:879–93.
- [24] Aittokoski T, Miettinen K. Efficient evolutionary approach to approximate the Pareto-optimal set in multiobjective optimization, UPS-EMOA. *Optimiz Methods Software* 2010;25:841–58.
- [25] Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: *Evolutionary methods for design, optimization and control*, Barcelona, Spain; 2002.
- [26] Sierra MR, Coello CAC. Multi-objective particle swarm optimizers: a survey of the state-of-the-art. *Int J Comput Intell Res* 2006;2:287–308.
- [27] Bureerat S, Cooper JE. Evolutionary optimization using evolutionary direction and pseudo-gradient. In: *1st ASMO UK/ISSMO Ilkley, UK*; 1999. p. 81–7.
- [28] Bureerat S. Hybrid population-based incremental learning using real codes. *Lect Notes Comput Sci* 2011;6683:379–91.
- [29] Kreyszig E. *Advanced engineering mathematics*. 9th ed. Hoboken, New Jersey: John Wiley & Sons; 2006.
- [30] Bandyopadhyay S, Saha S, Maulik U, Deb K. A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Trans Evol Comput* 2008;12:269–83.



Technical Note

Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms

Norapat Noilublao, Sujin Bureerat *

Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand

ARTICLE INFO

Article history:

Received 21 June 2011

Accepted 17 August 2011

Available online 13 September 2011

Keywords:

Truss optimisation

Multiobjective evolutionary algorithms

Topology optimisation

Structural vibration

Frequency response function

ABSTRACT

This paper presents an integrated design technique to carry out simultaneous topology, shape and sizing optimisation of a three-dimensional truss structure. Design objectives include mass, compliance, natural frequencies, frequency response function (FRF), and force transmissibility (FT). The Pareto fronts are explored by using: strength Pareto evolutionary algorithm (SPEA2), population-based incremental learning (PBIL), and archived multiobjective simulated annealing (AMOS). The results obtained from using the optimisers are compared based upon the hypervolume (HV) and generational distance (GD). It is shown that PBIL is the best for optimising compliance and natural frequency, while SPEA2 is superior when dealing with FRF and FT.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Trusses and frames are some of the most commonly used structures in daily life. Using such structures is said to be advantageous since they are simple and inexpensive to construct. It can be employed in many engineering purposes e.g. transmission towers, wind turbine towers, communication towers, civil engineering structures, and mechanical parts. Moreover, this type of structures can be built in difficult-to-access areas. Design/optimisation of trusses has been studied and researched worldwide. Throughout its history, over thousand research and technical papers have been published. For recent development, there have been efforts to employ evolutionary algorithms (EAs) and meta-heuristics (MHs) for truss optimisation problems. The most popular method is genetic algorithm and its variants [1–4]. Other methods include simulated annealing [5,6], particle swarm optimisation [7], charged system search [8], ant colony algorithm [9], and bee colony algorithm [10], etc. Mostly, the design problem is set for mass minimisation with safety constraints under static loads. Nevertheless, it has been recognised that practical design of trusses requires multiple design objectives with static and dynamic design criteria while only a handful of researchers focus on implementing multiobjective evolutionary algorithms (MOEAs) for truss and frame optimisation, e.g. [11,12]. If a multiobjective design problem of trusses can be posed and tackled, it would be more beneficial to this engineering research field.

In structural optimisation, design variables can be categorised into three groups as topology, shape and sizing variables. Topological design variables determine an initial structural layout whereas shape and sizing parameters give the shape and dimensions of structures respectively. Traditionally, a designer performs topological optimisation at the first design phase to obtain an initial structural configuration. The optimum shape and sizes of the structure are then found in the later design stage. This is often called multi-stage optimisation. Nevertheless, it has been found that the better design process is to perform topology, shape, and sizing optimisation simultaneously. This is easier said than done. The combination of shape and sizing design may be easily carried out; however, when combining topological variables to the design problem it is more difficult and complicated. However, such an integrated design approach can be attained and applied to both continuum [13] and skeletal structures [12].

The applications of such integrated design concept for skeleton structures (trusses and frames) have been studied for many years with the use of both gradient-based and population-based optimisers [4,14–17]. In the literature, design problems were mostly limited to single objective. Objective functions usually include structural weight, static and dynamic stiffness. Recent investigation [12] shows that the use of multiobjective evolutionary algorithms for this type of design problem is possible and effective. The work in [12] shows that multiobjective design of simple frames with combined topology, shape, and sizing design variables is achievable. The two best optimisers are real-code strength Pareto evolutionary algorithm and binary-code population-based incremental learning. However, the performance of these two methods

* Corresponding author.

E-mail address: sujbur@kku.ac.th (S. Bureerat).

in dealing with large-scale practical design has not been studied yet.

This work presents an approach to achieve simultaneous topology, shape and sizing optimisation of a three-dimensional truss structure. An optimisation problem is set to find structural layout, shape, and bar diameters simultaneously such that optimising multiple objective functions including mass, compliance, natural frequencies, frequency response function, and force transmissibility. The Pareto optimal solutions are explored by using the two best performer MOEAs from [12] i.e. SPEA2 and PBIL. Also, an efficient archived multiobjective simulated annealing is employed to solve the problem as the method has not been tested in the previous work. The results obtained from using the three optimisers are illustrated and compared.

2. Encoding/decoding design variables

One of the main contributions of this work is the strategy for encoding/decoding design variables. The design variables can be encoded as binary or real parameters as we can eventually transform them to be real numbers. Design variables in this paper are said to be mixed integer/continuous. The tower height is set to be $H = 50$ -m with a square cross-sectional area. The tower is allowed to have 10–20 tower levels. Fig. 1 displays 7 parameters $\{s_1, \dots, s_7\}$ (with the value of s_7 being fixed) that control the shape of the tower. Having 7 input parameters, the tower can be shaped by using an interpolation technique. The height of each tower level can also be varied in which four parameters t_1, t_2, t_3 , and t_4 are used to compute the heights of the tower levels.

A tower topology can be obtained by using an adaptive ground element approach. Fig. 2 shows the ground finite elements of a particular tower level. On each level, there are 8 nodal points, and 16 ground elements. Six design parameters are assigned as the cross-sectional areas of those elements to maintain structural symmetry as given in Table 1. The design variables for element cross-sections will be rounded off to their nearest integers $\{1, 2, 3, 4, 5\}$ and decoded to be corresponding finite element properties as given in Table 2 [15]. With the ground finite element approach, elements having too small cross-sectional areas (a design variable rounded off to be 1) are deleted from the structure. Although those elements are removed from the structure, they are included in the finite element model with small cross sectional area and Young modulus value in order to prevent matrix singularity in structural

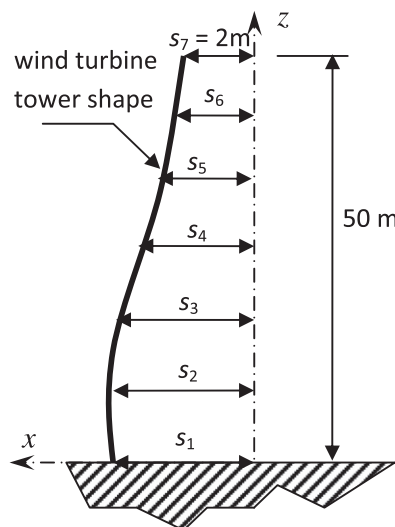


Fig. 1. Parameters for defining a tower shape.

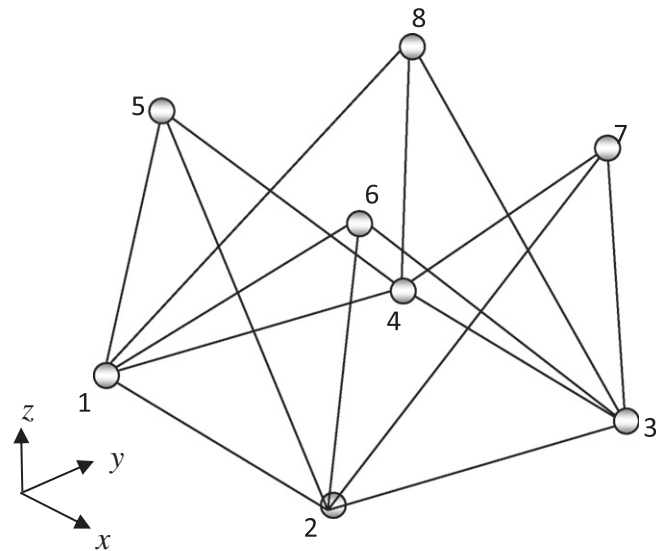


Fig. 2. Ground elements for each tower level.

Table 1
Element connectivity and cross-sectional variables.

Elements as node number connections	Cross-sectional areas
1–2 3–4 2–3 4–1	y_1
1–5 2–6 3–7 4–8	y_2
1–6 3–8	y_3
2–5 4–7	y_4
1–8 3–6	y_5
4–5 2–7	y_6

finite element analysis. This is a means to define truss topology and, at the same time, to define the elements' cross-sectional areas.

In the decoding process, there are four groups of design variable i.e. the first group determines the number of tower levels, the second group gives the variables to control tower shape as displayed in Fig. 1, the third group determines height distribution of tower levels, and the last group represents ground elements' cross-sectional areas. In the last design variable group, on each tower level, 6 design variables are assigned as the cross-sectional areas of 16 ground elements.

The pseudo-code for design variables decoding is displayed in Algorithm 1. The functions and parameters are defined as:

$\text{round}(X)$ rounds a real number X to the nearest integer.

$\mathbf{P} = \text{SplineInterp}(\mathbf{X}, \mathbf{Y})$ gives a piecewise polynomial form $\mathbf{P}$ by means of cubic spline interpolation where $\mathbf{X}$ and $\mathbf{Y}$ are sets of independent and dependent variables respectively.

$\mathbf{Y} = \text{SplineEval}(\mathbf{P}, \mathbf{X})$ predicts the values of dependent variables $\mathbf{Y}$ based on the input piecewise polynomial form $\mathbf{P}$, and independent variables $\mathbf{X}$.

n_{level} = the number of tower levels.

h_i = the height of level i .

Table 2

Parameters for cross-sections of truss elements.

Integer	Gross area (m ²)	Net area (m ²)	Radius of gyration (m)	Young modulus (N/m ²)
1	1×10^{-8}	1×10^{-8}	1×10^{-8}	214×10^5
2	0.000581	0.000426	0.0118	214×10^9
3	0.00108	0.000837	0.0157	214×10^9
4	0.00232	0.001894	0.0236	214×10^9
5	0.01108	0.009448	0.0695	214×10^9

W_i = the half width of level i .

Step 1 is used to assign the values of the parameters determining the number of tower levels, tower shape, and level height. Steps 2–5 are used to compute the heights of the tower levels. Step 5 is activated to ensure that $h_{i+1} > h_i$. Steps 6–8 are used for computing the widths of the tower levels. Step 9 creates $4(n_{level} + 1)$ nodal points from h_i and w_i . The n_{level} loops of Steps 10–11 define element connectivity of 16 ground elements for each level, and assign the element properties based on Tables 1 and 2. Steps 12–14 determine element connectivity and properties of the 4 remaining elements on the top of the tower.

As the maximum number of tower levels is set to be 20, the maximum number of design variables $\mathbf{x}$ is $6(20) + 12 = 132$. In the optimisation process, if the decoded number of tower levels is less than 20, only the first $6n_{level} + 12$ values of $\mathbf{x}$ are used for creating a finite element model of the tower. The output of Algorithm 1 is the required data for a finite element model.

Algorithm 1. Design variables decoding procedure

Input: $\mathbf{x}_{132 \times 1}$

Output: Nodal points, Truss elements, elements properties

1. Set $n_{level} = \text{round}(x_1)$, $\{s_1, \dots, s_6\} = \{x_2, \dots, x_7\}$, $s_7 = 2$, $\{t_1, \dots, t_4\} = \{x_8, \dots, x_{11}\}$
 2. Assign values: $\{Z_1, Z_2, Z_3, Z_4\} = \{0, H/3, 2H/3, H\}$
 3. Compute spline interpolation coefficients
 $\mathbf{P}_1 = \text{SplineInterp}(\{Z_1, \dots, Z_4\}, \{t_1, \dots, t_4\})$
 4. Determine $\{a_1, \dots, a_{n_{level}}\} = \text{SplineEval}(\mathbf{P}_1, \{H/n_{level}, 2H/n_{level}, \dots, H\})$
 5. Compute $h_i = H \sum_{j=1}^i a_j / \sum_{j=1}^{n_{level}} a_j$ the heights of the tower levels
 6. Assign values: $\{Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7\} = \{0, H/6, 2H/6, 3H/6, 4H/6, 5H/6, H\}$
 7. Compute spline interpolation coefficients
 $\mathbf{P}_2 = \text{SplineInterp}(\{Z_1, \dots, Z_7\}, \{s_1, \dots, s_7\})$
 8. Determine $\{w_0, \dots, w_{n_{level}}\} = \text{SplineEval}(\mathbf{P}_2, \{0, H/n_{level}, 2H/n_{level}, \dots, H\})$ the half size of the tower levels
 9. Determine $4(n_{level} + 1)$ nodal points from $\{0, h_1, \dots, h_{n_{level}}\}$ and $\{w_0, \dots, w_{n_{level}}\}$
 10. For $i = 1$ to n_{level}
 - 10.1. Assign element connectivity of 16 ground elements of level i
 - 10.2. Set $\{y_{6i-5}, \dots, y_{6i}\} = \text{round}(\{x_{11+6i-5}, \dots, x_{11+6i}\})$
 - 10.3. Assign elements properties based on Table 2
 11. End For
 12. Assign elements' connectivity for the four element cross-sections on the top of the tower
 13. Set the integer for four element cross-sections on the top
 $y_{6n_{level}+1} = \text{round}(x_{11+6n_{level}+1})$ where $y_{6n_{level}+1} \in \{2, 3, 4, 5\}$
 14. Assign the elements properties according to Table 2
-

3. Design testing problems

A 3D truss tower is used for this study. Four testing problems are posed to find the optimum tower shape, bar sizes, and topology such that minimising two objective functions while satisfying design constraints. The integrated topology, shape, and sizing optimisation problem can be expressed as:

$$\begin{aligned}
 &\min \quad [f_1, f_2], \\
 &\text{subject to} \quad \sigma_{\max} \leq \sigma_{\text{all}}, \\
 &\quad \lambda_i \leq 1, \\
 &\quad 10 \leq \text{round}(x_1) \leq 20, \\
 &\quad 4 \text{ m} \leq x_i \leq 6 \text{ m}; \quad \text{for } i = 2, \dots, 7, \\
 &\quad 0 \leq x_i \leq 1; \quad \text{for } i = 8, \dots, 11, \\
 &\quad \text{round}(x_i) \in 1, 2, 3, 4, 5; \quad \text{for } i = 12, \dots, (11 + 6\text{round}(x_1)), \\
 &\quad \text{round}(x_i) \in 2, 3, 4, 5; \quad \text{for } i = (12 + 6\text{round}(x_1))
 \end{aligned} \tag{11}$$

where f_1, f_2 are objective functions:

$\sigma_{\max}$ is the maximum stress on truss elements.

σ_{all} is an allowable stress to be defined.

λ_i is a buckling factor of each element (defined as the ratio of applied load to critical load) and $\mathbf{x}$ is a vector of simultaneous topology, shape, and sizing design variables.

The tower is subject to three static load cases where, for each case, four point loads are applied at the top four nodes. For the first load case, each point load has the magnitude of 1×10^4 N acting in the x direction. The point loads of the second and third cases have the magnitudes of 1×10^4 N, and 2×10^4 N acting in the y and z directions, respectively. Elements' weights are also included for each load case. Tower elements are made up of material with Young modulus $E = 214 \times 10^9$ N/m², density $\rho = 7850$ kg/m³, and yield strength $\sigma_{yt} = 360 \times 10^6$ N/m². The factor of safety for calculating the allowable stress is set to be 1.5. In the assembly process of a structural global stiffness matrix, Young modulus of elements with the integer design variable '1' is set to be $E = 214 \times 10^5$ N/m² as given in Table 2. This implies that elements that will be removed from the ground elements have an insignificant effect on a structural stiffness matrix.

The four bi-objective design problems have the same constraints as shown in Eq. (11). The objective functions of each design problems are given as follows:

- OPT1: f_1 = structural mass, $f_2 = \mathbf{F}_1^T \delta_1 + \mathbf{F}_2^T \delta_2 + \mathbf{F}_3^T \delta_3$.
- OPT2: f_1 = structural mass, $f_2 = \omega_1 + \omega_2 + \omega_3$.
- OPT3: f_1 = structural mass, $f_2 = \text{FRFCrest}(\omega_1) + \text{FRFCrest}(\omega_2) + \text{FRFCrest}(\omega_3)$.
- OPT4: f_1 = structural mass, $f_2 = \text{FTCrest}(\omega_1) + \text{FTCrest}(\omega_2) + \text{FTCrest}(\omega_3)$.

All of the design problems have the structural weight as the first design objective. The second objective of OPT1 is the total structural compliance due to the three load cases ($\mathbf{F}_1$, $\mathbf{F}_2$, and $\mathbf{F}_3$) where δ_i are the vector of structural displacements due to load case i . The second objective of OPT2 is the summation of the first three natural frequencies of the structure.

The OPT3 and OPT4 problems deal with harmonic response analysis. Fig. 3 displays the measure of harmonic response at the r th degree of freedom due to the input harmonic excitation at the s th degree of freedom of a particular structure. Harmonic response analyses in this paper include frequency response function and force transmissibility. All details of static and dynamic finite element analyses as well as FRF and FT computation are given in [12]. The function $FRFCrest(\omega_i)$ in OPT3 determines a FRF crest parameter with respect to the natural frequency ω_i . The so-called FRF crest parameter is shown in Fig. 4. In the figure, the frequency interval is $[\omega_1 - \Delta\omega, \omega_1 + \Delta\omega]$ where ω_1 is the first structural natural frequency. Such a parameter determines the shaded area of this small frequency interval, which implies that a design process attempts to reduce the peak amplitude of the first mode natural frequency on the FRF graph. The function $FTCrest(\omega_i)$ for a FT crest parameter used in OPT4 is defined in a similar manner.

The frequency response function herein is the point receptance measured at one of the top nodes in the x direction. For FT, the harmonic reaction at the bottom node in the z direction due to the harmonic excitation at the top node in the x direction is measured. This FT crest parameter is proposed to measure the force transmissibility from the tower top in the horizontal direction to the tower base in the vertical direction. Note that this is set for design demonstration. In practice, the positions and directions of the excitation force and response can be defined by a designer.

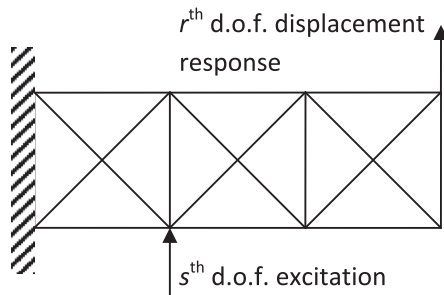


Fig. 3. Measurement of FRF.

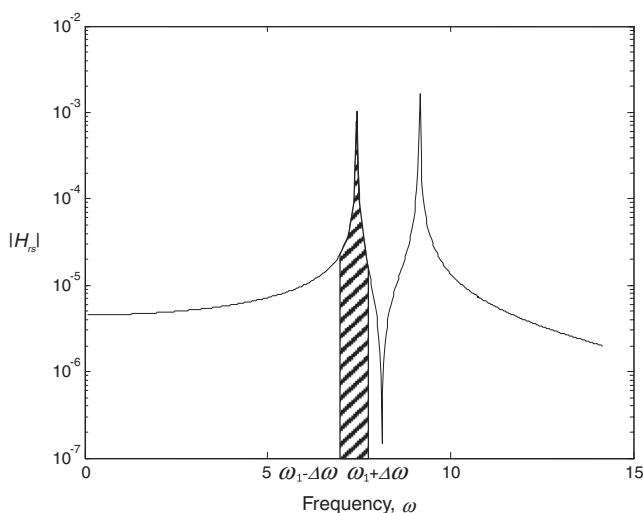


Fig. 4. FRF crest parameter.

For the multiobjective optimisers employed to tackle the problems, as SPEA [18] using real code, and PBIL [19,20] are the best performers in [12], the two optimisers are chosen for this comparative performance study. Also, AMOSA [21] one of the recently developed efficient MOEAs is implemented in this test. AMOSA search mechanism is based on a mutation operator. This means that we use three kinds of MOEAs for solving the problems: the crossover-based method (SPEA), mutation-based method (AMOSA) and the estimation of distribution algorithm (PBIL). They are implemented to solve the optimisation problems for ten runs for each design problem with the population size of 200, and number of generation as 300. The external Pareto archive size is set to be 200. The performance assessment is carried out by using the hypervolume and generational distance indicators [22]. The non-dominated sorting scheme proposed in [23] is used to deal with constraints. The optimisation settings are as follows:

- SPEA: The real-code crossover and mutation operators are detailed in [24]. The crossover and mutation probabilities are set as 1.0 and 0.5 respectively.
- PBIL uses the mutation shift of 0.2 where the mutation probability is set to be 0.1. Each design variable is represented by 4 binary bits. The probability matrix contains 40 probability vectors, which means one vector is used to produce 5 binary design solutions. The archive technique is the normal line method.
- AMOSA performs a simple downhill operator for the first 3 generations before entering the main procedure. In this paper, the method uses real codes where a candidate is obtained by means of mutation. The algorithm is terminated when the total number of function evaluations is equal to 200×300 . The clustering archive technique is activated to reduce the members in the external archive size to be 200 if the number of members in the archive exceeds 400.

4. Results

In comparing non-dominated fronts obtained from MOEAs, based on the indicator definitions, the higher value of a hypervolume indicates the better Pareto front whereas the lower value of a generational distance means the more advanced front. In order to have a reference front for computing the generational distance for each design problem, all of design solutions from 3×10 runs of the optimisers are put together. The non-dominated solutions of them are set as the reference front of the design problem. The hypervolumes and generational distance comparison is given in Table 3. Note that those values are normalised to ease in comparing. For each design problem, the average and standard deviation of 10 hypervolumes and generational distances of each optimiser

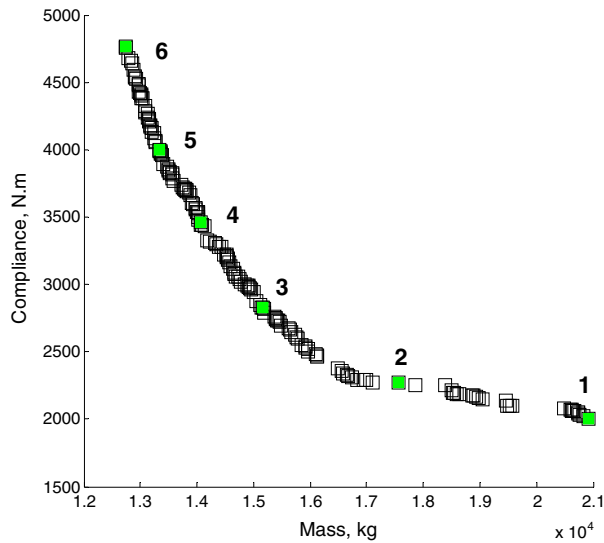
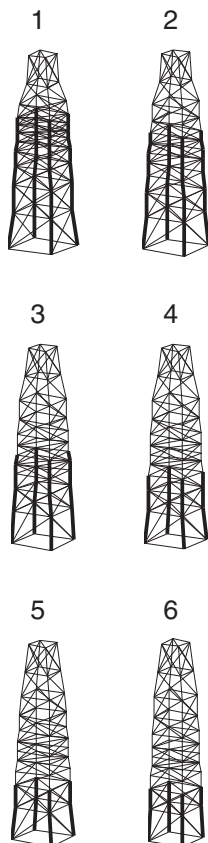
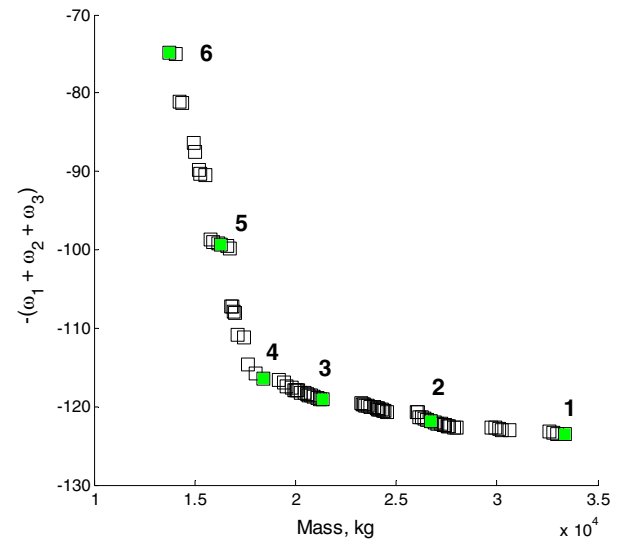
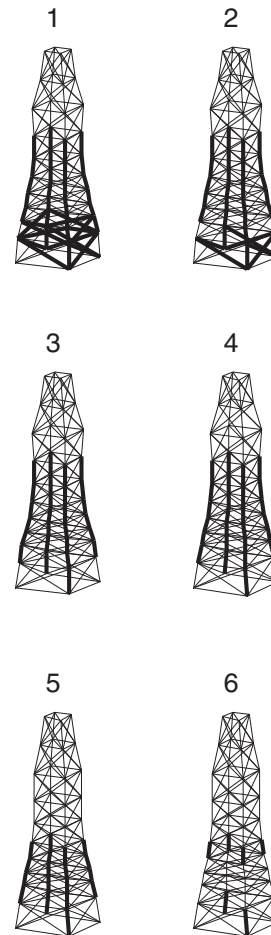
Table 3
Hypervolume and generational distance comparison.

		Hypervolume		Generational distance	
		AVG	STD	AVG	STD
OPT1	SPEA	0.8176	0.2008	0.0426	0.1348
	PBIL	0.9393	0.0300	0.0014	0.0045
	AMOSA	0.6020	0.2013	0.2944	0.4481
OPT2	SPEA	0.6453	0.1817	0.0083	0.0147
	PBIL	0.8055	0.0696	0.0245	0.0260
	AMOSA	0.4548	0.1985	0.3157	0.3007
OPT3	SPEA	0.9486	0.0235	0.0480	0.0378
	PBIL	0.8840	0.0524	0.3425	0.3676
	AMOSA	0.6632	0.1537	0.3012	0.2041
OPT4	SPEA	0.9767	0.0176	0.0295	0.0302
	PBIL	0.7672	0.1189	0.3203	0.2628
	AMOSA	0.8501	0.0743	0.3384	0.1609

Table 4

Average running time (min).

	OPT1	OPT2	OPT3	OPT4
SPEA	60.89	141.67	178.00	358.88
PBIL	66.10	117.00	168.50	188.78
AMOS	34.89	101.75	103.86	154.50

**Fig. 5.** The best front of OPT1 (SPEA).**Fig. 6.** Selected towers from the front in Fig. 5.**Fig. 7.** The best front of OPT2 (PBIL).**Fig. 8.** Selected towers of the front in Fig. 7.

are computed. The mean value is used to somewhat measure a convergence rate whereas the standard deviation suggests search consistency.

From the table, it can be seen that the best method for the first design problem OPT1 is PBIL since the average value of front hypervolumes produced by PBIL is the highest. It is also the most consistent optimiser based on the HV standard deviation. Based

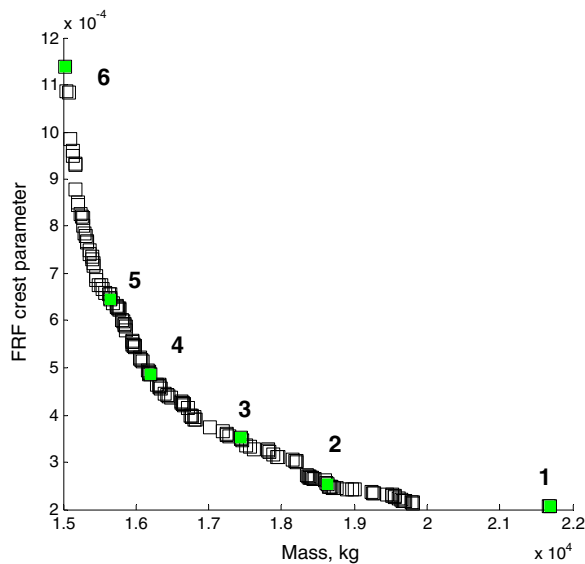


Fig. 9. The best front of OPT3 (SPEA).

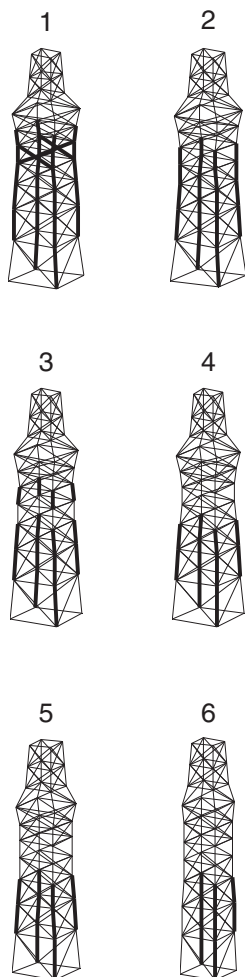


Fig. 10. Selected towers of the front in Fig. 9.

on the GD, the best method is still PBIL. For the OPT2 design problem, the best performer considering both HV average and standard deviation is PBIL while the best optimiser when considering GD is SPEA. This implies that, on average, PBIL produces better front

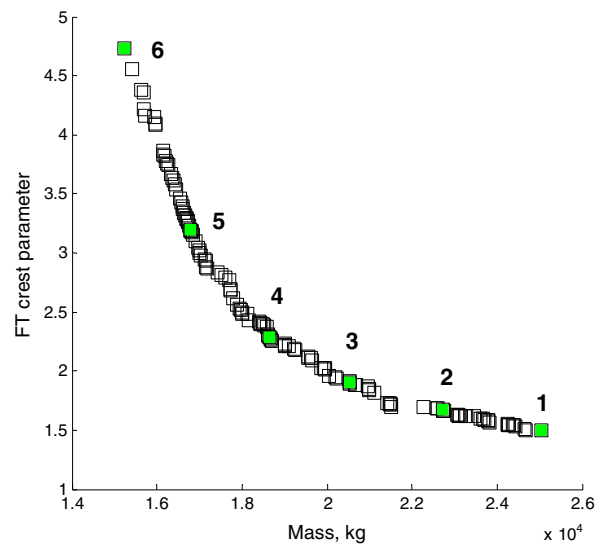


Fig. 11. The best front of OPT4 (SPEA).

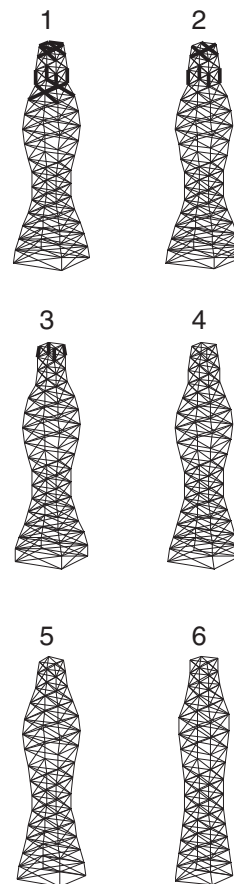


Fig. 12. Selected towers of the front in Fig. 11.

extension whereas SPEA produces more advanced fronts [25]. For the OPT3 and OPT4 design problems, SPEA is the best performer.

Table 4 shows the average computing time of each MOEA for solving the design problems using a computer with 2 Quad CPU at 2.40 GHz and 2.99 GB of RAM. It can be seen that AMOSA spent less computing time than the others. However, the computational time will not be used for performance assessment since this is an

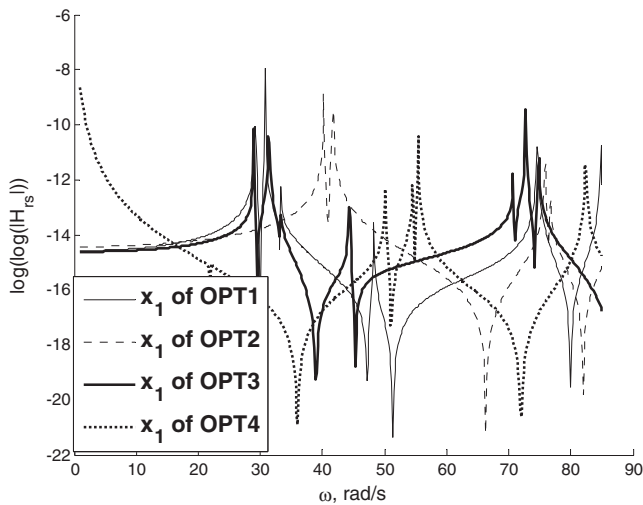


Fig. 13. FRF of the first towers from Figs. 6, 8, 10 and 12.

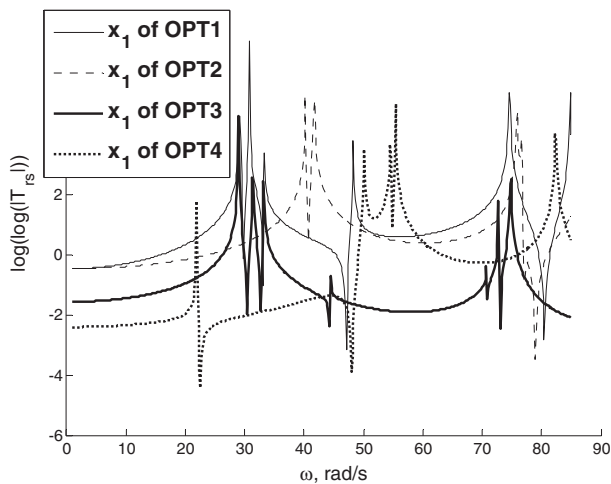


Fig. 14. FT of the first towers from Figs. 6, 8, 10 and 12.

offline optimisation process, which means the method that gives the best results will be more important.

Overall, the best optimiser is SPEA using real codes while the second best is PBIL. The optimiser that gives the worst performance is AMOSA. Nevertheless, it should be noted that this comparative results are based on optimisation parameter settings for each MOEA. It can be said that PBIL is better for an optimisation problem with a static design criterion but when dealing with dynamic design criteria, SPEA is superior.

The best front (from the best run) of OPT1 obtained from using SPEA is displayed in Fig. 5 where some selected design solutions are highlighted. The truss towers corresponding to the selected design solutions in Fig. 5 are shown in Fig. 6. For OPT2, the best front obtained from running PBIL is shown in Fig. 7 while the corresponding selected structures are illustrated in Fig. 8. The best front of OPT3 from SPEA is shown in Fig. 9 with 6 selected solutions while the corresponding structures are illustrated in Fig. 10. For the fourth testing problem OPT4, the best front obtained from using SPEA is depicted in Fig. 11 whereas the associated structures are illustrated in Fig. 12.

Figs. 13 and 14, respectively show the FRF and FT of solutions number 1 in Figs. 6, 8, 10 and 12. The double logarithmic scale is use on the y-axis so that the FRF and FT magnitudes from the

various towers can be visibly compared. From the figures, it can be seen that the FRF magnitudes around the first three natural frequencies of the solution from OPT3 is lower than those of the other towers. The solution obtained from optimising OPT2 gives the highest summation of the first three natural frequencies as expected. In Fig. 14, the design solution obtained from OPT4 is superior to the others when considering the effect of FT from the tower top to its base. The comparison in these figures reasonably shows that the use of MOEAs can achieve optimum solutions corresponding to the assigned design problems.

5. Conclusions and discussion

With the proposed numerical scheme for encoding/decoding design variables and the use of MOEAs, complicated truss design problems can be posed and tackled. The design criteria set in this study may not be sufficient for some real world applications. However, design problems can be altered to have more design objectives or different objectives. This is dependent on designer decisions, while MOEAs can still be applied to solve the problems.

From the comparative results, it can be concluded that PBIL is superior when optimising structural compliance (static stiffness) while the SPEA method is better when optimising FRF and FT. In cases of natural frequency maximisation or structural dynamic stiffness maximisation, both PBIL and SPEA are said to be equally good as one has a better HV and another has a better GD. The obtained truss structure are said to be practical if further detailed design phase is performed. The proposed design approach can be a powerful tool for truss and frame design. By using this design strategy, various optimum structures for decision making can be obtained within one optimisation run. Our future work will be the development of a new efficient MOEA that suits for solving this type of design problem. Extension of this design concept to other types of truss and frame structures will also be studied.

Acknowledgements

The authors are grateful for the support from the Thailand Research Fund (TRF) and the Sustainable Infrastructure Research and Development Centre (SIRDC), Khon Kaen University, Thailand.

References

- [1] Kahraman M, Erbatır F. A GA approach for simultaneous structural optimization. *Struct Eng Mech Comput* 2001;2:1147–54.
- [2] Balling RJ, Briggs RR, Gillman K. Multiple optimum size/shape/topology designs for skeletal structures using a genetic algorithm. *J Struct Eng* 2006;132:1158–65.
- [3] Xu T, Zuo W, Xu T, Song G, Li R. An adaptive reanalysis method for genetic algorithm with application to fast truss optimization. *Acta Mech Sinica* 2010;26:225–34.
- [4] Chan CM, Wong KM. Structural topology and element sizing design optimization of tall steel frameworks using a hybrid OC-GA method. *Struct Multidisc Optimiz* 2008;35:473–88.
- [5] Hasancebi O, Erbatır F. Layout optimisation of trusses using simulated annealing. *Adv Eng Softw* 2002;33:681–96.
- [6] Lamberti L. An efficient simulated annealing algorithm for design optimization of truss structures. *Comput Struct* 2008;86:1936–53.
- [7] Perez RE, Behdinan K. Particle swarm approach for structural design optimization. *Comput Struct* 2007;85:1579–88.
- [8] Kaveh A, Talatahari S. Optimal design of skeletal structures via the charged system search algorithm. *Struct Multidisc Optimiz* 2010;41:893–911.
- [9] Luh GC, Lin CY. Optimal design of truss structures using ant algorithm. *Struct Multidisc Optimiz* 2008;36:365–79.
- [10] Sonmez M. Artificial Bee Colony algorithm for optimization of truss structures. *Appl Soft Comput* 2011;11:2406–18.
- [11] Kicinger R, Obayashi S, Arciszewski T. Evolutionary multiobjective optimization of steel structural systems in tall buildings. *LNCS* 2007;4403:604–18.
- [12] Noilublao C, Bureerat S. Simultaneous topology shape and sizing optimisation of skeletal structures using multiobjective evolutionary algorithms. *Evolutionary computation*. In-Tech; 2009. ISBN 978-953-307-008-7 [Chapter 24, p. 487–580].

- [13] Zhou M, Pagaldipti N, Thomas HL, Shyy YK. An integrated approach to topology, sizing, and shape optimization. *Struct Multidisc Optimiz* 2004;26:308–17.
- [14] Wang X, Wang MY, Guo D. Structural shape and topology optimization in a level-set-based framework of region representation. *Struct Multidisc Optimiz* 2004;27:1–19.
- [15] Shea K, Smith IFC. Improving full-scale transmission tower design through topology and shape optimization. *J Struct Eng* 2006;132(5):781–90.
- [16] Tang W, Tong L, Gu Y. Improved genetic algorithm for design optimization of truss structures with sizing, shape and topology variables. *Int J Numer Methods Eng* 2005;62(13):1737–62.
- [17] Zhu J, Zhang W, Beckers P, Chen Y, Guo Z. Simultaneous design of components layout and supporting structures using coupled shape and topology optimization technique. *Struct Multidisc Optimiz* 2008;36:29–41.
- [18] Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Paper presented in evolutionary methods for design, optimization and control, Barcelona, Spain, 2002.
- [19] Bureerat S, Sriworamas K. Population-based incremental learning for multi-objective optimisation. *Adv Soft Comput* 2007;39:223–32.
- [20] Bureerat S, Srisomporn S. Optimum plate-fin heat sinks by using a multi-objective evolutionary algorithm. *Eng Optimiz* 2010;42:305–43.
- [21] Bandyopadhyay S, Saha S, Maulik U, Deb K. A Simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Trans Evolut Comput* 2008;12:269–83.
- [22] Tan KC, Yang YJ, Goh CK. A distributed cooperative coevolutionary algorithm for multiobjective optimization. *IEEE Trans Evolut Comput* 2006;10:527–49.
- [23] Deb K, Pratap A, Meyarivan T. Constrained test problems for multi-objective evolutionary optimization. *LNCS* 2001;1993:284–98.
- [24] Srisomporn S, Bureerat S. Geometrical design of plate-fin heat sinks using hybridization of MOEA and RSM. *IEEE Trans Compon Pack Technol* 2008;31:351–60.
- [25] Kunakote T, Bureerat S. Multiobjective topology optimization using evolutionary algorithms. *Eng Optimiz* 2011;43:541–57.