วิทยานิพนธ์

การใช้วิธีเชิงพันธุกรรมเพื่อแก้ไขปัญหาการจัดตารางการผลิตของเครื่องจักรขนานที่ไม่เหมือนกันแบบที่มีค่าใช้จ่ายเกิดจากผลิตเสร็จก่อนและผลิตเสร็จหลังวันกำหนดส่งมอบ

A GENETIC ALGORITHM FOR NON-IDENTICAL PARALLEL MACHINE SCHEDULING PROBLEMS WITH EARLINESS AND TARDINESS PENALTIES

นายกานต์ ปล้องอ่อน

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

พ.ศ. 2551

ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมอุตสาหการ)

ปริญญา

วิศวกรรมอุตสาหการ	วิศวกรรมอุตสาหการ
สาขา	**ภาควิชา**

เรื่อง การใช้วิธีเชิงพันธุกรรมเพื่อแก้ไขปัญหาการจัดตารางการผลิตของเครื่องจักรขนานที่ไม่เหมือนกันแบบที่มีค่าใช้จ่ายเกิดจากผลิตเสร็จก่อนและผลิตเสร็จหลังวันกำหนดส่งมอบ

A Genetic Algorithm for Non-Identical Parallel Machine Scheduling Problems with Earliness and Tardiness Penalties

นามผู้วิจัย นายกานต์ ปล้องอ่อน

ได้พิจารณาเห็นชอบโดย

ประธานกรรมการ .....

(อาจารย์วิสุทธิ์ สุพิทักษ์, Ph.D.)

กรรมการ .....

(อาจารย์พรเทพ อนุสสรนิติสาร, Ph.D.)

กรรมการ .....

(ผู้ช่วยศาสตราจารย์ นาวาโทสำราญ ทองเล็ก, บธ.ม.)

หัวหน้าภาควิชา .....

(ผู้ช่วยศาสตราจารย์อนันต์ มุ่งวัฒนา, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

วันที่ ..... **เดือน** ..... **พ.ศ.** .....

วิทยานิพนธ์

เรื่อง

การใช้วิธีเชิงพันธุกรรมเพื่อแก้ไขปัญหาการจัดตารางการผลิตของเครื่องจักรขนานที่ไม่เหมือนกัน แบบที่มีค่าใช้จ่ายเกิดจากผลิตเสร็จก่อนและผลิตเสร็จหลังวันกำหนดส่งมอบ

A Genetic Algorithm for Non-Identical Parallel Machine Scheduling Problems with Earliness and Tardiness Penalties

โดย

นายกานต์ ปล้องอ่อน

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมอุตสาหการ)
พ.ศ. 2551

กานต์ ปล้องอ่อน 2551: การใช้วิธีเชิงพันธุกรรมเพื่อแก้ไขปัญหาการจัดตารางการผลิตของเครื่องจักรขนานที่ไม่เหมือนกันแบบที่มีค่าใช้จ่ายเกิดจากผลิตเสร็จก่อนและผลิตเสร็จหลังวันกำหนดส่งมอบ ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมอุตสาหการ) สาขาวิศวกรรมอุตสาหการ ภาควิชาวิศวกรรมอุตสาหการ ประธานกรรมการที่ปรึกษา: อาจารย์วิสุทธิ์ สุพิทักษ์, Ph.D. 127 หน้า

งานวิจัยนี้จัดทำขึ้นเพื่อทำการวิเคราะห์การแก้ไขปัญหาการจัดตารางการผลิตของเครื่องจักรขนานที่ไม่เหมือนกัน (Non-identical Parallel Machine Scheduling) และมีวันส่งมอบที่แตกต่างกัน (Distinct Due Date) โดยมีฟังก์ชั่นวัตถุประสงค์คือ ผลรวมของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและผลิตเสร็จหลังวันกำหนดส่งมอบ (Earliness and Tardiness Costs) มีค่าน้อยที่สุด โดยนำวิธีเชิงพันธุกรรม (Genetic Algorithm) มาประยุกต์ในการจัดแบ่งงานและลำดับงานในแต่ละเครื่องจักรและใช้วิธีการจัดเวลาเริ่มงาน (Optimal Timing Algorithm) ในการหาเวลาเริ่มงานในแต่ละเครื่องจักร โดยวิธีนี้เรียกว่า วิธี GAOPT

วิธี GAOPT ที่นำเสนอในงานวิจัยนี้ ได้นำมาเปรียบเทียบกับวิธีฮิวริสติกส์อย่างง่าย 2 วิธีคือ วิธี EDDOPT และวิธี RNDOPT จากผลการทดสอบพบว่าวิธี GAOPT ให้ผลลัพท์ที่ดีกว่าวิธีฮิวริสติกส์ที่นำมาเปรียบเทียบทั้ง 2 วิธี ทั้งแบบขนาดปัญหาที่แตกต่างกันและแบบที่อัตราส่วนของอัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จหลังกำหนดส่งมอบงานต่ออัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จก่อนกำหนดส่งมอบงานที่แตกต่างกัน

_____ _____ ___ / ___ / ___

ลายมือชื่อนิสิต ลายมือชื่อประธานกรรมการ

Kan Plongon 2008: A Genetic Algorithm for Non-Identical Parallel Machine Scheduling Problems with Earliness and Tardiness Penalties. Master of Engineering (Industrial Engineering), Major Field: Industrial Engineering, Department of Industrial Engineering. Thesis Advisor: Mr. Wisut Supithak, Ph.D. 127 pages.

The paper considers the scheduling problem of non-identical parallel machines with distinct due date jobs. The objective is to determine the schedules such that the sum of earliness and tardiness costs of all jobs is minimized. In order to reduce the solution space, the problem is transformed to the sequencing problem. The genetic algorithm is applied to find the good job sequence. The optimal timing algorithm is then applied to determine the best schedule for each sequence. This heuristic is called the GAOPT.

The GAOPT heuristic proposed in this study is compared with the simple heuristics, EDDOPT and RNDOPT. The results showed that the GAOPT can outperform the other two heuristics at different problem sizes and ratio of earliness to tardiness penalties.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

ขอกราบขอบพระคุณอาจารย์วิสุทธิ์ สุพิทักษ์ ประธานกรรมการที่ปรึกษาที่ได้ช่วยเหลือในการวางแผนงานในวิทยานิพนธ์ฉบับนี้ ตลอดจนให้คำปรึกษาแนะนำและตรวจแก้ข้อบกพร่องต่าง ๆ ขอกราบขอบพระคุณ อาจารย์พรเทพ อนุสสรนิติสาร กรรมการวิชาเอก ผู้ช่วยศาสตราจารย์นาวา โทสำราญ ทองเล็ก กรรมการวิชารอง และผู้ช่วยศาสตราจารย์กชกร สุรเนาวรัตน์ผู้แทนบัณฑิตวิทยาลัย ที่ได้ให้ความกรุณาแนะนำ และตรวจแก้ไขวิทยานิพนธ์จนสำเร็จลุล่วงได้ด้วยดี

ขอกราบขอพระคุณอาจารย์ภาควิชาวิศวกรรมอุตสาหการและภาควิชาบริหารธุรกิจทุกท่านที่ได้อบรมสั่งสอนและมอบความรู้อันเป็นประโยชน์อย่างยิ่งในการนำไปใช้ประโยชน์ต่อไป รวมทั้งขอกราบขอบพระคุณคุณพ่อ คุณแม่และครอบครัวที่ได้ให้การสนับสนุนช่วยเหลือและเป็นกำลังใจในการทำวิทยานิพรนธ์จนสำเร็จลุล่วงได้ด้วยดี

กานต์ ปล้องอ่อน

พฤษภาคม 2551

สารบัญ

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

ตารางผนวกที่

สารบัญภาพ

สารบัญภาพ (ต่อ)

ภาพผนวกที่ **หน้า**

คำอธิบายสัญลักษณ์และคำย่อ

n	=	จำนวนของงานทั้งหมดที่นำมาพิจารณาในการจัดตาราง
m	=	จำนวนของเครื่องจักรทั้งหมดที่นำมาพิจาณาในการจัดตารางงาน
J_i	=	งานที่ i
M_j	=	เครื่องจักรที่ j
p_{ij}	=	เวลาทำงานหรือเวลาปฏิบัติงาน (Processing Time) หมายถึงเวลางานที่ i จะต้องใช้บนเครื่องจักร j
d_i	=	กำหนดส่งมอบ (Due Date) ของงาน i หมายถึง เวลาจัดส่งหรือเวลาเสร็จงานที่สัญญาไว้กับลูกค้า การทำงานเสร็จหลังจากกำหนดส่งมอบงานอาจเกิดขึ้นได้ แต่อาจจะมีค่าปรับเกิดขึ้น
C_i	=	เวลาเสร็จงาน (Completion Time) หมายถึง เวลาที่งาน i เสร็จสิ้นการทำงาน
t_i	=	เวลาเริ่มงาน (Starting Time) หมายถึง เวลาที่เริ่มปฏิบัติงาน i
L_i	=	เวลาสาย (Lateness) หมายถึง เวลาสายของงาน i ถ้างานใดมีค่า L_i เป็นบวก หมายความว่างานนั้นสาย แต่ถ้า L_i เป็นลบ แสดงว่างานนั้นเสร็จก่อนกำหนด
E_i	=	เวลาเสร็จก่อน (Earliness) หมายถึง เวลาที่งาน i เสร็จก่อนกำหนดส่งมอบ
T_i	=	เวลาล่าช้า (Tardiness) หมายถึง เวลาที่งาน i เสร็จหลังกำหนดส่งมอบ
α_i	=	อัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จก่อนกำหนดส่งมอบงาน (Weighted Earliness) หมายถึง อัตราค่าใช้จ่ายของงาน i เมื่องานเสร็จก่อนกำหนดส่งมอบ เช่น ค่าใช้จ่ายในการจัดเก็บวัสดุคงคลัง
β_i	=	อัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จหลังกำหนดส่งมอบงาน (Weighted Tardiness) หมายถึง อัตราค่าใช้จ่ายของงาน i เมื่องานเสร็จหลังกำหนดส่งมอบ เช่น ค่าใช้จ่ายที่เกิดขึ้นเนื่องจากการสูญเสียลูกค้า

การใช้วิธีเชิงพันธุกรรมเพื่อแก้ไขปัญหาการจัดตารางการผลิตของเครื่องจักรขนานที่ไม่เหมือนกันแบบที่มีค่าใช้จ่ายเกิดจากผลิตเสร็จก่อนและผลิตเสร็จหลังวันกำหนดส่งมอบ

A Genetic Algorithm for Non-Identical Parallel Machine Scheduling Problems with Earliness and Tardiness Penalties

คำนำ

ความเป็นมาและความสำคัญของปัญหา

ในภาวะเศรษฐกิจในปัจจุบันที่มีภาวะการแข่งขันค่อนข้างสูง ทำให้หลายๆองค์กรมุ่งเน้นที่จะลดค่าใช้จ่ายของตนลง เพื่อให้ต้นทุนของงานต่ำที่สุดและสามารถส่งมอบงานให้กับลูกค้าได้ตามกำหนดส่งมอบ โดยเฉพาะจากความต้องการที่เปลี่ยนแปลงที่เกิดขึ้นอยู่ตลอดเวลาของลูกค้านั้นเป็นสาเหตุหนึ่งที่ทำให้การวางแผนการผลิตดำเนินการได้ยาก แต่สำหรับการดำเนินธุรกิจที่มุ่งเน้นให้มีการดำเนินงานให้ตรงตามเป้าหมายของบริษัทแล้วยังต้องคำนึงถึงความพึงพอใจของลูกค้าอีกด้วย การวางแผนการผลิตจึงถือว่าเป็นส่วนสำคัญส่วนหนึ่งของอุตสาหกรรมในปัจจุบัน หากต้องมีการเร่งทำงานหรือเพิ่มชั่วโมงทำงานเพื่อให้ทันกำหนดส่งมอบ ซึ่งส่งผลต่อเนื่องไปถึงต้นทุนในการผลิต นอกจากนี้ งานที่ผลิตเสร็จก่อนวันส่งมอบ จะเกิดค่าใช้จ่ายในการจัดเก็บงานนั้น เนื่องจากลูกค้าจะไม่รับงานก่อนถึงวันกำหนดส่งมอบซึ่งจะทำให้เกิดค่าใช้จ่ายในการจัดเก็บงานของลูกค้าเอง ในทางตรงกันข้ามหากงานผลิตเสร็จหลังวันกำหนดส่งมอบก็จะเกิดค่าใช้จ่ายจากการปรับของลูกค้าเช่นกัน จึงต้องมีการวางแผนการผลิตที่เหมาะสมเพื่อให้เกิดค่าใช้จ่ายดังที่กล่าวมาข้างต้นน้อยที่สุด

งานวิจัยนี้จึงจัดทำขึ้นเพื่อทำการวิเคราะห์การแก้ไขปัญหาการจัดตารางการผลิตทำการพัฒนาโปรแกรมคอมพิวเตอร์เพื่อสนับสนุนการใช้งาน เพื่อให้เกิดความสะดวกและรวดเร็วในการทำงานมากยิ่งขึ้น

ขอบเขตของงานวิจัย

ขอบเขตของงานวิจัยนี้จะพิจารณาการแก้ไขปัญหาการจัดตารางการผลิตของเครื่องจักรขนานที่ไม่เหมือนกัน (Non-identical Parallel Machines Scheduling) โดยความเร็วของเครื่องจักรเป็นจำนวนเท่าของเลขจำนวนเต็ม (integer) , มีวันส่งมอบที่แตกต่างกัน (Distinct Due Date) , มีการถ่วงน้ำหนักของอัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จก่อนและหลังกำหนดส่งมอบงานและมีฟังก์ชั่นวัตถุประสงค์คือ ผลรวมของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและผลิตเสร็จหลังวันกำหนดส่งมอบ (Earliness and Tardiness Costs) มีค่าน้อยที่สุด โดยใช้วิธีเชิงพันธุกรรม (Genetic Algorithm) ในการจัดแบ่งงานและลำดับงานในแต่ละเครื่องจักร หลังจากที่ทราบจำนวนงานและลำดับงานที่แน่นอนในแต่ละเครื่องจักรแล้ว จึงใช้วิธีการจัดเวลาเริ่มงาน (Optimal Timming Algorithm) มาช่วยในการจัดหาเวลาเริ่มงาน

ประโยชน์ที่คาดว่าจะได้รับ

1. เพื่อให้ได้การจัดตารางการผลิตของเครื่องจักรขนานที่ไม่เหมือนกันที่มีประสิทธิภาพ
2. เพื่อพัฒนาโปรแกรมช่วยในการตัดสินใจในการจัดตารางการผลิต
3. เพื่อลดเวลาที่ใช้ในการจัดตารางการผลิต

วัตถุประสงค์

1. ทำการสร้างรูปแบบการแก้ปัญหาการจัดตารางการผลิตที่ใกล้เคียงกับสภาพแวดล้อมการผลิตในสถานการณ์จริงมากที่สุด

2. นำวิธีเชิงพันธุกรรมมาประยุกต์ในการแก้ปัญหาการจัดตารางการผลิต

3. พัฒนาโปรแกรมคอมพิวเตอร์เพื่อสนับสนุนการใช้งานด้านการจัดตารางการผลิต

การตรวจเอกสาร

ทฤษฎีที่เกี่ยวข้องกับงานวิจัยนี้แบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ ทฤษฎีการจัดตาราง โดยเน้นที่เครื่องจักรแบบขนานที่ไม่เหมือนกันและมีวันส่งมอบที่แตกต่างกัน โดยมีฟังก์ชั่นวัตถุประสงค์คือผลรวมของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและหลังวันกำหนดส่งมอบมีค่าน้อยที่สุด อีกส่วนคือทฤษฎีวิธีเชิงพันธุกรรม

ทฤษฎีการจัดตาราง

1. ความรู้เบื้องต้น

ปารเมศ (2546) กล่าวไว้ว่า ทฤษฎีการจัดตารางเกี่ยวข้องกับการสร้างและพัฒนาแบบจำลองทางคณิตศาสตร์ และการหาเทคนิคที่เหมาะสมในการแก้ปัญหาการจัดตาราง ซึ่งจะต้องอาศัยความรู้ทั้งในภาคทฤษฎีและภาคปฏิบัติร่วมกัน แล้วใช้การวิเคราะห์เชิงปริมาณเป็นเครื่องมือช่วย โดยแนวทางดังกล่าวนี้จะแปลงโครงสร้างของปัญหาการจัดตารางไปสู่รูปแบบของสมการทางคณิตศาสตร์ที่เหมาะสม ซึ่งกระบวนการนี้จะเกี่ยวข้องกับการแปลงเป้าหมายและความมีอยู่อย่างจำกัดของทรัพยากรในด้านต่างๆ ที่เกี่ยวข้องกับการตัดสินใจ ไปสู่ฟังก์ชั่นวัตถุประสงค์ (Objective Function) และข้อจำกัด (Constraint) ต่างๆ ซึ่งจะเขียนขึ้นมาอย่างชัดเจนในรูปแบบของสมการทางคณิตศาสตร์

ในทางทฤษฎีฟังก์ชั่นวัตถุประสงค์ของการจัดตารางควรจะประกอบด้วยค่าใช้จ่าย (Cost) ทั้งหมดที่เกิดขึ้นในระบบ ซึ่งค่าใช้จ่ายเหล่านี้จะได้รับผลกระทบโดยตรงจากการตัดสินใจจัดตารางในครั้งนี้ อย่างไรก็ตามในทางปฏิบัติแล้วค่าใช้จ่ายดังกล่าวอาจจะวัดออกมาเป็นตัวเลขได้ยากมาก ดังนั้นแทนที่จะแสดงฟังก์ชั่นวัตถุประสงค์ในรูปของค่าใช้จ่าย เราจะใช้เป้าหมาย 3 รูปแบบหลักในการตัดสินใจที่เกี่ยวข้องในการจัดตารางแทน นั่นคือ ประสิทธิภาพในการใช้สอยทรัพยากร (Resource Utilization) ความรวดเร็วในการสนองตอบต่ออุปสงค์ และการส่งมอบที่ตรงเวลา นอกจากนั้นแล้วเรายังอาจจะใช้ตัววัดสมรรถนะของระบบตัวอื่นๆแทนตัววัดที่เกิดจากค่าใช้จ่ายของระบบด้วยได้เช่นกัน ตัวอย่างเช่น เวลาเดินเปล่า (Idle Time) ของเครื่องจักร เวลารอคอยของงาน หรือเวลาสาย (Lateness) ของงาน เป็นต้น

ในการจัดตารางนั้น ข้อจำกัด 2 ประเภทที่พบเสมอก็คือ

1. ข้อจำกัดทางทรัพยากร (Resource Constraint) เกี่ยวข้องกับทรัพยากรที่มีความสามารถในการทำงานอย่างจำกัดในขณะใดขณะหนึ่ง เช่น เครื่องจักรเครื่องหนึ่งสามารถทำงานได้กับชิ้นงานเพียงชิ้นเดียวเท่านั้นที่เวลาใดเวลาหนึ่ง

2. ข้อจำกัดด้านเทคโนโลยี (Technological Constraint) เกี่ยวข้องกับความจำกัดในด้านลำดับก่อนหลังของการทำงาน (Precedence Constraint) เช่น เราต้องทำงานแรกบนเครื่องจักรเครื่องหนึ่งให้แล้วเสร็จก่อนที่จะเริ่มต้นทำงานที่ 2 บนเครื่องจักรเดียวกันนั้นได้

ดังนั้นตารางที่เป็นไปได้จริง ซึ่งเป็นผลลัพท์ของการแก้ปัญหาการจัดตาราง จะต้องเป็นไปตามเงื่อนไขของข้อจำกัดทั้งสองที่กล่าวมา และเพื่อทำให้สามารถตอบ 2 คำถามหลักที่เกี่ยวข้องกับการจัดตารางได้ กล่าวคือ จะใช้ทรัพยากรตัวไหนจากทรัพยากรหลายตัวที่มีอยู่พร้อมใช้งานเพื่อทำงานที่กำหนดให้และเราจะลงมือทำงานแต่ละงานเมื่อใด

ในทางปฏิบัติพบว่า บางครั้งปัญหาการจัดตารางอาจจะเกี่ยวข้องกับการตัดสินใจเพียงประเภทเดียวก็ได้ ทั้งนี้ขึ้นอยู่กับคุณลักษณะของระบบและปัญหาที่กำลังพิจารณาอยู่ เช่น ในการทำงานอย่างหนึ่งในโรงงาน พบว่ามีเครื่องจักรที่สามารถทำงานได้เพียงเครื่องจักรเดียวเท่านั้น ดังนั้นการตัดสินใจเกี่ยวกับการจัดสรรทรัพยากรจึงไม่เกิดขึ้น เนื่องจากไม่สามารถเลือกทำงานนี้บนเครื่องจักรได้ จะมีก็แต่การตัดสินใจเกี่ยวกับการจัดลำดับงานที่เหมาะสมเพื่อป้อนให้เครื่องจักรเครื่องนี้เท่านั้น หรือในทางตรงข้าม การทำงานอีกประเภทหนึ่งสามารถที่จะทำได้บนหลายเครื่องจักร แต่เมื่อได้จัดสรรและป้อนงานเหล่านี้ให้กับเครื่องจักรเครื่องใดเครื่องหนึ่งแล้ว จะไม่มีการเปลี่ยนแปลงลำดับงานที่อยู่บนแถวคอยหน้าเครื่องจักรอีก (โดยถือเอาลำดับของการจัดสรรงานให้กับเครื่องจักรเป็นลำดับของงานบนแถวคอยเลย) เนื่องจากการเปลี่ยนแปลงดังกล่าวอาจจะก่อให้เกิดความยุ่งยากและสับสนในการทำงานของคนงานเป็นอันมาก ดังนั้นการตัดสินใจในกรณีเช่นนี้ก็จะจำกัดอยู่เฉพาะที่เกี่ยวข้องกับการจัดสรรทรัพยากรเท่านั้น

2. โครงสร้างและคำอธิบาย

ในการพิจารณาปัญหาการจัดตาราง จำนวนของงานและเครื่องจักรจะถูกสมมุติให้มีจำนวนจำกัด โดยมีจำนวนงาน n งานและจำนวนเครื่องจักร m เครื่อง ตัวห้อย i จะอ้างอิงถึงงาน ขณะที่ตัวห้อย j จะอ้างอิงถึงเครื่องจักร เมื่อต้องการระบุว่างาน i ที่ทำบนเครื่องจักร j จะระบุเป็นคู่คือ (ij) นอกจากนี้ยังมีคำและสัญลักษณ์อื่นๆ ที่ใช้ในการจัดตารางอีกดังนี้

เวลาทำงานหรือเวลาปฏิบัติงาน (Processing Time; p_{ij}) หมายถึงเวลางานที่ i จะต้องใช้บนเครื่องจักร j ตัวห้อย j อาจจะถูกละเว้นได้ถ้าเวลาการทำงานของงาน i ไม่ขึ้นกับเครื่องจักรที่ใช้หรืองานนี้ทำได้เฉพาะกับเครื่องจักรเพียงเครื่องเดียวเท่านั้น

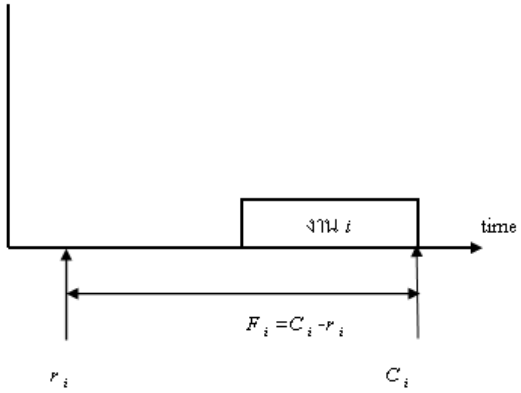
เวลาปล่อยงาน (Release Date; r_i) ของงาน i หมายถึง เวลาที่งาน i เข้ามาสู่ระบบ หรืออาจจะหมายถึง เวลาที่พร้อมจะเริ่มทำงาน (Ready Date) ได้ หรือเวลาที่เร็วที่สุดที่งาน i จะเริ่มได้

กำหนดส่งมอบ (Due Date; d_i) ของงาน i หมายถึง เวลาจัดส่งหรือเวลาเสร็จงานที่สัญญาไว้กับลูกค้า การทำงานเสร็จหลังจากกำหนดส่งมอบงานอาจเกิดขึ้นได้ แต่อาจจะมีค่าปรับเกิดขึ้น ส่วนคำว่าเส้นตาย (Dead Line) หมายถึง กำหนดส่งมอบที่ต้องทำให้ได้ ซึ่งถ้าไม่ได้แล้ว จะมีผลเสียต่อองค์กรเป็นอย่างมาก ทั้งในรูปตัวเงินและชื่อเสียง

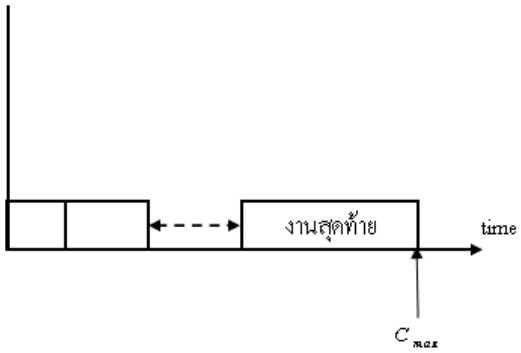
เวลาเสร็จงาน (Completion Time; C_{ij}) หมายถึง เวลาที่งาน i เสร็จสิ้นการทำงานบนเครื่องจักร j ถ้าตัวห้อย j ถูกละไว้ C_i จะหมายถึง เวลาที่งาน i ออกจากระบบ ข้อมูลที่เกี่ยวกับเวลาเสร็จงานจะมีลักษณะเป็นแบบพลวัตร (Dynamic) เพราะว่าเวลาเสร็จงานนี้จะขึ้นอยู่กับตารางที่สร้างขึ้นมา ซึ่งต่างกับข้อมูลที่กล่าวมาแล้วข้างต้นที่มีลักษณะไม่ขึ้นกับตารางที่จัดขึ้น ดังนั้นจึงมีลักษณะเป็นแบบคงที่ (Static)

เวลาไหลของงาน (Flow Time; F_i) หมายถึง เวลาทั้งหมดของงาน i ที่ใช้เวลาอยู่ในระบบ เขียนแทนด้วย $F_i = C_i - r_i$ ตามภาพที่ 1

เวลาปิดงานของระบบ (Makespan; C_{max}) หมายถึง เวลาที่ระบบทำงานชิ้นสุดท้ายเสร็จสิ้น ตามภาพที่ 2



ภาพที่ 1 เวลาไหลของงาน



ภาพที่ 2 เวลาปิดงานของระบบ

เวลาเสร็จก่อน (Earliness; E_i) หมายถึง เวลาที่งาน i เสร็จก่อนกำหนดส่งมอบ โดยหาได้จากสมการ (1)

$$E_i = \max(0, C_i - d_i) \quad (1)$$

เวลาล่าช้า (Tardiness; T_i) หมายถึง เวลาที่งาน i เสร็จหลังกำหนดส่งมอบโดยหาได้จากสมการ (2)

$$T_i = \max(0, d_i - C_i) \quad (2)$$

อัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จก่อนกำหนดส่งมอบงาน (Weighted Earliness; α_i) หมายถึง อัตราค่าใช้จ่ายของงาน *i* เมื่องานเสร็จก่อนกำหนดส่งมอบ

อัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จหลังกำหนดส่งมอบงาน (Weighted Tardiness; β_i) หมายถึง อัตราค่าใช้จ่ายของงาน *i* เมื่องานเสร็จหลังกำหนดส่งมอบ

ค่าใช้จ่ายเมื่องานเสร็จก่อนกำหนด (Earliness Cost) ของงาน *i* หมายถึงค่าใช้จ่ายที่เกิดขึ้นเมื่อเวลาเสร็จของงานเสร็จก่อนกำหนดส่งมอบ ซึ่งอาจหมายถึงค่าใช้จ่ายในการจัดเก็บวัสดุ หาได้โดย นำอัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จก่อนกำหนดส่งมอบงานคูณด้วยเวลาเสร็จก่อน ($\alpha_i E_i$)

ค่าใช้จ่ายเมื่องานเสร็จหลังกำหนด (Tardiness Cost) ของงาน *i* หมายถึงค่าใช้จ่ายที่เกิดขึ้นเมื่อเวลาเสร็จของงานเสร็จหลังกำหนดส่งมอบ ซึ่งอาจหมายถึงค่าปรับที่ต้องจ่ายให้กับลูกค้า หาได้โดย นำอัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จหลังกำหนดส่งมอบงานคูณด้วยเวลาล่าช้า ($\beta_i T_i$)

3. การจัดเรียงเครื่องจักร

รูปแบบที่สำคัญของการจัดเรียงเครื่องจักรมีอยู่หลายลักษณะ ขึ้นอยู่กับระบบการทำงานและปัจจัยแวดล้อมต่างๆขององค์กร ซึ่งสามารถแบ่งได้ดังนี้

เครื่องจักรเดี่ยว (Single Machine) ระบบนี้ประกอบไปด้วยเครื่องจักรเพียงเครื่องเดียว ซึ่งเป็นรูปแบบที่ง่ายที่สุดในรูปแบบการจัดเรียงเครื่องจักรที่เป็นไปได้ทั้งหมด

เครื่องจักรขนานที่เหมือนกัน (Identical Parallel Machines) ระบบนี้ประกอบไปด้วยเครื่องจักร *m* เครื่องที่เหมือนกัน ซึ่งมีการทำงานแบบขนานกัน ระบบผลิตจำนวนมากมีการทำงานแบบนี้ ตัวอย่างเช่น ในโรงงานแห่งหนึ่งมีสายการผลิตที่ประกอบไปด้วยหลายสถานีงาน ซึ่งแต่ละสถานีงานอาจประกอบไปด้วยเครื่องจักรที่ขนานกันอยู่จำนวนหลายเครื่องจักร เมื่องาน *j* มาถึงแต่ละสถานีงานที่มีเครื่องจักรขนานกันอยู่นั้น งาน j สามารถที่จะเลือกทำได้บนเครื่องจักรเครื่องใดก็ได้ใน m เครื่องจักรเหล่านี้

เครื่องจักรขนานที่อัตราการผลิตต่างกัน (Non- Identical Parallel Machines) ระบบนี้ประกอบไปด้วยเครื่องจักร *m* เครื่องที่เหมือนกัน ซึ่งมีการทำงานแบบขนานกัน แต่ทว่าเครื่องจักรแต่ละเครื่องมีความเร็วในการทำงานต่างกัน

เครื่องจักรขนานที่ไม่สัมพันธ์กัน (Unrelated Parallel Machines) ระบบนี้ประกอบไปด้วยเครื่องจักร *m* เครื่องที่มีการทำงานแบบขนานกัน เวลาที่ใช้ทำในแต่ละงานขึ้นอยู่กับงานนั้นๆที่ทำ การผลิตแบบไหลเลื่อน (Flow Shop) ระบบนี้ประกอบไปด้วยเครื่องจักร *m* เครื่อง งานทั้งหมดจะมีเส้นทางการไหลของงานเป็นรูปแบบเดียวกัน การดำเนินงานทั้งหมดที่อยู่ในลำดับเดียวกันจะต้องถูกดำเนินการโดยเครื่องจักรเครื่องเดียวกัน

การผลิตแบบไหลเลื่อนยืดหยุ่น (Flexible Flow Shop) ระบบนี้เป็นรูปแบบทั่วไปของระบบผลิตแบบไหลเลื่อนและระบบเครื่องจักรขนาน ในระบบนี้จะประกอบไปด้วย c ขั้นตอนการดำเนินงานที่เรียงลำดับกันอยู่ ในแต่ละขั้นตอนการดำเนินงานจะมีเครื่องจักรขนานที่เหมือนกันอยู่เป็นจำนวนหนึ่ง งานแต่ละงานจะต้องผ่านการดำเนินงานในขั้นที่ 1 ขั้นที่ 2 เรื่อยไปจนกระทั่งขั้นสุดท้าย ในแต่ละขั้นของการดำเนินงาน งานจะสามารถเลือกทำการดำเนินงานที่กำหนดไว้บนเครื่องจักรใดเครื่องจักรหนึ่งที่ขนานกันอยู่ได้

การผลิตแบบตามงาน (Job Shop) ระบบนี้ประกอบไปด้วยเครื่องจักร *m* เครื่อง แต่ละงานมีเส้นทางการไหลของงานเฉพาะของตนเองตามที่ผู้วางแผนกระบวนการกำหนดไว้เท่านั้น

การผลิตแบบตามสั่งยืดหยุ่น (Flexible Job Shop) ระบบนี้เป็นรูปแบบทั่วไปของระบบผลิตแบบตามงานและระบบเครื่องจักรขนาน ระบบนี้จะประกอบไปด้วย c สถานีงาน ในแต่ละสถานีงานจะมีเครื่องจักรขนานที่เหมือนกันอยู่เป็นจำนวนหนึ่ง แต่ละงานมีเส้นทางการไหลของงานเฉพาะของตนเอง และสามารถเลือกทำการดำเนินงานที่กำหนดบนเครื่องจักรใดเครื่องจักรหนึ่งที่ขนานกันอยู่และอยู่ในสถานีงานเดียวกันได้

การผลิตแบบเปิด (Open Shop) ระบบนี้ประกอบไปด้วยเครื่องจักร *m* เครื่อง แต่ละงานจะต้องมีการดำเนินงานแบบเวียนซ้ำบนเครื่องจักรแต่ละเครื่อง ซึ่งเวลาในการดำเนินงานนี้อาจจะเท่ากับศูนย์ก็ได้ ไม่มีข้อจำกัดเกี่ยวกับเส้นทางงานของแต่ละงาน ดังนั้นผู้จัดตารางจะเป็นผู้กำหนดเส้นทางงานให้กับแต่ละงาน และงานที่ต่างกันอาจจะมีเส้นทางงานที่ต่างกันได้

4. ตัววัดสมรรถนะ

ฟังก์ชั่นวัตถุประสงค์ที่จัดว่าเป็น “ตัววัดสมรรถนะแบบปกติ (Regular Measure of Performance)” เป็นฟังก์ชั่นของเซตเวลาเสร็จงาน และมีรูปแบบทั่วไป คือ

$$Z = f(C_1, C_2, \ldots, C_n) \qquad (3)$$

กล่าวคือ ตัววัดสมรรถภาพ Z เป็นแบบปกติได้ก็ต่อเมื่อ

1. วัตถุประสงค์ของการจัดตารางเพื่อทำให้ Z มีค่าต่ำที่สุด
2. Z จะมีค่าเพิ่มขึ้นได้ ก็ต่อเมื่อเวลาเสร็จงานอย่างน้อยหนึ่งค่าที่อยู่ในตารางมีค่าเพิ่มขึ้น

หรือกล่าวอีกนัยหนึ่งก็คือ สมมติว่า $Z = f(C_1, C_2, \ldots, C_n)$ เป็นตัวของค่าสมรรถนะของตาราง S และถ้า $Z' = f(C'_1, C'_2, \ldots, C'_n)$ เป็นค่าของตัววัดสมรรถนะตัวเดียวกัน แต่อยู่ภายใต้ S' ที่แตกต่างกันแล้ว ดังนั้น Z จะเป็นปกติก็ต่อเมื่อ ถ้า $Z' > Z$ แล้ว จะพบว่ามีงาน j บางงานที่มี $C'_j > C_j$

ในหลายๆ ปีที่ผ่านมางานวิจัยส่วนใหญ่มักจะเน้นไปที่เครื่องจักรเดียว(Single machine) และฟังก์ชั่นวัตถุประสงค์จัดเป็นตัววัดสมรรถนะแบบปกติ(Regular measure of performance) สำหรับฟังก์ชั่นวัตถุประสงค์ ผลรวมของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและหลังวันกำหนดส่งมอบมีค่าน้อยที่สุด จัดเป็นตัววัดสมรรถนะแบบไม่ปกติ(Non-regular measure of performance) เนื่องจาก เมื่อเวลาเสร็จงานอย่างน้อยหนึ่งค่าที่อยู่ในตารางมีค่าเพิ่มขึ้นค่า Z จะมีค่าเพิ่มขึ้นหรือลดลงก็ได้ ซึ่งที่ใช้กันทั่วไปยังสามารถแบ่งได้ออกเป็น 2 แบบ คือ การหาค่าที่น้อยที่สุดของผลรวม (Minsum)และการหาค่าที่น้อยที่สุดของค่าที่มากที่สุด(Minmax) โดยปัญหาการหาค่าที่น้อยที่สุดของผลรวมเป็นการหาผมรวมของค่าสัมบูรณ์ที่น้อยที่สุดของการเบี่ยงเบนของวันปิดงานกับวันกำหนดส่งมอบงานที่ถูกถ่วงน้ำหนัก(Minimize the sum weighted absolute deviation of job completion time about due date) ส่วนปัญหาการหาค่าที่น้อยที่สุดของค่าที่มากที่สุดเป็นการหาค่าที่น้อยที่สุดของค่าสัมบูรณ์ที่มากที่สุดของการเบี่ยงเบนของวันปิดงานกับวันกำหนดส่งมอบงานที่ถูกถ่วงน้ำหนัก(Minimize the maximum weighted absolute deviation of job completion time about due date)

Cheng and Sin (1990); Mokotoff (2001) ได้รวบรวมงานที่เกี่ยวกับปัญหาการจัดตารางให้กับงานจำนวนหนึ่งที่ต้องทำบนเครื่องจักรขนานกัน ได้แบ่งขั้นตอนประกอบการตัดสินใจหลักอยู่ 2 ชนิดด้วยกัน คือ การจัดสรรงาน (Allocation) และการจัดลำดับงาน (Sequencing) ส่วน Sugimoto *et al*. (1996); Tamaki *et al*. (1999); Tamaki *et al*. (2003) ได้แบ่งขั้นตอนเป็น 3 ชนิดด้วยกัน โดยเพิ่ม การจัดเวลาเริ่มงาน (Optimal Timing Algorithm) เข้าไปอีกส่วน ซึ่งเมื่อทำการจัดสรรงานแล้ว ในส่วนของการจัดลำดับงานและการจัดเวลาเริ่มงานสามารถพิจารณาเป็นเครื่องจักรเดี่ยวได้

งานที่รวบรวมโดย Cheng and Sin (1990) ได้แบ่งงานตามตัววัดสมรรถนะ (performance criteria) สำหรับเครื่องจักรขนานไว้ 3 แบบ คือ แบบที่หนึ่ง ตัววัดสมรรถนะบนพื้นฐานของเวลาปิดงาน (completion time based (CTB) performance measure) แบบที่สอง ตัววัดสมรรถนะบนพื้นฐานของวันกำหนดส่งมอบ (due date based (DDB) performance measure) และแบบที่สาม ตัววัดสมรรถนะบนพื้นฐานของเวลาไหลของงาน (flow time based (FTB) performance measure) ซึ่งเป็นกรณีที่เกิดขึ้นเมื่อวันกำหนดส่งมอบงานทั้งหมดเท่ากับศูนย์

Baker and Scudder (1990) ได้ทบทวนและรวบรวมงานที่เกี่ยวกับค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและหลังวันกำหนดส่งมอบ งานส่วนใหญ่เป็นเครื่องจักรเดียวและมีวันกำหนดส่งมอบเหมือนกัน(Common due date) ซึ่งผลที่ได้แสดงให้เห็นว่า เมื่อวันกำหนดส่งมอบเหมือนกันทุกงาน การจัดตารางที่ดีที่สุดจะเป็นรูปตัววี (V-shaped) และไม่มีเวลาเว้นว่าง(no inserted idle time) ระหว่างงาน มีงานเพียงงานเดียวที่เสร็จในวันส่งมอบ งานที่เสร็จก่อนกำหนดวันส่งมอบจะจัดลำดับแบบเวลาปฏิบัติงานมากที่สุด(Longest processing time; LPT) และงานที่เสร็จหลังกำหนดวันส่งมอบจะจัดลำดับแบบเวลาปฏิบัติงานน้อยที่สุด(Shortest processing time; SPT)

Gary *et al*. (1988) ได้ทำการพิสูจน์ว่าปัญหาเครื่องจักรเดียวและมีวันส่งมอบที่แตกต่างกันโดยมีฟังก์ชั่นวัตถุประสงค์คือ ผลรวมที่มีค่าน้อยที่สุดของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและหลังวันกำหนดส่งมอบมีอัตราเท่ากัน(symmetric earliness and tardiness penalties) เป็นปัญหาเอ็นพีบริบูรณ์ (NP-complete) Mandel and Mosheiov (1999)ได้ทำการพิสูจน์ว่าปัญหาเครื่องจักรขนานที่เหมือนกัน 2 เครื่องจักร โดยมีฟังก์ชั่นวัตถุประสงค์คือ ผลรวมที่มีค่าน้อยที่สุดของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนวันกำหนดส่งมอบ เป็นปัญหาเอ็นพีแบบยาก (NP-hard) ดังนั้น โดยสามัญสำนึก (ordinary sense) จึงอาจกล่าวได้ว่าปัญหาเครื่องจักรขนานที่ไม่เหมือนกันและมีวันส่งมอบที่

แตกต่างกัน โดยมีฟังก์ชั่นวัตถุประสงค์คือ ผลรวมของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและหลังวันกำหนดส่งมอบมีค่าน้อยที่สุด เป็นปัญหาเอ็นพีแบบยากเช่นกัน

4. การจัดเวลาเริ่มงาน (Optimal Timing Algorithm)

หลังจากที่จัดสรรงานให้กับแต่ละเครื่องจักรแล้ว สามารถพิจารณาแยกในแต่ละเครื่องจักรเป็นแบบเครื่องจักรเดียวได้ เมื่อทราบถึงลำดับงานที่แน่นอนแล้ว จึงมีการคิดวิธีการเพื่อหาวันที่เริ่มปฏิบัติงาน เพื่อให้งานเสร็จใกล้เคียงกับวันกำหนดส่งมอบมากที่สุดเพื่อให้ค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและหลังวันกำหนดส่งมอบมีค่าน้อยที่สุด งานจะถูกจัดเป็นบล๊อคๆ ซึ่งการจัดเวลาเริ่มงานทำให้เกิดเวลาว่างขึ้นระหว่างงาน(Insert idle time) โดยได้มีผู้วิจัยหลายท่านได้คิดค้นวิธีการจัดเวลาเริ่มงานไว้ เช่น Sidney (1976); Lakshminarayan *et al*. (1977); Davis and Kenet (1993); Lee and Choi (1995); Szwarc and Mukhhopadhyay (1995)

วิธีของ Lee and Choi (1995) จะมีการคำนวณที่ไม่มากเนื่องจากไม่ได้นำมาเปรียบเทียบกับฟังก์ชั่นวัตถุประสงค์ โดยให้ S_k เป็นตารางงานสำหรับ k งานเรียงตามลำดับกำหนดส่งมอบ โดยงานที่มีกำหนดส่งมอบเร็วที่สุดจะอยู่ลำดับที่ 1 ส่วนงานที่มีกำหนดส่งมอบช้าที่สุดจะอยู่ลำดับสุดท้าย สำหรับงานที่จัดเรียงต่อเนื่องใน S_k เรียกว่าบล็อค สมมติว่ามี t บล็อค $B_1, \ldots, B_t$ ในตาราง S_k เริ่มต้นโดยตาราง S_1 โดยจัดให้งาน J_1 เสร็จตามวันกำหนดส่งมอบ กำหนดตาราง S_k มาให้, งาน J_{k+1} จะถูกจัดตารางตามนี้คือ

ถ้า $C_k + p_{k+1} \le d_{k+1}$ แล้วงาน J_{k+1} จะเริ่มบล็อคใหม่ B_{t+1}

แต่ถ้า $C_k + p_{k+1} > d_{k+1}$ แล้วงาน J_{k+1} จะเริ่มต้นที่ C_k ในลำดับสุดท้ายของบล็อค B_t

พิจารณาฟังก์ชั่นค่าใช้จ่ายเพื่อใช้ในการเลื่อนบล็อค B_j โดยฟังก์ชั่นค่าใช้จ่ายของบล็อคที่น้อยที่สุดอยู่ที่ extreme point ซึ่งเป็นจุดที่ค่าความชันเริ่มมากขึ้นหรือเท่ากับศูนย์ เมื่อเจอจุดที่น้อยที่สุดของบล็อค B_j แล้ว ก็จะทำการเลื่อนไปยังจุดที่น้อยที่สุดนั้นจนกระทั่งเกิดกรณีใดกรณีหนึ่งใน 3 กรณีดังนี้

1. เวลาเริ่มต้นของงานแรกเป็นศูนย์
2. เจอจุดที่น้อยที่สุด
3. เวลาเริ่มต้นของงานแรกในบล็อค B_j เท่ากับเวลาปิดงานของงานสุดท้ายในบล็อค B_{j-1}

สำหรับในกรณีที่ 3 เมื่อบล็อค B_j ต่อพอดีกับบล็อค B_{j-1} จะได้ค่าที่น้อยที่สุดค่าใหม่สำหรับบล็อค B_{j-1} แล้วทำซ้ำต่อไปจนกระทั่งได้ตำแหน่งของแต่ละบล็อคที่ทำให้ได้ค่าใช้จ่ายที่น้อยที่สุดหรือเวลาเริ่มต้นของงานแรกเท่ากับศูนย์ จากวิธีการดังกล่าว ทำให้สามารถหาเวลาเริ่มต้นของแต่ละงานได้ตามในภาพที่ 3

```
procedure optimal_time;
    t:=first(1) := last(1) := 1;
    s1 := max(0, d1 – p1); t1 := max(d1, p1); B1 = {1};
    for i := 2 to n do
        if ti-1+ p1< d1 then
        { t := t + 1; first(t) := last(t) := i;
          si := di - pi ; ti = di ; Bi = {i};
        else if ti-1 + pi = di then
        { last(t) := i; si = ti-1; ti=di; Bi = Bi ∪{i}}
        else if ti-1 + pi > di then
        { last(t) := i; si = ti-1 ; ti = si + pi; Bi = Bi ∪{i} }
        until each block is located either at the minimum point or
                sfirst(t) = 0
                shift
        end until
        }
    end for
end optimal_time
```

ภาพที่ 3 ขั้นตอนของการจัดเวลาเริ่มงาน (Optimal Timing Algorithm)
ที่มา: Lee and Choi (1995)

ตัวอย่างที่ 1 พิจารณาปัญหาการจัดลำดับงานของงาน 4 งาน

i	1	2	3	4
p_i	3	5	5	2
d_i	6	9	12	17
α_i	1	1	1.2	1
β_i	1	1.5	1.5	1

กำหนด J_1 อยู่บล็อค B_1 เริ่มปฏิบัติงาน ที่ $t_1 = 3$, เวลาเสร็จงาน $C_1 = 6$

พิจารณา J_2, $C_2 = C_1 + p_2 = 11 > d_2$ ดังนั้น J_2 จัดเข้าบล็อค B_1 ตามหลัง J_1

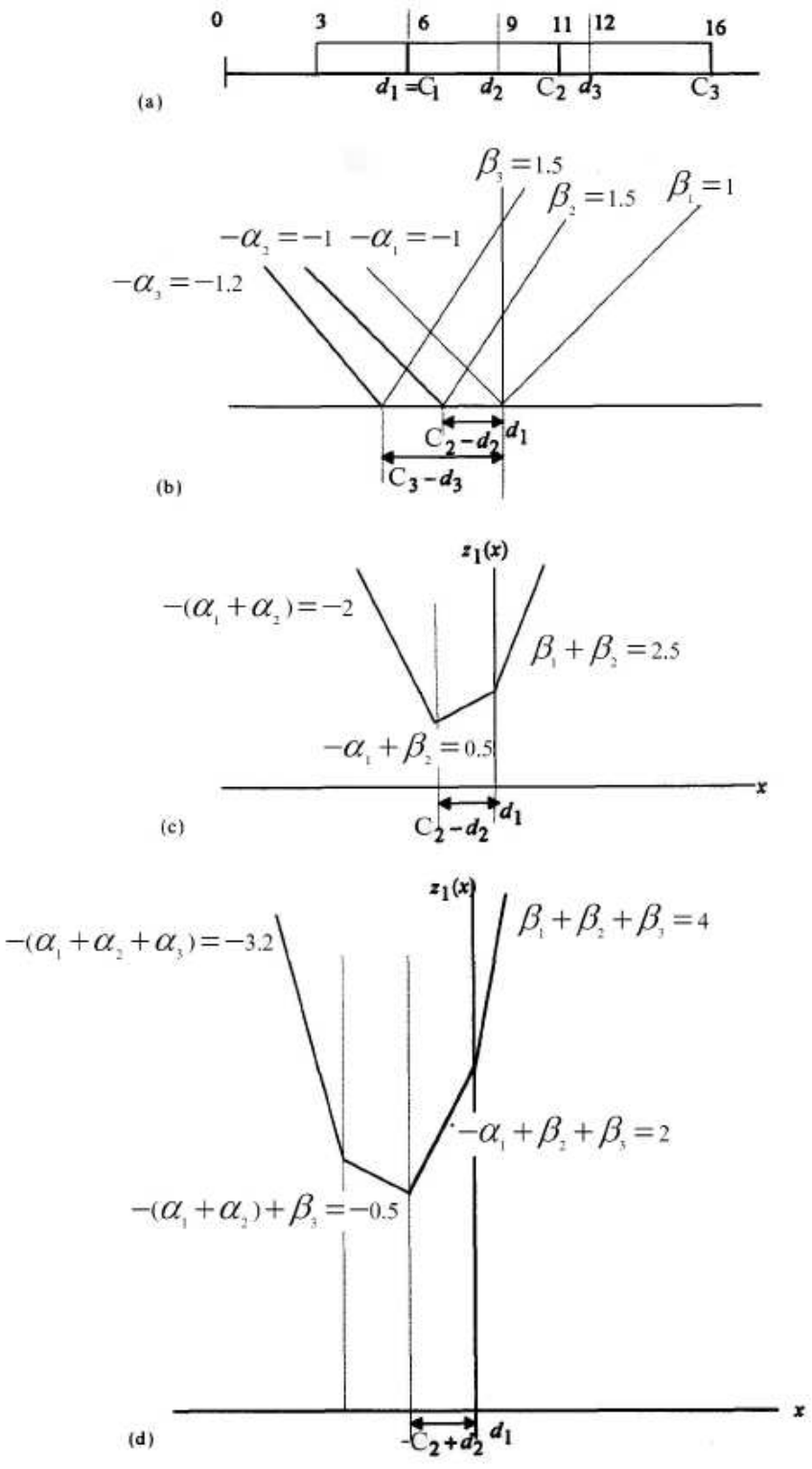
เลื่อนฟังก์ชั่นโดยพิจารณาจากจุดต่ำสุด (Minimum point) ของฟังก์ชั่นค่าใช้จ่าย (Cost Function) โดยการเปรียบเทียบความชัน (slope) เพื่อเลื่อนฟังก์ชั่นจะได้ผลตามภาพที่ 3(C) โดย $C_2 - d_2 = 2 < s_1$, บล็อค 1 เลื่อนไปทางซ้ายมือ 2 หน่วยเวลา ดังนั้นงานแรกของบล็อค 1 เสร็จก่อนกำหนดส่งมอบ 2 หน่วยเวลา เวลาเริ่มงานของงาน J_1คือ $t_1 = 1$ และเวลาเริ่มงานของงาน J_2 คือ $t_2 = 4$

พิจารณา J_3, $C_3 = C_2 + p_3 = 14 > d_3$ ดังนั้น J_3 จัดเข้าบล็อค B_1 ตามหลัง J_2

หาจุดต่ำสุด (Minimum point) ของฟังก์ชั่นค่าใช้จ่าย (Cost Function) โดยการเปรียบเทียบความชัน (slope) เพื่อเลื่อนฟังก์ชั่นจะได้ผลตามภาพที่ 3(D) โดยจุดต่ำสุด (Minimum point) ของฟังก์ชั่นค่าใช้จ่าย (Cost Function) อยู่ที่จุดเดียวกันกับภาพที่ 3(C) ดังนั้น เวลาเริ่มงานของงาน J_1คือ $t_1 = 1$, เวลาเริ่มงานของงาน J_2 คือ $t_2 = 4$ และเวลาเริ่มงานของงาน J_3 คือ $t_3 = 9$

พิจารณา J_4, $C_4 = C_3 + p_4 = 16 < d_4$ ดังนั้น J_4 จัดเข้าบล็อคใหม่ B_2 เนื่องจาก J_4 เป็นงานแรกของบล็อค ดังนั้น เวลาเริ่มงานของงาน J_4คือ $t_4 = 15$

จากตัวอย่างข้างบน แสดงให้เห็นว่าสามารถหาเวลาเริ่มงานของแต่ละงานได้ โดยพิจารณาเฉพาะความชันของฟังก์ชั่นค่าใช้จ่ายเพื่อหาจุดต่ำสุดได้ โดยที่ไม่ต้องคำนวณหาค่าของฟังก์ชั่นวัตถุประสงค์เลย



ภาพที่ 4 ฟังก์ชั่นค่าใช้จ่ายของบล็อค

ที่มา: Lee and Choi (1995)

ทฤษฎีวิธีเชิงพันธุกรรม

Pinedo (2002) กล่าวไว้ว่าขั้นตอนวิธีการเชิงพันธุกรรมเป็นการค้นหาแบบเฉพาะที่แบบหนึ่ง ซึ่งมีรูปแบบทั่วไปมากกว่าการค้นหาแบบจำลองการอ่อนตัว (Simulated annealing: SA) หรือการค้นหาแบบข้อห้าม (Tabu search) ในบางครั้งเราอาจกล่าวได้ว่า การค้นหาแบบจำลองการอ่อนตัว และการค้นหาแบบข้อห้าม เป็นกรณีพิเศษของการค้นหาโดยใช้ขั้นตอนวิธีเชิงพันธุกรรม

Man *et al*. (1996); Obitko (1998) ได้สรุปหลักการและพื้นฐานของวิธีเชิงพันธุกรรมซึ่งถูกพัฒนาขึ้นมาโดย Holland ว่าเป็นแนวคิดในการคำนวณโดยใช้หลักการทางชีววิทยาของทฤษฎีวิวัฒนาการว่า สิ่งมีชีวิตทุกชนิดประกอบด้วยเซลล์ ซึ่งแต่ละเซลล์จะมีกลุ่มของโครโมโซม (Chromosomes) ที่เหมือนกัน โครโมโซมประกอบด้วยยีนส์ (Genes) ซึ่งจะสะท้อนลักษณะเฉพาะของสิ่งมีชีวิตนั้น เช่น สีตา สีผม โดยยีนส์แต่ละตัวจะมีตำแหน่งของตนเองที่เรียกว่า Locus

เมื่อนำโครโมโซมทั้งหมดมาประกอบกันจะเรียกว่า Genome และเรียกกลุ่มของยีนส์ใน Genome นี้ว่าGenotype หรือรหัสพันธุกรรม ซึ่ง Genotypes นี้จะแปลงไปเป็นอวัยวะของสิ่งมีชีวิตต่อไปเรียกว่า Phenotype ตามทฤษฎีวิวัฒนาการสิ่งมีชีวิตที่แข็งแรงที่สุดจะมีโอกาสสืบทอดสายพันธุ์ที่แข็งแรงต่อไป การไขว้สายพันธุ์(Crossover) ของพ่อและแม่จะทำให้เกิดลูกซึ่งคัดลอกยีนส์ของพ่อแม่ที่ผสมกัน ทำให้เกิดสายพันธุ์ที่แข็งแรงยิ่งขึ้น บางครั้งการคัดลอกยีนส์ของพ่อแม่ไม่สมบูรณ์ อาจทำให้เกิดการผ่าเหล่า(Mutation) ในรุ่นลูก การผ่าเหล่าทำให้สิ่งมีชีวิตมีโอกาสพัฒนาสายพันธุ์ใหม่ที่ยิ่งเข้มแข็งขึ้น หากการผ่าเหล่าทำให้เกิดสายพันธุ์ด้อย สายพันธุ์ด้อยจะไม่สืบทอดต่อ

กระบวนการคำนวณเชิงพันธุกรรมมีขั้นตอนที่เกี่ยวข้องกับกระบวนการที่สำคัญ 3 ขั้นตอนหลักได้แก่ หนึ่ง สุ่มคำตอบที่เป็นไปได้ สอง สร้างประชากร (Population) รุ่นใหม่และประเมินคุณภาพของคำตอบด้วยฟังก์ชันการประเมินความเหมาะสม (Fitness) กระบวนการนี้จะวนรอบตามจำนวนรุ่นประชากร และ สาม แปลงคำตอบกลับมาอยู่ในรูปค่าจริงที่ตีความได้

ในขั้นตอนที่หนึ่ง ระบบจะแปลง (Encode) ผลลัพธ์ที่เป็นไปได้ของปัญหาคณิตศาสตร์ให้อยู่ในรูปโครโมโซม ตัวแปรที่ต้องการค้นหาหมายถึงยีนส์หนึ่งยีนส์เมื่อนำยีนส์มาประกอบกันจึงได้โครโมโซมหรือGenotype การแปลงค่าทำได้หลายวิธีเช่นวิธี Binary, Permutation, หรือ Value Encoding ตารางที่ 2 แสดงตัวอย่างการแปลงในรูปแบบต่างๆ การเลือกใช้การแปลงแบบใดขึ้นอยู่กับลักษณะของปัญหา เช่น การแปลงแบบ Binary เหมาะสำหรับการคำนวณหาค่าสูงสุดหรือต่ำสุดตามเงื่อนไข การแปลงแบบ Permutation เหมาะกับปัญหาการจัดอันดับเช่นการวางแผนเดินทาง ตัวเลขที่แสดงหมายถึงลำดับของจุดที่เดินทาง ในขณะที่วิธี Value มีการแปลงได้หลายแบบ เช่นเป็นค่าจริง (Real Number) ตัวอักษร หรือกลุ่มวัตถุ ก็ได้ การแปลงชนิดนี้เหมาะกับปัญหาลักษณะพิเศษต่างๆ เช่นการแปลงแบบค่าจริงอาจใช้ในการค้นหาน้ำหนักที่เหมาะสมสำหรับการคำนวณฟังก์ชันหนึ่งได้ เมื่อกำหนดวิธีการแปลงแล้ว ระบบจะสุ่มสร้างกลุ่มโครโมโซมนี้ขึ้นมาเป็นประชากร จำนวนโครโมโซมในประชากรแต่ละรุ่นขึ้นกับตามค่าพารามิเตอร์ที่กำหนด ความยาวของโครโมโซมขึ้นอยู่กับจำนวนตัวแปรที่ต้องการหาค่าในระบบสมการ และมีผลต่อการกำหนดพารามิเตอร์จำนวนขนาดประชากร เพื่อหาคำตอบที่ดีที่สุดด้วย

ตารางที่ 1 ตัวอย่างการ Encode โครโมโซมแบบต่างๆ

วิธีการแปลง (Encoding)	ตัวอย่างโครโมโซม
Binary	10110010110010101011100101
Permutation	1 5 3 2 6 4 7 9 8
Value	1.2324 5.3243 0.4556 2.3293 2.4545
	ABDJEIFJDHDIERJFDLDFLFEGT
	(back), (back), (right), (forward), (left)

ที่มา: Obitko (1998)

ประชากรเริ่มต้นจะถูกสุ่มสร้างขึ้นตามขั้นตอนดังภาพที่ 5 โดยหลังจากสร้างประชากรเริ่มต้นเรียบร้อยแล้ว จะทำการสุ่มคัดเลือกโครโมโซมพ่อแม่ (Parent) เพื่อนำไปสู่ กระบวนการสร้างประชากรรุ่นใหม่จะเกิดขึ้นตามลำดับต่อไปนี้

```
procedure: initialization
begin
        i ← 0;
        while (i < pop_size) do
        begin
            generate a job permutation list randomly;
            put m – 1 starts into the job list randomly;
            i ← i+1;
        end
end
```

ภาพที่ 5 ขั้นตอนการสร้างประชากรเริ่มต้น
ที่มา: Cheng *et al*. (1995)

1. การไขว้สายพันธุ์ (Crossover) จากโครโมโซมพ่อแม่ (Parent) จากตัวอย่างในตารางที่ 2 โครโมโซม 1 และ 2 เป็นพ่อและแม่ ระบบจะกำหนดจุดแบ่ง (Crossover Point) ซึ่งแสดงด้วยสัญลักษณ์ | เพื่อแบ่งโครโมโซมเป็นสองส่วน จากนั้นโครโมโซมลูกจะถูกสร้างขึ้นใหม่จากการไขว้สายพันธุ์พ่อและแม่ การไขว้สายพันธุ์เกิดจากการกำหนดพารามิเตอร์ ความน่าจะเป็นในการไขว้สายพันธุ์ (P_c) ซึ่งมีค่าตั้งแต่ 0-100% โดย 0 หมายถึงการนำคัดลอกยีนส์พ่อและแม่มาทั้งส่วน ในขณะที่ 100% คือการสร้างโครโมโซมของลูกขึ้นมาใหม่เลย

ตารางที่ 2 ตัวอย่างการสร้างโครโมโซมลูกจากการไขว้สายพันธุ์ พ่อและแม่

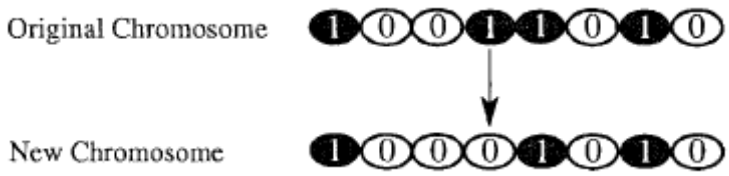
การสร้างโครโมโซม	ตัวอย่างโครโมโซม
Parent 1 (พ่อ)	**11011** | 00100110110
Parent 2 (แม่)	11001 | **11000011110**
Offspring 1 (ลูก)	**11011** | **11000011110**
Offspring 2 (ลูก)	11001 | 00100110110

ที่มา: Obitko (1998)

นอกจากนี้ยังสามารถกำหนดจุดแบ่ง(Crossover Point) ในการไขว้สายพันธุ์ ให้มีความซับซ้อนขึ้นอีกได้ ซึ่งจะทำให้เกิดผลลัพธ์ที่เป็นไปได้หลากหลายขึ้น เช่น Two Points Crossover ทำให้เกิดการไขว้สายพันธุ์ เป็นโครโมโซมลูกหรืออาจแบ่งเป็นแบบ multipoint

โดยวิธีการไขว้สายพันธุ์สำหรับปัญหาการจัดอันดับนั้นมีหลายแบบ เช่น uniform order-based crossover, partially matched crossover, cycle crossover เป็นต้น ซึ่ง Lee and Choi (1995) ได้กล่าวไว้ว่า วิธี uniform order-based crossover เป็นวิธีที่เหมาะสมที่สุดสำหรับปัญหาการจัดลำดับ

2. การผ่าเหล่า (Mutation) เมื่อเกิดการไขว้สายพันธุ์ จากพ่อและแม่เพื่อถ่ายไปสู่ลูก จะทำให้โครโมโซมของรุ่นลูกกระจุกอยู่ในกลุ่มเดียวกับพ่อแม่ ในเชิงคณิตศาสตร์ผลลัพธ์ที่ได้อาจเป็นค่าสูงสุดหรือต่ำสุดเฉพาะในขอบเขตจำกัดหนึ่งเท่านั้น หรือเรียกว่าค่าท้องถิ่น (Local Optimal Solution) มิใช่ค่าสูงสุดหรือต่ำสุดที่แท้จริง (Global Optimal Solution) เพื่อป้องกันปัญหานี้ จึงควรกำหนดความน่าจะเป็นในการผ่าเหล่า (P_m) ให้ยีนส์ของลูกมีการผ่าเหล่าได้ เช่นในกรณี Binary Encoding หากเราได้โครโมโซมลูกมาเป็น 110111100 แทนที่จะใช้รวมเป็นหนึ่งโครโมโซมของประชากรรุ่นลูก เราจะปรับยีนส์บางตำแหน่ง (Locus) ให้เป็นโครโมโซมใหม่เช่น 110011100 แล้วใช้ค่านี้แทน การทำเช่นนี้เพื่อทำให้การคำนวณเชิงพันธุกรรมมีโอกาสได้ค่าผลลัพธ์ที่ไม่ใช่ค่าท้องถิ่น ค่าความน่าจะเป็นในการผ่าเหล่า (P_m) กำหนดได้ตั้งแต่ 0-100% โดย 0 หมายถึงไม่มีการผ่าเหล่าเลย และ 100% หมายถึงมีการผ่าเหล่ายีนส์ทุกตัวในโครโมโซมลูก



ภาพที่ 6 การผ่าเหล่าในตำแหน่งที่ 4

3. ประเมินความเหมาะสมและคัดเลือก (Evaluation and selection) การประเมินความเหมาะสมของโครโมโซมในประชากรแต่ละชุด โดยพิจารณาจากฟังก์ชั่นวัตถุประสงค์ จากนั้นจะทำการคัดเลือกโครโมโซมที่เหมาะสมที่สุด 2 ชุดให้เป็นพ่อและแม่เพื่อใช้สืบทอดสายพันธุ์

(Reproduction) ให้แก่ประชากรรุ่นใหม่ อีกนัยหนึ่งประชากรรุ่นใหม่ที่จะสร้างขึ้นคือลูกที่เกิดจากการไขว้สายพันธุ์ ที่ดีที่สุดของประชากรรุ่นก่อนนั่นเอง

การคัดเลือกนี้สามารถทำได้หลายวิธี เช่น Roulette Wheel, Boltzman, Tournament, Rank หรือSteady State Selection หลักการกว้างๆ ของการคัดเลือกนี้คือโครโมโซมที่เหมาะสมที่สุดมีโอกาสที่จะได้รับการคัดเลือกมากกว่าโครโมโซมที่ด้อยกว่า ยกตัวอย่างเช่นวิธี Roulette Wheel เปรียบเสมือนการนำโครโมโซมทุกตัวของประชากรรุ่นนั้นมาเขียนเป็นช่องเหมือนกับวงล้อ Roulette ความกว้างในแต่ละช่องจะไม่เท่ากัน แต่เป็นสัดส่วนกับความเหมาะสมของโครโมโซมนั้น แล้วหมุนวงล้อ หากตกช่องไหน จะเลือกโครโมโซมนั้น เนื่องจากโครโมโซมที่เหมาะสมกว่าจะมีช่องขนาดใหญ่กว่า ดังนั้นโอกาสถูกเลือกจึงมีมากกว่า

4. Elitism บางครั้งการสร้างประชากรลูกโดยผสมยีนส์จากโครโมโซมพ่อและแม่ซึ่งเป็นผลลัพธ์ที่ดีที่สุดจากประชากรรุ่นก่อน อาจทำให้สูญเสียผลลัพธ์ที่ดีอยู่แล้วไปได้ การกำหนด Elitism จึงเป็นการให้คงโครโมโซมพ่อและแม่ให้เป็นส่วนหนึ่งของประชากรรุ่นลูก เพื่อให้มีโอกาสสร้างสายพันธุ์ใหม่ในประชากรรุ่นถัดไปด้วย

5. เงื่อนไขในการหยุดกระบวนการหาคำตอบ (Termination Criterion) กระบวนการสร้างประชากรรุ่นใหม่จะถูกทำซ้ำจนได้จำนวนโครโมโซมหรือผลลัพธ์เท่ากับที่กำหนดในพารามิเตอร์ Population การคำนวณจะย้อนกลับไปสู่การประเมินผลลัพธ์ของประชากรรุ่นใหม่ด้วยฟังก์ชันคณิตศาสตร์ เพื่อคัดเลือกโครโมโซมที่ดีที่สุดมาสร้างประชากรรุ่นใหม่ต่อไป การคำนวณจะวนเป็นรอบจนกระทั่งได้ค่าคำตอบที่ดีที่สุด หรือครบจำนวนรุ่นของประชากร หรือได้คำตอบที่มีค่าใกล้เคียงกับค่าที่ต้องการที่กำหนดในพารามิเตอร์ Generation

งานวิจัยที่เกี่ยวข้อง

Lee and Choi (1995) ใช้วิธีเชิงพันธุกรรมในการแก้ไขปัญหาเครื่องจักรเดียว ที่มีวันกำหนดส่งมอบที่แตกต่างกัน โดยมีฟังก์ชั่นวัตถุประสงค์คือ ผลรวมของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและหลังวันกำหนดส่งมอบมีค่าน้อยที่สุด โดยใช้การไขว้สายพันธุ์ แบบ uniform crossover ซึ่งผลที่ได้สามารถลดค่าใช้จ่ายได้ลง 12 – 33 % เมื่อเทียบกับวิธีฮิวลิสติกส์โดยขึ้นอยู่กับขนาดของปัญหา และสามารถหาค่า optimal solution ได้กับปัญหาที่มีงาน 15 – 20 งาน

Cheng *et al*. (1995) ใช้วิธีเชิงพันธุกรรมในการแก้ไขปัญหาเครื่องจักรขนานที่เหมือนกัน โดยมีฟังก์ชั่นวัตถุประสงค์คือหาค่าที่น้อยที่สุดของค่าสัมบูรณ์ของเวลาสายสูงสุดที่ถูกถ่วงน้ำหนัก (Mimimize the maximum weighted absolute lateness) โดยทดลองที่ 30 งาน 5 เครื่องจักร ซึ่งให้ผลลัพท์ที่ดีกว่าเมื่อเทียบกับวิธีฮิวลิสติกส์

Min and Cheng (1999) ใช้วิธีเชิงพันธุกรรมในการแก้ไขปัญหาเครื่องจักรขนานที่เหมือนกัน โดยมีฟังก์ชั่นวัตถุประสงค์คือหาค่าที่น้อยที่สุดของเวลาปิดงานของระบบ (minimize makespan) โดยเปรียบเทียบกับวิธีจัดลำดับแบบเวลาปฏิบัติงานมากที่สุด (LPT) และการค้นหาแบบจำลองการอ่อนตัว (SA) โดยทดลองกับปัญหาขนาดใหญ่สุดที่ 30 งาน 10 เครื่องจักร ผลที่ได้คือ วิธีเชิงพันธุกรรมให้ผลลัพท์ที่ดีกว่าทั้งสองวิธีที่นำมาเปรียบเทียบ

Guo (2005) ใช้วิธี parallel hybrid genetic algorithm สำหรับแก้ไขปัญหาเครื่องจักรขนานที่ไม่เหมือนกัน (Non-Identical Parallel Machine) โดยมีฟังก์ชั่นวัตถุประสงค์คือหาค่าที่น้อยที่สุดของเวลาปิดงานของระบบ (minimize makespan) โดยทดลองกับปัญหาขนาดใหญ่สุดที่ 100 งาน 15 เครื่องจักรเปรียบเทียบกับวิธีฮิวลิสติกส์ ซึ่งให้ผลลัพท์ที่ดีกว่าเมื่อเทียบกับวิธีฮิวลิสติกส์

Huang *et al*. (2005) ใช้วิธี hybrid genetic algorithm สำหรับแก้ไขปัญหาเครื่องจักรขนานที่ไม่เหมือนกัน โดยมีฟังก์ชั่นวัตถุประสงค์คือหาค่าที่น้อยที่สุดของช่วงเวลาที่สายและเวลาปิดงานของระบบ (minimizing the range of lateness and makespan) โดยทดลองกับปัญหาขนาดใหญ่สุดที่ 60 งาน 9 เครื่องจักรเปรียบเทียบกับวิธีฮิวลิสติกส์ ซึ่งให้ผลลัพท์ที่ดีกว่าเมื่อเทียบกับวิธีฮิวลิสติกส์

อุปกรณ์และวิธีการ

อุปกรณ์

วัสดุและอุปกรณ์ที่ใช้ในการวิจัยมีดังนี้

1. คอมพิวเตอร์ AMD Athlon XP-M 2800++, 2.13 GHz, Ram 512 MB

2. โปรแกรม Dev-C++ Version 4.9.9.2

3. โปรแกรม MINITAB Release 14.1

วิธีการ

วิธีการในการดำเนินการวิจัยนี้เพื่อให้บรรลุตามวัตถุประสงค์มีขั้นตอนดังนี้

1. กำหนดปัญหา / เก็บรวมรวมและศึกษาข้อมูลอ้างอิง

1.1. งานแต่ละงานปฏิบัติเสร็จขั้นตอนเดียว (Single operation) ใน 1 เครื่องจักร

1.2. ไม่อนุญาติให้มีการแทรกงานอื่นใดในระหว่างการผลิต

1.3. ไม่มีการยกเลิกหรือหยุดพักเมื่องานกำลังถูกกระทำอยู่ งานแต่ละงานต้องถูกทำต่อเนื่องจนเสร็จ

1.4. เวลาการปฏิบัติงานไม่ขึ้นอยู่กับลำดับงาน

1.5. งานสามารถรอจนเครื่องจักรว่างได้

1.6. อนุญาติให้มีเวลาว่างบนเครื่องจักรได้

1.7. เครื่องจักรแต่ละเครื่องไม่สามารถปฏิบัติงานได้มากกว่าหนึ่งงานในเวลาเดียวกัน

1.8. ไม่พิจารณากรณีเครื่องจักรเสีย

1.9. งานทุกงานพร้อมที่จะเริ่มงานทันที

1.10. ไม่มีการสุ่ม รู้จำนวนงานที่แน่นอน n งาน, รู้จำนวนเครื่องจักรที่แน่นอน m เครื่องจักร, รู้เวลาการปฏิบัติงานที่แน่นอน, รู้เวลากำหนดส่งมอบที่แน่นอน

1.11. ความเร็วของเครื่องจักรแต่ละเครื่องเป็นจำนวนเท่าของจำนวนเต็ม

2. วิเคราะห์ปัญหา

ปัญหาการจัดตารางการผลิตที่มีเครื่องจักรแบบขนานที่ไม่เหมือนกัน *m* เครื่องจักร มีงานที่ต้องจัดตาราง *n* งาน ซึ่งแต่ละงานมีการดำเนินงานเพียงขั้นตอนเดียวเท่านั้น จะประกอบด้วยการตัดสินใจหลักอยู่ 2 ส่วนด้วยกัน โดยส่วนแรกเป็นการจัดแบ่งงานและลำดับงานในแต่ละเครื่องจักร ส่วนที่สองเป็นการหาเวลาเริ่มงานของแต่ละงาน ทำให้การจัดตารางการผลิตที่ดีที่สุดเป็นสิ่งที่ยากยิ่งขึ้นกว่ากรณีของเครื่องจักรเดียว โดย Gary *et al*. (1988) ได้พิสูจน์ว่าปัญหาเครื่องจักรเดียวและมีวันส่งมอบที่แตกต่างกันโดยมีฟังก์ชั่นวัตถุประสงค์คือ ผลรวมที่มีค่าน้อยที่สุดของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและหลังวันกำหนดส่งมอบมีอัตราเท่ากันเป็นปัญหาเอ็นพีบริบูรณ์ (NP-complete) และ Mandel and Mosheiov (1999)ได้พิสูจน์ว่าปัญหาเครื่องจักรขนานที่เหมือนกัน 2 เครื่องจักร โดยมีฟังก์ชั่นวัตถุประสงค์คือ ผลรวมที่มีค่าน้อยที่สุดของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนวันกำหนดส่งมอบ เป็นปัญหาเอ็นพีแบบยาก (NP-hard) ดังนั้นปัญหาเครื่องจักรขนานที่ไม่เหมือนกันและมีวันส่งมอบที่แตกต่างกันโดยมีฟังก์ชั่นวัตถุประสงค์คือ ผลรวมของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและหลังวันกำหนดส่งมอบมีค่าน้อยที่สุดจึงเป็นปัญหาเอ็นพีเช่นกัน ซึ่งปัญหาเอ็นพีนั้น เป็นปัญหาที่ไม่สามารถจะหาคำตอบได้ในเวลาเชิงพหุนาม (nondeterministic polynomial time) และยังไม่มี algorithm ใดที่สามารถนำมาแก้ไขปัญหาได้ จึงได้นำ metaheuristic มาใช้แก้ไขปัญหา

ให้ *II* เป็นชุดของตารางการผลิตที่เป็นไปได้ (set of feasible schedule) สำหรับตารางการผลิต $S \in II$ ให้ c_j เป็นเวลาเสร็จงานของงาน *J* ตามตารางการผลิต *S* และ *f(S)* เป็นค่าของฟังก์ชั่นวัตถุประสงค์ โดยฟังก์ชั่นวัตถุประสงค์คือ ผลรวมของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและหลังวันกำหนดส่งมอบมีค่าน้อยที่สุด สามารถแสดงในรูปแบบสมการเชิงคณิตศาสตร์ได้ตามสมการที่ (4) โดยสมการที่ (5) เป็นค่าของฟังก์ชั่นวัตถุประสงค์

$$Z = \min_{S \in II} f(S) \qquad (4)$$

$$f(S) = \sum_{i=1}^{n} (\alpha_i E_i + \beta_i T_i) \qquad (5)$$

3. วิธีการแก้ปัญหา / โปรแกรม

โดยใช้ภาษาซีในการเขียนโปรแกรม และใช้โปรแกรม Dev-C++ Version 4.9.9.2 เป็นตัวคอมไพล์และทดสอบโปรแกรม ซึ่งในงานวิจัยนี้จะเรียกโปรแกรมนี้ว่าวิธี GAOPT เป็นการนำวิธีเชิงพันธุกรรมมาประยุกต์ใช้ร่วมกับการจัดเวลาเริ่มงาน (Optimal Timing Algorithm) เพื่อแก้ไขปัญหาการจัดตารางการผลิตนั้น โดยวิธีเชิงพันธุกรรมจะใช้ในการจัดแบ่งงานและลำดับงานในแต่ละเครื่องจักร หลังจากที่ทราบจำนวนงานและลำดับงานที่แน่นอนในแต่ละเครื่องจักรแล้ว จึงใช้วิธีการจัดเวลาเริ่มงานของ Lee and Choi (1995) มาช่วยในการจัดหาเวลาเริ่มงานตามตัวอย่างที่ 1 ดังได้กล่าวมาก่อนหน้านี้

การนำวิธีเชิงพันธุกรรมมาประยุกต์ใช้ร่วมกับการจัดเวลาเริ่มงาน จึงสามารถสรุปเป็นขั้นตอนได้ดังนี้

3.1. กำหนดค่าพารามิเตอร์เริ่มต้น เช่น จำนวนประชากร, ความน่าจะเป็นในการไขว้สายพันธุ์ (P_c), ความน่าจะเป็นในการผ่าเหล่า (P_m)

3.2. ทำการสุ่มสร้างประชากรเริ่มต้น

3.3. สร้างประชากรจากการไขว้สายพันธุ์โดยใช้ uniform order-based crossover

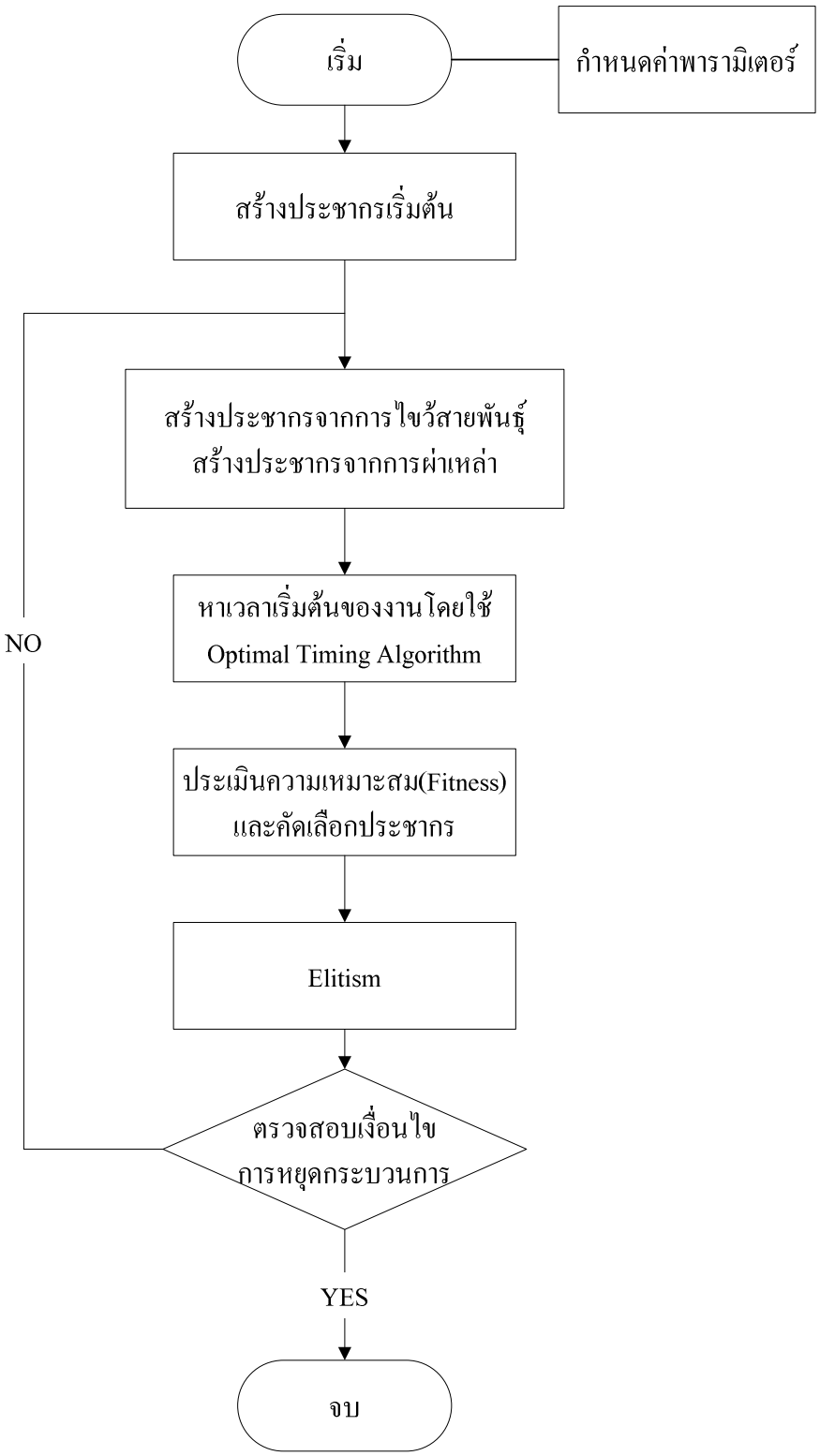
3.4. สร้างประชากรจากการผ่าเหล่า

3.5. หาเวลาเริ่มต้นของงานโดยใช้ Optimal Timing Algorithm

3.6. ประเมินค่าความเหมาะสมและทำการคัดเลือกประชากรรุ่นถัดไป

3.7. ใช้ elitism เพื่อรักษาโครโมโซมที่ดีที่สุด 1 ชุดให้เป็นประชากรในรุ่นถัดไป

3.8. หยุดเมื่อเข้าเงื่อนไขในการหยุดกระบวนการหาคำตอบ



ภาพที่ 7 ขั้นตอนการนำวิธีเชิงพันธุกรรมมาประยุกต์ใช้ร่วมกับการจัดเวลาเริ่มงาน

องค์ประกอบที่สำคัญของวิธีเชิงพันธุกรรมที่ใช้ในงานวิจัยนี้มี 8 ส่วนดังนี้

1. การแปลงค่าและการนำเสนอ (Encoding and Representation) ในงานวิจัยนี้เป็นปัญหาเกี่ยวกับการจัดอันดับ จึงเลือกใช้การแปลงแบบ Permutation ตามวิธีของ Cheng *et al*. (1995) โดยใช้สัญลักษณ์ของงานแทนด้วยตัวเลขซึ่งเป็นอันดับของงานและใช้สัญลักษณ์ * ในการแบ่งงานสำหรับแต่ละเครื่องจักร เช่น มีงาน 9 งาน และเครื่องจักร 3 เครื่อง เครื่องจักรที่ 1 ทำงานที่ 5 งานที่ 8 และงานที่ 1 เครื่องจักรที่ 2 ทำงานที่ 3 งานที่ 7 งานที่ 4 และงานที่ 6 เครื่องจักรที่ 3 ทำงานที่ 2 และงานที่ 9 โครโมโซมที่ได้จะมีลักษณะดังนี้

[5 8 1 * 3 7 4 6 * 2 9]

โดยทั่วไป สำหรับสำหรับปัญหางาน n งานและเครื่องจักร m เครื่อง โครโมโซมจะประกอบไปด้วยสัญลักษณ์ของงาน n งานและ $m - 1$ สำหรับสัญลักษณ์ของการแบ่งเครื่องจักร โครโมโซมจึงมีจำนวนยีนส์เป็น $(n + m - 1)$

2. การสร้างประชากรเริ่มต้น (Initial Population) เป็นการกระทำอันดับแรกก่อนที่จะเข้าสู่กระบวนการของวิธีเชิงพันธุกรรม ประชากรเริ่มต้นเกิดจากการสุ่ม (Random) เพื่อนำประชากรเข้าไปในกระบวนการ ในการสุ่มจะต้องสุ่มให้ได้จำนวนเท่ากับขนาดของรุ่นที่ได้กำหนดไว้ โดยที่ยังไม่มีการสนใจค่าความเหมาะสมของแต่ละโครโมโซม

3. การไขว้สายพันธุ์ (Crossover) ใช้วิธีเดียวกันกับของ Lee and Choi (1995) โดยใช้ uniform order-based crossover ซึ่งโครโมโซม 1 คู่จะได้ offspring 2 ตัว โดยแสดงขั้นตอนของการใช้งานตามตัวอย่างที่ 2

ตัวอย่างที่ 2 พิจารณา โครโมโซมพ่อแม่และมี bit string ซึ่งมีขนาดเดียวกันกับโครโมโซมพ่อแม่

Parent 1	1	2	3	*	4	5	6
Parent 2	6	3	*	5	2	4	1

Binary Template	0	1	1	1	0	1	0

uniform order-based crossover จะแทนที่บางตำแหน่งของ Offspring 1 (หรือ Offspring 2) โดยการคัดลอกจาก Parent 1 (หรือ Parent 2) เมื่อค่าของ bit string เป็น “1” (หรือ “0”) ดังนี้

Offspring 1	_	2	3	*	_	5	_
Offspring 2	6	_	_	_	2	_	1

ในตำแหน่งที่เหลือของ Offspring 1 (หรือ Offspring 2) จะคัดลอกโดยเรียงตามลำดับตามที่ปรากฏจาก Parent 2 (หรือ Parent 1) จึงได้ Offspring 1(หรือ Offspring 2) ดังนี้

Offspring 1	6	2	3	*	4	5	1
Offspring 2	6	3	*	4	2	5	1

4. การผ่าเหล่า (Mutation) ใช้วิธีการสลับตำแหน่งแบบสุ่ม (random exchange) โดยจะทำการสุ่มตำแหน่งที่ต้องสลับกัน 2 ตำแหน่งแล้วทำการสลับกัน ตามตัวอย่างที่ 3

ตัวอย่างที่ 3 ทำการสุ่มได้ตำแหน่งที่ 2 และ 5 จึงได้ Offspring 1 ดังนี้

Parent 1	1	2	3	*	4	5	6
Offspring 1	1	4	3	*	2	5	6

5. การประเมินความเหมาะสมของประชากร (Fitness Function) เนื่องจากในแต่ละรุ่นของวิธีเชิงพันธุกรรมโครโมโซมจะถูกประเมินความเหมาะสม ซึ่งความเหมาะสมหาได้จากฟังก์ชั่นวัตถุประสงค์และในงานวิจัยนี้มีฟังก์ชันวัตถุประสงค์คือผลรวมของค่าใช้จ่ายมีค่าน้อยที่สุด ซึ่ง

หมายความว่าโครโมโซมที่มีค่าใช้จ่ายน้อยที่สุดต้องมีความเหมาะสมมากที่สุด ดังนั้นความเหมาะสมของโครโมโซมแต่ละตัวจึงหาได้จากส่วนกลับของค่าใช้จ่ายของแต่ละโครโมโซม

$$eval(v_t) = \frac{1}{f(v_t)} \qquad (6)$$

โดย $eval(v_t)$ เป็นฟังก์ชั่นการประเมินความเหมาะสมของโครโมโซมที่ t และ $f(v_t)$ เป็นค่าของฟังก์ชั่นวัตถุประสงค์ตามสมการที่ (5)

6. การคัดเลือกโครโมโซม (Selection) โดยใช้วิธี Roulette Wheel เปรียบเสมือนการนำโครโมโซมทุกตัวของประชากรรุ่นนั้นมาเขียนเป็นช่องเหมือนกับวงล้อ Roulette ความกว้างในแต่ละช่องจะไม่เท่ากัน โดยเป็นสัดส่วนกับความเหมาะสมของโครโมโซมนั้น และมีการหมุนวงล้อเท่ากับจำนวนประชากรที่กำหนดไว้ในแต่ละรุ่น เมื่อวงล้อหมุนตกช่องไหน จะเลือกโครโมโซมนั้น เนื่องจากโครโมโซมที่เหมาะสมกว่าจะมีช่องขนาดใหญ่กว่า ดังนั้นโอกาสถูกเลือกจึงมีมากกว่า โดยแสดงตามตัวอย่างที่ 4

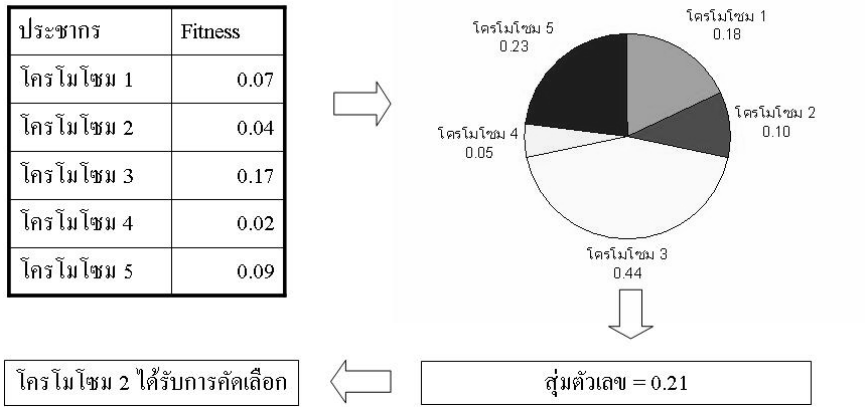
ตัวอย่างที่ 4 คัดเลือกโครโมโซมโดยใช้วิธี Roulette Wheel จากโครโมโซม 5 ชุด

โครโมโซม	:	1	2	3	4	5
ค่าใช้จ่าย	:	15	27	6	52	11

ความเหมาะสมของโครโมโซมแต่ละตัวหาได้ตามสมการที่ (6)

ความเหมาะสม	:	0.07	0.04	0.17	0.02	0.09
สัดส่วนกับความเหมาะสม	:	0.18	0.10	0.44	0.05	0.23
ความเหมาะสมสะสม	:	0.18	0.28	0.72	0.77	1.00

ทำการสุ่มตัวเลขจาก 0.00 ถึง 1.00 ได้ 0.21 ดังนั้นโครโมโซมที่ 2 ได้รับการคัดเลือก



ภาพที่ 8 การคัดเลือกโครโมโซม

7. Elitism เนื่องจากในการคัดเลือกบางครั้งโครโมโซมที่ดีที่สุดอาจไม่ได้รับการคัดเลือกให้เป็นประชากรในรุ่นถัดไป elitism จึงเป็นการคัดเลือกโครโมโซมที่ดีที่สุดไว้เพื่อรักษาให้คงอยู่ โดยทำการสุ่มแทนที่โครโมโซมที่ได้รับการคัดเลือกไว้ก่อนหน้า

8. เงื่อนไขในการหยุดกระบวนการหาคำตอบ (Stopping Criteria) วิธีเชิงพันธุกรรมจะทำการวนหาคำตอบต่อไปจนกระทั่งเกิดกรณีใดกรณีหนึ่งใน 3 กรณีดังนี้

8.1. ค่าใช้จ่ายเป็นศูนย์

8.2. ค่าใช้จ่ายที่ดีที่สุดของรุ่นปัจจุบันเปรียบเทียบกับค่าใช้จ่ายที่ดีที่สุดของรุ่นก่อนหน้านี้ 300 รุ่น มีเปอร์เซ็นต์ความแตกต่างค่าน้อยกว่า 0.01 เช่น รุ่นที่ 301 เปรียบเทียบกับรุ่นที่ 1 หรือรุ่นที่ 302 เปรียบเทียบกับรุ่นที่ 2 เป็นต้น

8.3. จำนวนรุ่นของประชากรสูงสุดมากกว่า 10,000 รุ่น

การปรับแต่ง โปรแกรม

ทำการปรับแต่งโปรแกรมจากการสร้างและพัฒนาโปรแกรมสำหรับการจัดตารางการผลิตแล้วนั้น เพื่อทำการทดสอบค่า P_c และ P_m ที่เหมาะสมกับโปรแกรม โดยปัญหาที่ถูกจำลองขึ้นนั้นมี 30 ปัญหา แบ่งออกเป็น 6 แบบ แบบที่ 1 $P_c = 0.6$ และ $P_m = 0.1$, แบบที่ 2 $P_c = 0.6$ และ $P_m = 0.2$, แบบที่ 3 $P_c = 0.7$ และ $P_m = 0.1$, แบบที่ 4 $P_c = 0.7$ และ $P_m = 0.2$, แบบที่ 5 $P_c = 0.8$ และ $P_m = 0.1$

และแบบที่ 6 $P_c = 0.8$ และ $P_m = 0.2$ โดยทั้ง 7 แบบมีชุดปัญหา 30 ปัญหา, โดยมี $\beta = 0.5\alpha$ 10 ปัญหา, $\beta = \alpha$ 10 ปัญหา, $\beta = 2\alpha$ 10 ปัญหา, ทุกปัญหามี 30 งาน, เครื่องจักร 2 เครื่อง ประชากรขนาด 100 โครโมโซมโดยในแต่ละปัญหาจะทำการทดสอบ 3 ครั้งและเลือกค่าที่ดีที่สุดสำหรับแต่ละปัญหาซึ่งได้ผลการทดสอบตามตารางที่ 3

ตารางที่ 3 ค่าที่ดีที่สุดของผลการทดสอบการปรับแต่งโปรแกรม

ปัญหาที่	แบบที่ 1	แบบที่ 2	แบบที่ 3	แบบที่ 4	แบบที่ 5	แบบที่ 6
1	9.98	9.16	8.90	10.50	9.45	9.98
2	19.08	21.57	19.44	25.21	25.93	17.68
3	20.31	33.65	39.69	26.25	26.01	37.40
4	6.81	1.02	6.10	1.08	2.74	1.08
5	13.48	15.03	14.05	18.01	12.42	23.09
6	17.82	10.84	11.94	13.75	18.74	5.16
7	59.19	44.61	62.42	60.58	62.87	63.53
8	5.69	13.07	9.64	6.94	15.90	9.64
9	4.01	2.64	1.30	0.85	5.80	2.57
10	7.62	8.13	7.70	10.09	4.15	4.64
11	52.50	62.56	58.42	58.20	47.71	60.83
12	28.02	27.23	27.46	22.82	19.97	26.41
13	51.43	77.78	66.22	65.52	65.61	78.19
14	9.59	2.33	2.39	15.04	3.14	0.00
15	18.20	22.72	11.72	17.83	16.49	15.39
16	14.76	19.09	21.12	18.10	20.71	20.64
17	105.37	110.33	112.17	117.04	114.95	90.37
18	8.88	9.84	11.80	11.40	18.36	12.42
19	50.30	41.83	59.88	42.59	45.19	47.79
20	25.14	31.26	37.85	27.01	35.90	28.86
21	22.48	23.05	9.40	22.52	23.87	21.35
22	36.59	29.38	22.28	29.38	33.22	28.74

ตารางที่ 3 (ต่อ)

ปัญหาที่	แบบที่ 1	แบบที่ 2	แบบที่ 3	แบบที่ 4	แบบที่ 5	แบบที่ 6
23	79.05	96.35	124.15	100.43	69.16	88.57
24	217.17	203.37	255.53	226.27	128.21	240.91
25	74.53	34.17	44.6	51.19	68.3	61.29
26	336.53	187.24	337.13	209.61	338.16	202.5
27	19.39	24.36	27.17	16.42	20.71	25.03
28	179.18	215.45	206.97	191.58	225.85	216.19
29	138.59	207.96	326.58	283.49	244.52	210.68
30	40.12	54.71	61.31	86.3	32.04	81.01

จากผลการการทดสอบ พิจารณาถึงจำนวนครั้งที่แบบการทดลองแต่ละแบบสามารถให้ค่าที่ดีที่สุดตามตารางที่ 4 โดยพบว่าแบบการทดลองแบบที่ 1 สามารถหาค่าที่ดีที่สุดเป็นจำนวนครั้งที่มากที่สุด จึงเลือกใช้ค่า $P_c = 0.6$ และ $P_m = 0.1$ สำหรับวิธี GAOPT

ตารางที่ 4 จำนวนครั้งของการทดลองที่ให้ค่าที่ดีที่สุด

แบบที่	จำนวนครั้ง
1	8
2	5
3	3
4	2
5	7
6	5

วิธีฮิวลิสติกส์ที่นำมาทดสอบเปรียบเทียบ

1. วิธี EDDOPT โดยจะทำการจัดอันดับงานตามกำหนดวันส่งมอบแล้วทำการจัดสรรงานทีละงานให้กับเครื่องจักรทีละเครื่อง เมื่อครบทุกเครื่องแล้ว ให้จัดงานที่เหลือทีละงานโดยเริ่มที่เครื่องจักรเครื่องแรกอีกครั้งจนกว่าจะครบทุกงาน (ตามตัวอย่างที่ 3) แล้วหาเวลาเริ่มต้นของงานโดยใช้ Optimal Timing Algorithm

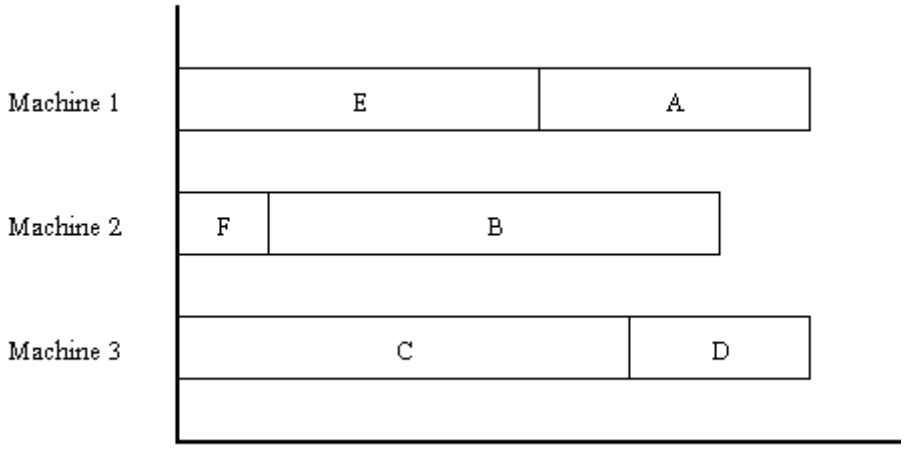
ตัวอย่างที่ 3 พิจารณาปัญหาการจัดสรรงาน 6 งาน ให้กับเครื่องจักร 2 เครื่อง

i	A	B	C	D	E	F
p_i	3	5	5	2	4	1
d_i	10	12	9	17	5	7

จัดเรียงลำดับงานใหม่ตามกำหนดวันส่งมอบ

i	E	F	C	A	B	D
p_i	4	1	5	3	5	2
d_i	5	7	9	10	12	17

สามารถจัดสรรงานให้แต่ละเครื่องจักรตามภาพที่ 9



ภาพที่ 9 การจัดสรรงานสำหรับวิธี EDDOPT

2. วิธี RNDOPT โดยการสุ่มการจัดอันดับงานให้กับเครื่องจักร โดยการสุ่มการจัดตารางการผลิตจำนวน 100 ตารางการผลิต หลังจากนั้นจึงใช้วิธี Optimal Timing Algorithm เพื่อหาเวลาเริ่มงาน ทำการคำนวณค่าใช้จ่ายที่เกิดขึ้น โดยวนซ้ำจำนวน 10,000 รอบ แต่ไม่ได้นำตัวดำเนินการทางพันธุกรรม (genetic operator) มาใช้ในการคำนวณในแต่ละรอบและเก็บคำตอบที่ดีที่สุดหรือคำตอบที่ให้ค่าใช้จ่ายต่ำที่ไว้ และใช้ปัญหาเดียวกับที่ใช้วิธี GAOPT

ผลและวิจารณ์

ผล

จากการสร้างโปรแกรมการจัดตารางการผลิตหรือวิธี GAOPT นั้นได้นำมาทดสอบเปรียบเทียบกับวิธีฮิวลิสติกส์อีก 2 วิธีคือวิธี EDDOPT และวิธี RNDOPT โดยในการเปรียบเทียบใช้จำนวนประชากรขนาด 100 โครโมโซมในแต่ละรุ่นของประชากร เวลาทำงานหรือเวลาปฏิบัติงานสุ่มสร้างโดยมีการแจกแจงแบบเอกรูปชนิดไม่ต่อเนื่อง (Discrete random uniform distribution) ในช่วง [1, 5] กำหนดส่งมอบของงาน i (d_i) สุ่มสร้างโดยมีการแจกแจงแบบเอกรูปชนิดไม่ต่อเนื่องในช่วงปิดค่าที่น้อยที่สุดของเวลาทำงานถึง 1.2 เท่าของผลรวมของเวลาทำงานต่อจำนวนเครื่องจักร [$\min p$, (1.2 x $\sum_{i=0}^{n} p_i$)/m] อัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จก่อนกำหนดส่งมอบงาน α_i สุ่มสร้างโดยมีการแจกแจงแบบเอกรูปชนิดต่อเนื่อง(Continuous random uniform distribution) ในช่วง (0, 10] อัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จหลังกำหนดส่งมอบงาน β_i เป็น $0.5\alpha_i$, α_i และ $2\alpha_i$ สำหรับแต่ละแบบ โดยแต่ละแบบมีชุดปัญหา 30 ปัญหา ปัญหาละ 20 งาน ทดสอบที่เครื่องจักร 2 เครื่อง, 5 เครื่องและ 10 เครื่อง อัตราการผลิตของเครื่องจักรสุ่มสร้างโดยมีการแจกแจงแบบเอกรูป [1, 2, 3] รวมทั้งสิ้นมีการทดสอบ 9 แบบ 270 ปัญหา

ผลการทดสอบทั้ง 9 แบบแสดงตามตารางผนวกที่ ก1 ถึงตารางผนวกที่ ก9 และเนื่องจากค่าที่ดีที่สุดที่ได้จากวิธี GAOPT มีค่าเป็น 0 บางค่า (optimal) จึงกำหนดให้ค่าที่ได้จากวิธีฮิวลิสติกส์เป็น 100 % โดยเปอร์เซ็นต์เปรียบเทียบของวิธี GAOPT กับวิธี EDDOPT หาได้จากสมการ (7) และเปอร์เซ็นต์เปรียบเทียบของวิธี GAOPT กับวิธี RNDOPT หาได้จากสมการ (8) ตามลำดับ

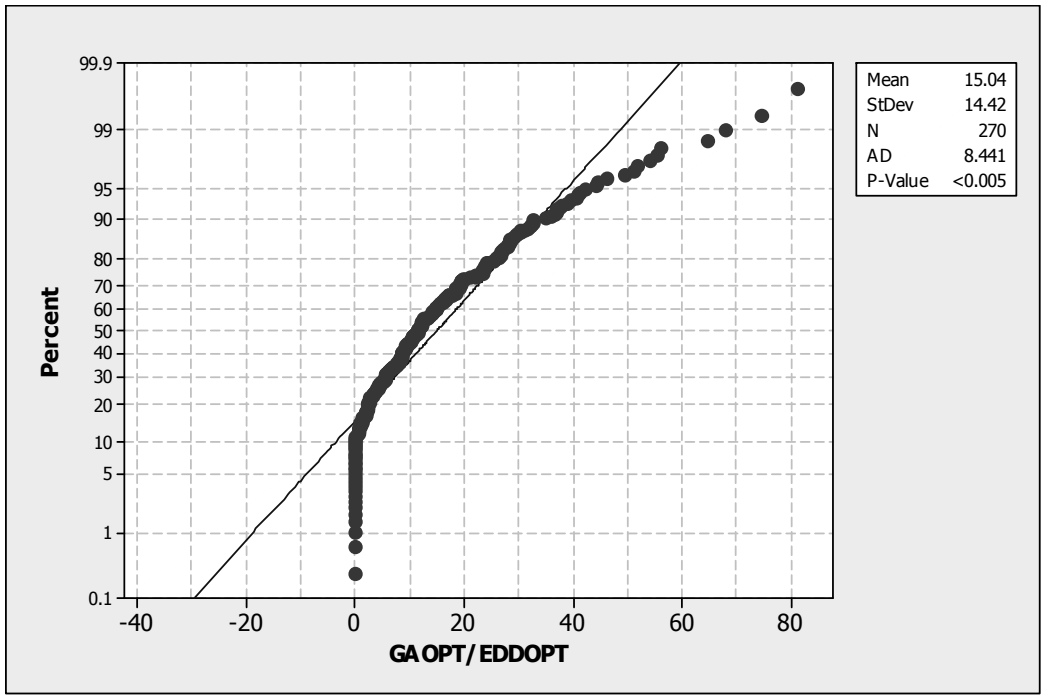
$$\%compare = 100 \times \left(\frac{GAOPT}{EDDOPT} \right) \tag{7}$$

$$\%compare = 100 \times \left(\frac{GAOPT}{RNDOPT} \right) \tag{8}$$

โดยหากค่าเปอร์เซ็นต์เปรียบเทียบที่ได้มีค่าน้อยกว่า 100 แสดงว่าวิธี GAOPT สามารถหาค่าที่ดีที่สุดได้น้อยกว่า

เปรียบเทียบวิธี GAOPT กับวิธี EDDOPT

นำเปอร์เซ็นต์เปรียบเทียบของวิธี GAOPT กับวิธี EDDOPT มาทำ Normality Test พบว่าไม่ใช่การแจกแจงแบบปกติตามภาพที่ 10



ภาพที่ 10 Normality Test ของวิธี GAOPT กับวิธี EDDOPT

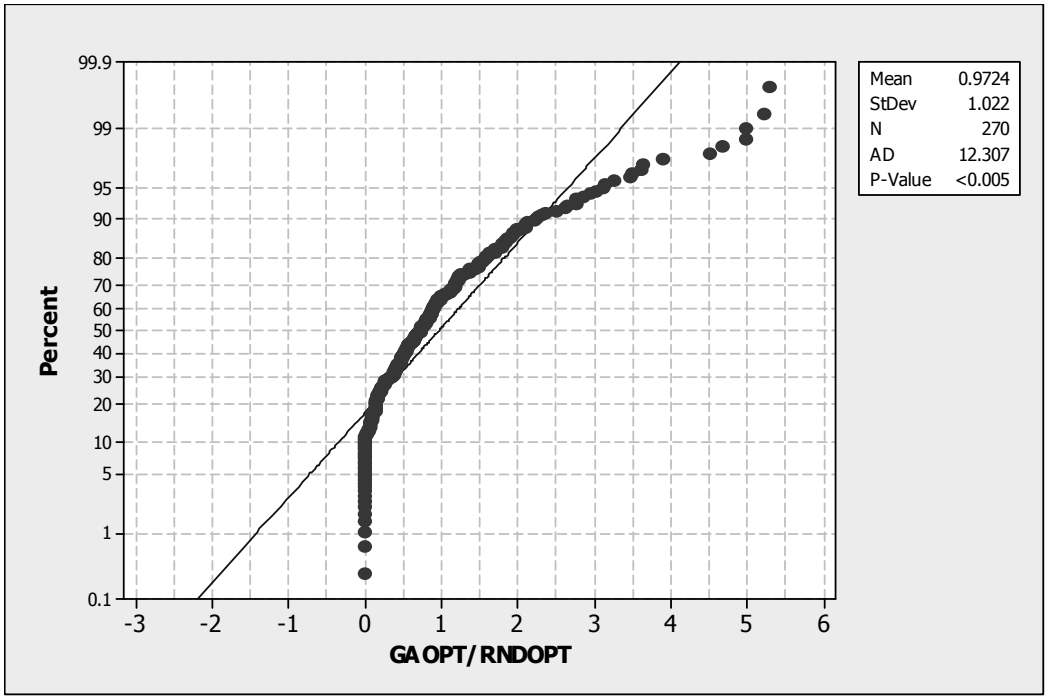
เนื่องจากไม่ใช่การแจกแจงปกติ จึงต้องใช้วิธี Kruskal-Wallis test ในการวิเคราะห์ความแตกต่าง โดยให้จำนวนเครื่องจักรที่ใช้และอัตราส่วนของค่าใช้จ่ายที่ใช้เป็น factor ให้เปอร์เซ็นต์เปรียบเทียบเป็นค่า response ได้ค่า p-Value ตามตารางที่ 5

ตารางที่ 5 ค่า p-Value จากการเปรียบเทียบของวิธี GAOPT กับวิธี EDDOPT

Factor	p-Value
จำนวนเครื่องจักร	0.000
อัตราส่วนของค่าใช้จ่าย	0.558

เปรียบเทียบวิธี GAOPT กับวิธี RNDOPT

นำเปอร์เซ็นต์เปรียบเทียบของวิธี GAOPT กับวิธี RNDOPT มาทำ Normality Test พบว่าไม่ใช่การแจกแจงแบบปกติตามภาพที่ 11



ภาพที่ 11 Normality Test ของวิธี GAOPT กับวิธี RNDOPT

เนื่องจากไม่ใช่การแจกแจงปกติ จึงต้องใช้วิธี Kruskal-Wallis test ในการวิเคราะห์ความแตกต่าง โดยให้จำนวนเครื่องจักรที่ใช้และอัตราส่วนของค่าใช้จ่ายที่ใช้เป็น factor ให้เปอร์เซ็นต์เปรียบเทียบเป็นค่า response ได้ค่า p-Value ตามตารางที่ 6

ตารางที่ 6 ค่า p-Value จากการเปรียบเทียบของวิธี GAOPT กับวิธี RNDOPT

Factor	p-Value
จำนวนเครื่องจักร	0.000
อัตราส่วนของค่าใช้จ่าย	0.165

นำเปอร์เซ็นต์เปรียบเทียบของสถานการณ์ทั้ง 9 แบบแบบละ 30 ปัญหามาหาค่าเฉลี่ย ได้ตามตารางที่ 7

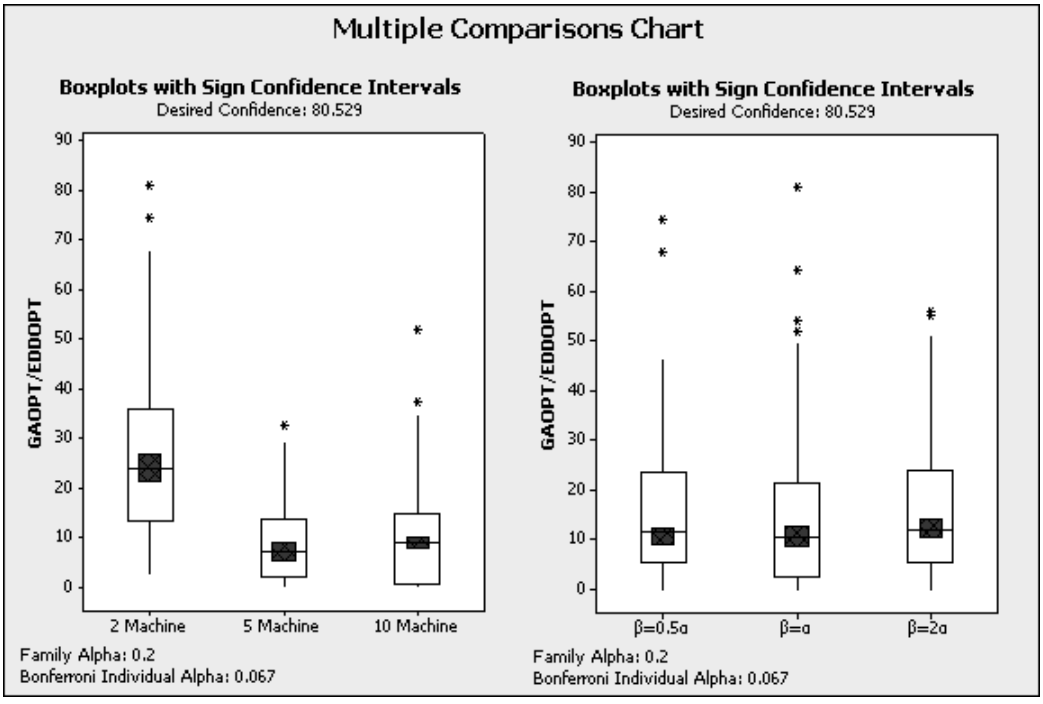
ตารางที่ 7 ค่าเฉลี่ยของเปอร์เซ็นต์เปรียบเทียบของค่าที่ดีที่สุด

เครื่องจักร	อัตราค่าใช้จ่าย	GAOPT/EDDOPT	GAOPT/RNDOPT
2	$\beta = 0.5\alpha$	26.37	1.53
	$\beta = \alpha$	26.68	0.99
	$\beta = 2\alpha$	24.74	1.34
เฉลี่ย 2 เครื่องจักร		25.93	1.29
5	$\beta = 0.5\alpha$	10.34	0.76
	$\beta = \alpha$	7.71	0.50
	$\beta = 2\alpha$	8.77	0.60
เฉลี่ย 5 เครื่องจักร		8.94	0.62
10	$\beta = 0.5\alpha$	9.01	0.88
	$\beta = \alpha$	9.39	0.97
	$\beta = 2\alpha$	12.33	1.18
เฉลี่ย 10 เครื่องจักร		10.25	1.01
เฉลี่ยทั้งหมด		15.04	0.97

วิจารณ์

จากการเปรียบเทียบวิธี GAOPT กับวิธี EDDOPT โดยจำนวนเครื่องจักรเป็น factor ให้เปอร์เซ็นต์เปรียบเทียบเป็นค่า response ได้ค่า p-Value = 0.000 แสดงว่าจำนวนเครื่องจักรที่ใช้มีนัยสำคัญและจาก boxplot ในภาพที่ 11 พบว่าที่เครื่องจักร 5 เครื่องและเครื่องจักร 10 เครื่องให้ค่าเปอร์เซ็นต์เปรียบเทียบที่ไม่แตกต่างกัน แต่สำหรับที่เครื่องจักร 2 เครื่องจะให้ค่าเปอร์เซ็นต์เปรียบเทียบที่มากกว่าเมื่อเปรียบเทียบกับที่เครื่องจักร 5 เครื่องและเครื่องจักร 10 เครื่อง

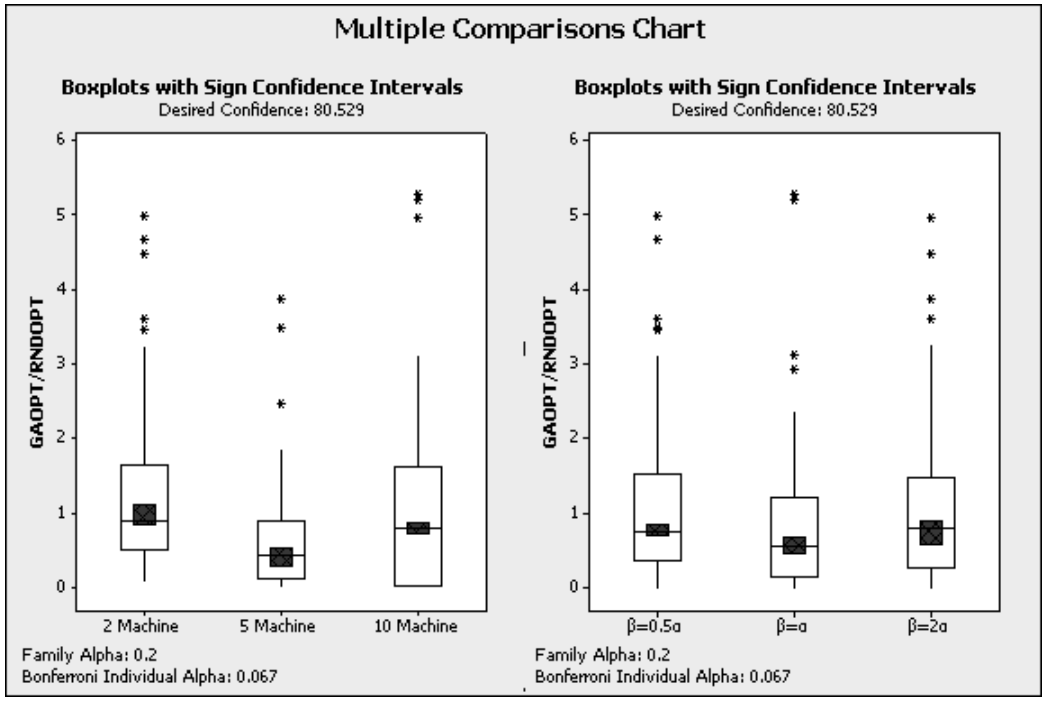
จากการเปรียบเทียบวิธี GAOPT กับวิธี EDDOPT โดยอัตราส่วนของค่าใช้จ่ายเป็น factor ให้เปอร์เซ็นต์เปรียบเทียบเป็นค่า response ได้ค่า p-Value = 0.558 แสดงว่าแสดงว่าอัตราส่วนของค่าใช้จ่ายที่ใช้ไม่มีนัยสำคัญซึ่งสอดคล้องกับ boxplot ที่แสดงในภาพที่ 12



ภาพที่ 12 เปรียบเทียบวิธี GAOPT กับวิธี EDDOPT

จากการเปรียบเทียบวิธี GAOPT กับวิธี RNDOPT โดยจำนวนเครื่องจักรเป็น factor ให้เปอร์เซ็นต์เปรียบเทียบเป็นค่า response ได้ค่า p-Value = 0.000 แสดงว่าจำนวนเครื่องจักรที่ใช้มีนัยสำคัญและจาก boxplot ในภาพที่ 12 พบว่าที่เครื่องจักร 2 เครื่องและเครื่องจักร 10 เครื่องให้ค่าเปอร์เซ็นต์เปรียบเทียบที่ไม่แตกต่างกัน แต่สำหรับที่เครื่องจักร 5 เครื่องจะให้ค่าเปอร์เซ็นต์เปรียบเทียบที่น้อยกว่าเมื่อเปรียบเทียบกับที่เครื่องจักร 2 เครื่องและเครื่องจักร 10 เครื่อง

จากการเปรียบเทียบวิธี GAOPT กับวิธี RNOPT โดยอัตราส่วนของค่าใช้จ่ายเป็น factor ให้เปอร์เซ็นต์เปรียบเทียบเป็นค่า response ได้ค่า p-Value = 0.165 แสดงว่าแสดงว่าอัตราส่วนของค่าใช้จ่ายที่ใช้ไม่มีนัยสำคัญซึ่งสอดคล้องกับ boxplot ที่แสดงในภาพที่ 13



ภาพที่ 13 เปรียบเทียบวิธี GAOPT กับวิธี RNDOPT

จากผลการทดลอง วิธี GAOPT ให้ผลที่ดีกว่าในทุกปัญหา แสดงว่าวิธีเชิงพันธุกรรมที่นำมาใช้แบ่งงานและจัดลำดับงานในแต่ละเครื่องจักรไปประยุกต์ใช้ร่วมกับวิธีการจัดเวลาเริ่มงานเพื่อจัดเวลาเริ่มงานในแต่ละเครื่องจักรนั้นให้ประสิทธิภาพที่ค่อนข้างดีเมื่อนำมาใช้แก้ไขปัญหาการจัดตารางการผลิตของเครื่องจักรขนานที่ไม่เหมือนกันและมีวันส่งมอบที่แตกต่างกันโดยมีฟังก์ชั่นวัตถุประสงค์คือ ผลรวมของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและหลังวันกำหนดส่งมอบมีค่า

น้อยที่สุด และใช้เวลาในการคำนวณสูงสุดน้อยกว่า 5 นาที ซึ่งเป็นเวลาที่ไม่มากนักสำหรับการนำไปใช้งานจริงในงานอุตสาหกรรม

สำหรับวิธี EDDOPT พบว่าเมื่อจำนวนเครื่องจักรมากขึ้น ค่าเฉลี่ยของเปอร์เซ็นต์เปรียบเทียบมีค่าลดลง เป็นผลมาจากการใช้วันกำหนดส่งมอบมาเป็นวิธีการจัดแบ่งงานและลำดับงานให้แต่ละเครื่องจักร โดยเมื่อเครื่องจักรมีจำนวนมากขึ้น ทำให้วันกำหนดส่งมอบของแต่ละงานในแต่ละเครื่องจักรเดียวกันมีความห่างกันมากขึ้น ทำให้การจัดเวลาเริ่มงานในแต่ละเครื่องจักรเพื่อให้ใกล้เคียงกับวันส่งมอบของแต่ละงานทำได้ง่ายขึ้น

สำหรับวิธี RNDOPT ซึ่งมีส่วนคล้ายกับวิธี GAOPT แต่ตัดในส่วนวิธีเชิงพันธุกรรมออกไปนั้น ถึงแม้จะทำการรันโปรแกรมถึง 10,000 รอบ แต่ก็ให้ค่าเฉลี่ยของค่าที่ดีที่สุดแย่กว่าวิธี GAOPT แสดงว่าวิธีเชิงพันธุกรรมที่นำมาประยุกต์ใช้มีผลอย่างมากต่อการจัดตารางการผลิต

จากผลของวิธี GAOPT ที่ให้ผลที่ดีกว่าในทุกปัญหา แสดงว่าโปรแกรมวิธี GAOPT ที่เขียนขึ้นสามารถใช้แก้ปัญหาที่มีอัตราส่วนของอัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จหลังกำหนดส่งมอบงาน (β) ต่ออัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จก่อนกำหนดส่งมอบงาน (α) ได้ทั้งกรณีที่มีอัตราส่วนมากกว่า อัตราส่วนเท่ากันและอัตราส่วนที่น้อยกว่า

สรุปและข้อเสนอแนะ

สรุป

จากการใช้โปรแกรม GAOPT ที่พัฒนาขึ้น เพื่อใช้แก้ไขปัญหาการจัดตารางการผลิตของเครื่องจักรขนานที่ไม่เหมือนกันและมีวันส่งมอบที่แตกต่างกันโดยมีฟังก์ชั่นวัตถุประสงค์คือผลรวมของค่าใช้จ่ายที่เกิดขึ้นเมื่อผลิตเสร็จก่อนและหลังวันกำหนดส่งมอบมีค่าน้อยที่สุด โดยนำวิธีเชิงพันธุกรรมมาใช้แบ่งงานและจัดลำดับงานในแต่ละเครื่องจักร เมื่อทราบจำนวนงานและลำดับงานที่แน่นอนในแต่ละเครื่องจักรแล้วจึงนำไปร่วมประยุกต์ใช้ร่วมกับวิธีการจัดเวลาเริ่มงานเพื่อจัดเวลาเริ่มงานของงานในแต่ละเครื่องจักรนั้น พบว่าโปรแกรม GAOPT ให้ผลที่ดีกว่าการใช้วิธีฮิวลิสติกส์ที่นำมาเปรียบเทียบทั้ง 2 วิธีและใช้เวลาในการประมวลผลน้อยกว่า 5 นาทีสำหรับขนาดปัญหาที่มีงาน 30 งานและมีเครื่องจักรไม่เกิน 10 เครื่อง ซึ่งเป็นเวลาที่ไม่มากนัก สามารถนำไปประยุกต์ใช้วางแผนการผลิตในอุตสาหกรรมจริงได้

ข้อเสนอแนะ

เนื่องจากในขั้นตอนการสร้างประชากรเริ่มต้นเป็นการสร้างแบบสุ่มขึ้นมา ซึ่งบางครั้งอาจได้โครโมโซมเริ่มต้นที่ไม่ดีมากนัก ทำให้การใช้โปรแกรม GAOPT ใช้เวลาในการคำนวณมาก ดังนั้นอาจนำวิธีการแบ่งงานและจัดลำดับงานให้กับแต่ละเครื่องจักรในวิธี EDDOPT มาใช้เป็นส่วนหนึ่งของการสร้างประชากรเริ่มต้น เพื่อให้ได้โครโมโซมเริ่มต้นที่ดีขึ้น ซึ่งจะทำให้การคำนวณใช้เวลาลดน้อยลง

การใช้งานโปรแกรมสำหรับการจัดตารางการผลิตนี้เป็นเพียงการคำนวณทางตัวเลขเพื่อเป็นแนวทางในการจัดตารางการผลิตเท่านั้น เนื่องจากในสถานการณ์จริงนั้น ยังมีปัจจัยอีกหลายอย่างที่ไม่สามารถคำนวณเป็นตัวเลขได้ เช่น อาจมีคำสั่งโดยตรงจากผู้บริหารให้ผลิตงานบางงานก่อนโดยที่ไม่สนใจค่าใช่จ่ายที่เกิดขึ้นหรือต้องผลิตให้กับลูกค้าเก่าก่อนเพื่อรักษาสายสัมพันธ์ที่มีกันมานาน

และหากมองถึงในสภาพความเป็นจริง ค่าใช้จ่ายที่เกิดจากการผลิตเสร็จก่อนวันกำหนดส่งมอบนั้น เป็นค่าใช้จ่ายในส่วนที่ผู้ผลิตต้องรับผิดชอบเองอยู่แล้ว ซึ่งเป็นส่วนที่สามารถประเมินหรือคาดการณ์ล่วงหน้าได้ แต่สำหรับค่าใช้จ่ายที่เกิดจากการผลิตเสร็จหลังวันกำหนดส่งมอบนั้น บางครั้งไม่สามารถประเมินเป็นมูลค่าได้ เช่น การส่งมอบที่ล่าช้าทำให้เกิดความไม่น่าเชื่อถือหรือส่งผลเสียหายต่อลูกค้า ซึ่งอาจส่งผลในระยะยาวสำหรับการตัดสินใจในการสั่งซื้อหรือสั่งผลิตในครั้งถัดไปของลูกค้า ซึ่งในกรณีนี้ สามารถปรับแต่งในส่วนของโปรแรม GAOPT ได้ โดยปรับแต่งในส่วนข้อมูลของอัตราค่าใช้งานของงานเมื่อผลิตเสร็จก่อนวันกำหนดส่งมอบให้มีอัตราส่วนที่ลดลงหรือให้เป็นศูนย์เนื่องจากเป็นค่าใช้จ่ายของผู้ผลิตเอง

นอกจากนี้ในงานบางงานอาจมีเวลาที่ใช้ในการตั้งเครื่อง (Set up time) และค่าใช้จ่ายในการตั้งเครื่อง (Set up cost) เข้ามาเกี่ยวข้อง ในการพัฒนาโปรแกรมต่อถัดไป ควรต้องนำส่วนนี้มาพิจารณาเพิ่มเติมในส่วนของการแก้ปัญหาและการเขียนโปรแกรม

เอกสารและสิ่งอ้างอิง

ปารเมศ ชุติมา. 2546. **เทคนิคการจัดตารางการดำเนินงาน.** ครั้งที่ 1. สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, บริษัท แอคทีฟ พริ้นท์ จำกัด, กรุงเทพฯ.

Baker, K.R and G.D Scudder. 1990. Sequencing with Earliness and Tardiness Penalties: A Review. **Operations Research** 38 (1): 22-36.

Cheng, R., M. Gen and T. Tosawa. 1995. Minmax Earliness/Tardiness Scheduling in Identical Parallel Machine System Using Algorithms. **Computers ind. Engng** 1995 (29): 513-517.

Cheng, T.C.E. and C.C.S. Sin. 1990. A State-of-the-Art Review of Parallel-Machine Scheduling Research. **European Journal of Operational Research** 47: 271-293.

Davis, J.S. and J.J. Kanet. 1993. Single-Machine Scheduling with Early and Tardy Completion Costs. **Naval Research Logistics** 40: 85-101.

Garey, M.R., R.E. Tarjan and G.T. Wilfong. 1988. One-Processor Scheduling with Symmetric Earliness and Tardiness Penalties. **Mathematics of Operations Research** 13 (2): 330-348.

Gao, J. 2005. A Parallel Hybrid Genetic Algorithm for Solving a Kind of Non-Identical Parallel Machine Scheduling Problems, pp. 469-473. *In* **Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region** . IEEE, Zhejiang, China.

Huang, D., H. Guo and N. Qian. 2005. Hybrid Genetic Algorithm For Minimizing The Range Of Lateness And Make-Span On Non-Identical Parallel Machines, pp. 150-154. *In* **2005 International Conference on Neural Network and Brain October 13-15, 2005.** Beijing, China.

Lakshminarayan, S., R. Lakshmanan, R.L. Papineau and R. Rochette. 1978. Optimal Single-Machine Scheduling with Earliness and Tardiness Penalties. **Operations Research** 26 (6): 1079-1083.

Lee, C.Y. and J.Y. Choi. 1995. A Genetic Algorithm for Job Sequencing Ploblems with Distinct Due Dates and General Early-Tardy Penalty Weights. **Computers & Operations Research** 22 (8): 857-869.

Man, K.F., K.S. Tang and S. Kwong. 1996. Genetic Algorithms: Concepts and Applications. **IEEE Transactions on Industrial Electronics** 43 (5): 519-534.

Mandel, M. and G. Mossheiov. 2001. Minimizing Maximum Earliness on Parallel Identical Machines. **Computers & Operations Research** 28: 317-327.

Min, L. and W. Cheng. 1999. A Genetic Algorithm for Minimizing the Makespan in the Case of Scheduling Identical Parallel Machines. **Artificial Intelligence in Engineering** 13: 399-403.

Mokotoff, E. 2001. Parallel Machine Scheduling Ploblems: A Survey. **Asia-Pacific Journal of Operation Research** 18: 193-243.

Obitko, M. 1998. **Genetic Algorithms.** Introduction to Genetic Algorithms with JAVA Appets. Available Source: http://cs.felk.cvut.cz/~xobitko/ga/, April 20, 2006.

Pinedo, M. 2003. **Scheduling Theory, Algorithms, and Systems.** 2nd ed. Prentice-Hall, Inc, United States of America.

Sidney, J.B. 1977. Optimal Single-Machine Scheduling with Earliness and Tardiness Penalties. **Operations Research** 25 (1): 62-69.

Sukimoto, T., H. Tamaki and M. Araki. 1996. Identical Parallel Machine Scheduling with a Non-Regular Objective Function, pp. 2741-2743. *In* **Proceedings of the 35th Conference on Decision and Control .** IEEE, Kobe, Japan.

Szwarc, W. and S.K. Mukhopadhyay. 1995. Optimal Timing Schedules in Earliness-Tardiness Single Machine Sequencing. **Naval Research Logistics** 42: 1109-1114.

Tamaki, H, H. Murao and S. Kitamura. 2003. A Heuristic-Based Hybrid Solution for Parallel Machine Scheduling Problems with Earliness and Tardiness Penalties, pp. 239-244. *In* **Proceedings of the 9th IEEE International Conference on Emerging Technologies and Factory Automation**. IEEE, Kobe, Japan.

_____, T. Komori and S. Abe. 1999. A Heuristic Approach to Parallel Machine Scheduling with Earliness and Tardiness Penalties, pp. 1367-1370. *In* **Proceedings of the 9th IEEE International Conference on Emerging Technologies and Factory Automation** . IEEE, Barcelona, Spain.

ภาคผนวก

ภาคผนวก ก

ผลการทดลอง

ผลการทดลอง

ผลการทดสอบทั้ง 9 แบบแสดงตามตารางผนวกที่ ก1 ถึงตารางผนวกที่ ก9 โดยช่องเวลาในตารางมีหน่วยเป็นวินาที

ตารางผนวกที่ ก1 ผลการทดสอบวิธี GAOPT เปรียบเทียบกับวิธีฮิวริสติกส์ ที่ 2 เครื่องจักร, $\beta = 0.5\alpha$

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
1	557	10.24	129.91	85.05	<0.01	1429.68	3.25	12.04	0.72
2	1227	30.40	199.80	65.76	<0.01	1321.14	3.16	46.23	2.30
3	740	40.65	196.50	174.22	<0.01	1953.37	3.81	23.33	2.08
4	1270	5.20	209.74	32.99	<0.01	2455.53	4.74	15.76	0.21
5	1044	14.61	200.63	98.19	<0.01	1725.24	3.84	14.88	0.85
6	871	11.99	156.89	71.95	<0.01	2902.47	2.19	16.66	0.41
7	907	58.93	153.02	181.03	<0.01	1962.72	1.67	32.55	3.00
8	687	9.39	151.25	65.89	<0.01	1459.24	1.72	14.25	0.64
9	1189	2.80	168.97	32.70	<0.01	1776.85	1.72	8.56	0.16
10	975	2.11	207.97	46.79	<0.01	1813.44	1.78	4.51	0.12
11	579	29.32	124.49	94.66	<0.01	1982.12	1.69	30.97	1.48
12	701	62.20	159.91	173.84	<0.01	2370.42	1.72	35.78	2.62
13	1073	13.17	176.56	114.71	<0.01	2128.24	2.28	11.48	0.62
14	1005	62.05	173.94	91.52	<0.01	1724.28	1.67	67.80	3.60
15	1000	10.44	231.22	44.20	0.02	995.02	1.80	23.62	1.05

ตารางผนวกที่ ก1 (ต่อ)

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
16	1174	21.43	201.56	28.77	<0.01	1510.30	1.72	74.49	1.42
17	1013	46.29	225.36	104.42	<0.01	2220.55	1.67	44.33	2.08
18	491	12.23	150.67	143.54	<0.01	2439.82	1.67	8.52	0.50
19	574	13.27	163.42	32.81	<0.01	2678.99	1.67	40.44	0.50
20	789	24.53	184.38	90.84	<0.01	2089.33	1.64	27.00	1.17
21	864	14.50	163.63	78.55	<0.01	2033.83	1.67	18.46	0.71
22	619	43.12	124.09	109.25	<0.01	1387.98	1.66	39.47	3.11
23	737	21.66	153.66	84.71	0.02	1955.91	1.69	25.57	1.11
24	724	60.39	194.58	163.62	<0.01	1738.34	1.69	36.91	3.47
25	1059	0.82	166.69	34.30	<0.01	1291.12	1.67	2.39	0.06
26	1188	87.92	209.94	226.57	0.02	1884.40	1.69	38.80	4.67
27	681	7.08	124.89	101.90	<0.01	1918.44	1.69	6.95	0.37
28	915	12.84	157.88	48.44	<0.01	1072.72	1.67	26.51	1.20
29	1790	15.89	253.41	82.35	<0.01	2088.98	1.67	19.30	0.76
30	764	88.72	170.19	377.99	<0.01	1783.03	1.64	23.47	4.98

ตารางผนวกที่ ก2 ผลการทดสอบวิธี GAOPT เปรียบเทียบกับวิธีฮิวลิสติกส์ ที่ 2 เครื่องจักร, $\beta = \alpha$

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
31	840	37.65	162.94	158.50	<0.01	4862.43	1.66	23.75	0.77
32	561	19.97	196.76	481.97	<0.01	2171.98	1.66	4.14	0.92
33	851	80.15	216.34	99.00	<0.01	5248.87	1.67	80.96	1.53
34	572	3.24	112.75	98.44	0.01	4090.55	1.67	3.29	0.08
35	906	16.06	138.28	97.73	<0.01	3052.62	1.70	16.43	0.53
36	869	18.80	135.69	123.24	<0.01	1881.15	1.63	15.25	1.00
37	1073	109.55	214.58	170.05	0.02	6861.00	1.67	64.42	1.60
38	1195	12.32	194.47	44.24	<0.01	2517.49	1.69	27.85	0.49
39	850	47.55	177.17	307.20	<0.01	3710.12	1.88	15.48	1.28
40	816	30.49	170.55	127.28	0.02	5056.99	1.80	23.96	0.60
41	943	8.55	133.39	278.92	<0.01	2239.90	1.75	3.07	0.38
42	1304	77.57	180.89	157.41	<0.01	5260.82	1.76	49.28	1.47
43	625	55.09	164.30	291.15	<0.01	3570.77	1.70	18.92	1.54
44	585	17.88	129.00	89.15	<0.01	7893.38	1.91	20.06	0.23
45	1142	143.29	178.11	492.86	0.01	4887.10	1.64	29.07	2.93

ตารางผนวกที่ ก2 (ต่อ)

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
46	640	94.24	160.61	254.78	<0.01	4010.34	1.66	36.99	2.35
47	634	29.28	127.56	210.56	<0.01	2422.48	1.67	13.91	1.21
48	1061	3.47	135.98	31.26	<0.01	5319.74	1.67	11.10	0.07
49	668	20.66	156.24	243.05	<0.01	4386.41	1.69	8.50	0.47
50	1134	16.44	184.53	84.42	<0.01	3967.88	1.69	19.47	0.41
51	1543	20.97	218.31	57.44	<0.01	4730.35	1.74	36.51	0.44
52	1011	72.43	196.24	256.30	<0.01	3605.90	1.67	28.26	2.01
53	1066	50.11	180.08	190.99	<0.01	5384.01	1.73	26.24	0.93
54	638	103.94	194.42	323.36	<0.01	5704.23	3.19	32.14	1.82
55	637	28.42	127.77	64.50	<0.01	5786.69	3.38	44.06	0.49
56	713	20.65	152.94	241.40	<0.01	3044.27	3.19	8.55	0.68
57	888	44.07	178.92	116.65	<0.01	5134.89	3.25	37.78	0.86
58	938	56.78	225.67	105.27	<0.01	4261.64	3.09	53.94	1.33
59	743	4.05	122.86	79.21	<0.01	2340.44	3.23	5.11	0.17
60	1629	73.42	212.63	174.74	<0.01	6676.54	3.09	42.02	1.10

ตารางผนวกที่ ก3 ผลการทดสอบวิธี GAOPT เปรียบเทียบกับวิธีฮิวริสติกส์ ที่ 2 เครื่องจักร, $\beta = 2\alpha$

ปัญหาที่	GAOPT		EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
61	850	23.28	211.58	82.88	<0.01	4300.43	2.39	28.09	0.54
62	661	29.56	120.56	97.18	<0.01	5303.09	1.72	30.42	0.56
63	713	100.29	127.63	181.45	<0.01	8309.51	1.66	55.27	1.21
64	908	202.36	160.17	396.87	0.02	12845.52	1.77	50.99	1.58
65	1242	42.15	156.84	656.68	<0.01	4836.07	1.88	6.42	0.87
66	950	174.14	124.75	671.01	<0.01	9773.91	1.84	25.95	1.78
67	624	17.51	112.66	61.66	0.01	6693.29	1.86	28.40	0.26
68	665	217.52	135.41	388.77	<0.01	9614.70	1.83	55.95	2.26
69	1346	233.61	232.45	983.51	<0.01	7212.61	1.81	23.75	3.24
70	579	50.15	135.22	153.26	<0.01	9339.23	1.86	32.72	0.54
71	816	52.76	170.84	458.36	<0.01	13390.18	1.73	11.51	0.39
72	831	151.72	142.00	476.25	0.02	4192.43	1.69	31.86	3.62
73	737	12.11	137.22	151.06	<0.01	7951.04	1.78	8.02	0.15
74	669	258.27	268.08	633.27	<0.01	9057.59	1.73	40.78	2.85
75	1065	46.45	198.28	194.75	<0.01	5210.84	1.66	23.85	0.89

ตารางผนวกที่ ก3 (ต่อ)

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
76	705	21.95	196.63	132.84	<0.01	6680.64	1.67	16.52	0.33
77	517	17.83	128.14	270.90	<0.01	3841.75	1.69	6.58	0.46
78	1136	50.87	157.61	458.59	<0.01	8074.49	1.81	11.09	0.63
79	1007	143.25	166.55	620.35	<0.01	10570.64	1.63	23.09	1.36
80	921	36.99	205.08	368.99	<0.01	4335.61	1.66	10.02	0.85
81	880	247.56	174.58	845.38	<0.01	11076.38	1.67	29.28	2.24
82	903	143.14	159.78	503.25	<0.01	11777.12	1.67	28.44	1.22
83	1001	119.09	163.11	830.67	<0.01	7514.48	1.66	14.34	1.58
84	699	42.99	171.44	377.62	<0.01	8251.86	1.67	11.38	0.52
85	796	39.09	153.01	758.11	<0.01	4389.74	1.69	5.16	0.89
86	534	35.74	135.94	86.64	0.02	4167.84	2.20	41.25	0.86
87	830	167.54	161.06	552.79	<0.01	6382.29	3.06	30.31	2.63
88	886	94.12	147.72	350.76	<0.01	10102.44	2.77	26.83	0.93
89	1334	182.76	200.22	966.50	<0.01	4070.86	2.47	18.91	4.49
90	675	41.52	148.81	278.97	<0.01	10663.70	2.59	14.88	0.39

ตารางผนวกที่ ก4 ผลการทดสอบวิธี GAOPT เปรียบเทียบกับวิธีฮิวลิสติกส์ ที่ 5 เครื่องจักร, $\beta = 0.5\alpha$

ปัญหาที่	GAOPT		EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
91	609	4.41	142.14	38.07	<0.01	554.81	3.72	11.58	0.79
92	662	5.49	192.55	47.75	<0.01	988.41	2.31	11.50	0.56
93	499	0.00	123.02	19.09	<0.01	883.05	2.23	0.00	0.00
94	1277	16.32	251.24	63.44	<0.01	468.65	2.20	25.73	3.48
95	948	11.33	176.72	51.67	<0.01	636.95	2.23	21.93	1.78
96	1068	6.21	269.61	59.16	0.02	672.13	2.23	10.50	0.92
97	428	1.13	229.42	43.21	<0.01	943.34	2.23	2.62	0.12
98	452	0.35	194.97	31.17	<0.01	248.64	2.20	1.12	0.14
99	921	7.20	168.25	66.74	0.02	765.41	2.28	10.79	0.94
100	1041	5.56	201.34	72.50	<0.01	728.65	2.20	7.67	0.76
101	1150	2.90	180.58	66.84	0.01	702.53	2.19	4.34	0.41
102	622	4.54	157.53	76.01	<0.01	1000.58	2.16	5.97	0.45
103	1335	7.31	224.94	90.75	<0.01	1380.29	2.19	8.06	0.53
104	521	4.10	242.28	16.10	0.02	460.53	2.27	25.47	0.89
105	646	3.02	130.59	70.27	<0.01	993.07	2.16	4.30	0.30

ตารางผนวกที่ ก4 (ต่อ)

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
106	748	11.74	166.50	64.21	<0.01	994.32	2.19	18.28	1.18
107	751	3.18	231.03	128.82	0.02	757.03	2.20	2.47	0.42
108	780	0.43	205.78	31.40	<0.01	456.03	2.22	1.37	0.09
109	551	6.12	161.94	89.73	<0.01	924.29	2.41	6.82	0.66
110	595	0.46	111.84	29.18	<0.01	538.87	2.44	1.58	0.09
111	783	5.83	154.09	49.02	<0.01	1446.66	2.52	11.89	0.40
112	586	0.83	122.86	61.61	<0.01	576.50	2.27	1.35	0.14
113	630	8.95	262.16	27.50	<0.01	1247.77	2.19	32.55	0.72
114	1373	4.97	192.89	58.93	<0.01	717.43	2.16	8.43	0.69
115	451	0.00	134.17	11.82	0.01	776.90	2.20	0.00	0.00
116	723	0.00	177.16	44.95	<0.01	787.10	2.19	0.00	0.00
117	604	14.85	162.50	109.93	<0.01	599.59	2.17	13.51	2.48
118	649	14.53	162.44	116.48	0.02	1203.94	2.20	12.47	1.21
119	768	11.90	152.50	40.31	<0.01	1053.78	2.20	29.52	1.13
120	836	8.41	162.64	45.93	<0.01	519.49	2.20	18.31	1.62

ตารางผนวกที่ ก5 ผลการทดสอบวิธี GAOPT เปรียบเทียบกับวิธีฮิวริสติกส์ ที่ 5 เครื่องจักร, $\beta = \alpha$

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
121	465	0.42	96.70	82.00	<0.01	1174.02	2.25	0.51	0.04
122	955	19.40	186.09	120.06	<0.01	1155.01	2.23	16.16	1.68
123	427	1.42	127.47	62.69	<0.01	1726.29	2.22	2.27	0.08
124	914	1.37	146.41	64.12	<0.01	716.37	2.22	2.14	0.19
125	732	11.22	130.74	120.58	<0.01	1133.40	2.25	9.31	0.99
126	678	0.00	156.23	45.91	<0.01	1169.86	2.27	0.00	0.00
127	530	3.58	147.89	89.88	<0.01	1379.48	2.17	3.98	0.26
128	491	3.61	183.66	81.43	<0.01	4600.39	2.19	4.43	0.08
129	855	3.50	173.13	108.87	<0.01	1671.70	2.20	3.21	0.21
130	853	2.68	221.59	208.27	0.02	1705.28	2.17	1.29	0.16
131	394	0.14	101.42	54.95	<0.01	1347.57	2.25	0.25	0.01
132	733	4.01	157.92	33.92	<0.01	1701.17	2.19	11.82	0.24
133	1000	3.35	191.34	61.29	<0.01	1666.20	2.22	5.47	0.20
134	688	8.30	118.52	152.32	<0.01	1321.52	2.27	5.45	0.63
135	993	9.48	182.75	35.62	0.01	1552.08	2.23	26.61	0.61

ตารางผนวกที่ ก5 (ต่อ)

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
136	499	18.62	171.09	84.78	<0.01	1370.88	2.31	21.96	1.36
137	834	2.32	152.70	126.36	<0.01	1554.11	2.22	1.84	0.15
138	947	23.91	193.20	135.27	<0.01	2026.20	2.27	17.68	1.18
139	394	0.00	101.81	37.04	<0.01	1138.42	2.16	0.00	0.00
140	996	16.41	171.52	84.07	<0.01	1952.92	2.28	19.52	0.84
141	507	12.71	126.00	148.22	<0.01	1025.68	2.28	8.58	1.24
142	678	15.40	150.77	99.58	<0.01	1365.90	2.33	15.46	1.13
143	478	0.02	105.97	27.29	<0.01	935.95	3.36	0.07	0.00
144	636	22.29	204.14	150.96	<0.01	1529.40	3.30	14.77	1.46
145	954	1.60	155.58	67.48	<0.01	991.12	2.19	2.37	0.16
146	438	2.87	115.38	50.63	<0.01	1162.96	2.38	5.67	0.25
147	1896	9.99	210.25	61.91	<0.01	2227.82	2.31	16.14	0.45
148	778	10.31	164.48	260.37	<0.01	1758.99	2.20	3.96	0.59
149	1027	13.52	294.86	144.01	<0.01	2129.63	2.20	9.39	0.63
150	583	0.94	129.44	108.29	<0.01	1448.79	2.22	0.87	0.06

ตารางผนวกที่ ก6 ผลการทดสอบวิธี GAOPT เปรียบเทียบกับวิธีฮิวริสติกส์ ที่ 5 เครื่องจักร, $\beta = 2\alpha$

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
151	981	2.75	136.58	78.00	<0.01	1561.13	2.22	3.53	0.18
152	481	2.64	180.25	105.98	<0.01	2885.86	2.27	2.49	0.09
153	439	3.02	96.53	131.61	<0.01	2635.88	2.19	2.29	0.11
154	478	11.41	137.31	68.05	<0.01	5191.78	2.34	16.77	0.22
155	1434	101.46	220.31	380.06	<0.01	8689.73	2.25	26.70	1.17
156	524	12.14	126.06	277.74	<0.01	1818.63	2.25	4.37	0.67
157	678	21.87	140.94	146.01	<0.01	4008.02	2.36	14.98	0.55
158	1050	62.18	185.74	468.96	<0.01	3344.72	2.19	13.26	1.86
159	872	16.58	211.50	160.23	<0.01	1382.04	2.22	10.35	1.20
160	569	9.07	143.20	163.83	<0.01	2451.17	2.33	5.54	0.37
161	579	6.84	178.89	93.65	<0.01	1600.61	2.31	7.30	0.43
162	1020	26.50	175.05	212.09	<0.01	2983.11	2.25	12.49	0.89
163	610	19.38	201.52	210.76	<0.01	3806.83	2.27	9.20	0.51
164	826	11.83	170.45	129.73	<0.01	1488.62	2.25	9.12	0.79
165	514	44.82	123.70	229.32	<0.01	2990.40	2.22	19.54	1.50

ตารางผนวกที่ ก6 (ต่อ)

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
166	445	2.02	105.98	37.14	<0.01	3646.70	2.28	5.44	0.06
167	1127	2.48	145.80	345.71	0.02	1963.62	2.28	0.72	0.13
168	830	10.33	170.20	55.88	<0.01	2818.37	2.30	18.49	0.37
169	530	6.35	119.25	335.23	<0.01	3191.90	2.28	1.89	0.20
170	507	0.84	121.44	138.16	<0.01	2443.33	2.19	0.61	0.03
171	532	4.44	130.47	166.02	<0.01	1668.03	2.27	2.67	0.27
172	486	16.69	147.33	168.60	0.01	2545.46	2.39	9.90	0.66
173	1171	39.85	233.13	146.24	<0.01	3850.40	2.27	27.25	1.03
174	1194	92.68	204.81	481.94	<0.01	2391.72	2.28	19.23	3.88
175	711	3.24	163.67	168.32	<0.01	2513.80	2.23	1.92	0.13
176	569	0.00	129.78	29.41	<0.01	5443.03	2.27	0.00	0.00
177	588	0.00	162.89	229.72	0.02	4152.80	2.20	0.00	0.00
178	593	0.00	139.03	60.92	<0.01	1747.42	2.28	0.00	0.00
179	572	13.56	126.27	177.69	<0.01	2969.40	2.19	7.63	0.46
180	564	2.40	139.72	25.67	<0.01	1932.98	2.19	9.35	0.12

ตารางผนวกที่ ก7 ผลการทดสอบวิธีGAOPT เปรียบเทียบกับวิธีฮิวริสติกส์ ที่ 10 เครื่องจักร, $\beta = 0.5\alpha$

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
181	918	6.22	211.53	17.82	<0.01	225.96	4.53	34.90	2.75
182	530	3.36	160.91	59.00	<0.01	417.63	3.23	5.69	0.80
183	903	7.44	193.88	44.62	<0.01	929.84	3.11	16.67	0.80
184	855	16.14	222.02	82.14	<0.01	906.92	2.94	19.65	1.78
185	1010	0.00	209.20	48.67	<0.01	405.93	3.20	0.00	0.00
186	894	0.00	177.25	59.30	0.01	333.38	2.92	0.00	0.00
187	589	1.82	226.73	21.13	<0.01	376.24	2.92	8.61	0.48
188	801	3.92	205.72	51.55	<0.01	457.28	2.91	7.60	0.86
189	546	4.49	132.92	37.45	0.02	226.94	2.89	11.99	1.98
190	657	10.55	142.95	72.01	<0.01	550.75	2.89	14.65	1.92
191	856	5.85	200.84	34.14	<0.01	295.78	2.91	17.14	1.98
192	780	5.80	160.81	60.14	0.02	507.75	2.86	9.64	1.14
193	620	0.40	189.27	36.49	<0.01	1241.46	2.86	1.10	0.03
194	812	4.74	153.01	46.42	<0.01	262.79	2.89	10.21	1.80
195	938	6.63	185.52	35.82	0.02	392.59	2.91	18.51	1.69

ตารางผนวกที่ ก7 (ต่อ)

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
196	470	2.37	137.76	37.76	<0.01	370.47	2.89	6.28	0.64
197	396	0.00	110.99	18.69	<0.01	219.23	2.89	0.00	0.00
198	502	3.60	119.89	41.34	<0.01	408.35	3.22	8.71	0.88
199	549	0.00	99.69	17.63	<0.01	392.27	3.23	0.00	0.00
200	412	0.00	87.83	31.02	<0.01	391.51	3.70	0.00	0.00
201	636	2.87	194.42	49.43	<0.01	291.79	3.11	5.81	0.98
202	1579	3.64	292.81	40.53	<0.01	494.36	2.94	8.98	0.74
203	1105	2.93	167.03	47.85	0.02	353.75	3.00	6.12	0.83
204	837	4.97	186.74	54.83	<0.01	691.74	3.02	9.06	0.72
205	582	0.00	133.56	25.00	<0.01	342.32	2.89	0.00	0.00
206	603	6.61	203.41	53.79	0.01	315.89	2.95	12.29	2.09
207	521	2.58	137.74	33.23	<0.01	317.71	3.00	7.76	0.81
208	571	1.81	221.48	68.19	<0.01	448.77	2.95	2.65	0.40
209	582	1.20	150.47	5.03	<0.01	401.40	3.02	23.86	0.30
210	494	0.40	137.11	16.50	<0.01	313.22	3.13	2.42	0.13

ตารางผนวกที่ ก8 ผลการทดสอบวิธีGAOPT เปรียบเทียบกับวิธีฮิวริสติกส์ ที่ 10 เครื่องจักร, $\beta = \alpha$

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
211	528	5.80	154.53	67.69	<0.01	678.58	2.97	8.57	0.85
212	462	0.00	146.48	18.32	<0.01	828.52	2.94	0.00	0.00
213	152	0.00	127.48	52.61	<0.01	980.21	2.95	0.00	0.00
214	806	17.21	267.89	161.62	0.01	330.47	3.00	10.65	5.21
215	914	7.59	210.73	72.03	<0.01	1109.81	3.00	10.54	0.68
216	437	1.32	139.83	67.03	<0.01	913.36	3.25	1.97	0.14
217	263	0.00	108.19	85.21	<0.01	402.68	3.11	0.00	0.00
218	1139	25.30	245.09	129.71	<0.01	1509.50	3.03	19.51	1.68
219	667	4.99	129.78	147.58	0.02	1906.36	2.89	3.38	0.26
220	600	15.66	168.59	128.35	<0.01	974.59	3.17	12.20	1.61
221	744	0.00	141.41	47.39	<0.01	968.42	2.88	0.00	0.00
222	855	1.02	151.80	125.96	<0.01	297.45	2.91	0.81	0.34
223	540	6.59	140.55	136.17	<0.01	564.91	2.95	4.84	1.17
224	645	9.48	216.72	94.61	<0.01	1093.74	3.30	10.02	0.87
225	167	0.00	88.72	43.02	<0.01	776.35	2.91	0.00	0.00

ตารางผนวกที่ ก8 (ต่อ)

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
226	90	0.00	60.36	11.77	0.02	607.88	2.92	0.00	0.00
227	378	13.12	132.51	56.52	<0.01	1340.53	2.97	23.21	0.98
228	506	0.59	145.92	79.27	0.01	3108.02	3.14	0.74	0.02
229	666	4.88	183.64	39.51	<0.01	1055.76	2.94	12.35	0.46
230	938	3.39	148.16	6.55	<0.01	454.36	2.97	51.76	0.75
231	754	19.04	175.03	85.69	<0.01	1019.24	3.05	22.22	1.87
232	456	0.00	88.42	20.13	0.02	576.17	2.92	0.00	0.00
233	589	9.44	135.94	72.96	<0.01	768.91	2.94	12.94	1.23
234	625	0.94	180.61	152.50	<0.01	1534.06	2.92	0.62	0.06
235	1425	33.39	245.09	119.27	<0.01	631.99	2.84	28.00	5.28
236	681	22.99	163.73	155.46	<0.01	733.92	2.88	14.79	3.13
237	520	15.46	165.25	73.59	<0.01	842.57	2.88	21.01	1.83
238	479	0.00	126.80	82.27	<0.01	404.59	2.89	0.00	0.00
239	423	1.54	117.80	42.32	<0.01	1268.90	2.92	3.64	0.12
240	864	3.30	154.48	40.99	<0.01	620.90	2.86	8.05	0.53

ตารางผนวกที่ ก9 ผลการทดสอบวิธีGAOPT เปรียบเทียบกับวิธีฮิวริสติกส์ ที่ 10 เครื่องจักร, $\beta = 2\alpha$

ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
241	563	43.62	128.41	116.25	<0.01	1587.72	2.91	37.52	2.75
242	622	17.68	147.38	217.16	<0.01	1961.65	2.91	8.14	0.90
243	1168	35.72	227.42	260.88	<0.01	1862.25	2.89	13.69	1.92
244	973	41.50	248.59	385.25	<0.01	1883.82	2.91	10.77	2.20
245	839	22.26	162.88	91.72	<0.01	1502.64	2.91	24.27	1.48
246	607	0.00	152.70	174.26	0.02	1125.04	2.91	0.00	0.00
247	539	35.99	142.80	251.98	<0.01	2090.82	2.92	14.28	1.72
248	483	18.20	149.91	180.82	<0.01	957.16	2.92	10.07	1.90
249	490	30.74	211.97	253.22	<0.01	1943.20	3.19	12.14	1.58
250	422	18.28	112.95	129.16	<0.01	2307.17	3.88	14.15	0.79
251	603	21.96	174.95	183.00	0.02	1039.84	2.95	12.00	2.11
252	1071	70.74	286.89	223.17	<0.01	1425.70	2.78	31.70	4.96
253	630	11.08	181.73	47.63	<0.01	820.65	2.84	23.26	1.35
254	404	8.28	134.92	187.96	<0.01	1490.95	2.88	4.41	0.56
255	877	0.00	158.86	42.19	<0.01	1708.74	3.11	0.00	0.00

ตารางผนวกที่ ก9 (ต่อ)

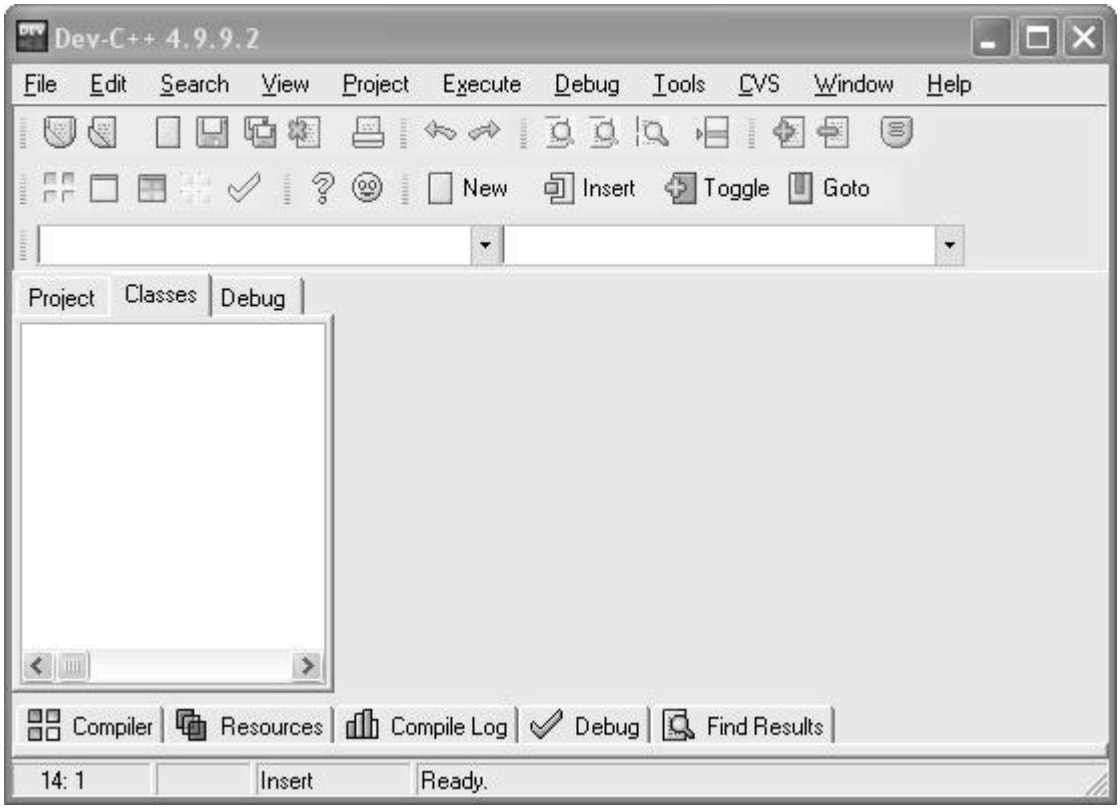
ปัญหาที่	GAOPT			EDDOPT		RNDOPT		เปอร์เซ็นต์เปรียบเทียบ	
	รอบที่หยุด	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	ค่าที่ดีที่สุด	เวลา	GAOPT/EDDOPT	GAOPT/RNDOPT
256	532	5.30	212.66	248.86	0.01	1423.38	2.89	2.13	0.37
257	476	37.02	202.72	129.36	<0.01	5238.51	2.86	28.62	0.71
258	666	16.38	174.20	136.75	<0.01	1375.80	2.83	11.98	1.19
259	152	0.00	87.42	19.94	<0.01	4927.05	2.88	0.00	0.00
260	915	15.18	199.78	83.74	<0.01	1025.80	2.92	18.13	1.48
261	554	11.24	122.31	124.68	<0.01	1009.38	2.97	9.02	1.11
262	467	18.88	121.64	231.84	<0.01	1789.58	2.91	8.14	1.05
263	684	0.00	139.01	245.96	<0.01	1236.80	2.97	0.00	0.00
264	462	0.00	106.89	150.92	<0.01	904.90	2.88	0.00	0.00
265	423	0.00	100.41	135.30	<0.01	1471.40	3.03	0.00	0.00
266	1027	0.00	185.84	128.00	<0.01	1347.91	3.20	0.00	0.00
267	1435	67.32	255.67	251.94	<0.01	2453.80	2.91	26.72	2.74
268	980	33.24	252.76	180.28	<0.01	3535.06	3.00	18.44	0.94
269	543	23.74	139.47	140.06	<0.01	3261.65	2.89	16.95	0.73
270	784	38.68	149.72	287.08	<0.01	3950.48	2.89	13.47	0.98

ภาคผนวก ข

การใช้งานโปรแกรม GAOPT

การใช้งานโปรแกรม GAOPT

1. เริ่มต้นด้วยการติดตั้งโปรแกรม Dev-C++ Version 4.9.9.2 ซึ่งเป็นโปรแกรมแจกฟรี ดาน์วโหลดได้จาก http://www.bloodshed.net/dev/devcpp.html หน้าจอของโปรแกรมหลังการติดตั้งจะได้ตามภาพผนวกที่ ข1



ภาพผนวกที่ ข1 หน้าจอโปรแกรม Dev-C++

2. เปิดไฟล์ GAOPT.c ขึ้นมาเพื่อปรับแต่งค่าก่อนนำไปใช้งาน โดยจะมีอยู่ 4 ตำแหน่ง เพื่อให้ตรงกับปัญหาที่จะนำโปรแกรมมาใช้ในการแก้ไขในบรรทัดที่ 9, 10, 12 และ 13 ตามภาพผนวกที่ ข2

2.1 บรรทัดที่ 9 ระบุถึงตำแหน่งของไฟล์ปัญหาที่จะนำมาแก้ไข จากภาพผนวกที่ ข2 ไฟล์ปัญหามีชื่อว่า job.txt เก็บอยู่ในโฟลเดอร์ problem ในไดร์ K หากต้องการเป็นตำแหน่งที่เก็บไฟล์

หรือชื่อไฟล์ สามารถเปลี่ยนได้ เช่น เปลี่ยนชื่อไฟล์เป็น job2.txt และอยู่ในไดร์ C ได้ดังนี้ “C:\\job2.txt”

```
/*-----------------------------------------------------------------*/
/*          Change detail of file and detail of job                */
/*-----------------------------------------------------------------*/
#define FILE1 "k:\\problem\\job.txt"      /* input file */
#define FILE4 "k:\\result\\startjob.txt"  /* output file */

#define MACHINES  2
#define MAXJOBS   30

#define PROBLEM_NO 1
/*-----------------------------------------------------------------*/
/*-----------------------------------------------------------------*/
```

ภาพผนวกที่ ข2 โปรแกรมส่วนที่ต้องปรับแต่ง

2.2 บรรทัดที่ 10 ระบุถึงตำแหน่งของไฟล์ผลการจัดตารางปัญหาที่นำมาแก้ไข จากภาพผนวกที่ 2 ไฟล์ไฟล์ผลการจัดตารางมีชื่อว่า startjob.txt เก็บอยู่ในโฟลเดอร์ result ในไดร์ K หากต้องการเป็นตำแหน่งที่เก็บไฟล์หรือชื่อไฟล์ สามารถเปลี่ยนได้เลยเช่น เปลี่ยนชื่อไฟล์เป็น startjob2.txt อยู่ในโฟลเดอร์ result2 และอยู่ในไดร์ C ได้ดังนี้ “C:\\result2\\job2.txt”

2.3 บรรทัดที่ 12 ระบุจำนวนเครื่องจักรให้ตรงกับปัญหา

2.4 บรรทัดที่ 13 ระบุจำนวนงานให้ตรงกับปัญหา

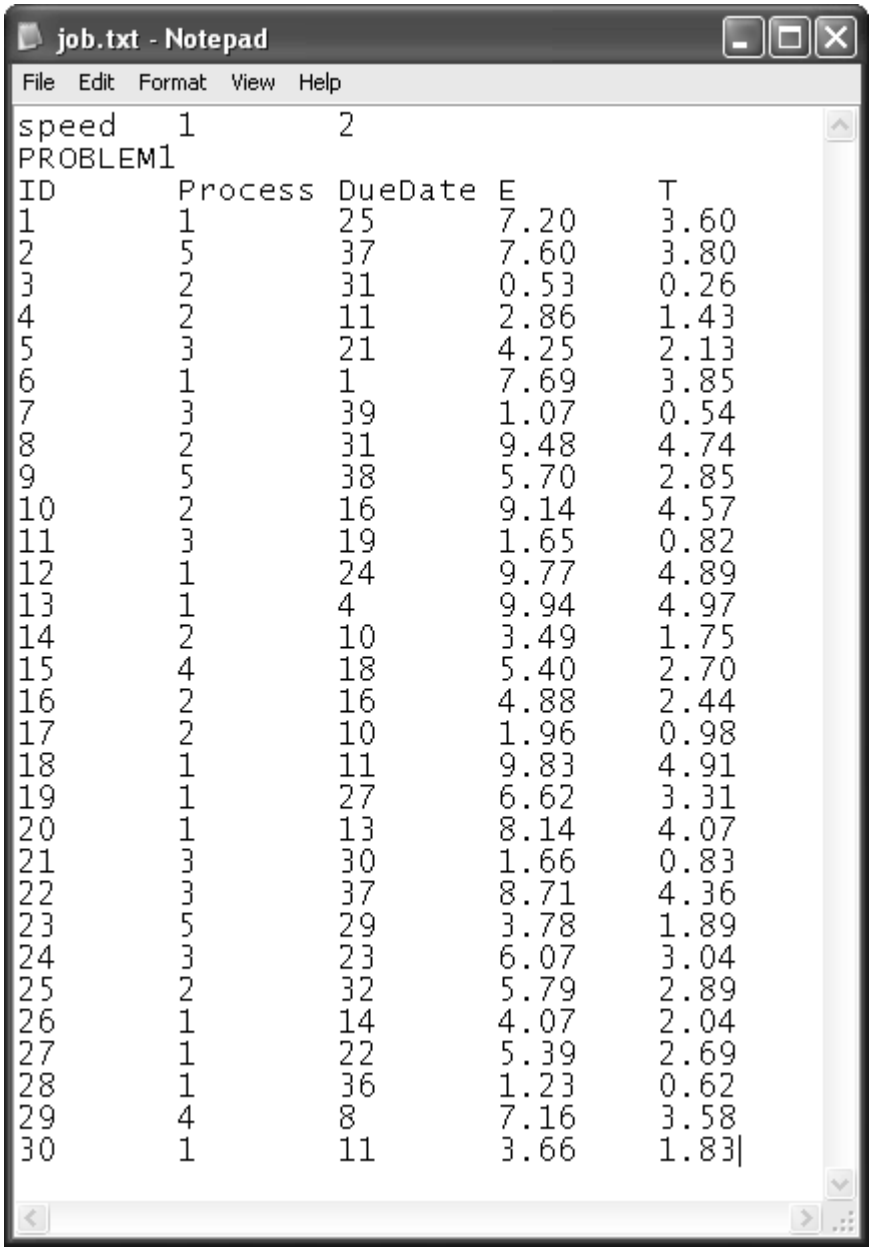
2.5 บรรทัดที่ 15 ระบุจำนวนของปัญหา ใช้แก้ไขในกรณีที่ทำมากกว่า 1 ปัญหาพร้อมกัน

หลังจากทำการแก้ไขแล้วให้ทำการคอมไพล์โปรแกรมก่อนนำไปใช้งาน โดยไปที่เมนูเลือก Execute และเลือก Compile

3. จัดเตรียมไฟล์ปัญหา โดยตั้งชื่อและเก็บไฟล์ไว้ให้ตรงกับที่ระบุในข้อที่ 2 ซึ่งไฟล์ที่ใช้สำหรับ GAOPT มีนามสกุลเป็น .txt ซึ่งจะระบุอัตราความเร็วของเครื่องจักร, หมายเลขของงาน, เวลาที่ใช้ในการทำงาน, วันกำหนดส่งมอบ, อัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จก่อนกำหนดส่งมอบงาน, อัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จหลังกำหนดส่งมอบงาน ดังนี้

3.1 บรรทัดที่ 1 ระบุอัตราความเร็วของเครื่องจักรให้เท่ากับจำนวนเครื่องจักรที่ระบุในข้อ 2 เช่น ในภาพผนวกที่ ข3 มีเครื่องจักร 2 เครื่อง มีอัตราความเร็วของเครื่องจักรเป็น 1 และ 2

ตามลำดับ โดยอัตราความเร็วที่มากกว่า หมายความว่า มีความเร็วในการปฏิบัติงานมากกว่าเป็นจำนวนเท่าของอัตราความเร็วปกติ นั่นคือใช้เวลาปฏิบัติงานน้อยลง โดยเศษทศนิยมที่เกินมาคิดเป็น 1 วันเต็ม



ภาพผนวกที่ ข3 ไฟล์ปัญหา

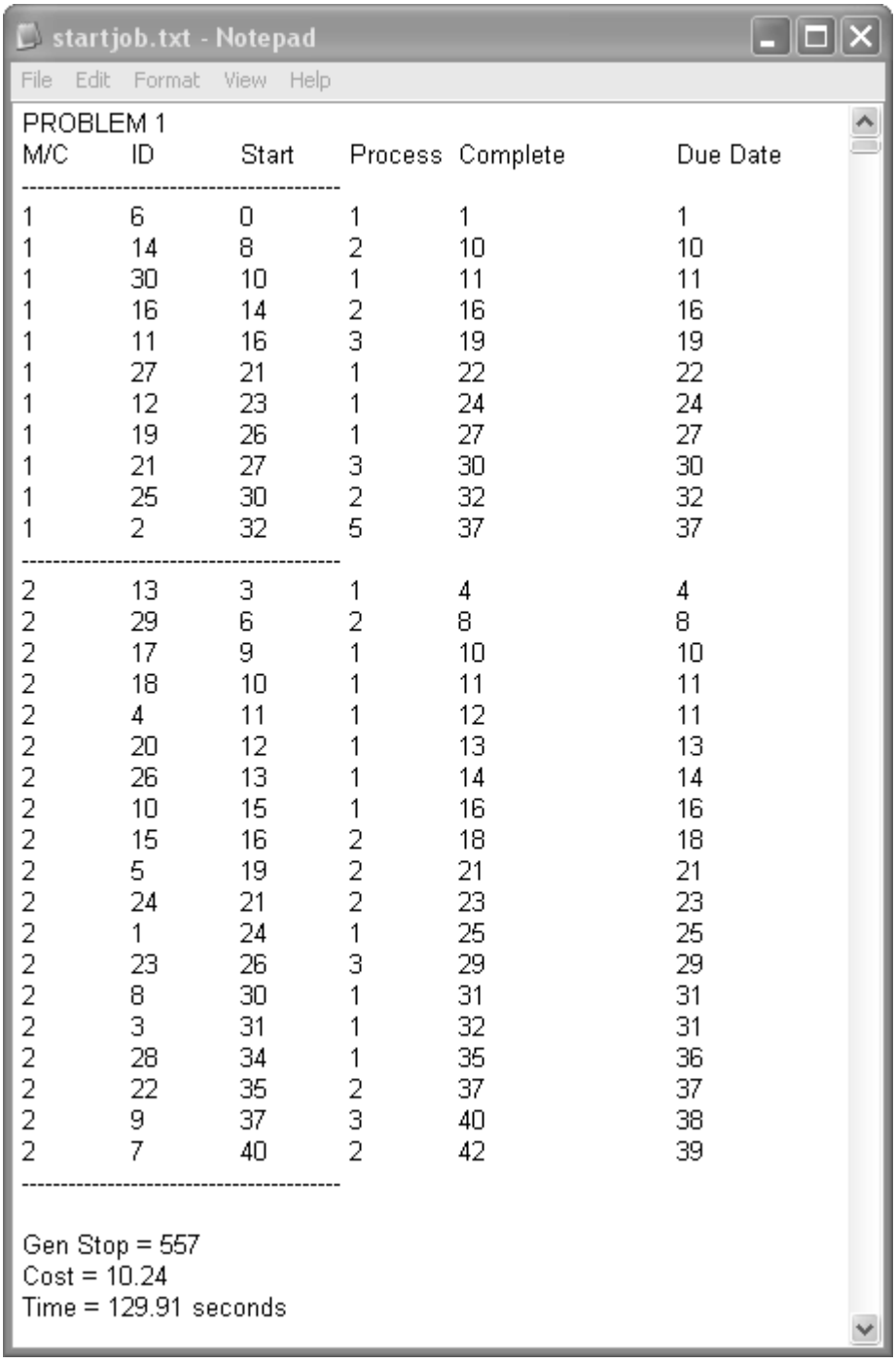
3.2 บรรทัดที่ 4 เป็นต้นไปจะระบุรายละเอียดของงาน งานละ 1 บรรทัด โดยคอลัมน์ที่ 1 เป็นหมายเลขของงาน, คอลัมน์ที่ 2 เป็นเวลาที่ใช้ในการทำงาน, คอลัมน์ที่ 3 เป็นวันกำหนดส่งมอบ,

คอลัมน์ที่ 4 เป็นอัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จก่อนกำหนดส่งมอบงานและคอลัมน์ที่ 5 เป็นอัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จหลังกำหนดส่งมอบงาน เช่น ในภาพผนวกที่ ข3 มีงาน 30 งาน โดยงานที่ 1 มีเวลาที่ใช้ในการทำงานเท่ากับ 1, วันกำหนดส่งมอบเท่ากับ 25, อัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จก่อนกำหนดส่งมอบงานเท่ากับ 7.20 และอัตราค่าใช้จ่ายต่อเวลาเมื่องานเสร็จหลังกำหนดส่งมอบงานเท่ากับ 3.60

4. ทำการรันโปรแกรมเพื่อแก้ไขปัญหา โดยไปที่เมนู เลือก Execute เลือก Run

5. เปิดไฟล์ผลการจัดตารางตามชื่อและตำแหน่งที่ระบุในข้อ 2 จะเห็นไฟล์ตามภาพผนวกที่ ข4 โดยที่ M/C หมายถึงเครื่องจักร, ID หมายถึงหมายเลขงาน, Start หมายถึงเวลาเริ่มงาน, Process หมายถึงเวลาที่ใช้ในการปฏิบัติงาน, Complete หมายถึงวันที่ปฏิบัติงานเสร็จ, Due Date หมายถึงวันกำหนดส่งมอบ

ตัวอย่างเช่น จากภาพผนวกที่ ข4 งานที่ 13 ทำบนเครื่องจักรที่ 2 โดยเริ่มทำงานวันที่ 3 ใช้เวลาปฏิบัติงาน 1 วัน งานเสร็จวันที่ 4 กำหนดส่งมอบวันที่ 4 โดยการทำงานทั้งหมดมีค่าใช้จ่ายรวม 10.24 โดยโปรแกรมใช้เวลาคำนวณ 129.91 วินาที



startjob.txt - Notepad

File Edit Format View Help

PROBLEM 1

M/C	ID	Start	Process	Complete	Due Date
1	6	0	1	1	1
1	14	8	2	10	10
1	30	10	1	11	11
1	16	14	2	16	16
1	11	16	3	19	19
1	27	21	1	22	22
1	12	23	1	24	24
1	19	26	1	27	27
1	21	27	3	30	30
1	25	30	2	32	32
1	2	32	5	37	37
2	13	3	1	4	4
2	29	6	2	8	8
2	17	9	1	10	10
2	18	10	1	11	11
2	4	11	1	12	11
2	20	12	1	13	13
2	26	13	1	14	14
2	10	15	1	16	16
2	15	16	2	18	18
2	5	19	2	21	21
2	24	21	2	23	23
2	1	24	1	25	25
2	23	26	3	29	29
2	8	30	1	31	31
2	3	31	1	32	31
2	28	34	1	35	36
2	22	35	2	37	37
2	9	37	3	40	38
2	7	40	2	42	39

Gen Stop = 557
Cost = 10.24
Time = 129.91 seconds

ภาพผนวกที่ ข4 ผลการจัดตาราง

ภาคผนวก ค

code ภาษาซีของโปรแกรม GAOPT

```
1: #include <stdio.h>
2: #include <string.h>
3: #include <stdlib.h>
4: #include <sys\timeb.h>
5:
6: /*-------------------------------------------------------------*/
7: /* Change detail of file and detail of job */
8: /*-------------------------------------------------------------*/
9: #define FILE1 "k:\\problem\\job.txt" /* input file */
10: #define FILE4 "k:\\result\\startjob1.txt" /* output file */
11:
12: #define MACHINES 2
13: #define MAXJOBS 30
14:
15: #define PROBLEM_NO 1
16: /*-------------------------------------------------------------*/
17: /*-------------------------------------------------------------*/
18:
19: #define randomize() srand((unsigned)time(NULL)) /* seed number */
20:
21:
22: #define POP_SIZE 100 /*population size */
23: #define C_OFF_SIZE (2*POP_SIZE) /*crossover offspring size*/
24: #define M_OFF_SIZE (1*POP_SIZE) /*mutation offspring size*/
25: #define All_POP_SIZE (POP_SIZE+C_OFF_SIZE+M_OFF_SIZE)
26: /*all pop size */
27: #define CHROM_LENGTH (MAXJOBS+MACHINES-1) /* Chromosome size */
28: #define pCross 0.6 /* probability of Crossover */
29: #define pMut 0.1 /* probability of Mutation */
30: #define MAX_GEN 10000 /* Maximum generations */
```

```
31: #define LOOP 3
32:
33:
34: /* Job structure declaration */
35: typedef struct /* JOB_INFO */
36: {
37: char id[10];
38: int ProcessingTime;
39: int DueDate;
40: float EarlinessCost;
41: float TardinessCost;
42: float StartingTime;
43: float CompletionTime;
44: int Block;
45: int Order;
46: float SumE;
47: } job_info;
48:
49: /* POPULATION structure declaration */
50: typedef struct
51: {
52: job_info schd[CHROM_LENGTH]; /* Chromosome of schedule */
53: float Cost ; /* cost of schedule */
54: double Fitness ; /* fitness for selection */
55: }population;
56:
57: typedef struct /* statistics */
58: {
59: float min_cost ;
60: double avg_Fitness ;
```

```
61: }statistics;
62:
63: population pop[POP_SIZE];
64: population BEST;
65:
66: population c_offspring[C_OFF_SIZE];
67: population m_offspring[M_OFF_SIZE];
68: population all_pop[All_POP_SIZE];
69:
70: statistics stat[MAX_GEN];
71:
72: float rate[MACHINES];
73:
74: void shufle_job();
75:
76: void optimal_timing (population temp[], int);
77: void single_job ( population temp[], int, int, int );
78: void empty_machine ( population temp[], int, int );
79: void opt_single (population temp[], int, int, int, int);
80: void check_con (population temp[], int, int, int, int);
81: int B_First_Check (population temp[], int, int, int);
82: int Min_Point (population temp[], int, int, int);
83: int Check_Condition (population temp[], int);
84: void Shift_Block (population temp[], int, int, int, int);
85:
86: void crossover ();
87: void mutation();
88: void reproduction();
89: void evaluate (population temp[], int, int );
90:
```

```
91: FILE *fp1; /* input data of job */
92: FILE *fp4; /* output schedule */
93:
94: int Shift;
95: float BEST_COST;
96:
97: void select(int, double);
98: int selected_p[POP_SIZE];
99: int gen;
100: double improve (int);
101:
102:
103:/*== ================================================*/
104: /* Main */
105: /*================================================*/
106: main ()
107: {
108: int i, k, p, r,l;
109: int cnt, m_cnt;
110: float min_cost, best_cost;
111: char c[15];
112: int gen_stop, best_gen;
113: double sum_fitness, avg_fitness ;
114: double improvement ;
115: double t_diff;
116: struct timeb t_start, t_current;
117: population p1;
118:
119: /* Open data of job file */
120: fp1 = fopen(FILE1, "r");
```

```
121: if (fp1 == NULL)
122: {
123: printf ("** ERROR: cannot open file %s\n", FILE1);
124: printf ("\n**** End of run ****");
125: printf ("\n**** Press any key to EXIT ****");
126: getchar();
127: exit (EXIT_FAILURE);
128: }
129:
130: /* Open output schedule */
131: fp4 = fopen(FILE4, "w+");
132: if (fp4 == NULL)
133: {
134: printf ("** ERROR: cannot open file %s\n", FILE4);
135: printf ("\n**** End of run ****");
136: printf ("\n**** Press any key to EXIT ****");
137: getchar();
138: exit (EXIT_FAILURE);
139: }
140:
141: for (p = 1; p <= PROBLEM_NO ; p++)
142: {
143:
144: ftime(&t_start);
145:
146: printf("\nPROBLEM %d\n", p );
147: fprintf(fp4,"PROBLEM %d\n", p , FILE4 );
148:
149: /* initialize_population */
150: /* get speed of machine */
```

```
151: m_cnt = 0 ;
152: fscanf(fp1, "%s", c ); /*read "speed" from text file*/
153:
154: while( !feof(fp1) )
155: {
156: if(m_cnt == (MACHINES))
157: break;
158:
159: fscanf(fp1, "%f", &rate[m_cnt] );
160: m_cnt++; /* Read next machine */
161: }
162:
163: /* get job */
164: cnt = 0;
165: fscanf(fp1, "%s", c ); /*read "PROBLEM" from text file*/
166: fscanf(fp1, "%s", c ); /*read "ID" from text file*/
167: fscanf(fp1, "%s", c ); /*read "Process" from text file*/
168: fscanf(fp1, "%s", c ); /*read "DueDaye" from text file*/
169: fscanf(fp1, "%s", c ); /*read "E" from text file*/
170: fscanf(fp1, "%s", c ); /*read "E" from text file*/
171:
172: /* While NOT e.o.f., do: */
173: while( !feof(fp1) )
174: {
175: if(cnt == (MAXJOBS ))
176: break;
177:
178: fscanf(fp1, "%s %d %d %f %f ", p1.schd[cnt].id,
179: &p1.schd[cnt].ProcessingTime, &p1.schd[cnt].DueDate,
180: &p1.schd[cnt].EarlinessCost, &p1.schd[cnt].TardinessCost);
```

```
181:
182: cnt++; /* Read next job */
183: }
184:
185: /* Add machine */
186: for (i = 0; i < MACHINES - 1 ; i
187: {
188: strcpy( p1.schd[cnt].id , "*" );
189: cnt++;
190: }
191:
192: /* run 3 time per problem */
193: for (l = 1; l <= LOOP ; l++)
194: {
195:
196: pop[0] = p1;
197: optimal_timing(pop, 0);
198: evaluate( pop, 1 , cnt);
199: shufle_job();
200:
201: gen = 0;
202: improvement = 2.00 ;
203:
204: while ( (gen < MAX_GEN) && ( improvement >= 1.01 ) )
205: {
206: printf ("\nGen : %4d,", gen );
207:
208: /* create offspring (extend population) */
209: crossover();
210: mutation();
```

```
211: reproduction();
212: evaluate(all_pop ,All_POP_SIZE , CHROM_LENGTH);
213:
214: sum_fitness = 0.00;
215: for ( k = 0; k < All_POP_SIZE ; k++)
216: sum_fitness += all_pop[k].Fitness;
217:
218: avg_fitness = sum_fitness/All_POP_SIZE ;
219: stat[gen].avg_Fitness = avg_fitness ;
220:
221: select( gen, sum_fitness );
222:
223: improvement = improve (gen) ;
224:
225: gen_stop = gen ;
226: gen++;
227:
228: }
229:
230:
231: printf ("\n\nGen Stop : %4d", gen_stop );
232:
233: min_cost = best_fit (pop ,POP_SIZE, 1 );
234:
235: } /* end of loop */
236:
237: ftime(&t_current);
238: t_diff = (double)(1000.0 * (t_current.time - t_start.time)
239: + (t_current.millitm - t_start.millitm))/1000;
240:
```

```
241:
242: printf ("\n\n\nIt took %.2lf seconds to run application.\n", t_diff );
243: printf("************************************\n\n" );
244:
245: fprintf(fp4,"Time = %.2lf seconds\n\n", t_diff , FILE4 );
246: fprintf(fp4,"************************************\n\n", FILE4 );
247:
248: } /* end of problem */
249:
250: /* Close file */
251: fclose(fp1);
252: fclose(fp4);
253:
254: printf ("\n**** End of run ****");
255: printf ("\n**** Press any key to EXIT ****");
256: getchar();
257:
258: return 0;
259: } /* end main */
260:
261: /*-------------------------------------------------------------------*/
262: /* function job shuffle. */
263: /*-------------------------------------------------------------------*/
264: void shufle_job()
265: {
266: int i,j,k;
267: population temp;
268: temp = pop[0];
269:
270: randomize();
```

```
271:
272: for (k = 1; k < POP_SIZE; k++)
273: {
274: pop[k] = pop[k-1];
275:
276: for (i = 0; i < CHROM_LENGTH; i++)
277: {
278: j = rand() % CHROM_LENGTH;
279: // swap the job at the current position with
280: // the one at the randomly selected position
281: temp.schd[i] = pop[k].schd[i];
282: pop[k].schd[i] = pop[k].schd[j];
283: pop[k].schd[j] = temp.schd[i];
284: } /* end of for (i) */
285:
286: optimal_timing(pop, k);
287: } /* end of for (k) */
288:
289: evaluate(pop ,POP_SIZE , CHROM_LENGTH);
290:
291: } /* end of shufle_job */
292:
293: /*-------------------------------------------------------------------*/
294: /* function Optimal timing algorithm for multi machines. */
295: /*-------------------------------------------------------------------*/
296: void optimal_timing( population temp[], int k)
297: {
298: int m,a,b,c;
299: int start;
300: int machine;
```

```
301:
302: machine = 0 ; // machine 1
303: start = 0; // start chromosome (job)
304:
305: for ( m = 0; m < CHROM_LENGTH ; m++)
306: {
307: a = strcmp (temp[k].schd[m].id , "*");
308: b = strcmp (temp[k].schd[m - 1].id , "*");
309: c = strcmp (temp[k].schd[m - 2].id , "*");
310:
311: if ( m == 0 )
312: { if ( a == 0)
313: { empty_machine ( temp , k, m);
314: machine++ ;
315: start = m + 1 ; }
316: }
317:
318: else if ( m == 1 )
319: { if ( a == 0)
320: { empty_machine ( temp , k, m);
321:
322: { if ( b == 0)
323: start = m + 1 ;
324: else
325: single_job ( temp , k, m - 1, machine );
326: }
327: machine++;
328: start = m + 1 ;
329: }
330: }
```

```
331:
332: else
333: { if ( a == 0)
334: { empty_machine ( temp , k, m);
335: { if ( b == 0)
336: start = m + 1 ;
337: else
338: { if ( c == 0)
339: single_job ( temp , k, m - 1, machine );
340: else
341: opt_single(temp, k, start, m, machine ); }
342: }
343: machine++;
344: start = m + 1 ; }
345: else
346: { if ( m == (CHROM_LENGTH - 1) )
347: { if ( b == 0)
348: single_job ( temp , k, m, machine );
349: else
350: opt_single(temp, k, start, m+1, machine ); }
351: }
352: }
353:
354:
355: } /* end of for (m) */
356: } /* end of optimal_timing */
357:
358: /*-----------------------------------------------------------------*/
359: /* function for sigle job per machine. */
360: /*-----------------------------------------------------------------*/
```

```
361: void single_job ( population temp[], int k, int m , int speed)
362: {
363: temp[k].schd[m].StartingTime =
364: temp[k].schd[m].CompletionTime = 0;
365: temp[k].schd[m].Block = 1;
366: temp[k].schd[m].Order = 1;
367: temp[k].schd[m].SumE = 0;
368: {
369: if ( temp[k].schd[m].DueDate - rate[speed]*temp[k].schd[m].ProcessingTime >0)
370: { temp[k].schd[m].StartingTime = temp[k].schd[m].DueDate
371: - ceil(rate[speed]*temp[k].schd[m].ProcessingTime); }
372:
373: else temp[k].schd[m].StartingTime = 0;}
374: {
375: if ( (rate[speed]*temp[k].schd[m].ProcessingTime) > temp[k].schd[m].DueDate)
376: { temp[k].schd[m].CompletionTime =
377: (rate[speed]*temp[k].schd[m].ProcessingTime); }
378:
379: else temp[k].schd[m].CompletionTime = temp[k].schd[m].DueDate;}
380: }
381:
382: /*-------------------------------------------------------------------*/
383: /* function for empty machine. */
384: /*-------------------------------------------------------------------*/
385: void empty_machine ( population temp[], int k, int m )
386: {
387: temp[k].schd[m].StartingTime = 0 ;
388: temp[k].schd[m].CompletionTime = 0 ;
389: temp[k].schd[m].Block = 0 ;
390: temp[k].schd[m].Order = 0 ;
```

```
391: }
392:
393: /*--------------------------------------------------------------------*/
394: /* function Optimal timing algorithm for single machine. */
395: /*--------------------------------------------------------------------*/
396: void opt_single(population temp[], int k, int s, int l, int speed)
397: {
398: int i ;
399: float a ;
400:
401: temp[k].schd[s].Block = 1;
402: temp[k].schd[s].Order = 1;
403:
404: /* StartingTime = max(0, DueDate - ProcessingTime) */
405: if ( temp[k].schd[s].DueDate - (rate[speed]*temp[k].schd[s].ProcessingTime) > 0)
406: { temp[k].schd[s].StartingTime = (temp[k].schd[s].DueDate
407: - (rate[speed]*temp[k].schd[s].ProcessingTime)); }
408: else
409: temp[k].schd[s].StartingTime = 0;
410:
411: /*CompletionTime = max(DueDate,ProcessingTime) */
412: if ( (rate[speed]*temp[k].schd[s].ProcessingTime) > temp[k].schd[s].DueDate)
413: { temp[k].schd[s].CompletionTime =
414: (rate[speed]*temp[k].schd[s].ProcessingTime); }
415: else
416: temp[k].schd[s].CompletionTime = temp[k].schd[s].DueDate;
417:
418: temp[k].schd[s].SumE = -temp[k].schd[s].EarlinessCost;
419:
420: for ( i = s+1 ; i < l ; i++)
```

```
421: {
422: a = temp[k].schd[i-1].CompletionTime
423: + (rate[speed]*temp[k].schd[i].ProcessingTime) ;
424:
425: /*CompletionTime[i-1] + ProcessingTime[i] < DueDate[i] */
426: if ( a < temp[k].schd[i].DueDate)
427: {temp[k].schd[i].Block = temp[k].schd[i-1].Block +1; /* start new block */
428: temp[k].schd[i].Order = 1;
429: temp[k].schd[i].StartingTime = temp[k].schd[i].DueDate
430: - (rate[speed]*temp[k].schd[i].ProcessingTime);
431: temp[k].schd[i].CompletionTime = temp[k].schd[i].DueDate;
432: temp[k].schd[i].SumE = -temp[k].schd[i].EarlinessCost; }
433:
434: else /* CompletionTime[i-1] + ProcessingTime[i] > DueDate[i] */
435: {
436: if ( a > temp[k].schd[i].DueDate)
437: {temp[k].schd[i].Block = temp[k].schd[i-1].Block;
438: temp[k].schd[i].Order = temp[k].schd[i-1].Order + 1;
439: temp[k].schd[i].StartingTime = temp[k].schd[i-1].CompletionTime;
440: temp[k].schd[i].CompletionTime = temp[k].schd[i].StartingTime
441: + (rate[speed]*temp[k].schd[i].ProcessingTime);
442: temp[k].schd[i].SumE = temp[k].schd[i-1].SumE
443: - temp[k].schd[i].EarlinessCost; }
444: else /* CompletionTime[i-1] + ProcessingTime[i] = DueDate[i] */
445: {
446: temp[k].schd[i].Block = temp[k].schd[i-1].Block;
447: temp[k].schd[i].Order = temp[k].schd[i-1].Order + 1;
448: temp[k].schd[i].StartingTime = temp[k].schd[i-1].CompletionTime;
449: temp[k].schd[i].CompletionTime = temp[k].schd[i].DueDate;
450: temp[k].schd[i].SumE = temp[k].schd[i-1].SumE
```

```
451: - temp[k].schd[i].EarlinessCost;
452: }
453:
454: check_con ( temp, k, i, s, speed);
455:
456: }
457: }
458: } /* end of optimal_timing */
459:
460: /*--------------------------------------------------------------------*/
461: /* function for condition check. */
462: /*--------------------------------------------------------------------*/
463: void check_con ( population temp[], int k, int i, int s, int speed)
464: {
465: int b, m ;
466:
467: b = B_First_Check (temp, k , i, s);
468: m = Min_Point (temp, k , i, s);
469:
470: if ( !( b || m ) )
471: { Shift_Block(temp, k , i, s,speed); }
472:
473: } /* end of check_con */
474:
475: /*--------------------------------------------------------------------*/
476: /* function for checking the starting time of first job in the block */
477: /*--------------------------------------------------------------------*/
478: int B_First_Check (population temp[], int k, int i, int s)
479: {
480: int start;
```

```
481:
482: if ( temp[k].schd[i].Block == 1 )
483: start = temp[k].schd[s].StartingTime;
484: else
485: {
486: while (temp[k].schd[i].Order != 1)
487: i-- ;
488:
489: start = temp[k].schd[i].StartingTime;
490: }
491:
492: if ( start == 0)
493: return 1 ;
494: else
495: return 0 ;
496: } /* enf of B_First_Check */
497:
498: /*---------------------------------------------------------------------*/
499: /* function check Block Located good or not ? */
500: /*---------------------------------------------------------------------*/
501: int Min_Point (population temp[], int k, int i, int s)
502: {
503: int slope,r;
504: r = i;
505:
506: slope = temp[k].schd[r].SumE + temp[k].schd[r].EarlinessCost
507: + temp[k].schd[r].TardinessCost ;
508:
509: while (( slope < 0 ) && ( r >= s ))
510: {
```

```
511: r-- ;
512: slope = slope + temp[k].schd[r].EarlinessCost
513: + temp[k].schd[r].TardinessCost ;
514: } /* end while */
515:
516: if ( r != i )
517: { Shift = temp[k].schd[r].CompletionTime - temp[k].schd[r].DueDate;
518: return 0 ;
519: }
520: else
521: return 1 ;
522: } /* end Min_Point */
523:
524: /*-------------------------------------------------------------------*/
525: /* function Shift Block */
526: /*-------------------------------------------------------------------*/
527: void Shift_Block (population temp[], int k, int i, int s, int speed)
528: {
529: int j, l, a ;
530: l = i ;
531:
532: if ( temp[k].schd[i].Block == 1 ) /* if 1 */
533: {if (temp[k].schd[s].StartingTime - Shift >= 0 ) /* if 2 */
534: {
535: temp[k].schd[s].StartingTime = temp[k].schd[s].StartingTime - Shift ;
536: temp[k].schd[s].CompletionTime = temp[k].schd[s].StartingTime
537: + (rate[speed]*temp[k].schd[s].ProcessingTime) ;
538:
539: for ( j = s+1 ; j <= l ; j++ )
540: {
```

```
541: temp[k].schd[j].StartingTime = temp[k].schd[j-1].CompletionTime ;
542: temp[k].schd[j].CompletionTime = temp[k].schd[j].StartingTime
543: + (rate[speed]*temp[k].schd[j].ProcessingTime) ;
544: } /* end of for */
545: } /* end if 2 */
546: } /* end if 1 */
547:
548: else /* else 1 */
549: {
550:
551: if ( temp[k].schd[s].StartingTime - Shift
552: >= temp[k].schd[s-1].CompletionTime ) /* if 4 */
553: {
554: temp[k].schd[s].StartingTime = temp[k].schd[s].StartingTime - Shift ;
555: temp[k].schd[s].CompletionTime = temp[k].schd[s].StartingTime
556: + (rate[speed]*temp[k].schd[s].ProcessingTime) ;
557:
558: for ( j = s+1 ; j <= 1 ; j++ )
559: {
560: temp[k].schd[j].StartingTime = temp[k].schd[j-1].CompletionTime ;
561: temp[k].schd[j].CompletionTime = temp[k].schd[j].StartingTime
562: + (rate[speed]*temp[k].schd[j].ProcessingTime) ;
563: } /* end of for */
564: } /* end if 4 */
565: } /* end of else 1 */
566:
567: } /* end Shift_Block */
568:
569: /*---------------------------------------------------------------------*/
570: /* function to evaluate the cost of schdule. */
```

```
571: /*-------------------------------------------------------------------*/
572: void evaluate (population temp[],int size, int length)
573: {
574: int i, k ;
575: float max_cost ;
576: float min_cost ;
577:
578: /* cost */
579: for ( k = 0; k < size ; k++ )
580: {
581: temp[k].Cost = 0 ;
582: for (i = 0; i < length ; i++ )
583: {
584: if (temp[k].schd[i].DueDate > temp[k].schd[i].CompletionTime)
585: {
586: temp[k].Cost += temp[k].schd[i].EarlinessCost*(temp[k].schd[i].DueDate
587: - temp[k].schd[i].CompletionTime);
588: }
589: else
590: if (temp[k].schd[i].DueDate < temp[k].schd[i].CompletionTime)
591: {
592: temp[k].Cost += temp[k].schd[i].TardinessCost*(temp[k].schd[i].CompletionTime
593: - temp[k].schd[i].DueDate);
594: }
595: else
596: {temp[k].Cost += 0 ;}
597: } /* end for (i) */
598: }/* end for (k) */
599:
600: /* fitness */
```

```
601: max_cost = 0 ;
602: for ( k = 0; k < size ; k++)
603: {
604: if ( temp[k].Cost >= max_cost )
605: { max_cost = temp[k].Cost ; }
606: }
607: for ( k = 0; k < size ; k++)
608: {
609: if ( temp[k].Cost == 0 )
610: temp[k].Fitness = 0.00;
611: else
612: temp[k].Fitness = (double)max_cost/temp[k].Cost;
613: }
614:
615: } /* end of evaluate */
616:
617: /*--------------------------------------------------------------------*/
618: /* function to crossover. */
619: /*--------------------------------------------------------------------*/
620: void crossover ()
621: {
622: int i, j, temp, k, x, a, b;
623: int parent1 , parent2 ;
624: int child1, child2 ;
625: int m_cnt;
626: int c_cnt;
627: double prob;
628:
629: population bitmask;
630:
```

```
631: c_cnt = 0 ;
632:
633: while ( c_cnt < C_OFF_SIZE )
634: {
635: prob =((double)rand())/RAND_MAX;
636:
637: if ( prob<pCross )
638: {
639: parent1 = rand()%(POP_SIZE);
640: parent2 = rand()%(POP_SIZE);
641:
642: while ( parent2 == parent1 )
643: parent2 = rand()%(POP_SIZE);
644:
645: child1 = c_cnt ;
646:
647: /* creat bitmask */
648: for (i = 0; i < CHROM_LENGTH ; i++)
649: {
650: x = rand()%2;
651: strcpy( bitmask.schd[i].id , "0");
652:
653: if ( x == 1)
654: strcpy( bitmask.schd[i].id , "1");
655:
656: } /* end of for (i) */
657:
658: /* child 1 */
659:
660: for (i = 0; i < CHROM_LENGTH ; i++)
```

```
661: {
662: if ( (strcmp (bitmask.schd[i].id , "1") == 0 ))
663: { c_offspring[child1].schd[i] = pop[parent1].schd[i]; }
664:
665: } /* end of for (i) */
666:
667: /* child 1 */
668: for (k = 0; k < CHROM_LENGTH ; k++) /* bitmask */
669: { m_cnt = 0;
670: if ((strcmp (bitmask.schd[k].id , "0") == 0 ))
671: {
672: for (j = CHROM_LENGTH -1 ; j >= 0 ; j--) /* parent2 */
673: {
674: for (i = 0 ; i < CHROM_LENGTH ; i++) /* child1 */
675: {
676: a = strcmp (pop[parent2].schd[j].id, "*") ;
677: b = strcmp (pop[parent2].schd[j].id, c_offspring[child1].schd[i].id) ;
678:
679: if ( a == 0 )
680: {
681: if ( b == 0 )
682: m_cnt = m_cnt + 1;
683:
684: if ( m_cnt >= (MACHINES -1 ) )
685: break ;
686: else
687: {
688: if ( i == (CHROM_LENGTH - 1) )
689: {c_offspring[child1].schd[k] = pop[parent2].schd[j];
690: break ;}
```

```
691: }
692: }
693: else /* else 1 */
694: {
695: if ( b == 0)
696: break ;
697: else /* else 2 */
698: {
699: if ( i == (CHROM_LENGTH - 1) )
700: { c_offspring[child1].schd[k] = pop[parent2].schd[j];
701: break ;}
702:
703: }/* end else 2 */
704: }/* end else 1 */
705: } /* end of while (i) child1 */
706: } /* end of for (j) parent2 */
707: }/* end of if (compair with "0") */
708: }/* end of for (k) bitmask */
709:
710: optimal_timing(c_offspring, child1);
711:
712: c_cnt++;
713: child2 = c_cnt ;
714:
715: if ( c_cnt++ < C_OFF_SIZE )
716: {
717: /* child 2 */
718: for (i = 0; i < CHROM_LENGTH ; i++)
719: {
720: if ( (strcmp (bitmask.schd[i].id , "0") == 0 ))
```

```
721: { c_offspring[child2].schd[i] = pop[parent2].schd[i]; }
722: } /* end of for (i) */
723:
724:
725: for (k = 0; k < CHROM_LENGTH ; k++) /* bitmask */
726: { m_cnt = 0;
727: if ((strcmp (bitmask.schd[k].id , "1") == 0 ))
728: {
729: for (j = CHROM_LENGTH -1 ; j >= 0 ; j--) /* parent2 */
730: {
731: for (i = 0 ; i < CHROM_LENGTH ; i++) /* child1 */
732: {
733: a = strcmp (pop[parent1].schd[j].id, "*") ;
734: b = strcmp (pop[parent1].schd[j].id, c_offspring[child2].schd[i].id) ;
735:
736: if ( a == 0 )
737: {
738: if ( b == 0 )
739: m_cnt = m_cnt + 1;
740:
741: if ( m_cnt >= (MACHINES -1 ) )
742: break ;
743: else
744: {
745: if ( i == (CHROM_LENGTH - 1) )
746: {c_offspring[child2].schd[k] = pop[parent1].schd[j];
747: break ;}
748: }
749: }
750: else /* else 1 */
```

```
751: {
752: if ( b == 0)
753: break ;
754: else /* else 2 */
755: {
756: if ( i == (CHROM_LENGTH - 1) )
757: { c_offspring[child2].schd[k] = pop[parent1].schd[j];
758: break ;}
759:
760: }/* end else 2 */
761: }/* end else 1 */
762: } /* end of while (i) child2 */
763: } /* end of for (j) parent1 */
764: }/* end of if (compair with "0") */
765: }/* end of for (k) bitmask */
766:
767: optimal_timing(c_offspring, child2);
768: }
769: } /* end of if */
770: }
771:
772: } /* end of crossover */
773:
774: /*-------------------------------------------------------------------*/
775: /* function to mutation. */
776: /*-------------------------------------------------------------------*/
777: void mutation()
778: {
779: int i, j, x_point1, x_point2,temp;
780: int r ;
```

```
781: double prob;
782:
783: population gtemp[M_OFF_SIZE];
784:
785: i = 0 ;
786:
787: while ( i < M_OFF_SIZE )
788: {
789: prob =((double)rand())/RAND_MAX;
790:
791: if ( prob<pMut )
792: {
793: r = (int)rand()%POP_SIZE;
794: m_offspring[i] = pop[r];
795:
796: x_point1 = (int)rand()%CHROM_LENGTH;
797: x_point2 = (int)rand()%CHROM_LENGTH;
798:
799: while ( x_point2 == x_point1 )
800: x_point2 = (int)rand()%(CHROM_LENGTH);
801:
802:
803: gtemp[i].schd[0] = m_offspring[i].schd[x_point1];
804: m_offspring[i].schd[x_point1] = m_offspring[i].schd[x_point2];
805: m_offspring[i].schd[x_point2] = gtemp[i].schd[0];
806:
807: optimal_timing(m_offspring, i);
808: i++;
809: } /* end of if (prob) */
810: } /* end of while (i) */
```

```
811:
812: } /* end of mutation */
813:
814: /*-----------------------------------------------------------------------*/
815: /* function to selection. */
816: /*-----------------------------------------------------------------------*/
817: void select(int gen, double sum_fitness)
818: {
819: int a, i, k , m , s,x;
820: float min_cost ;
821: int min_pos ;
822: int elit , replace ;
823: double r, part_sum ;
824:
825: for ( k = 0; k < POP_SIZE ; k++)
826: {
827: part_sum = 0;
828:
829: r = rand()%(int)sum_fitness ; /* spin the roulette*/
830:
831: for ( i=0; i < All_POP_SIZE, part_sum <=r ; i++)
832: {
833: if ( all_pop[i].Fitness != 0 )
834: {
835: m = i ;
836: part_sum += all_pop[i].Fitness ;
837: }
838: } /* end for i */
839: selected_p[k] = m;
840: } /* end for k */
```

```
841:
842: /* find min cost */
843: min_cost = all_pop[0].Cost ;
844: min_pos = 0 ;
845:
846: for ( a = 1; a < All_POP_SIZE ; a++)
847: {
848: if ( all_pop[a].Cost <= min_cost )
849: {
850: min_pos = a ;
851: min_cost = all_pop[a].Cost ;
852: }
853: } /* end for a */
854:
855: stat[gen].min_cost = min_cost ;
856:
857: /* elitist */
858: elit = 1 ;
859: for ( k = 0; k < POP_SIZE ; k++)
860: {
861: if ( selected_p[k] == min_pos )
862: { elit = 0 ; }
863: }
864:
865: if ( elit == 1 )
866: {
867: replace = rand()%(int)POP_SIZE ;
868: selected_p[replace] = min_pos ;
869: }
870:
```

```
871: /* new population */
872: for ( k = 0; k < POP_SIZE ; k++)
873: {
874: i = selected_p[k];
875: pop[k] = all_pop[i];
876: }
877:
878: } /* end of selection */
879:
880: /*--------------------------------------------------------------------*/
881: /* function to reproduce. (Add N parent with n offspring .) */
882: /*--------------------------------------------------------------------*/
883: void reproduction()
884: {
885: int k ;
886:
887: for (k = 0; k < POP_SIZE ; k++)
888: all_pop[k] = pop[k];
889:
890: for (k = 0; k < C_OFF_SIZE ; k++)
891: all_pop[POP_SIZE + k] = c_offspring[k];
892:
893: for (k = 0; k < M_OFF_SIZE ; k++)
894: all_pop[POP_SIZE + C_OFF_SIZE + k] = m_offspring[k];
895:
896: for (k = 0; k < All_POP_SIZE ; k++)
897: {
898: all_pop[k].Cost = 0 ;
899: all_pop[k].Fitness = 0;
900: }
```

```
901:
902: } /* end of reproduction */
903:
904: /*--------------------------------------------------------------------*/
905: /* function to find the best cost of schdule. */
906: /*--------------------------------------------------------------------*/
907: int best_fit (population temp[],int size, int l )
908: {
909: int i, m, k ;
910: float min_cost ;
911: int mc;
912:
913: min_cost = temp[0].Cost ;
914:
915: if ( l == 1 )
916: { BEST_COST = min_cost;
917: BEST = temp[0];}
918:
919: m = 0 ;
920:
921: for ( k = 1; k < size ; k++)
922: {
923: if ( temp[k].Cost < min_cost )
924: { min_cost = temp[k].Cost;
925: m = k;}
926: }
927:
928: if ( min_cost < BEST_COST )
929: { BEST_COST = min_cost;
930: BEST = temp[m];}
```

```
931:
932: if ( l == LOOP )
933: {
934: mc = 0 ;
935:
936: fprintf(fp4,"M/C\tID\tStart\tProcess\tComplete\tDue Date\n",FILE4);
937: fprintf(fp4,"----------------------------------------\n",FILE4);
938: for (i = 0; i < CHROM_LENGTH; i++)
939: {
940: if ( strcmp (BEST.schd[i].id, "*") == 0 )
941: { mc++;
942: fprintf(fp4,"----------------------------------------\n",FILE4);
943: }
944: else
945: {
946: fprintf(fp4,"%d\t%s\t%.f\t%.f\t%.f\t\t%d\n",mc+1,
947: BEST.schd[i].id, BEST.schd[i].StartingTime,
948: (rate[mc]*BEST.schd[i].ProcessingTime),
949: BEST.schd[i].CompletionTime, BEST.schd[i].DueDate,
950: FILE4);
951: }
952: }
953: fprintf(fp4,"----------------------------------------\n\n",FILE4);
954:
955: fprintf(fp4,"Gen Stop = %d\n", gen , FILE4 );
956: fprintf(fp4,"Cost = %.2f\n", BEST_COST , FILE4 );
957:
958: }
959:
960: return (0);
```

```
961:
962: } /* end of best_loop */
963:
964: /*--------------------------------------------------------------------*/
965: /* function for improvement of schdule. */
966: /*--------------------------------------------------------------------*/
967: double improve (gen)
968: {
969: int i ;
970: double sum_cost, avg_cost ;
971: double standard_dev ;
972: double c, improve ;
973:
974: sum_cost = 0.00 ;
975: avg_cost = 0.00 ;
976: standard_dev = 0.00 ;
977: improve = 0.00 ;
978: c = 0.00 ;
979:
980: if ( stat[gen].min_cost == 0 )
981: { return 0 ; }
982: else
983: {
984: if ( gen < 500 )
985: { return 5 ; }
986: else
987: {
988: for ( i = gen -299 ; i <= gen ; i++ )
989: sum_cost += (double)stat[i].min_cost ;
990:
```

```
991: avg_cost = (double)sum_cost/(double)300 ;
992:
993: for ( i = gen -299 ; i <= gen ; i++ )
994: c += pow ( (double)stat[i].min_cost - avg_cost , 2 ) ;
995:
996: standard_dev = sqrt ( c / (double)299 ) ;
997:
998: improve = ( (double)stat[gen].min_cost + standard_dev )
999: / (double)stat[gen].min_cost ;
1000:
1001: return ( improve );
1002:
1003: }
1004: }
1005: }
1006:
```

ภาคผนวก ง

ตัวอย่างปัญหาและผลการใช้โปรแกรม GAOPT

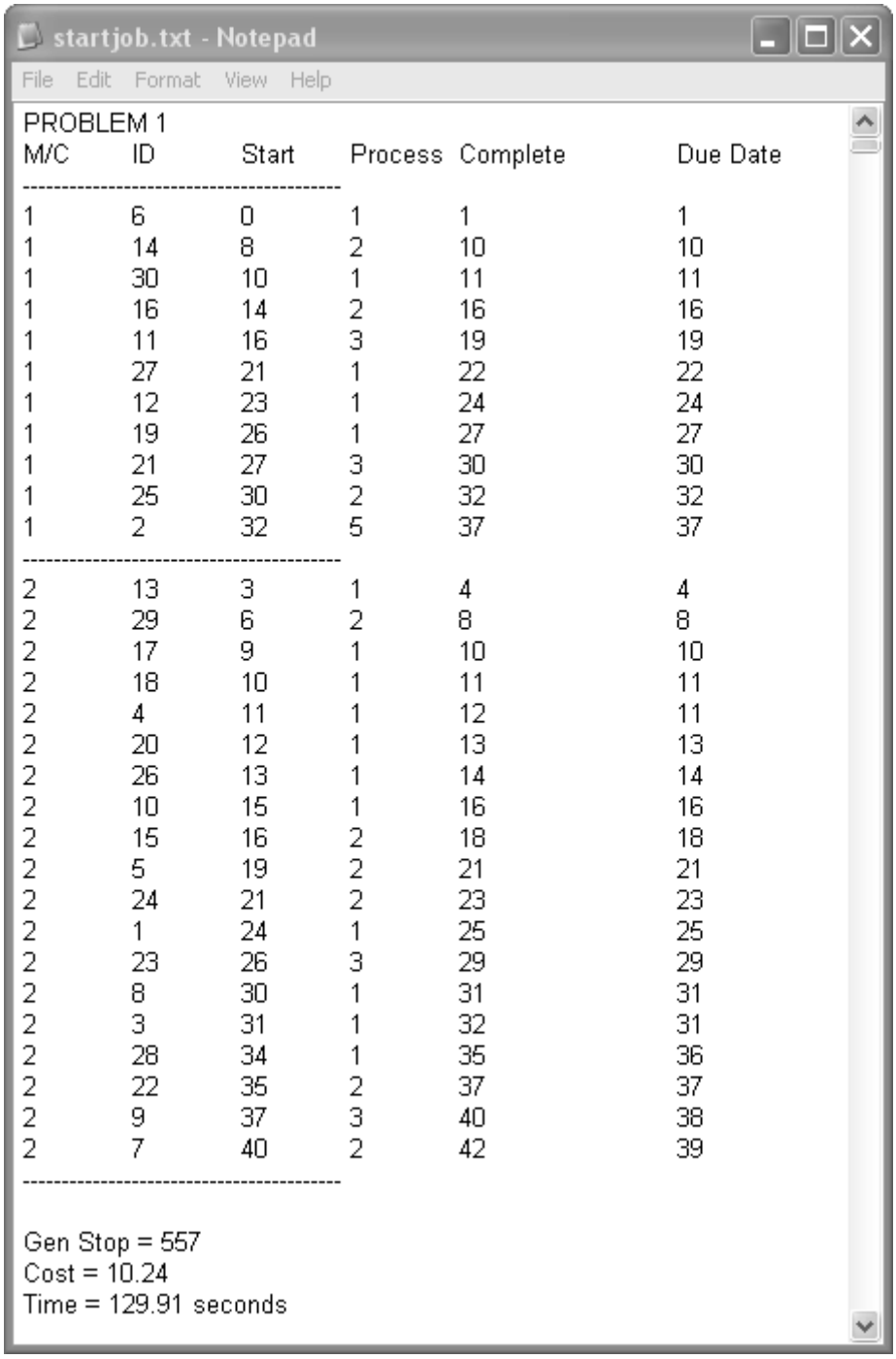
ตัวอย่างปัญหาและผลการใช้โปรแกรม GAOPT

job.txt - Notepad

File Edit Format View Help

```
speed    1        2
PROBLEM1
ID       Process  DueDate  E        T
1        1        25       7.20     3.60
2        5        37       7.60     3.80
3        2        31       0.53     0.26
4        2        11       2.86     1.43
5        3        21       4.25     2.13
6        1        1        7.69     3.85
7        3        39       1.07     0.54
8        2        31       9.48     4.74
9        5        38       5.70     2.85
10       2        16       9.14     4.57
11       3        19       1.65     0.82
12       1        24       9.77     4.89
13       1        4        9.94     4.97
14       2        10       3.49     1.75
15       4        18       5.40     2.70
16       2        16       4.88     2.44
17       2        10       1.96     0.98
18       1        11       9.83     4.91
19       1        27       6.62     3.31
20       1        13       8.14     4.07
21       3        30       1.66     0.83
22       3        37       8.71     4.36
23       5        29       3.78     1.89
24       3        23       6.07     3.04
25       2        32       5.79     2.89
26       1        14       4.07     2.04
27       1        22       5.39     2.69
28       1        36       1.23     0.62
29       4        8        7.16     3.58
30       1        11       3.66     1.83
```

ภาพผนวกที่ ง1 ปัญหาที่ 1 ที่ 2 เครื่องจักรและ $\beta = 0.5\alpha$



startjob.txt - Notepad

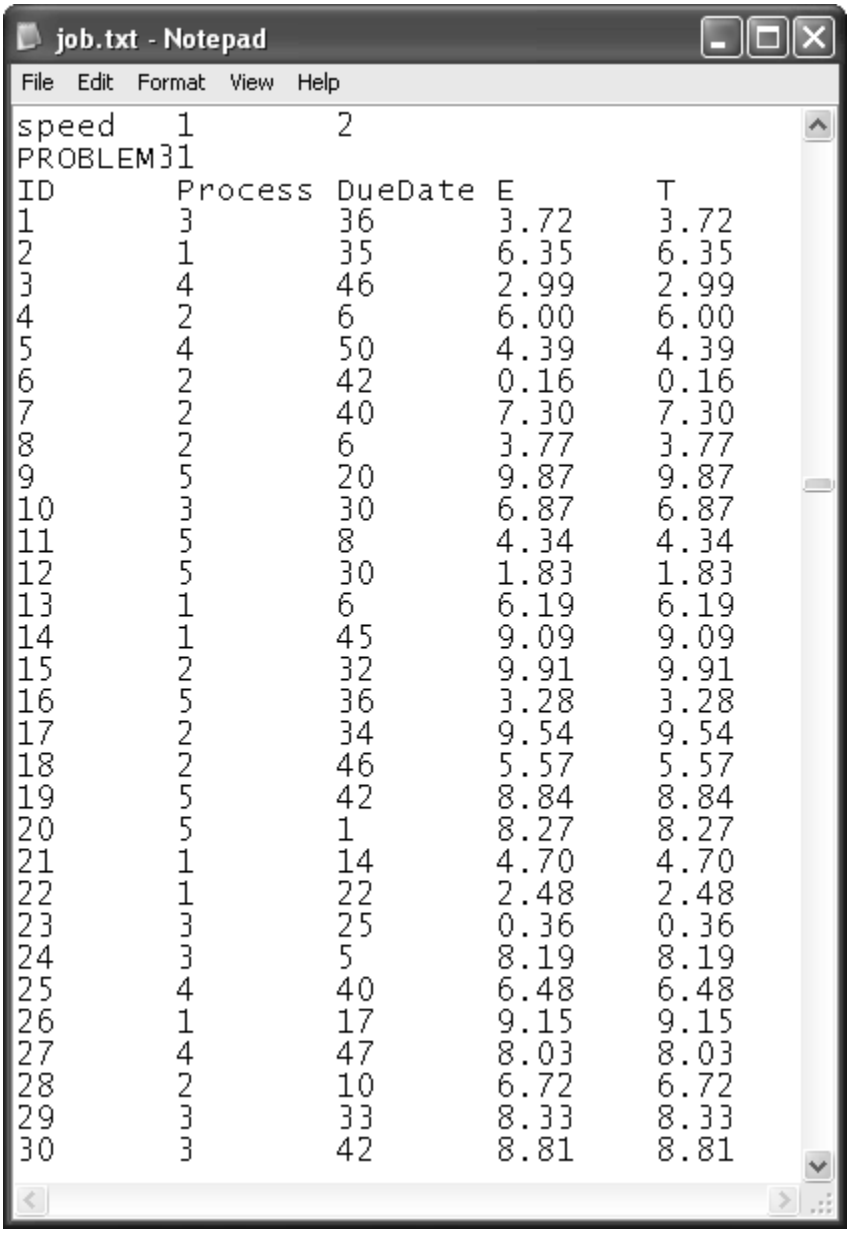
File Edit Format View Help

PROBLEM 1

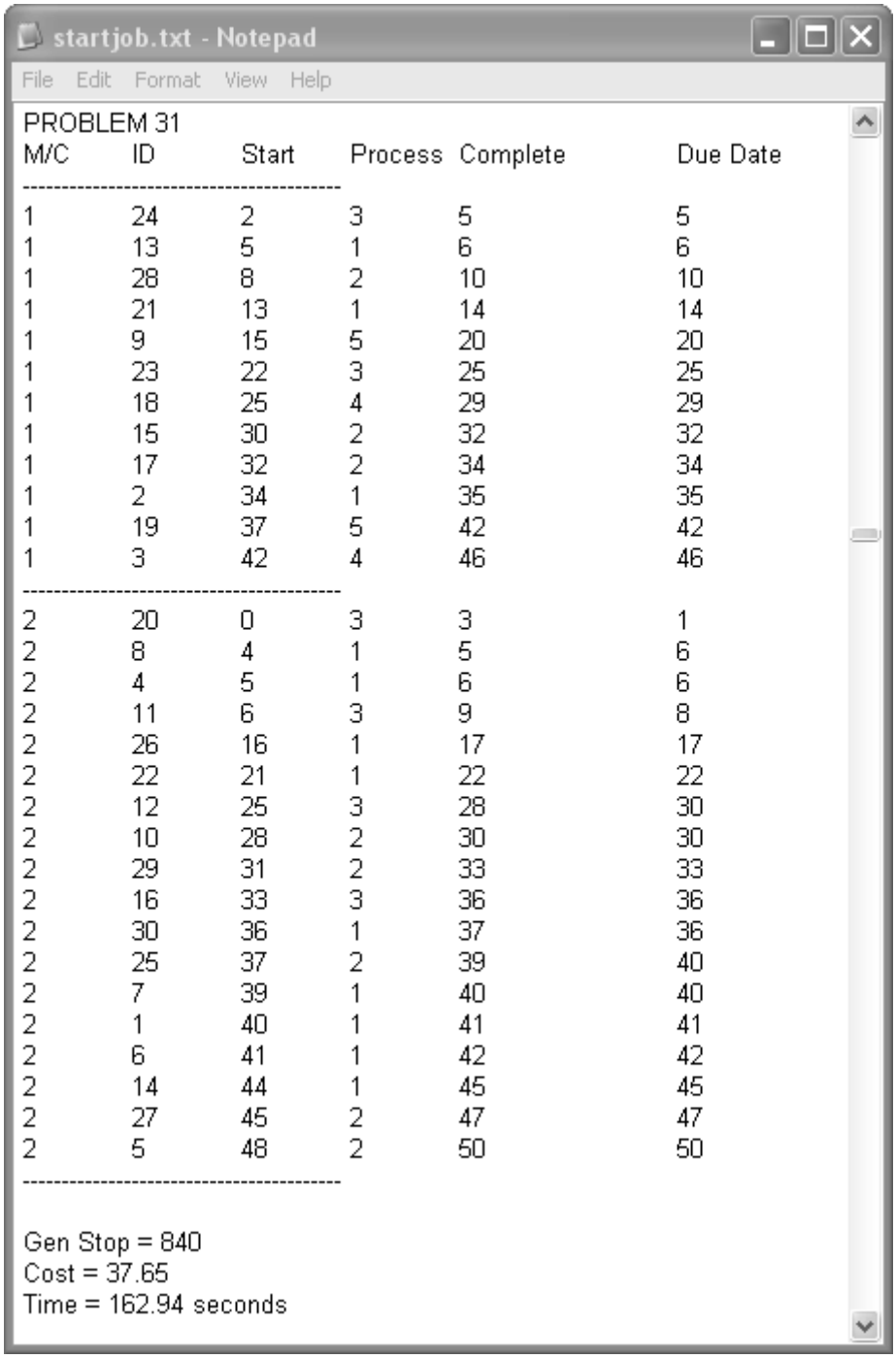
M/C	ID	Start	Process	Complete	Due Date
1	6	0	1	1	1
1	14	8	2	10	10
1	30	10	1	11	11
1	16	14	2	16	16
1	11	16	3	19	19
1	27	21	1	22	22
1	12	23	1	24	24
1	19	26	1	27	27
1	21	27	3	30	30
1	25	30	2	32	32
1	2	32	5	37	37
2	13	3	1	4	4
2	29	6	2	8	8
2	17	9	1	10	10
2	18	10	1	11	11
2	4	11	1	12	11
2	20	12	1	13	13
2	26	13	1	14	14
2	10	15	1	16	16
2	15	16	2	18	18
2	5	19	2	21	21
2	24	21	2	23	23
2	1	24	1	25	25
2	23	26	3	29	29
2	8	30	1	31	31
2	3	31	1	32	31
2	28	34	1	35	36
2	22	35	2	37	37
2	9	37	3	40	38
2	7	40	2	42	39

Gen Stop = 557
Cost = 10.24
Time = 129.91 seconds

ภาพผนวกที่ ง2 ผลการใช้โปรแกรม GAOPT ของปัญหาที่ 1 ที่ 2 เครื่องจักรและ $\beta = 0.5\alpha$



ภาพผนวกที่ ง3 ปัญหาที่ 31 ที่ 2 เครื่องจักรและ $\beta = \alpha$



startjob.txt - Notepad

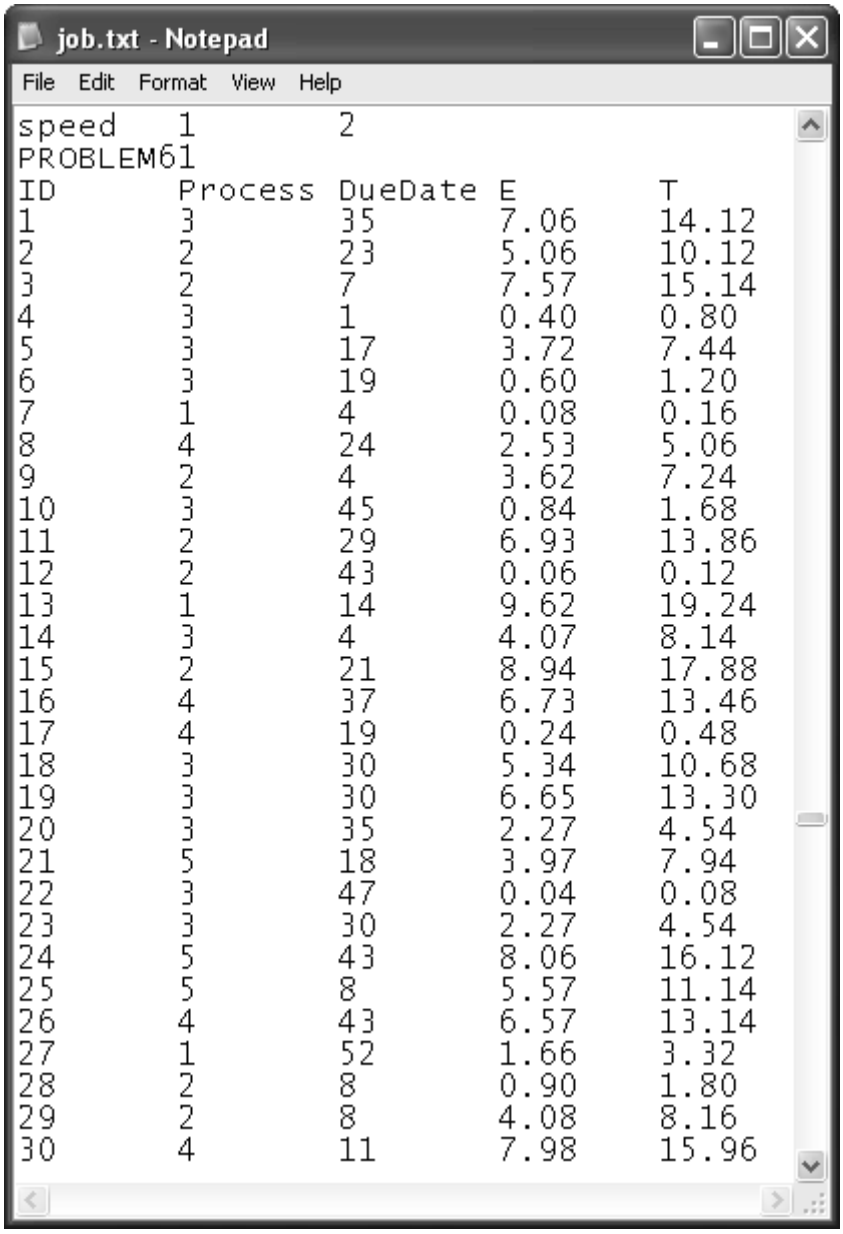
File Edit Format View Help

PROBLEM 31

M/C	ID	Start	Process	Complete	Due Date
1	24	2	3	5	5
1	13	5	1	6	6
1	28	8	2	10	10
1	21	13	1	14	14
1	9	15	5	20	20
1	23	22	3	25	25
1	18	25	4	29	29
1	15	30	2	32	32
1	17	32	2	34	34
1	2	34	1	35	35
1	19	37	5	42	42
1	3	42	4	46	46
2	20	0	3	3	1
2	8	4	1	5	6
2	4	5	1	6	6
2	11	6	3	9	8
2	26	16	1	17	17
2	22	21	1	22	22
2	12	25	3	28	30
2	10	28	2	30	30
2	29	31	2	33	33
2	16	33	3	36	36
2	30	36	1	37	36
2	25	37	2	39	40
2	7	39	1	40	40
2	1	40	1	41	41
2	6	41	1	42	42
2	14	44	1	45	45
2	27	45	2	47	47
2	5	48	2	50	50

Gen Stop = 840
Cost = 37.65
Time = 162.94 seconds

ภาพผนวกที่ ง4 ผลการใช้โปรแกรม GAOPT ของปัญหาที่ 31 ที่ 2 เครื่องจักรและ $\beta = \alpha$

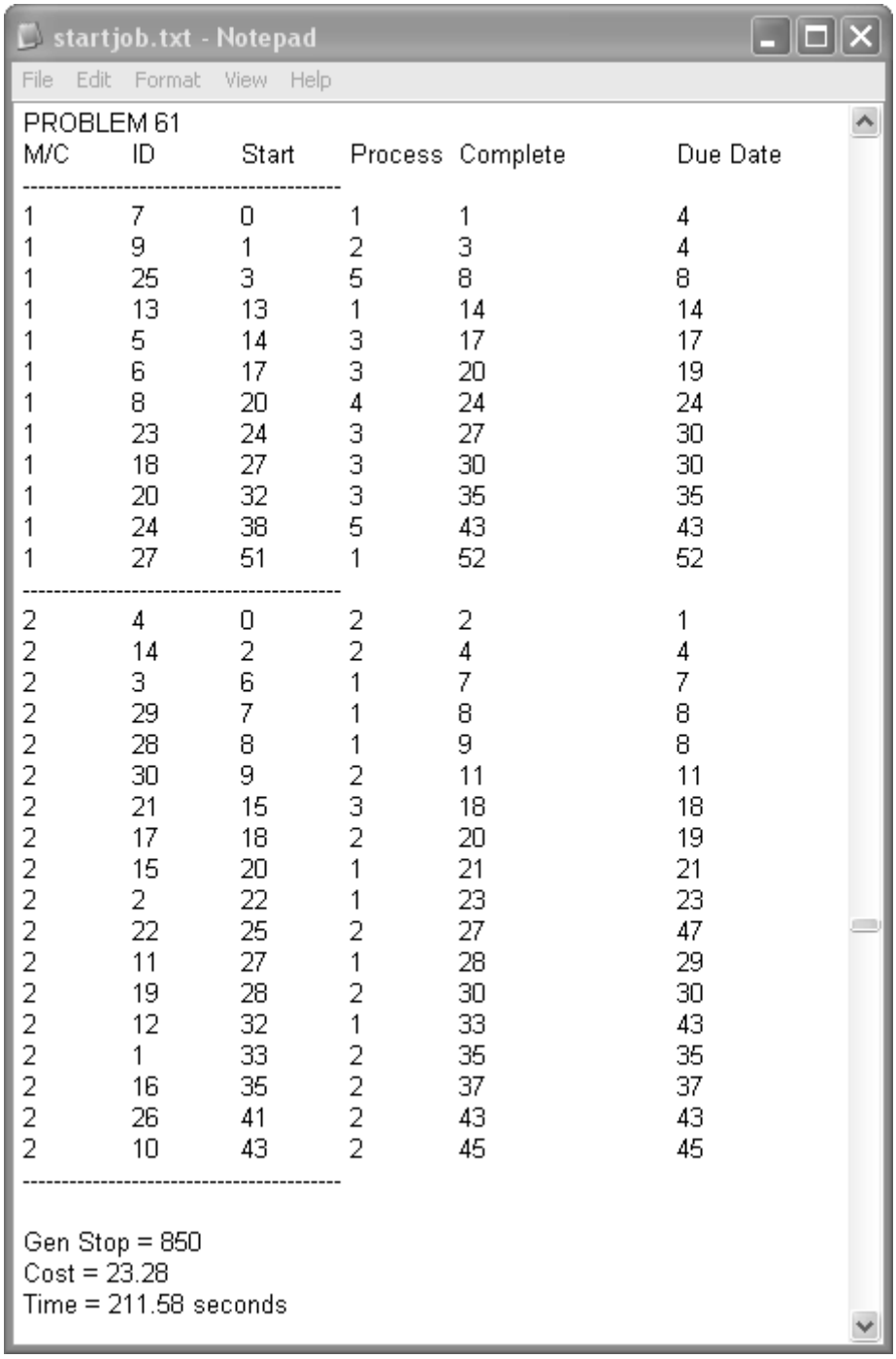


job.txt - Notepad

File Edit Format View Help

```
speed    1        2
PROBLEM61
ID       Process  DueDate  E        T
1        3        35       7.06     14.12
2        2        23       5.06     10.12
3        2        7        7.57     15.14
4        3        1        0.40     0.80
5        3        17       3.72     7.44
6        3        19       0.60     1.20
7        1        4        0.08     0.16
8        4        24       2.53     5.06
9        2        4        3.62     7.24
10       3        45       0.84     1.68
11       2        29       6.93     13.86
12       2        43       0.06     0.12
13       1        14       9.62     19.24
14       3        4        4.07     8.14
15       2        21       8.94     17.88
16       4        37       6.73     13.46
17       4        19       0.24     0.48
18       3        30       5.34     10.68
19       3        30       6.65     13.30
20       3        35       2.27     4.54
21       5        18       3.97     7.94
22       3        47       0.04     0.08
23       3        30       2.27     4.54
24       5        43       8.06     16.12
25       5        8        5.57     11.14
26       4        43       6.57     13.14
27       1        52       1.66     3.32
28       2        8        0.90     1.80
29       2        8        4.08     8.16
30       4        11       7.98     15.96
```

ภาพผนวกที่ ง5 ปัญหาที่ 61 ที่ 2 เครื่องจักรและ $\beta = 2\alpha$



startjob.txt - Notepad

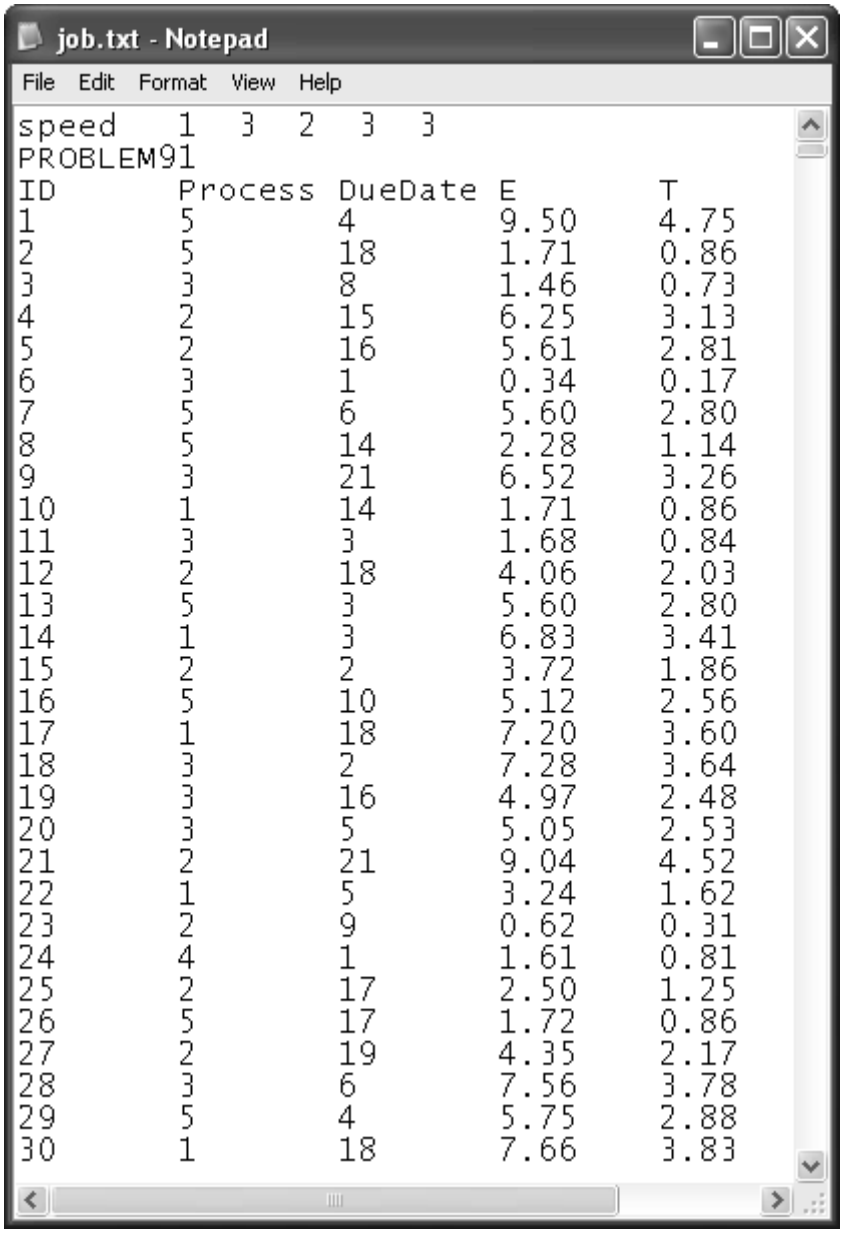
File Edit Format View Help

PROBLEM 61

M/C	ID	Start	Process	Complete	Due Date
1	7	0	1	1	4
1	9	1	2	3	4
1	25	3	5	8	8
1	13	13	1	14	14
1	5	14	3	17	17
1	6	17	3	20	19
1	8	20	4	24	24
1	23	24	3	27	30
1	18	27	3	30	30
1	20	32	3	35	35
1	24	38	5	43	43
1	27	51	1	52	52
2	4	0	2	2	1
2	14	2	2	4	4
2	3	6	1	7	7
2	29	7	1	8	8
2	28	8	1	9	8
2	30	9	2	11	11
2	21	15	3	18	18
2	17	18	2	20	19
2	15	20	1	21	21
2	2	22	1	23	23
2	22	25	2	27	47
2	11	27	1	28	29
2	19	28	2	30	30
2	12	32	1	33	43
2	1	33	2	35	35
2	16	35	2	37	37
2	26	41	2	43	43
2	10	43	2	45	45

Gen Stop = 850
Cost = 23.28
Time = 211.58 seconds

ภาพผนวกที่ ง6 ผลการใช้โปรแกรม GAOPT ของปัญหาที่ 61 ที่ 2 เครื่องจักรและ $\beta = 2\alpha$

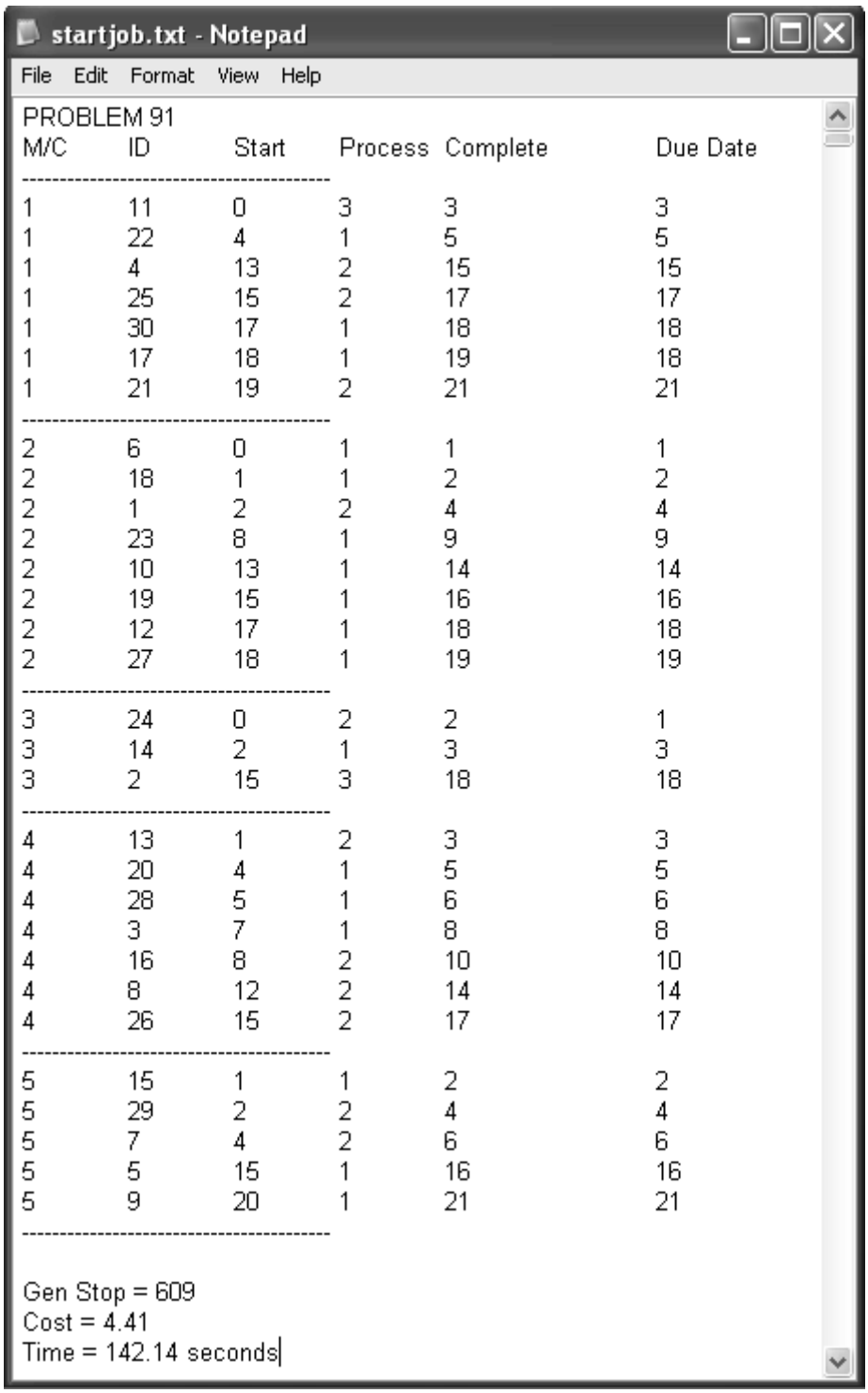


job.txt - Notepad

File Edit Format View Help

```
speed    1   3   2   3   3
PROBLEM91
ID       Process  DueDate  E        T
1        5        4        9.50     4.75
2        5        18       1.71     0.86
3        3        8        1.46     0.73
4        2        15       6.25     3.13
5        2        16       5.61     2.81
6        3        1        0.34     0.17
7        5        6        5.60     2.80
8        5        14       2.28     1.14
9        3        21       6.52     3.26
10       1        14       1.71     0.86
11       3        3        1.68     0.84
12       2        18       4.06     2.03
13       5        3        5.60     2.80
14       1        3        6.83     3.41
15       2        2        3.72     1.86
16       5        10       5.12     2.56
17       1        18       7.20     3.60
18       3        2        7.28     3.64
19       3        16       4.97     2.48
20       3        5        5.05     2.53
21       2        21       9.04     4.52
22       1        5        3.24     1.62
23       2        9        0.62     0.31
24       4        1        1.61     0.81
25       2        17       2.50     1.25
26       5        17       1.72     0.86
27       2        19       4.35     2.17
28       3        6        7.56     3.78
29       5        4        5.75     2.88
30       1        18       7.66     3.83
```

ภาพผนวกที่ ง7 ปัญหาที่ 91 ที่ 5 เครื่องจักรและ $\beta = 0.5\alpha$



startjob.txt - Notepad

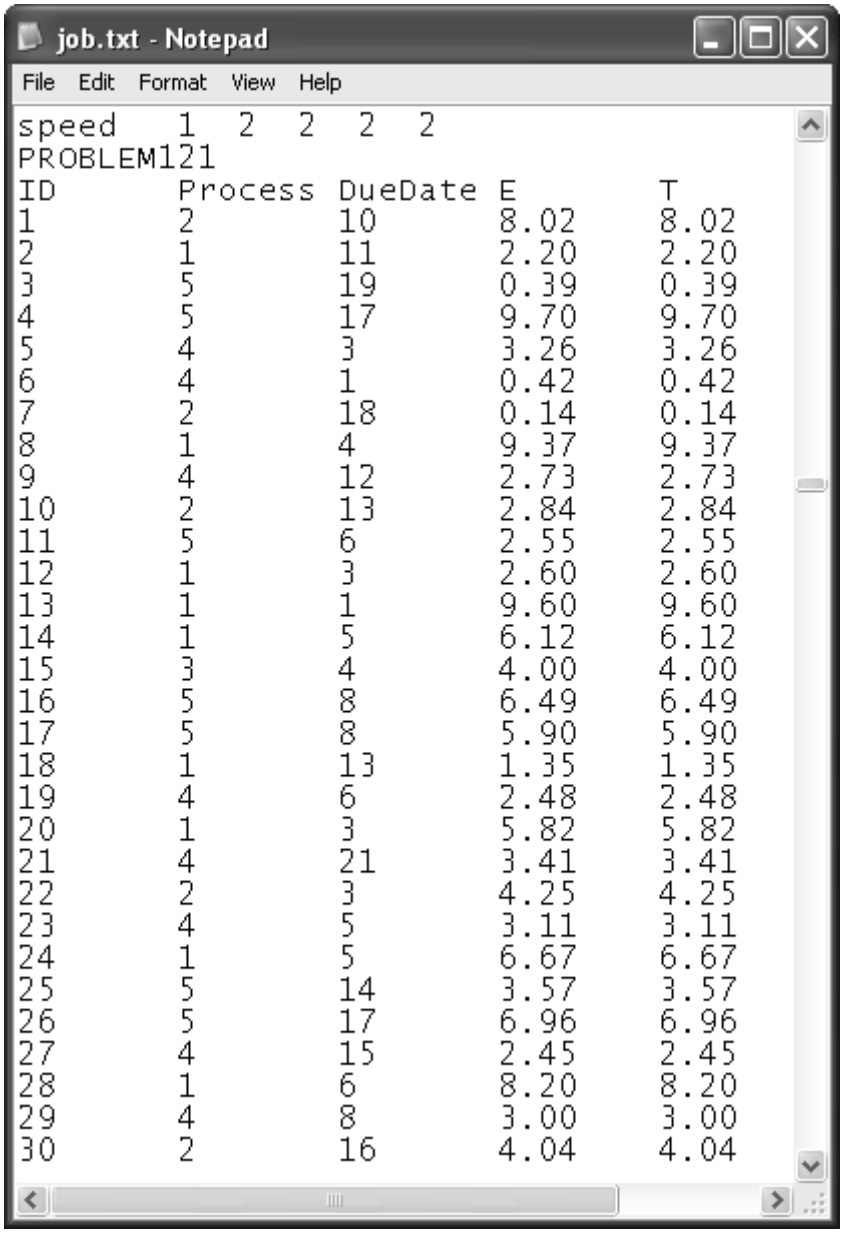
File Edit Format View Help

PROBLEM 91

M/C	ID	Start	Process	Complete	Due Date
1	11	0	3	3	3
1	22	4	1	5	5
1	4	13	2	15	15
1	25	15	2	17	17
1	30	17	1	18	18
1	17	18	1	19	18
1	21	19	2	21	21
2	6	0	1	1	1
2	18	1	1	2	2
2	1	2	2	4	4
2	23	8	1	9	9
2	10	13	1	14	14
2	19	15	1	16	16
2	12	17	1	18	18
2	27	18	1	19	19
3	24	0	2	2	1
3	14	2	1	3	3
3	2	15	3	18	18
4	13	1	2	3	3
4	20	4	1	5	5
4	28	5	1	6	6
4	3	7	1	8	8
4	16	8	2	10	10
4	8	12	2	14	14
4	26	15	2	17	17
5	15	1	1	2	2
5	29	2	2	4	4
5	7	4	2	6	6
5	5	15	1	16	16
5	9	20	1	21	21

Gen Stop = 609
Cost = 4.41
Time = 142.14 seconds

ภาพผนวกที่ ง8 ผลการใช้โปรแกรม GAOPT ของปัญหาที่ 91 ที่ 5 เครื่องจักรและ $\beta = 0.5\alpha$

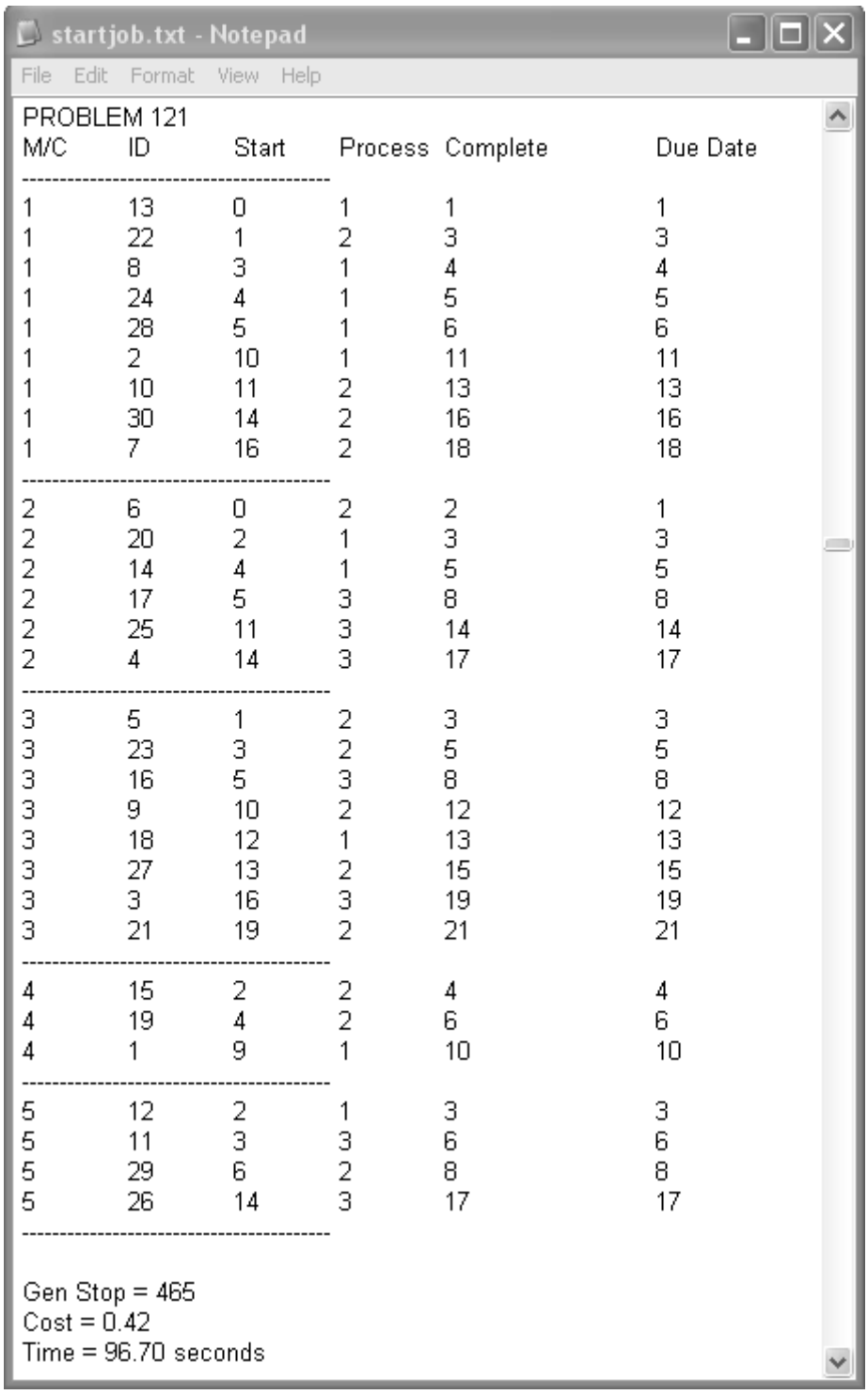


job.txt - Notepad

File Edit Format View Help

```
speed    1  2  2  2  2
PROBLEM121
ID       Process DueDate E        T
1        2       10      8.02     8.02
2        1       11      2.20     2.20
3        5       19      0.39     0.39
4        5       17      9.70     9.70
5        4       3       3.26     3.26
6        4       1       0.42     0.42
7        2       18      0.14     0.14
8        1       4       9.37     9.37
9        4       12      2.73     2.73
10       2       13      2.84     2.84
11       5       6       2.55     2.55
12       1       3       2.60     2.60
13       1       1       9.60     9.60
14       1       5       6.12     6.12
15       3       4       4.00     4.00
16       5       8       6.49     6.49
17       5       8       5.90     5.90
18       1       13      1.35     1.35
19       4       6       2.48     2.48
20       1       3       5.82     5.82
21       4       21      3.41     3.41
22       2       3       4.25     4.25
23       4       5       3.11     3.11
24       1       5       6.67     6.67
25       5       14      3.57     3.57
26       5       17      6.96     6.96
27       4       15      2.45     2.45
28       1       6       8.20     8.20
29       4       8       3.00     3.00
30       2       16      4.04     4.04
```

ภาพผนวกที่ ง9 ปัญหาที่ 121 ที่ 5 เครื่องจักรและ $\beta = \alpha$



startjob.txt - Notepad

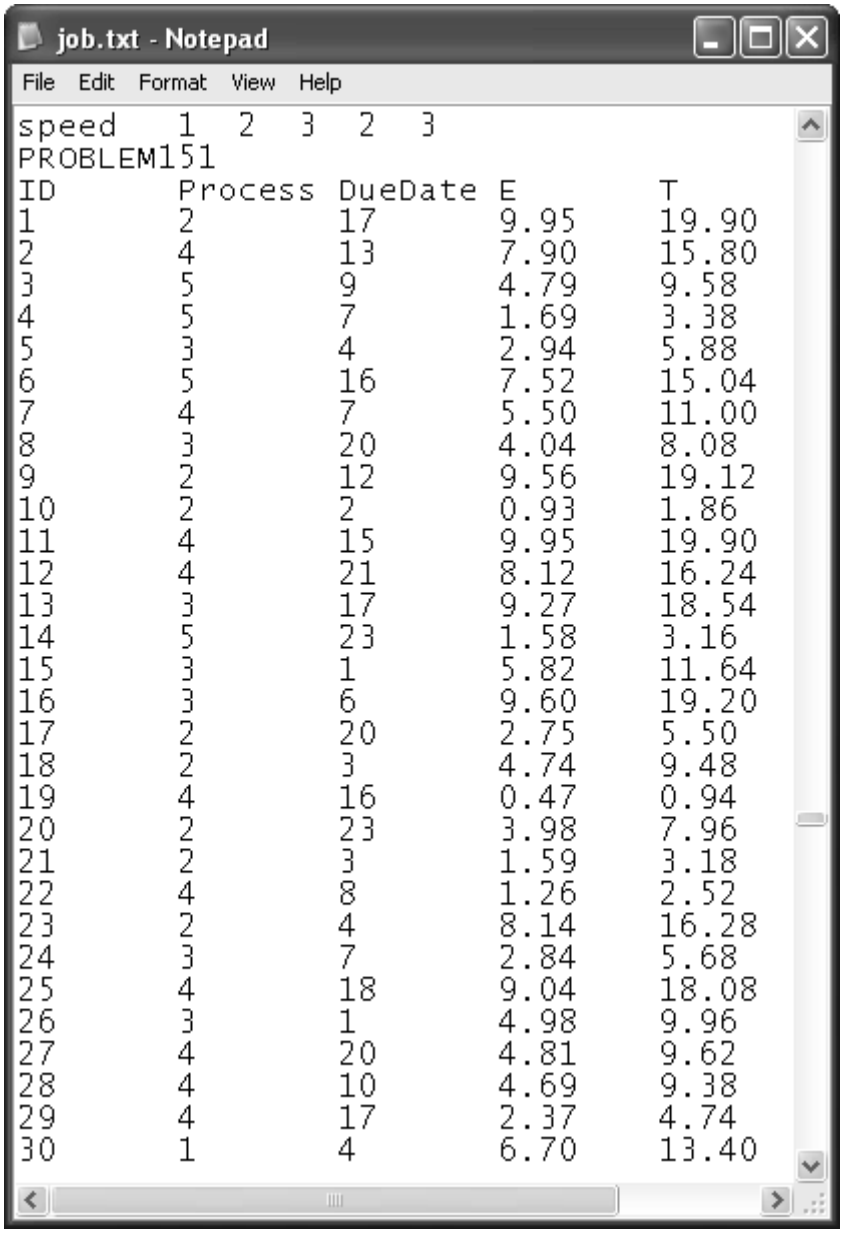
File Edit Format View Help

PROBLEM 121

M/C	ID	Start	Process	Complete	Due Date
1	13	0	1	1	1
1	22	1	2	3	3
1	8	3	1	4	4
1	24	4	1	5	5
1	28	5	1	6	6
1	2	10	1	11	11
1	10	11	2	13	13
1	30	14	2	16	16
1	7	16	2	18	18
2	6	0	2	2	1
2	20	2	1	3	3
2	14	4	1	5	5
2	17	5	3	8	8
2	25	11	3	14	14
2	4	14	3	17	17
3	5	1	2	3	3
3	23	3	2	5	5
3	16	5	3	8	8
3	9	10	2	12	12
3	18	12	1	13	13
3	27	13	2	15	15
3	3	16	3	19	19
3	21	19	2	21	21
4	15	2	2	4	4
4	19	4	2	6	6
4	1	9	1	10	10
5	12	2	1	3	3
5	11	3	3	6	6
5	29	6	2	8	8
5	26	14	3	17	17

Gen Stop = 465
Cost = 0.42
Time = 96.70 seconds

ภาพผนวกที่ ง10 ผลการใช้โปรแกรม GAOPT ของปัญหาที่ 121 ที่ 5 เครื่องจักรและ $\beta = \alpha$

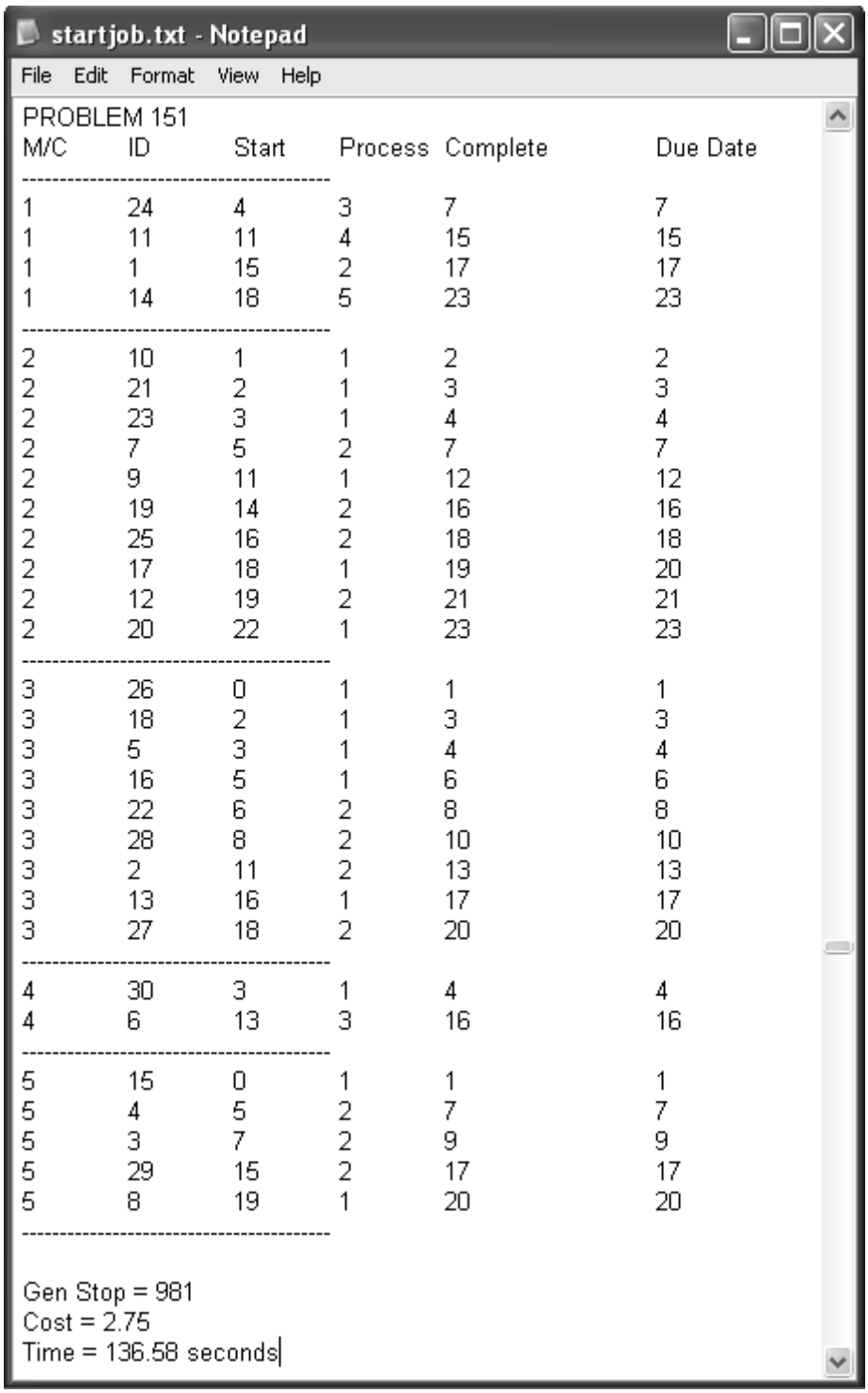


job.txt - Notepad

File Edit Format View Help

```
speed    1  2  3  2  3
PROBLEM151
ID       Process DueDate E        T
1        2       17      9.95     19.90
2        4       13      7.90     15.80
3        5       9       4.79     9.58
4        5       7       1.69     3.38
5        3       4       2.94     5.88
6        5       16      7.52     15.04
7        4       7       5.50     11.00
8        3       20      4.04     8.08
9        2       12      9.56     19.12
10       2       2       0.93     1.86
11       4       15      9.95     19.90
12       4       21      8.12     16.24
13       3       17      9.27     18.54
14       5       23      1.58     3.16
15       3       1       5.82     11.64
16       3       6       9.60     19.20
17       2       20      2.75     5.50
18       2       3       4.74     9.48
19       4       16      0.47     0.94
20       2       23      3.98     7.96
21       2       3       1.59     3.18
22       4       8       1.26     2.52
23       2       4       8.14     16.28
24       3       7       2.84     5.68
25       4       18      9.04     18.08
26       3       1       4.98     9.96
27       4       20      4.81     9.62
28       4       10      4.69     9.38
29       4       17      2.37     4.74
30       1       4       6.70     13.40
```

ภาพผนวกที่ ง11 ปัญหาที่ 151 ที่ 5 เครื่องจักรและ $\beta = 2\alpha$



startjob.txt - Notepad

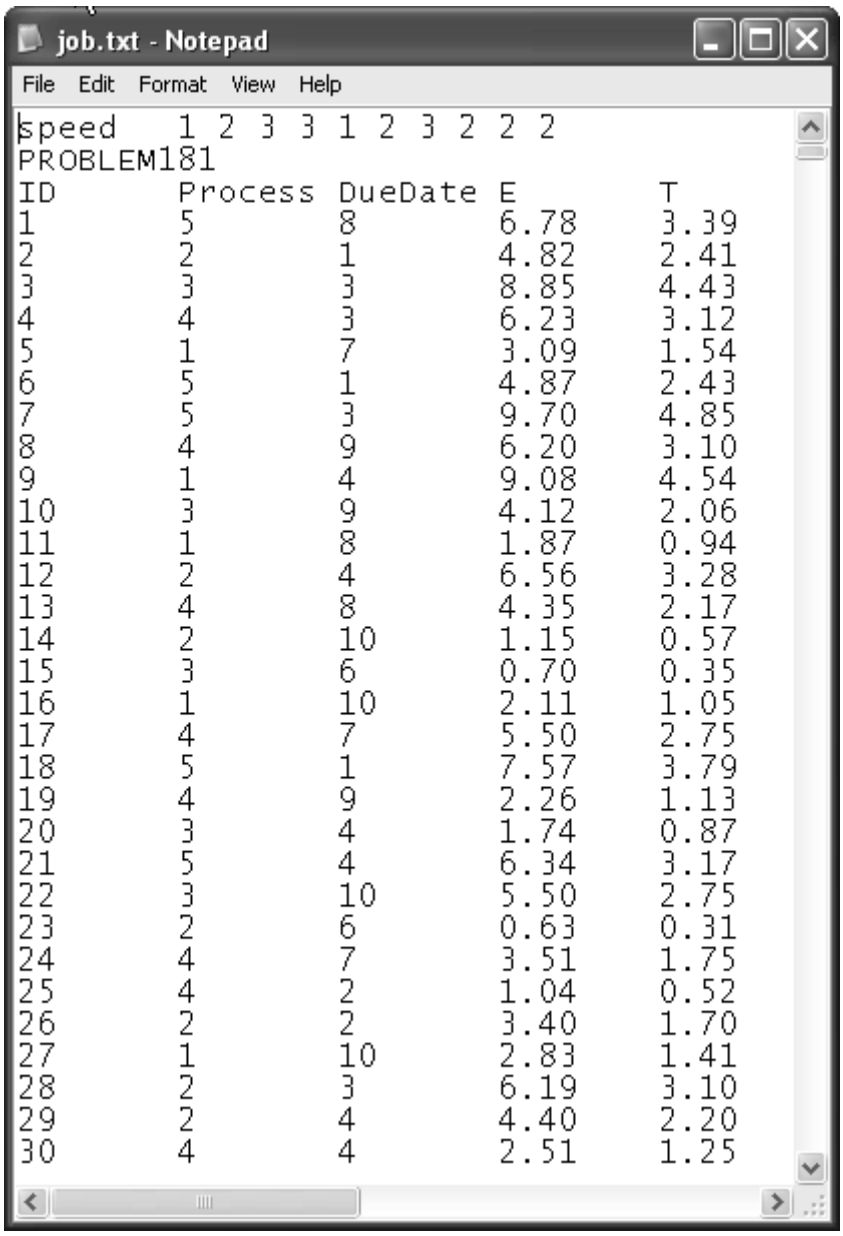
File Edit Format View Help

PROBLEM 151

M/C	ID	Start	Process	Complete	Due Date
1	24	4	3	7	7
1	11	11	4	15	15
1	1	15	2	17	17
1	14	18	5	23	23
2	10	1	1	2	2
2	21	2	1	3	3
2	23	3	1	4	4
2	7	5	2	7	7
2	9	11	1	12	12
2	19	14	2	16	16
2	25	16	2	18	18
2	17	18	1	19	20
2	12	19	2	21	21
2	20	22	1	23	23
3	26	0	1	1	1
3	18	2	1	3	3
3	5	3	1	4	4
3	16	5	1	6	6
3	22	6	2	8	8
3	28	8	2	10	10
3	2	11	2	13	13
3	13	16	1	17	17
3	27	18	2	20	20
4	30	3	1	4	4
4	6	13	3	16	16
5	15	0	1	1	1
5	4	5	2	7	7
5	3	7	2	9	9
5	29	15	2	17	17
5	8	19	1	20	20

Gen Stop = 981
Cost = 2.75
Time = 136.58 seconds

ภาพผนวกที่ ง12 ผลการใช้โปรแกรม GAOPT ของปัญหาที่ 151 ที่ 5 เครื่องจักรและ $\beta = 2\alpha$

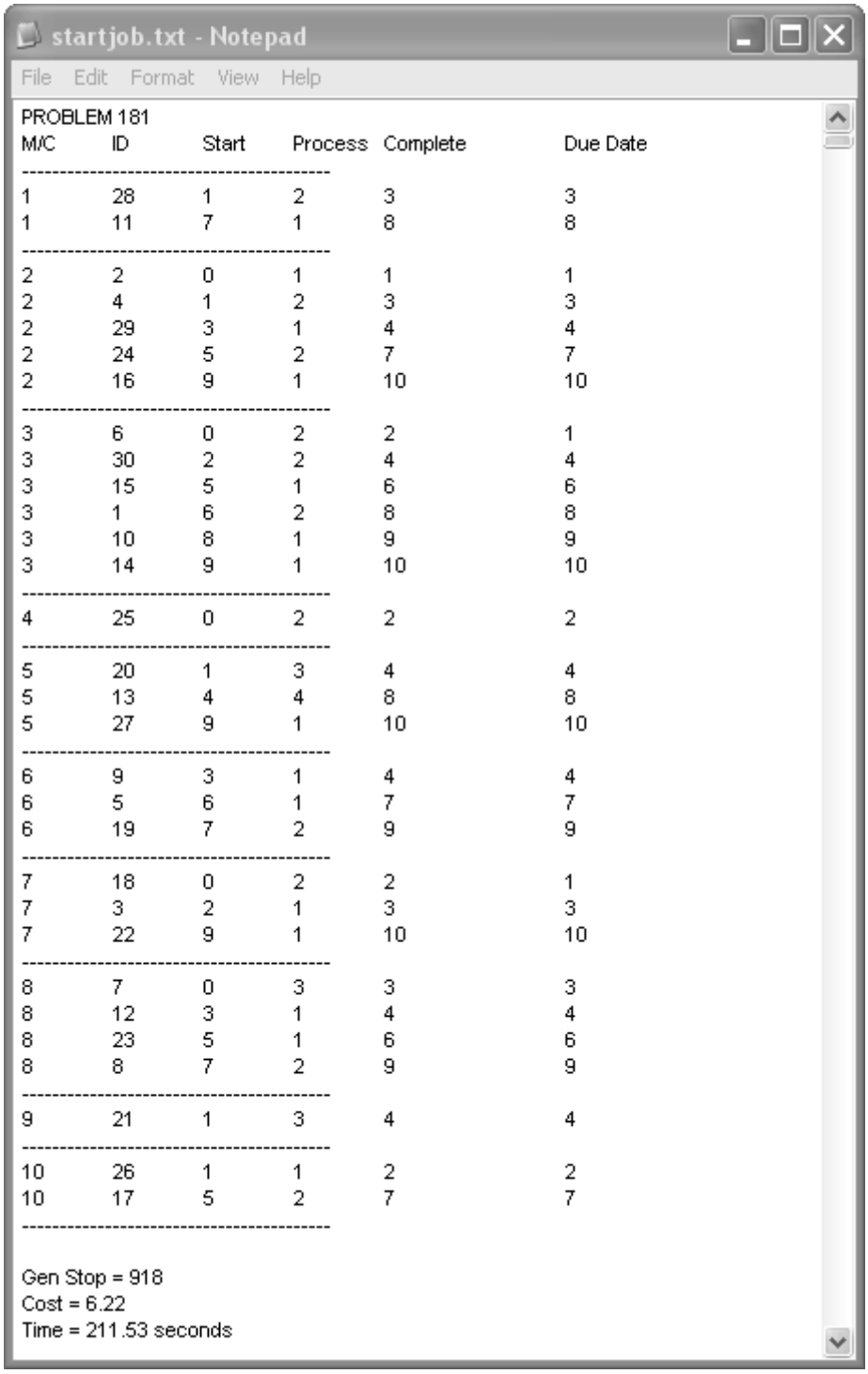


job.txt - Notepad

File Edit Format View Help

```
speed    1 2 3 3 1 2 3 2 2 2
PROBLEM181
ID       Process  DueDate  E        T
1        5        8        6.78     3.39
2        2        1        4.82     2.41
3        3        3        8.85     4.43
4        4        3        6.23     3.12
5        1        7        3.09     1.54
6        5        1        4.87     2.43
7        5        3        9.70     4.85
8        4        9        6.20     3.10
9        1        4        9.08     4.54
10       3        9        4.12     2.06
11       1        8        1.87     0.94
12       2        4        6.56     3.28
13       4        8        4.35     2.17
14       2        10       1.15     0.57
15       3        6        0.70     0.35
16       1        10       2.11     1.05
17       4        7        5.50     2.75
18       5        1        7.57     3.79
19       4        9        2.26     1.13
20       3        4        1.74     0.87
21       5        4        6.34     3.17
22       3        10       5.50     2.75
23       2        6        0.63     0.31
24       4        7        3.51     1.75
25       4        2        1.04     0.52
26       2        2        3.40     1.70
27       1        10       2.83     1.41
28       2        3        6.19     3.10
29       2        4        4.40     2.20
30       4        4        2.51     1.25
```

ภาพผนวกที่ ง13 ปัญหาที่ 181 ที่ 10 เครื่องจักรและ $\beta = 0.5\alpha$



startjob.txt - Notepad

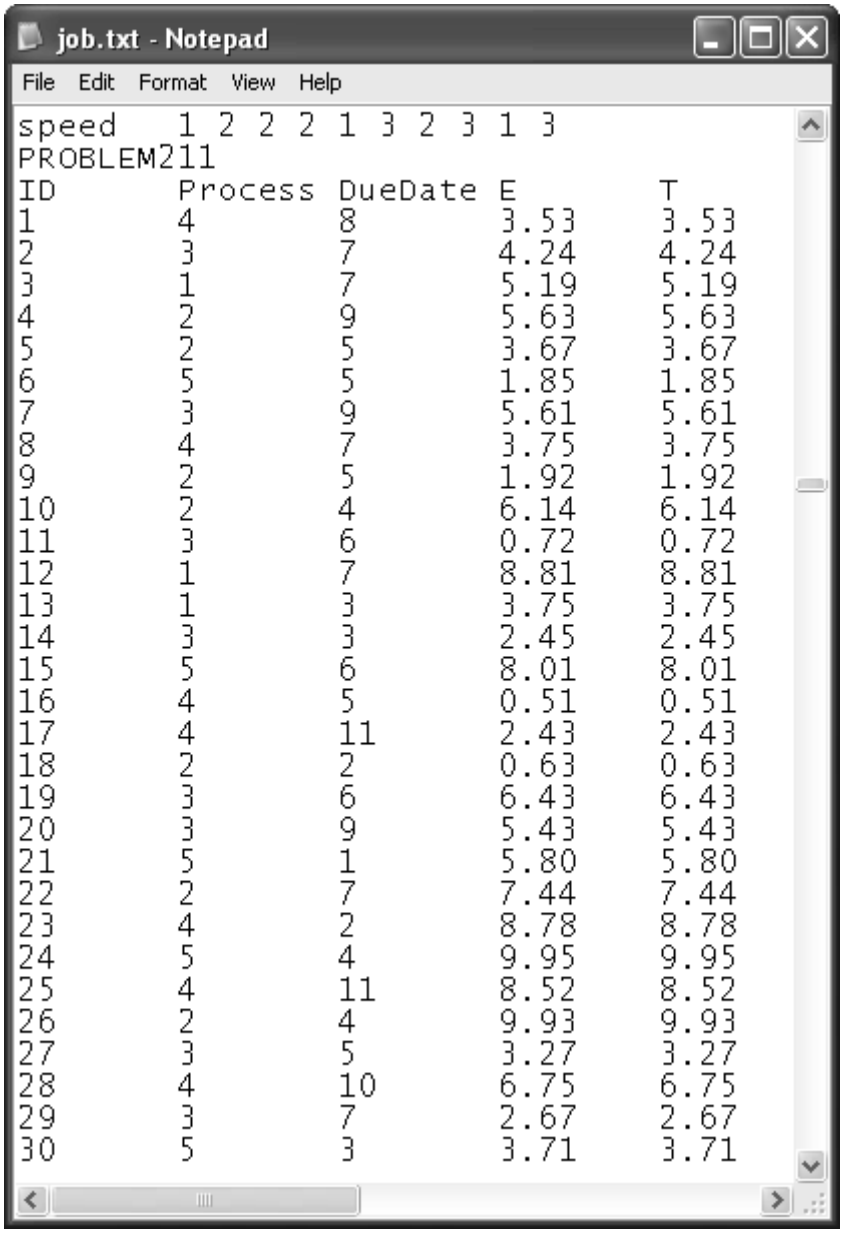
File Edit Format View Help

PROBLEM 181

M/C	ID	Start	Process	Complete	Due Date
1	28	1	2	3	3
1	11	7	1	8	8
2	2	0	1	1	1
2	4	1	2	3	3
2	29	3	1	4	4
2	24	5	2	7	7
2	16	9	1	10	10
3	6	0	2	2	1
3	30	2	2	4	4
3	15	5	1	6	6
3	1	6	2	8	8
3	10	8	1	9	9
3	14	9	1	10	10
4	25	0	2	2	2
5	20	1	3	4	4
5	13	4	4	8	8
5	27	9	1	10	10
6	9	3	1	4	4
6	5	6	1	7	7
6	19	7	2	9	9
7	18	0	2	2	1
7	3	2	1	3	3
7	22	9	1	10	10
8	7	0	3	3	3
8	12	3	1	4	4
8	23	5	1	6	6
8	8	7	2	9	9
9	21	1	3	4	4
10	26	1	1	2	2
10	17	5	2	7	7

Gen Stop = 918
Cost = 6.22
Time = 211.53 seconds

ภาพผนวกที่ ง14 ผลการใช้โปรแกรม GAOPT ของปัญหาที่ 181 ที่ 10 เครื่องจักรและ $\beta = 0.5\alpha$



job.txt - Notepad

File Edit Format View Help

```
speed    1 2 2 2 1 3 2 3 1 3
PROBLEM211
ID       Process DueDate E        T
1        4       8       3.53     3.53
2        3       7       4.24     4.24
3        1       7       5.19     5.19
4        2       9       5.63     5.63
5        2       5       3.67     3.67
6        5       5       1.85     1.85
7        3       9       5.61     5.61
8        4       7       3.75     3.75
9        2       5       1.92     1.92
10       2       4       6.14     6.14
11       3       6       0.72     0.72
12       1       7       8.81     8.81
13       1       3       3.75     3.75
14       3       3       2.45     2.45
15       5       6       8.01     8.01
16       4       5       0.51     0.51
17       4       11      2.43     2.43
18       2       2       0.63     0.63
19       3       6       6.43     6.43
20       3       9       5.43     5.43
21       5       1       5.80     5.80
22       2       7       7.44     7.44
23       4       2       8.78     8.78
24       5       4       9.95     9.95
25       4       11      8.52     8.52
26       2       4       9.93     9.93
27       3       5       3.27     3.27
28       4       10      6.75     6.75
29       3       7       2.67     2.67
30       5       3       3.71     3.71
```

ภาพผนวกที่ ง15 ปัญหาที่ 211 ที่ 10 เครื่องจักรและ $\beta = \alpha$

startjob.txt - Notepad

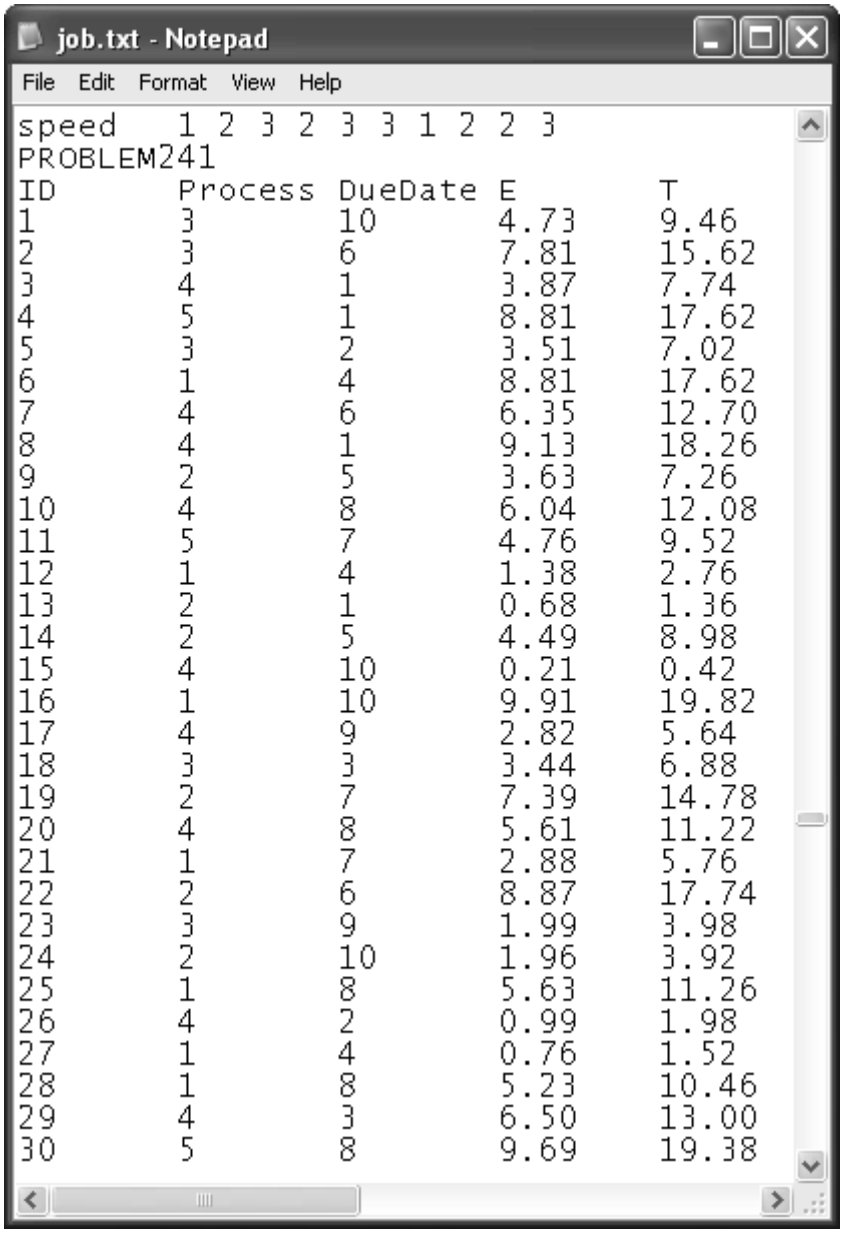
File Edit Format View Help

PROBLEM 211

M/C	ID	Start	Process	Complete	Due Date
1	13	2	1	3	3
1	19	3	3	6	6
1	12	6	1	7	7
1	17	7	4	11	11
2	23	0	2	2	2
2	10	3	1	4	4
3	16	3	2	5	5
3	2	5	2	7	7
3	7	7	2	9	9
4	24	1	3	4	4
4	9	4	1	5	5
4	29	5	2	7	7
4	11	8	3	11	11
5	6	0	5	5	5
5	22	5	2	7	7
5	4	7	2	9	9
6	30	1	2	3	3
6	15	4	2	6	6
6	28	8	2	10	10
7	26	3	1	4	4
7	8	5	2	7	7
8	3	1	1	2	2
8	14	2	1	3	3
8	20	8	1	9	9
9	18	0	2	2	2
9	5	3	2	5	5
10	21	0	2	2	1
10	27	4	1	5	5
10	1	6	2	8	8
10	25	9	2	11	11

Gen Stop = 528
Cost = 5.80
Time = 154.53 seconds

ภาพผนวกที่ ง16 ผลการใช้โปรแกรม GAOPT ของปัญหาที่ 211 ที่ 10 เครื่องจักรและ $\beta = \alpha$

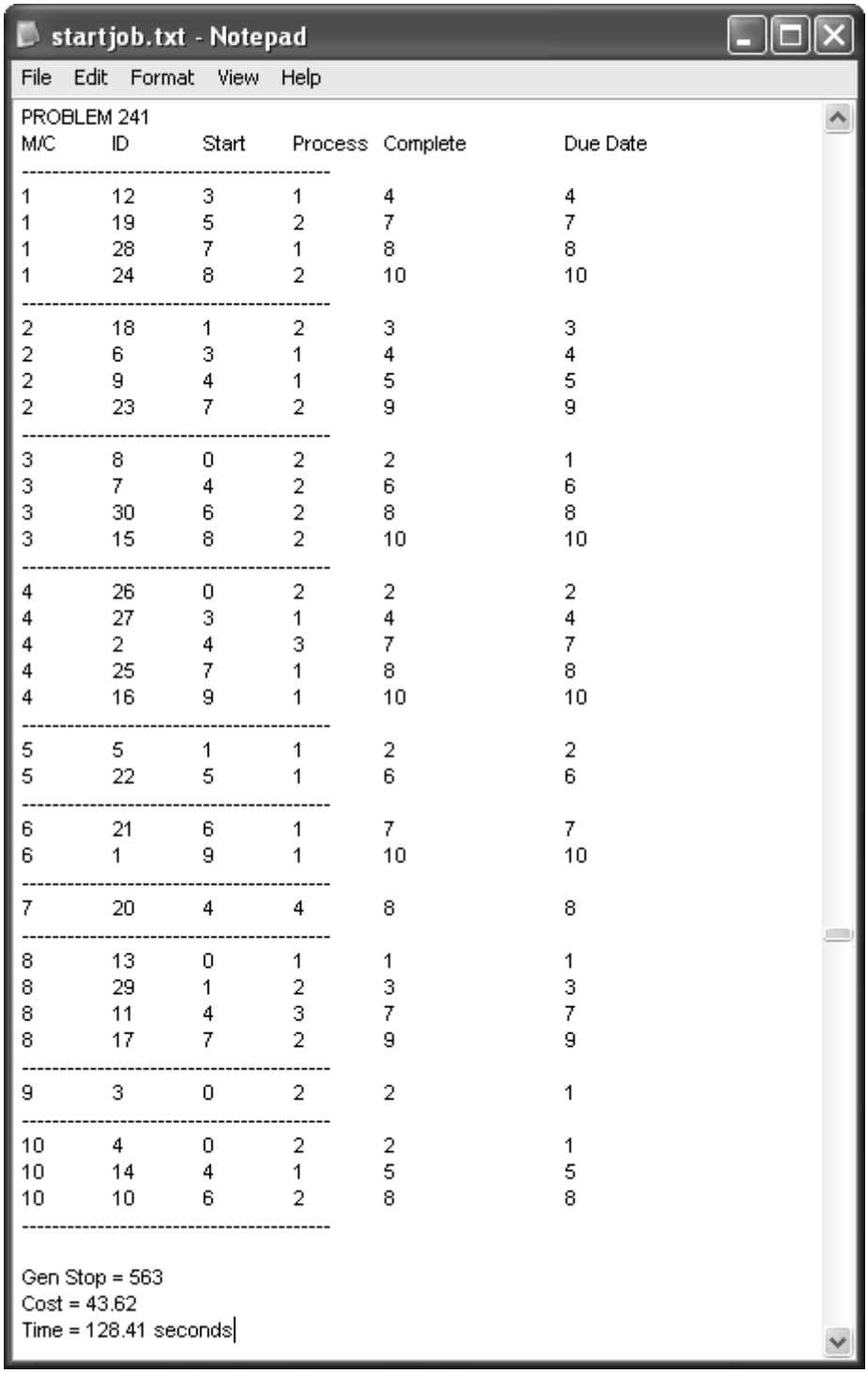


job.txt - Notepad

File Edit Format View Help

```
speed    1 2 3 2 3 3 1 2 2 3
PROBLEM241
ID       Process DueDate E        T
1        3       10      4.73     9.46
2        3       6       7.81     15.62
3        4       1       3.87     7.74
4        5       1       8.81     17.62
5        3       2       3.51     7.02
6        1       4       8.81     17.62
7        4       6       6.35     12.70
8        4       1       9.13     18.26
9        2       5       3.63     7.26
10       4       8       6.04     12.08
11       5       7       4.76     9.52
12       1       4       1.38     2.76
13       2       1       0.68     1.36
14       2       5       4.49     8.98
15       4       10      0.21     0.42
16       1       10      9.91     19.82
17       4       9       2.82     5.64
18       3       3       3.44     6.88
19       2       7       7.39     14.78
20       4       8       5.61     11.22
21       1       7       2.88     5.76
22       2       6       8.87     17.74
23       3       9       1.99     3.98
24       2       10      1.96     3.92
25       1       8       5.63     11.26
26       4       2       0.99     1.98
27       1       4       0.76     1.52
28       1       8       5.23     10.46
29       4       3       6.50     13.00
30       5       8       9.69     19.38
```

ภาพผนวกที่ ง17 ปัญหาที่ 241 ที่ 10 เครื่องจักรและ $\beta = 2\alpha$



startjob.txt - Notepad

File Edit Format View Help

PROBLEM 241

M/C	ID	Start	Process	Complete	Due Date
1	12	3	1	4	4
1	19	5	2	7	7
1	28	7	1	8	8
1	24	8	2	10	10
2	18	1	2	3	3
2	6	3	1	4	4
2	9	4	1	5	5
2	23	7	2	9	9
3	8	0	2	2	1
3	7	4	2	6	6
3	30	6	2	8	8
3	15	8	2	10	10
4	26	0	2	2	2
4	27	3	1	4	4
4	2	4	3	7	7
4	25	7	1	8	8
4	16	9	1	10	10
5	5	1	1	2	2
5	22	5	1	6	6
6	21	6	1	7	7
6	1	9	1	10	10
7	20	4	4	8	8
8	13	0	1	1	1
8	29	1	2	3	3
8	11	4	3	7	7
8	17	7	2	9	9
9	3	0	2	2	1
10	4	0	2	2	1
10	14	4	1	5	5
10	10	6	2	8	8

Gen Stop = 563
Cost = 43.62
Time = 128.41 seconds

ภาพผนวกที่ ง18 ผลการใช้โปรแกรม GAOPT ของปัญหาที่ 241 ที่ 10 เครื่องจักรและ $\beta = 2\alpha$

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	นายกานต์ ปล้องอ่อน
วัน เดือน ปี ที่เกิด	วันที่ 2 ธันวาคม 2517
สถานที่เกิด	พัทลุง
ประวัติการศึกษา	วศ.บ.(เครื่องกล) มหาวิทยาลัยสงขลานครินทร์
ตำแหน่งหน้าที่การงานปัจจุบัน	ผู้ช่วยผู้ควบคุมงาน
สถานที่ทำงานปัจจุบัน	บริษัท ทีพีไอ โพลีน จำกัด (มหาชน)