วิทยานิพนธ์

การใช้วิธีเชิงฮิวริสติกส์เพื่อแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีคลังสินค้าหลายแห่ง

HEURISTIC APPROACH FOR MULTI-DEPOT VEHICLE ROUTING PROBLEM

นางสาวอุบลรัตน์ เธียรธนาคม

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

พ.ศ. 2551

ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมอุตสาหการ)

ปริญญา

วิศวกรรมอุตสาหการ — **สาขา**

วิศวกรรมอุตสาหการ — **ภาควิชา**

เรื่อง การใช้วิธีเชิงฮิวริสติกส์เพื่อแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีคลังสินค้าหลายแห่ง

Heuristic Approach for Multi-depot Vehicle Routing Problem

นามผู้วิจัย นางสาวอุบลรัตน์ เธียรธนาคม

ได้พิจารณาเห็นชอบโดย

ประธานกรรมการ (รองศาสตราจารย์อนันต์ มุ่งวัฒนา, Ph.D.)

กรรมการ (อาจารย์วิสุทธิ์ สุพิทักษ์, Ph.D.)

กรรมการ (ผู้ช่วยศาสตราจารย์สำราญ ทองเล็ก, บธ.ม.)

หัวหน้าภาควิชา (รองศาสตราจารย์อนันต์ มุ่งวัฒนา, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

วันที่ 23 **เดือน** พฤษภาคม **พ.ศ.** 2551

วิทยานิพนธ์

เรื่อง

การใช้วิธีเชิงฮิวริสติกส์เพื่อแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีคลังสินค้าหลายแห่ง

Heuristic Approach for Multi-depot Vehicle Routing Problem

โดย

นางสาวอุบลรัตน์ เธียรธนาคม

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมอุตสาหการ)
พ.ศ. 2551

อุบลรัตน์ เธียรธนาคม 2551: การใช้วิธีเชิงฮิวริสติกส์เพื่อแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีคลังสินค้าหลายแห่ง ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมอุตสาหการ) สาขาวิศวกรรมอุตสาหการ ภาควิชาวิศวกรรมอุตสาหการ ประธานกรรมการที่ปรึกษา: รองศาสตราจารย์อนันต์ มุ่งวัฒนา, Ph.D. 83 หน้า

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาวิธีการในการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีคลังสินค้าหลายแห่ง ซึ่งประกอบไปด้วยสามขั้นตอนหลัก โดยมีเป้าหมายเพื่อให้ระยะทางรวมของระบบน้อยที่สุด ในการหาผลเฉลยเบื้องต้น ลูกค้าจะถูกแจกจ่ายไปยังคลังสินค้าที่ใกล้ที่สุดโดยวิธีใกล้ไหนไปนั่น (Arbitrary Nearest Neighbor Algorithm) จากนั้นเส้นทางของยานพาหนะในแต่ละคลังสินค้าจะถูกสร้างขึ้นโดยใช้อัลกอริทึมแบบประหยัด (Saving Algorithm) และเส้นทางนั้นๆ จะถูกพัฒนาต่อมาโดยวิธี Arbitrary Insertion ในขั้นตอนสุดท้าย อัลกอริทึมเชิงพันธุกรรม (Genetic Algorithm) จะถูกนำมาประยุกต์ใช้ในการปรับปรุงผลเฉลยของระบบ ผลจากการคำนวณแสดงให้เห็นถึงการลดลงของระยะทางรวมของระบบในแต่ละขั้นตอน

น.ส. อุบลรัตน์ เธียรธนาคม
ลายมือชื่อนิสิต

ลายมือชื่อประธานกรรมการ

14 / 05 / 51

Ubonrat Theantanakhom 2008: Heuristic Approach for Multi-depot Vehicle Routing Problem. Master of Engineering (Industrial Engineering), Major Field: Industrial Engineering, Department of Industrial Engineering. Thesis Advisor: Associate Professor Anan Mungwattana, Ph.D. 83 pages.

The objective of this research is to develop a method to solve the multi-depot vehicle routing problems (MDVRP). The method for solving the MDVRP consists of three major steps with the goal of minimizing the total distrance. First, customers are clustered for each depot (or warehouse) by using the Arbitrary Nearest Neighbor Algorithm. Then a route for each depot is constructed using the Saving Algorithm and improved by the Arbitrary Insertion technique. Last a Genetic Algorithm is applied to improve the route. The computational results have shown substantial improvement in each step.

Ubonrat Theantanakhom
Student's signature

Anan Mungwattana
Thesis Advisor's signature

14 / 05 / 02

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จได้ด้วยความกรุณาเอาใจใส่เป็นอย่างดีจาก ผศ.ดร.อนันต์ มุ่งวัฒนา ประธานกรรมการ ที่ได้ให้ความช่วยเหลือ แนะนำแนวทางต่างๆ รวมทั้งให้โอกาสได้ทำงานวิจัยในครั้งนี้ และ อ.ดร.วิสุทธิ์ สุพิทักษ์ กรรมการสาขาวิชาเอก ที่ให้คำแนะนำและแนวทางในการทำงานวิจัย ผศ.นท.สำราญ ทองเล็ก กรรมการสาขาวิชารอง และ อ.ดร.ชัยชาญ สุทธิกานต์ ผู้แทนบัณฑิตวิทยาลัยที่ได้กรุณาให้คำแนะนำปรึกษา ตลอดจนแก้ไขวิทยานิพนธ์ให้สมบูรณ์ยิ่งขึ้น ขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้

ข้าพเจ้าขอขอบพระคุณบิดา มารดา และครอบครัวของข้าพเจ้าที่ให้กำลังใจ และสนับสนุนรวมทั้งความอดทน และความเหนื่อยยากของบิดามารดาที่เป็นแรงผลักดันให้ข้าพเจ้าสามารถลุกขึ้นสู้กับปัญหาทุกเรื่องอย่างไม่ท้อแท้ และเป็นกำลังใจที่ดีให้กับข้าพเจ้าจนทำให้วิทยานิพนธ์นี้สำเร็จได้ด้วยดี และขอขอบคุณทุกท่านที่มิได้เอ่ยนามที่มีส่วนในการช่วยเหลือให้งานวิทยานิพนธ์นี้สำเร็จได้ด้วยดี

อุบลรัตน์ เธียรธนาคม

มกราคม 2551

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

การใช้วิธีเชิงฮิวริสติกส์เพื่อแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีคลังสินค้าหลายแห่ง

Heuristic Approach for Multi-depot Vehicle Routing Problem

คำนำ

ปัญหาการจัดเส้นทางยานพาหนะเป็นปัญหาหนึ่งที่มีความสำคัญมากต่อระบบเศรษฐกิจของประเทศ ทั้งในภาคอุตสาหกรรม ภาคการขนส่ง ภาคเกษตรกรรม ระบบสาธารณูปโภค ตลอดจนการบริโภคในครัวเรือน และเป็นปัญหาที่ยุ่งยากมีปัจจัยเกี่ยวข้องหลายประการ เช่น จำนวนยานพาหนะที่ใช้ในการขนส่งสินค้า ขีดจำกัดด้านความจุของยานพาหนะ จำนวนคลังสินค้า การจำกัดระยะทางในแต่ละเส้นทาง ความต้องการในการขนส่ง เป็นต้น ดังนั้นการวางแผนการจัดเส้นทางยานพาหนะที่ดีจะทำให้การขนส่งมีประสิทธิภาพมากขึ้น อีกทั้งเป็นการช่วยลดค่าใช้จ่ายด้านการขนส่งลงอีกวิธีหนึ่ง เพื่อรับมือกับสภาวะการเปลี่ยนแปลงของราคาน้ำมันในตลาดโลกที่มีแนวโน้มการปรับตัวสูงขึ้นอย่างต่อเนื่องและรวดเร็ว

งานวิจัยนี้จึงจัดทำขึ้นเพื่อทำการวิเคราะห์ปัญหาการจัดเส้นทางยานพาหนะ เพื่อให้ระยะทางที่ใช้ในการขนส่งสินค้าน้อยที่สุด โดยจะนำหลักการของอัลกอริทึมเชิงพันธุกรรมมาประยุกต์ในการแก้ปัญหาการจัดเส้นทางยานพาหนะ และใช้โปรแกรม Mathemetica เป็นเครื่องมือในการประมวลผลตัวแบบของปัญหาในงานวิจัยนี้

วัตถุประสงค์

สร้างรูปแบบการแก้ปัญหาการขนส่ง ซึ่งมีการพิจารณาปัญหาการจัดเส้นทางยานพาหนะ โดยมีวัตถุประสงค์ดังนี้

1. ทำการสร้างตัวแบบของปัญหาการจัดเส้นทางยานพาหนะ เพื่อระยะทางในการขนส่งสินค้าน้อยที่สุด โดยมีข้อจำกัดของปัญหาดังนี้
   1.1 มีคลังสินค้ากลางที่ใช้ส่งสินค้าหลายจุด
   1.2 ยานพาหนะที่ใช้ในการขนส่งขนาดเดียว และไม่จำกัดจำนวน
   1.3 เมื่อยานพาหนะส่งสินค้าหมด ยานพาหนะจะกลับไปยังคลังสินค้าเดิม
   1.4 พื้นที่ในการจอดรถในแต่ละคลังสินค้ามีไม่จำกัด
   1.5 จำนวนสินค้าในแต่ละคลังสินค้ามีเพียงพอกับความต้องการของลูกค้าในแต่ละคลังสินค้านั้นๆ
2. นำฮิวริสติกส์มาประยุกต์ในการแก้ปัญหาการขนส่งที่มีขนาดใหญ่
3. พัฒนาโปรแกรมคอมพิวเตอร์เพื่อสนับสนุนการใช้งานด้านการขนส่ง

ประโยชน์ที่คาดว่าจะได้รับ

1. ลดระยะทางที่ใช้ในการขนส่ง
2. ตอบสนองความต้องการลูกค้าได้ดีขึ้น
3. ลดจำนวนรถที่ใช้ในการขนส่ง
4. มีแบบแผนและโปรแกรมที่ใช้ในการขนส่งที่มีประสิทธิภาพ
5. การใช้โปรแกรมคอมพิวเตอร์ในการจัดเส้นทางขนส่งช่วยลดเวลาที่ใช้ในการจัดเส้นทาง

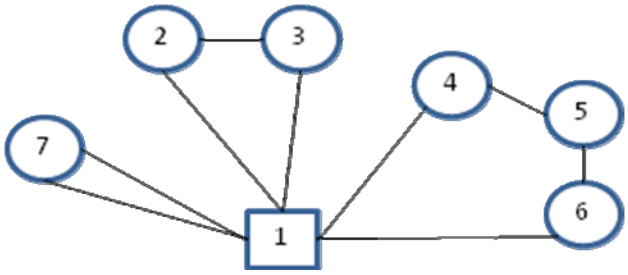
การตรวจเอกสาร

1. ทฤษฎีที่เกี่ยวข้องกับปัญหาการขนส่ง

1.1 ปัญหาการจัดเส้นทางยานพาหนะ (Vehicle Routing Problem; VRP)

ปัญหาการจัดเส้นทางยานพาหนะ คือการกำหนดกลุ่มของเส้นทางการขนส่งจากคลังสินค้ากลางไปยังความต้องการที่จุดต่างๆ โดยที่ค่าใช้จ่ายหรือระยะทางในการขนส่งมีค่าน้อยที่สุด ภายใต้เงื่อนไขหรือข้อจำกัดในด้านต่างๆ ด้วย เช่น เวลา จำนวนและขีดจำกัดของยานพาหนะ เป็นต้น

Golden et al. (1977) ได้เสนอปัญหาการจัดเส้นทางของยานพาหนะจากคลังสินค้าไปยังลูกค้าหลายจุด ซึ่งมีปริมาณความต้องแตกต่างกัน ภายใต้เงื่อนไขคือ ระยะทางต่ำที่สุด และทุกๆ ยานพาหนะจะเริ่มต้นและสิ้นสุดที่คลังสินค้ากลาง โดยมีข้อจำกัดในความจุของยานพาหนะที่ใช้ในการขนส่งและระยะเวลาสูงสุดในการขนส่งหนึ่งรอบของเส้นทางการจัดส่ง ถ้าไม่คำนึงถึงข้อจำกัดในระยะเวลาสูงสุดในการขนส่ง จะเป็นปัญหาการจัดเส้นทางยานพาหนะมาตรฐาน (Standard Vehicle Routing Problem : VRP) ภาพที่ 1 เป็นเครือข่ายที่แสดงลักษณะเส้นทางยานพาหนะโดยกลุ่มยานพาหนะ



ภาพที่ 1 เครือข่ายเส้นทางยานพาหนะ

โดยแสดงเป็นสมการทางคณิตศาสตร์ดังนี้

Minimize $$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{v=1}^{NV} C_{ij}X_{ij}^{v} \quad (1)$$

Subject to $$\sum_{i=1}^{n}\sum_{v=1}^{NV} X_{ij}^{v} = 1 \quad (j = 2, \ldots, n) \quad (2)$$

$$\sum_{j=1}^{n}\sum_{v=1}^{NV} X_{ij}^{v} = 1 \quad (i = 2, \ldots, n) \quad (3)$$

$$\sum_{i=1}^{n} X_{ip}^{v} - \sum_{j=1}^{n} X_{pj}^{v} = 0 \quad (v = 1,\ldots, NV ; p = 1,\ldots,n) \quad (4)$$

$$\sum_{i=1}^{n} D_i \left(\sum_{j=1}^{n} X_{ij}^{v} \right) \le K_v \quad (v = 1, \ldots, NV) \quad (5)$$

$$\sum_{i=1}^{n} t_{i}^{v} \sum_{j=1}^{n} X_{ij}^{v} + \sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}^{v} X_{ij}^{v} \le T_v \quad (v = 1, \ldots, NV) \quad (6)$$

$$\sum_{j=2}^{n} X_{1j}^{v} \le 1 \quad (v = 1, \ldots, NV) \quad (7)$$

$$\sum_{i=2}^{n} X_{i1}^{v} \le 1 \quad (v = 1, \ldots, NV) \quad (8)$$

$$X \in S \quad (9)$$

$$X_{ij}^{v} = 0 \text{ or } 1 \text{ for all i, j, v} \quad (10)$$

โดยที่

n คือจำนวนของปม (Node)

NV คือจำนวนยานพาหนะ

K_v คือความจุยานพาหนะ (v)

T_v คือข้อจำกัดเวลาสูงสุดของเส้นทางยานพาหนะ v

d_i คือปริมาณความต้องการของปม i (จุดส่งสินค้า i) $d_1 = 0$

t_1^v คือเวลาที่ต้องการสำหรับยานพาหนะ v ในการส่งมอบ ณ ปม i ($t_1^v = 0$)
ปม i = 1 คือ คลังสินค้ากลาง

t_{ij}^v คือเวลาในการเดินทางสำหรับยานพาหนะ v จากปม iไปยังปม j ($t_{ij}^v = \infty$)

C_{ij} คือค่าใช้จ่ายในการเดินทางจากปม i ไปปม j

$X_{ij}^v = 1$ ถ้าเส้นทาง i-j ถูกเชื่อมเส้นทางโดยยานพาหนะ v (ยานพาหนะ v เดินทางจากปม i ไปยังปม j) ถ้าไม่เช่นนั้น $X_{ij}^v = 0$, X = เมตริกซ์ของ $X_{ij} \equiv \sum_{v=1}^{NV} X_{ij}^v$ แสดงการเชื่อมโยงกันของปม, S เป็นเส้นทางของยานพาหนะแต่ละคันซึ่งไม่รวมจุดเริ่มต้นปม 1 (คลังสินค้ากลาง)

1.2 ปัญหาการจัดเส้นทางยานพาหนะที่มีคลังสินค้าหลายแห่ง (The multi-depot vehicle routing problem; MDVRP)

เป็นรูปแบบของปัญหาที่พัฒนามาจากปัญหาการจัดเส้นทางยานพาหนะ (VRP) ซึ่งจะมีคลังสินค้าเพียงแห่งเดียว โดยแสดงเป็นสมการทางคณิตศาสตร์ได้ดังต่อไปนี้

Minimize $$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{v=1}^{NV} C_{ij} X_{ij}^v \qquad (11)$$

Subject to $$\sum_{i=1}^{n}\sum_{v=1}^{NV} X_{ij}^v = 1 \quad (j = M+1, \ldots, n) \qquad (12)$$

$$\sum_{j=1}^{n}\sum_{v=1}^{NV} X_{ij}^v = 1 \quad (i = M+1, \ldots, n) \qquad (13)$$

$$\sum_{i=1}^{n} X_{ip}^{v} - \sum_{j=1}^{n} X_{pj}^{v} = 0 \quad (v = 1,\ldots, NV ; p = 1,\ldots,n) \tag{14}$$

$$\sum_{i=1}^{n} D_i \left(\sum_{j=1}^{n} X_{ij}^{v} \right) \leq K_v \quad (v = 1, \ldots, NV) \tag{15}$$

$$\sum_{i=1}^{n} t_i^{v} \sum_{j=1}^{n} X_{ij}^{v} + \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij}^{v} X_{ij}^{v} \leq T_v \quad (v = 1, \ldots, NV) \tag{16}$$

$$\sum_{i=1}^{M} \sum_{j=M+1}^{n} X_{ij}^{v} \leq 1 \qquad (v = 1, \ldots, NV) \tag{17}$$

$$\sum_{p=1}^{M} \sum_{i=M+1}^{n} X_{ip}^{v} \leq 1 \qquad (v = 1, \ldots, NV) \tag{18}$$

$$X \in S \tag{19}$$

$$X_{ij}^{v} = 0 \text{ or } 1 \text{ for all i, j, v} \tag{20}$$

โดยที่

M	คือจำนวนคลังสินค้า
n	คือจำนวนของปม (Node)
NV	คือจำนวนยานพาหนะ
K_v	คือความจุยานพาหนะ (v)
T_v	คือข้อจำกัดเวลาสูงสุดของเส้นทางยานพาหนะ v
d_i	คือปริมาณความต้องการของปม i (จุดส่งสินค้า i) $d_1 = 0$
t_1^v	คือเวลาที่ต้องการสำหรับยานพาหนะ v ในการส่งมอบ ณ ปม i ($t_1^v = 0$) ปม i = 1 คือ คลังสินค้ากลาง
t_{ij}^v	คือเวลาในการเดินทางสำหรับยานพาหนะ v จากปม iไปยังปม j ($t_{ij}^v = \infty$)
C_{ij}	คือค่าใช้จ่ายในการเดินทางจากปม i ไปปม j

รูปแบบทางคณิตศาสตร์ที่ทำการพิจารณามีวัตถุประสงค์ในการหาระยะทางที่น้อยที่สุด ดังแสดงในสมการที่ (11) โดยมีสมการข้อจำกัดดังนี้ สมการที่ (12) และ (13) เป็นการควบคุมการส่งสินค้าไปยังลูกค้าด้วยยานพาหนะเพียงหนึ่งคัน และเพื่อให้การเดินทางเป็นไปอย่างต่อเนื่องนั้น จึงใช้สมการที่ (14) เป็นตัวควบคุม คือเมื่อยานพาหนะเข้าไปส่งสินค้ายังลูกค้าเรียบร้อยแล้วจะต้องออกจากจุดส่งสินค้านั้นด้วย สมการที่ (15) คือข้อจำกัดในด้านความจุของยานพาหนะ เช่นเดียวกับสมการที่ (16) ซึ่งที่ข้อจำกัดในด้านเวลาที่ให้ในการเดินทาง สมการที่ (17) และ (18) ยืนยันว่าความต้องการจะไม่เกินจำนวนยานพาหนะที่สามารถใช้งานได้ สมการที่ (19) คือข้อจำกัดในเรื่องของการเกิดเส้นทางย่อย (Subtour breaking) สมการที่ (20) แสดงการเชื่อมโยงระหว่างปม i และ j โดยจะให้ค่าเท่ากับ 1 เมื่อมีการเชื่อมโยงระหว่างปม i กับ j และให้ค่าเท่ากับ 0 เมื่อไม่เกิดการเชื่อมโยงเกิดขึ้น

ปัญหาการเกิดเส้นทางย่อยสามารถกำจัดออกไปได้โดยการเลือกใช้ข้อกำหนดใดๆ ดังต่อไปนี้

(1) $S = \{(x_{ij}) : \sum_{i \in Q} \sum_{j \notin Q} x_{ij} \geq 1$ สำหรับทุกๆ เซตย่อย Q = {1, 2, …, n}
ซึ่งมีปม $1, 2, \ldots, M\}$

(2) $S = \{(x_{ij}) : \sum_{i \in R} \sum_{j \in R} x_{ij} \leq |R| - 1$ สำหรับทุกๆเซตย่อย R ที่ไม่ใช่เซตว่าง
ของปม $\{M+1,\ M+2,\ \ldots,\ n\ \}\}$

(3) $S = \{(x_{ij}) : y_i - y_j + n x_{ij} \leq n - 1$ สำหรับ $M = 1 \leq i \neq j \leq n$
และจำนวนจริงใด y_i

เนื่องจากปัญหาที่เกี่ยวกับการขนส่งนั้นมีขอบเขตที่หลากหลาย และขึ้นกับการประยุกต์ใช้ในการปฏิบัติงาน การทำความเข้าใจในรายละเอียดของปัญหาให้ดีจึงเป็นสิ่งจำเป็นที่จะทำให้วิเคราะห์ได้อย่างถูกต้อง ดังนั้น Bodin and Golden (1981) ได้นำเสนอลักษณะเฉพาะของสิ่งที่ทำให้ปัญหาการจัดเส้นทางการขนส่งมีความแตกต่างกัน ดังแสดงในตารางที่ 1

ตารางที่ 1 ลักษณะของปัญหาการจัดเส้นทางสำหรับยานพาหนะ

ลักษณะของปัญหา	ทางเลือกที่เป็นไปได้
1. จำนวนของยานพาหนะ (Fleet)	- จำนวน 1 คัน
	- จำนวนหลายคัน
2. ประเภทของยานพาหนะที่มี (Vehicle type)	- ประเภทเดียวกันหมด
	- หลายประเภท
	- ยานพาหนะแบบพิเศษ
3. ที่จอดยานพาหนะ (Depot) หรือคลังสินค้า	- คลังสินค้ากลาง 1 แห่ง
(Warehouse)	- คลังสินค้ากลางมากกว่า 1 แห่ง
4. ความต้องการในการขนส่ง (Transport demand)	- ความต้องการที่แน่นอน (Deterministic)
	- ความต้องการที่ไม่แน่นอน (Stochastic)
	- ขึ้นกับความพอใจบางส่วน
5. จุดกำเนิดความต้องการ (Demand location)	- ที่ตำแหน่ง (Node หรือ point)
	- ที่เส้นทาง (Arc หรือ route)
	- ที่ตำแหน่งและเส้นทาง (Mix)
6. ความสามารถในการบรรทุกยานพาหนะ	- เท่ากันหมด
(Vehicle capicity)	- ไม่เท่ากัน
7. เวลาในการขนส่งที่ยอมรับได้มากที่สุด	- กำหนด (เท่ากันทุกเส้นทาง)
(Maximun route time)	- กำหนด (ต่างกันตามเส้นทาง)
	- ไม่กำหนด (ไม่จำกัดความจุ)
8. ข้อจำกัดในด้านเวลาในการขนส่ง	- แบบด้านเดียว (Single-sided)
(Time windows)	- แบบสองด้าน (Double-sided)
10. ต้นทุน	- ต้นทุนผันแปรหรือต้นทุนเส้นทาง
	- ต้นทุนคงที่หรือต้นทุนยานพาหนะ
	- ต้นทุนขนส่งร่วม

2. ทฤษฎีที่เกี่ยวข้องในงานวิจัย

หลักการพื้นฐานของอัลกอริทึมอบเหนียวจำลอง (Simulated annealing) คือการใช้วิธีการค้นหาแบบสุ่ม ซึ่งจะไม่เพียงแค่ยอมรับการเปลี่ยนแปลงใดๆ ที่ฟังก์ชันในแต่ละสถานะเพิ่มขึ้น แต่มันยังยอมรับการเปลี่ยนแปลงที่ฟังก์ชันลดลงอีกด้วย การยอมรับการเปลี่ยนแปลงนี้เกิดจากการใช้ความน่าจะเป็นเข้ามาพิจารณาร่วมด้วย

ขั้นตอนของอัลกอริทึมอบเหนียวจำลอง สามารถอธิบายได้ดังต่อไปนี้ ในแต่ละรอบของอัลกอริทึม สถานะใหม่ของระบบจะถูกสร้างขึ้นมาจากสถานะปัจจุบัน โดยการสลับตำแหน่งของอนุภาคสองอนุภาคที่ถูกสุ่มเลือก ถ้าพลังงานของสถานะใหม่ต่ำกว่าพลังงานของสถานะปัจจุบัน การสลับตำแหน่งนั้นถูกยอมรับ และสถานะใหม่นั้นกลายมาเป็นสถานะปัจจุบัน อย่างไรก็ตาม ถ้าพลังงานของสถานะใหม่สูงกว่าพลังงานของสถานะปัจจุบันโดย ΔE ความน่าจะเป็นของการเปลี่ยนแปลงจากสถานะปัจจุบันไปเป็นสถานะใหม่คือ

$$\exp\left(-\frac{\Delta E}{kT}\right)$$

โดยที่ k คือค่าคงที่โบสมานน์ (Boltzmann constant) และ T คืออุณหภูมิสัมบูรณ์ปัจจุบัน ขั้นตอนทั้งหมดที่กล่าวมาข้างต้นจะถูกดำเนินการซ้ำอย่างไม่มีกำหนด ซึ่งสามารถแสดงได้ว่าวิธีการสร้างสถานะปัจจุบันนี้นำไปสู่การกระจายตัวของสถานะที่ซึ่ง ความน่าจะเป็นของสถานะที่ถูกเลือกต่อสถานะปัจจุบันคือ

$$\frac{\exp\left(-E_i / kT\right)}{\sum_j \exp\left(-E_j / kT\right)}$$

E_i คือพลังงานของสถานะที่ถูกเลือก และฟังก์ชันความน่าจะเป็นนี้ถูกเรียกว่า ความหนาแน่นโบสมานน์ (Boltzmann density) หนึ่งในคุณสมบัติของฟังก์ชันนี้คือ สำหรับระบบที่อุณหภูมิสูงมาก แต่ละสถานะจะมีโอกาสใกล้เคียงกันในการเปลี่ยนเป็นสถานะปัจจุบัน อย่างไรก็ตาม ที่อุณภูมิต่ำสถานะที่มีพลังงานต่ำเท่านั้นจะมีโอกาสสูงในการเป็นสถานะปัจจุบัน

จากการทดลองจริง ได้แสดงให้เห็นว่า ระบบที่มีอุณหภูมิต่ำมากๆ ไม่จำเป็นจะต้องอยู่ในสถานะพื้น (ground state) หรือในสถานะที่ใกล้เคียง ดังนั้น โดยปกติแล้วอัลกอริทึมอบเหนียวจำลอง จะเริ่มจากอุณหภูมิสูง แล้วค่อยๆ ลดอุณหภูมิอย่างช้าๆ จนกระทั่งระบบมีช่วงของสถานะพื้นจำนวนมาก ฟังก์ชันการลดลงของอุณหภูมิคือ

$$T_{k+1} = \frac{T_k}{1+\tau\sqrt{T_k}}$$

โดยที่ T_k คืออุณหภูมิปัจจุบันของการคำนวณรอบที่ k และ τ คือค่าคงที่เวลาที่กำหนดขึ้นเอง $\sqrt{T_k}$ ถูกกำหนดเพื่อที่จะเพิ่มความเร็วในการลดอุณหภูมิ

กระบวนการของอัลกอริทึมอบเหนียว ที่ใช้ในการแก้ปัญหาการจัดเส้นทางยานพาหนะที่มีคลังสินค้าหลายแห่ง (MDVRP) มีดังนี้

1. สร้างเส้นทางเริ่มต้นขึ้นมา ($route_k = \{cust_1, cust_2, ..., cust_i, ..., cust_j, ..., cust_N\}$)
2. เพื่อที่จะสร้างเส้นทางใหม่ $route_{k+1}$
   2.1 เลือกลูกค้าขึ้นมาอย่างสุ่มสองคน ($cust_i$ และ $cust_j$)
   2.2 สลับตำแหน่งของลูกค้าทั้งสองเพื่อสร้างเส้นทางทดลอง ($route_{tr} = \{cust_1, cust_2, ..., cust_j, ..., cust_i, ..., cust_N\}$)
   2.3 คำนวณระยะทางของเส้นทางทดลอง $E(route_{tr})$
   2.4 ถ้า $E(route_{tr}) \le E(route_k)$ ให้ยอมรับเส้นทางทดลองนี้โดยกำหนด $route_{k+1} = route_{tr}$ แต่ถ้า $E(route_{tr}) > E(route_k)$
      2.4.1 ให้คำนวณความน่าจะเป็น $P = \exp(-\Delta E / T)$ โดยที่ $\Delta E = E(route_{tr}) - E(route_k)$
      2.4.2 สุ่มจำนวนจริงขึ้นมาหนึ่งค่า r โดยที่ $0 \le r \le 1$
      2.4.3 ถ้า $P \ge r$ ให้ยอมรับเส้นทางทดลองนี้ โดยกำหนด $route_{k+1} = route_{tr}$ แต่ถ้า $P < r$ ให้ปฏิเสธเส้นทางทดลอง โดยกำหนด $route_{k+1} = route_k$
3. ทำการเปลี่ยนแปลงอุณหภูมิของระบบโดยการใช้ฟังก์ชัน

$$T_{k+1} = \frac{T_k}{1+\tau\sqrt{T_k}}$$

ข้อดีและข้อด้อยของอัลกอริทึมอบเหนียวจำลอง

ข้อดี – สามารถประยุกต์ใช้กับระบบต่างๆได้ง่าย เนื่องจากว่าอัลกอริทึมอบเหนียวจำลองไม่ขึ้นกับคุณสมบัติเฉพาะของระบบใดๆ และไม่ขึ้นกับความซับซ้อนของปัญหา นอกจากนี้ อัลกอริทึมอบเหนียวจำลองยังรับประกันการหาผลเฉลยที่ดีที่สุด

ข้อเสีย - ปัญหาหลักของอัลกอริทึมอบเหนียวจำลองคือ ไม่สามารถบอกได้ว่า จะพบผลเฉลยที่ดีที่สุดหรือไม่ นอกจากนี้ วิธีการอบเหนียวจำลองยังใช้ได้ไม่ดีกับปัญหาที่ซึ่งพลังงานต่อเนื่อง (ไม่มีการก้าวกระโดดของพลังงาน หรือการก้าวกะโดดของเส้นทางในปัญหาการจัดเส้นทางยานพาหนะ)

Holland (1975) เสนอแนวคิดในการคำนวณหาคำตอบที่เรียกว่าอัลกอริทึมเชิงพันธุกรรม (Genetic algorithm: GA) โดยใช้หลักการในการเลียนแบบลักษณะทางวิวัฒนาการ (Evolutionary algorithm) ทางชีววิทยาโดยตั้งอยู่บนแนวความคิดของการเลือกเผ่าพันธุ์ธรรมชาติ (Natural selection) และวิธีการทางพันธุกรรม (Genetics) ซึ่งทฤษฎีวิวัฒนาการมีว่า สิ่งมีชีวิตทุกชนิดประกอบด้วยเซลล์ ซึ่งแต่ละเซลล์จะมีกลุ่มของโครโมโซม (Chromosome) ที่เหมือนกัน โครโมโซมประกอบด้วยยีนส์ (Gene) ซึ่งจะสะท้อนลักษณะเฉพาะของสิ่งมีชีวิตนั้น เช่น สีตา สีผม โดยยีนส์แต่ละตัวจะมีตำแหน่งของตนเอง

เมื่อนำโครโมโซมทั้งหมดมาประกอบกันจะเรียกว่า Genome และเรียกกลุ่มของยีนส์ใน Genome นี้ว่า Genotype หรือรหัสพันธุกรรม ซึ่ง Genotypes นี้จะแปลงไปเป็นอวัยวะของสิ่งมีชีวิตต่อไปเรียกว่า Phenotype ตามทฤษฎีวิวัฒนาการสิ่งมีชีวิตที่แข็งแรงที่สุดจะมีโอกาสสืบทอดสายพันธุ์ที่แข็งแรงต่อไป การผสมยีนส์ของพ่อและแม่จะทำให้เกิดลูกซึ่งคัดลอกยีนส์ของพ่อแม่ที่ผสมกัน ทำให้เกิดสายพันธุ์ที่แข็งแรงยิ่งขึ้น บางครั้งการคัดลอกยีนส์ของพ่อแม่ไม่สมบูรณ์ อาจทำให้เกิดการผ่าเหล่า (Mutation) ในรุ่นลูก การผ่าเหล่าทำให้สิ่งมีชีวิตมีโอกาสพัฒนาสายพันธุ์ใหม่ที่ยิ่งเข้มแข็งขึ้น หากการผ่าเหล่าทำให้เกิดสายพันธุ์ที่ด้อยกว่า สายพันธุ์นั้นจะไม่ถูกสืบทอดต่อไป

กระบวนการเชิงพันธุกรรมมีขั้นตอนที่สำคัญ 3 ขั้นตอนคือ สุ่มคำตอบที่เป็นไปได้, สร้างประชากร (Population)รุ่นใหม่ และประเมินคุณภาพของคำตอบด้วยฟังก์ชันการประเมินความ

เหมาะสม (Fitness) ซึ่งกระบวนการนี้จะวนรอบตามจำนวนรุ่นของประชากร และท้ายสุดคือการแปลงคำตอบให้อยู่ในรูปของค่าจริงที่ตีความได้

ในขั้นตอนที่หนึ่ง ระบบจะแปลง (Encode) ผลลัพธ์ที่เป็นไปได้ของปัญหาคณิตศาสตร์ให้อยู่ในรูปโครโมโซม ตัวแปรที่ต้องการค้นหาหมายถึงยีนส์หนึ่งยีนส์เมื่อนำยีนส์มาประกอบกันจึงได้โครโมโซมหรือ Genotype การแปลงค่าทำได้หลายวิธีเช่น แบบไบนารี (Binary), แบบวางสลับเปลี่ยนลำดับ (Permutation) หรือแบบการใช้ค่าของตัวแปร (Value) การเลือกแปลงขึ้นอยู่กับลักษณะของปัญหา เช่น การแปลงค่าแบบไบนารี (Binary encoding) เหมาะสำหรับการคำนวณหาค่าสูงสุดหรือต่ำสุดตามเงื่อนไข การแปลงแบบวางสลับเปลี่ยนลำดับ (Permutation encoding) เหมาะสำหรับปัญหาการจัดอันดับเช่น ในการวางแผนเดินทาง ตัวเลขที่แสดงหมายถึงลำดับของจุดที่เดินทาง ในขณะที่วิธีการแปลงค่าแบบใช้ค่าของตัวแปร (Value encoding) สามารถแปลงค่าได้หลายแบบ เช่นเป็นค่าจริง ตัวอักษร หรือกลุ่มวัตถุ การแปลงชนิดนี้เหมาะกับปัญหาลักษณะพิเศษต่างๆ เช่นการแปลงแบบค่าจริงอาจใช้ในการค้นหาน้ำหนักที่เหมาะสมสำหรับการคำนวณฟังก์ชันหนึ่งก็ได้ เมื่อกำหนดวิธีการแปลงแล้ว ระบบจะสุ่มสร้างกลุ่มโครโมโซมนี้ขึ้นมาเป็นประชากรซึ่งจำนวนประชากรในแต่ละรุ่นจะขึ้นอยู่กับค่าพารามิเตอร์ที่กำหนด ความยาวของโครโมโซมนี้ขึ้นอยู่กับจำนวนตัวแปรที่ต้องการหาค่าในระบบสมการ และมีผลต่อการกำหนดพารามิเตอร์จำนวนประชากร เพื่อหาผลเฉลยที่ดีที่สุดด้วย

ระบบจะประเมินความเหมาะสม (Fitness) ของโครโมโซมประชากรแต่ละชุด โดยพิจารณาจากสมการเป้าหมาย จากนั้นจะทำการคัดเลือกโครโมโซมที่เหมาะสมที่สุด 2 โครโมโซมให้เป็นพ่อและแม่เพื่อใช้สืบทอดสายพันธุ์ (Reproduction) ให้แก่ประชากรรุ่นใหม่ อีกนัยหนึ่งประชากรรุ่นใหม่ที่จะสร้างขึ้นคือลูกที่เกิดจากการผสมยีนส์ที่ดีที่สุดของประชากรรุ่นก่อนนั้นเอง

การคัดเลือกสามารถทำได้หลายวิธีเช่น การคัดเลือกโครโมโซมโดยใช้วงล้อสัดส่วนค่าความเหมาะสม (Roulette wheel selection) การคัดเลือกโครโมโซมแบบ Tournament, การคัดเลือกแบบ Rank หรือการคัดเลือกแบบ Elitise หลักการกว้างๆของการคัดเลือกของวงล้อสัดส่วนค่าความเหมาะสมคือ โครโมโซมที่เหมาะสมที่สุดมีโอกาสที่จะได้รับการคัดเลือกมากกว่าโครโมโซมที่ด้อยกว่า ขนาดพื้นที่ของวงล้อสัดส่วนคือสัดส่วนของค่าความเหมาะสมที่เหมาะสมของทุกโครโมโซม เมื่อมีการหมุนวงล้อ โครโมโซมที่มีค่าความเหมาะสมมากจะมีโอกาสถูกเลือกมากตามไปด้วย ส่วนการคัดเลือกแบบ Ranking คือ เลือกประชากรที่มีค่าความเหมาะสมที่ดีที่สุด โดยที่ไม่สนใจ

ประชากรตัวอื่นเลย และการคัดเลือกแบบ Elitise เป็นแนวคิดที่ป้องกันการสูญหายของเส้นทางที่ดีที่สุด หมายความว่ามีการคัดลอกโครโมโซมที่ดีที่สุดไว้ก่อน ส่วนประชากรส่วนที่เหลือจะใช้วิธีการเลือกแบบอื่นๆ

หลังจากคัดเลือกโครโมโซมพ่อแม่ได้แล้ว กระบวนการสร้างประชากรรุ่นใหม่จะเกิดขึ้นตามลำดับต่อไปนี้

การข้ามสายพันธุ์ (Crossover) เป็นกระบวนการที่สำคัญของอัลกอริทึมเชิงพันธุกรรม ซึ่งเมื่อเกิดการข้ามสายพันธุ์ขึ้นแล้ว จะทำให้เกิดการเปลี่ยนของสิ่งมีชีวิตที่หลากหลายขึ้น ซึ่งการข้ามสายพันธุ์จะต้องอาศัยกระบวนการวิวัฒนาการที่เป็นเวลานาน ในทำนองเดียวกันในทางการแก้ปัญหาจะทำให้เกิดความหลากหลายของคำตอบที่ได้ ทำให้เราได้รับคำตอบที่หลากหลาย จึงสามารถเลือกเอาคำตอบที่เหมาะสมกับความต้องการได้มากที่สุด ขั้นตอนในการข้ามสายพันธุ์จะนำโครโมโซมพ่อและโครโมโซมแม่มาผสมกันเพื่อให้ได้โครโมโซมใหม่ขึ้นมา

การผ่าเหล่า (Mutation) เป็นกระบวนการที่เกิดขึ้นหลังกระบวนการข้ามสายพันธุ์ นั่นหมายความว่าได้รุ่นลูกที่เกิดจากรุ่นพ่อแม่แล้ว จึงนำรุ่นลูกมาดำเนินการผ่าเหล่า ในการผ่าเหล่านั้นในทางพันธุศาสตร์จะทำให้เกิดการเปลี่ยนแปลงหรือทำให้เกิดลักษณะใหม่ๆขึ้น และทำให้เกิดวิวัฒนาการ ในการแก้ปัญหานั้นการเกิดผลลัพธ์ในลักษณะที่แตกต่างออกไปจากเดิมจะทำหน้าที่ป้องกันข้อผิดพลาดของวิธีการแก้ปัญหาทั้งหมด เมื่อเกิดการผสมยีนส์จากพ่อและแม่เพื่อถ่ายไปสู่ลูก จะทำให้โครโมโซมของรุ่นลูกกระจุกอยู่ในกลุ่มเดียวกับพ่อแม่ ในเชิงคณิตศาสตร์ผลลัพธ์ที่ได้อาจเป็นค่าสูงสุดหรือต่ำสุดเฉพาะในขอบเขตจำกัดหนึ่งเท่านั้น หรือเรียกว่าค่าท้องถิ่น (Local Optimal Solution) ไม่ใช่ค่าสูงสุดหรือต่ำสุดที่แท้จริง

ในบางครั้งการสร้างประชากรลูกโดยผสมยีนส์จากโครโมโซมพ่อและแม่ซึ่งเป็นผลลัพธ์ที่ดีที่สุดจากประชากรรุ่นก่อน อาจทำให้สูญเสียผลลัพธ์ที่ดีอยู่แล้วไปก็ได้ การกำหนดค่า Elitism จึงเป็นการคงซึ่งโครโมโซมพ่อและแม่ให้เป็นส่วนหนึ่งของประชากรรุ่นลูก เพื่อให้มีโอกาสสร้างสายพันธุ์ใหม่ในประชากรรุ่นถัดไป

ในการหยุดกระบวนการหาคำตอบของอัลกอริทึมเชิงพันธุกรรมนั้นมีได้หลายวิธีด้วยกัน เช่น ครบตามจำนวนรอบที่กำหนดไว้, พบเป้าหมายหรือคำตอบที่ต้องการ หรือพบคำตอบที่ใกล้เคียงกับที่ต้องการเช่น โครโมโซมมีค่า ณ ตำแหน่งของยีนส์เดียวกันเหมือนกันถึงร้อยละ 95

คำศัพท์ทางพันธุกรรมและการเชื่อมโยงกับความหมายทางการคำนวณเพื่อให้ผู้อ่านทบทวนแนวคิดและสร้างความเข้าใจการคำนวณเชิงพันธุกรรมได้ง่ายขึ้น ดังแสดงในตารางที่ 2

ตารางที่ 2 คำศัพท์ทางพันธุกรรมและศัพท์การคำนวณทางพันธุกรรม

ศัพท์ทางพันธุกรรม	ศัพท์การคำนวณทางพันธุกรมม
ยีนส์ (Gene)	ตัวแปรที่ต้องการหาคำตอบ
โครโมโซม (Chromosome)	กลุ่มผลลัพธ์ที่เป็นไปได้ของตัวแปรต่างๆ ในระบบในรูปแบบโครโมโซม
จำนวนประชากร (Population)	จำนวนผลลัพธ์ที่เป็นไปได้สำหรับการทดสอบหาค่าที่ดีที่สุดในแต่ละรุ่นของประชากร
จำนวนรุ่นของประชากร (Generation)	จำนวนการทดลองทำซ้ำเพื่อค้นหาผลลัพธ์ที่ดีที่สุด
การคัดเลือกทางธรรมชาติ (Natural selection)	การกำหนดฟังก์ชันทางคณิตศาสตร์ด้วยวิธีต่างๆ เช่น วงล้อสัดส่วนค่าความเหมาะสม (Roulette wheel) เพื่อประเมินความเหมาะสมของผลลัพธ์ที่ได้
การสืบทอดเผ่าพันธุ์ (Reproduction) และการข้ามสายพันธุ์ (Crossover)	การกำหนดจุดแบ่งในโครโมโซมพ่อและโครโมโซมแม่ จากผลลัพธ์ที่ถูกเลือกของประชากรรุ่นก่อน เพื่อสร้างโครโมโซมใหม่ของลูกหรือผลลัพธ์กลุ่มใหม่
การผ่าเหล่า (Mutation)	วิธีการปรับผลลัพธ์ที่เป็นไปได้เพื่อหลีกเลี่ยงการได้ผลลัพธ์ที่เป็นค่าท้องถิ่น (Local optimal solution) ของสมการเป้าหมาย

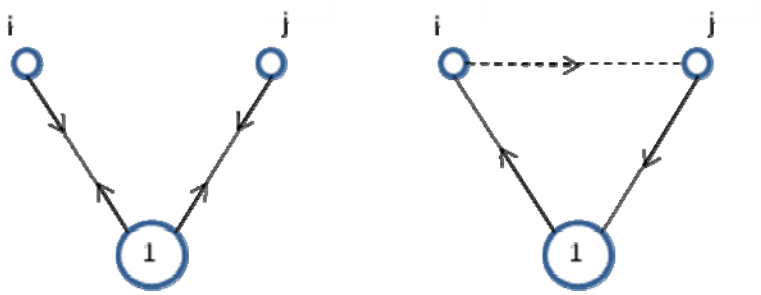
ตารางที่ 2 (ต่อ)

ศัพท์ทางพันธุกรรม	ศัพท์การคำนวณทางพันธุกรมม
Elitism	การรักษาผลลัพธ์ที่ดีที่สุดจากประชากรรุ่นก่อนให้เป็นทางเลือกของการสร้างประชากรรุ่นใหม่

Tan et al. (2001) ได้เสนอวิธีการแก้ปัญหาเส้นทางการขนส่งสินค้าให้ลูกค้า โดยทราบปริมาณความต้องการของลูกค้า และมีข้อจำกัดของเวลาในการจัดส่งสามารถจัดส่งได้และ ข้อจำกัดเกี่ยวกับความสามารถของรถ ในการศึกษาได้สร้างฮิวริสติกส์โดยการประยุกต์ใช้อัลกอริทึมเชิงพันธุกรรม เพื่อหาคำตอบของปัญหาที่ทำให้ต้นทุนรวมต่ำที่สุด

Clarke and Wright (1964) นำเสนออัลกอริทึมแบบประหยัด (Saving Algorithm) ของปัญหาการจัดเส้นทางยานพาหนะที่มีคลังสินค้าเพียงแห่งเดียวไปยังลูกค้าต่างๆ โดยพิจารณาหาเส้นทางเดินรถที่เหมาะสมจากระยะทางที่ประหยัดที่สุด (Saving) ของแต่ละคู่ของลูกค้า ซึ่งมีขั้นตอนดังต่อไปนี้

1. เลือกคลังสินค้าให้เป็นปมเริ่มต้นลำดับที่ 1
2. คำนวณระยะทางที่ประหยัดในการขนส่งจาก $S_{ij} = C_{1i} - C_{ij} + C_{j1}$ สำหรับทุกคู่ของลูกค้า i และ j ซึ่งจะเป็นผลทำให้เกิดการเชื่อมต่อ (i, j) เพื่อให้เกิดเส้นทางใหม่ขึ้นคือ (1, i, j, 1) แทนที่การแบ่งเส้นทางเป็น 2 เส้นคือ (1, i, 1) และ (1, j, 1) ดังแสดงในภาพที่ 2



(ก) (ข)

ภาพที่ 2 การเชื่อมต่อแบบประหยัด
(ก) สถานะเริ่มต้น
(ข) สถานะหลังการเชื่อมต่อ (i, j)

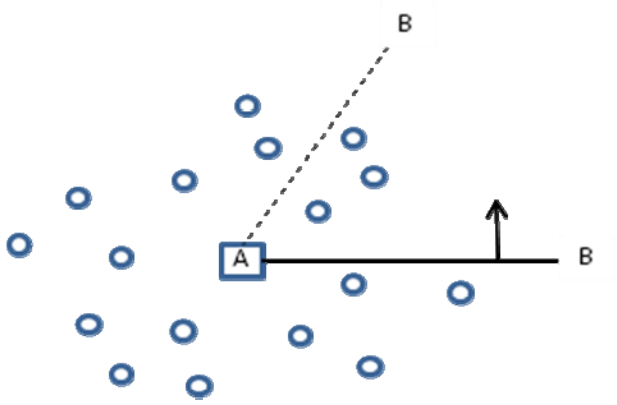
3. เรียงลำดับค่า S_{ij} จากค่ามากไปค่าน้อย

4. เริ่มทำการเชื่อมต่อปม i และ j ตั้งแต่ค่าแรกของชุดข้อมูลแบบประหยัดไปจนถึงค่าสุดท้ายกล่าวคือการเชื่อมโยงเส้นทางแรกที่มีค่า S_{ij} มากที่สุด

5. ถ้าไม่สามารถเพิ่มการเชื่อมต่อเข้าไปในเส้นทางได้อีก อันเนื่องมาจากข้อจำกัดด้านความจุของยานพาหนะหรือระยะเวลาในการเดินทางแล้ว จึงจะทำการสร้างเส้นทางใหม่ขึ้นมา

6. ทำซ้ำในข้อ 5 และข้อ 6 จนกระทั่งไม่สามารถเชื่อมโยงได้อีก

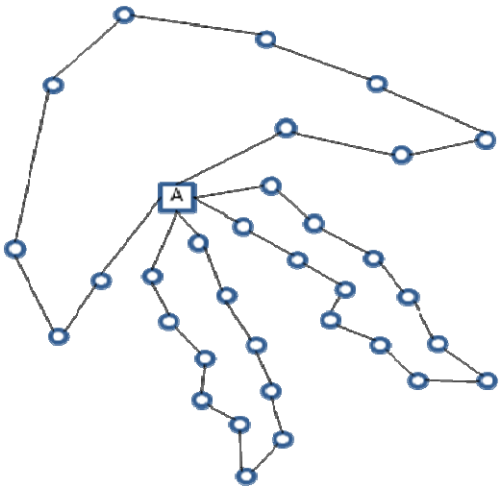
Gillett and Miller (1974) เสนอวิธีฮิวริสติกที่เรียกว่า Sweeping Approach ซึ่งเป็นวิธีที่เหมาะกับปัญหาที่มีจะขนส่งมากถึง 250 ปม ซึ่งมีขั้นตอนดังนี้ กำหนดจุดเดินทางและบอกลำดับที่ต้องเดินทางผ่านจะต่างๆ โดยที่ตำแหน่งที่ตั้งของจุดต่างๆ เป็น Polar Coordinate และคลังสินค้ากลางเป็นจุด A โดยจะเลือกจุดเริ่มต้นแบบสุ่มและกวาดแขน AB (หมุนทวนหรือตามเข็มนาฬิกา) จากคลังสินค้าไปยังจุดต่างๆ เพื่อตอบสนองความต้องการในแต่ละจุด ดังแสดงในภาพที่ 3



ภาพที่ 3 แสดงการดำเนินการของวิธี Sweeping Approach

ข้อบกพร่องของวิธีการ Sweeping Approach คือ

1. ในกรณีที่คลังสินค้าไม่ได้อยู่ที่จุดศูนย์กลางของพื้นที่จะทำให้ได้เส้นทางมีขนาดไม่สมดุลย์ แสดงให้เห็นว่าวิธีนี้ไม่สามารถจ่ายงานให้รถแต่ละคันได้อย่างสมดุล ดังแสดงในภาพที่ 4
2. วิธีการนี้ไม่คำนึงถึงถนน ทำให้จุดที่ใกล้เคียงกันที่อยู่บนถนนเส้นเดียวกันอาจจะไม่อยู่บนเส้นเดียวกัน



ภาพที่ 4 แสดงเส้นทางที่ไม่สมดุลย์ของเส้นทางเดินรถย่อย

อุปกรณ์และวิธีการ

อุปกรณ์

อุปกรณ์ที่ใช้ในงานวิจัยประกอบด้วย

1. คอมพิวเตอร์ส่วนบุคคล
   - ระบบปฏิบัติการ Windows XP Professional Service Pack 2
   - หน่วยประมวลผลกลาง Intel(R) Xeon(TM) CPU 3.20 GHz
   - หน่วยความจำ 3 GB
2. โปรแกรม Mathematica 6.0

วิธีการ

ในงานวิจัยนี้นำเสนอวิธีการแก้ปัญหาการจัดเส้นทางยานพาหนะโดยเริ่มจากการสร้างชุดข้อมูลแบบสุ่มโดยใช้โปรแกรม Mathematica 6.0 จากนั้นจึงพัฒนาวิธีเชิงฮิวริสติกในโปรแกรมเดียวกัน ซึ่งมีรายละเอียดดังต่อไปนี้

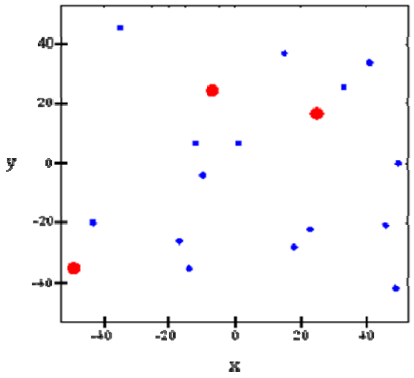
1. การสร้างชุดข้อมูลนำเข้า

ชุดข้อมูลนำเข้าที่ใช้ในตัวแบบที่พัฒนาขึ้นนี้ จะแบ่งเป็น 2 ชุดข้อมูลคือ ชุดข้อมูลของระยะทางและชุดข้อมูลความต้องการของลูกค้า โดยพัฒนาโปรแกรม Mathematica 6.0 ขึ้นมาเพื่อความสะดวกในการสร้างชุดข้อมูลแบบสุ่ม ซึ่งจะทำการสุ่มตำแหน่งของคลังสินค้าและลูกค้าในรูปแบบของพิกัด (x, y) ก่อนแล้วจึงเปลี่ยนเป็นให้อยู่ในรูปของระยะทางภายหลัง ในงานวิจัยนี้เลือกใช้ระบบพิกัดเนื่องจากต้องการดูลักษณะการจัดกลุ่มและการกระจายตัวของลูกค้าและคลังสินค้า อีกทั้งเพื่อความชัดเจนในการศึกษาเส้นทางและการแก้ปัญหาต่อไป อย่างไรก็ตามในการนำไปประยุกต์ใช้จริงแล้ว จะใช้ชุดข้อมูลนำเข้าเป็นชุดข้อมูลของระยะทางแทนชุดข้อมูลในระบบพิกัด เพื่อความสะดวกในประยุกต์ใช้งานจริง ซึ่งขั้นตอนในการสร้างชุดข้อมูลมีดังนี้

1.1 การสร้างชุดข้อมูลระยะทาง

1.1.1 ป้อนจำนวนคลังสินค้าและจำนวนลูกค้า

1.1.2 โปรแกรมจะทำการสุ่มพิกัด (x, y) ให้กับคลังสินค้าและลูกค้า แสดงดังภาพที่ 5 ทำการสุ่มคลังสินค้า 3 แห่ง และลูกค้าจำนวน 15 ราย



ภาพที่ 5 การสุ่มพิกัดของคลังสินค้าและลูกค้า

1.1.3 คำนวณระยะทางระหว่างจุด Aและจุด B ($dist(A,B)$) โดยใช้สูตรในการคำนวณดังนี้

$$dist(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

1.1.4 สร้างเมตริกซ์ระยะทางระหว่างคลังสินค้า → คลังสินค้า ,คลังสินค้า → ลูกค้า และลูกค้า →ลูกค้า ดังแสดงในภาพที่ 6

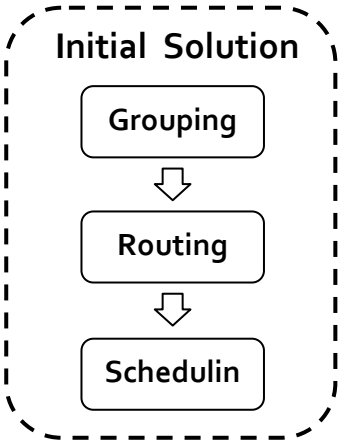
คลังสินค้า1	0	90	33	22	12	65	41	39	77	46	64	23	30	67	60	43	38	26
คลังสินค้า2	90	0	73	96	102	35	50	73	16	67	98	113	105	82	33	96	56	65
คลังสินค้า3	33	73	0	25	40	60	29	56	58	59	87	49	62	35	52	70	19	20
ลูกค้า 1	22	96	25	0	21	78	48	60	81	65	86	26	51	51	71	66	40	33
ลูกค้า 2	12	102	40	21	0	77	52	49	89	56	70	11	31	71	72	49	49	37
ลูกค้า 3	65	35	60	78	77	0	31	39	33	33	63	88	73	84	9	62	42	45
ลูกค้า 4	41	50	29	48	52	31	0	38	37	37	70	64	60	56	23	59	11	16
ลูกค้า 5	39	73	56	60	49	39	38	0	66	8	33	59	35	89	40	23	45	36
ลูกค้า 6	77	16	58	81	89	33	37	66	0	62	95	100	95	66	27	89	41	52
ลูกค้า 7	46	67	59	65	56	33	37	8	62	0	34	66	43	91	35	29	46	39
ลูกค้า 8	64	98	87	86	70	63	70	33	95	34	0	76	42	122	68	21	78	69
ลูกค้า 9	23	113	49	26	11	88	64	59	100	66	76	0	35	77	83	55	59	48
ลูกค้า10	30	105	62	51	31	73	60	35	95	43	42	35	0	97	72	21	62	49
ลูกค้า11	67	82	35	51	71	84	56	89	66	91	122	77	97	0	74	105	45	53
ลูกค้า12	60	33	52	71	72	9	23	40	27	35	68	83	72	74	0	63	33	38
ลูกค้า13	43	96	70	66	49	62	59	23	89	29	21	55	21	105	63	0	64	53
ลูกค้า14	38	56	19	40	49	42	11	45	41	46	78	59	62	45	33	64	0	13
ลูกค้า15	26	65	20	33	37	45	16	36	52	39	69	48	49	53	38	53	13	0

ภาพที่ 6 ตัวอย่างเมตริกซ์ระยะทางของคลังสินค้า 3 แห่ง และลูกค้า 15 ราย

1.1.5 แปลงข้อมูลจากเมตริกซ์ระยะทางให้อยู่ในรูปของไฟล์ข้อความ (text file) เพื่อในคำนวณในตัวแบบ

1.2 การสร้างชุดข้อมูลความต้องการของลูกค้า ทำการสุ่มตัวเลขขึ้นมาให้เท่ากับจำนวนลูกค้า

2. การหาผลเฉลยเบื้องต้น



ภาพที่ 7 แผนภาพการหาผลเฉลยเบื้องต้น

ในการหาผลเฉลยเบื้องต้น (Initial Solution) ประกอบด้วย 3 ขั้นตอน ดังแสดงในภาพที่ 7 โดยขั้นตอนแรกจะเริ่มจากการจัดกลุ่ม (Grouping) ลูกค้าให้กับคลังสินค้าแต่ละแห่ง ซึ่งจะพิจารณาจากระยะทางของการกระจัดระหว่างลูกค้ากับคลังสินค้าเป็นหลัก โดยดูจากระยะทางการกระจัดของลูกค้าว่าใกล้กับคลังสินค้าไหนมากที่สุดก็จะจัดให้ลูกค้าอยู่ในกลุ่มของคลังสินค้านั้นๆ จากนั้นจึงทำการจัดเส้นทาง (Routing) เบื้องต้นของลูกค้าให้กับรถในแต่ละคลังสินค้าโดยใช้อัลกอริทึมแบบประหยัด (Saving Algorithm) และขั้นตอนสุดท้ายจะทำการจัดเรียงลำดับ (Scheduling) ลูกค้าในเส้นทางโดยใช้วิธี Arbitrary Insertion เนื่องจากเป็นวิธีการจัดลำดับลูกค้าในเส้นทางที่มีขั้นตอนการคำนวณไปในทางเดียวกัน ไม่ซับซ้อนเกินไปนัก อย่างไรก็ตามผลที่ได้จากวิธีนี้ให้คำตอบใกล้เคียงคำตอบที่ดีที่สุด

เพื่อให้สามารถเข้าใจอัลกอริทึมแบบประหยัดได้ง่ายขึ้น เราจึงเสนอตัวอย่างในการคำนวณค่า Saving Matrix ดังต่อไปนี้

เริ่มจากการค้นหาเมตริกซ์ระยะทาง (Distance Matrix) ซึ่งบ่งบอกระยะทางระหว่างคู่ใดๆ ของตำแหน่งที่ยานพาหนะจะผ่าน ระยะทางถูกใช้เป็นการแทนที่ค่าใช้จ่ายของการเดินทางระหว่างคู่ใดๆ ของตำแหน่ง ถ้าเรารู้ค่าใช้จ่ายการขนส่งระหว่างคู่ใดๆ ของตำแหน่ง ค่าใช้จ่ายนั้นสามารถถูกใช้แทนที่ระยะทาง ระยะทาง $dist(A,B)$ ระหว่างตำแหน่ง A ที่อยู่บนพิกัด (x_A, y_A) และตำแหน่ง B ที่อยู่บนพิกัด (x_B, y_B) สามารถคำนวณได้จาก

$$dist(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

ระยะทางระหว่างคลังสินค้า (Depot) และทุกๆ คู่ของของตำแหน่งของลูกค้าถูกแสดงในตารางที่ 1 ระยะทางระหว่างทุกๆ คู่ของตำแหน่งถูกใช้เพื่อที่จะหาเมตริกซ์แบบประหยัด (Saving Matrix) ดังแสดงในตารางที่ 3

ตารางที่ 3 เมตริกซ์ระยะทาง

	Depot	ลูกค้า 1	ลูกค้า 2	ลูกค้า 3	ลูกค้า 4
ลูกค้า 1	12	0			
ลูกค้า 2	8	9	0		
ลูกค้า 3	17	8	10	0	
ลูกค้า 4	15	9	8	4	0

จากนั้นจึงทำการหาเมตริกซ์แบบประหยัด แสดงถึงการรวบรวมลูกค้าให้มาอยู่เส้นทางเดียวกัน การประหยัด (Saving) อาจจะถูกคำนวณในรูปแบบของระยะทาง เวลา หรือราคา อย่างไรก็ตาม ในตัวอย่างนี้ จะแสดงเมตริกซ์แบบประหยัด ในรูปแบบของระยะทาง เส้นทางใดๆ ถูกกำหนดเป็นลำดับของตำแหน่งต่างๆ ที่ยานพาหนะเดินทางผ่านเส้นทาง คลังสินค้า $\rightarrow$ ลูกค้า x $\rightarrow$ คลังสินค้า เริ่มต้นที่คลังสินค้าส่งของให้ลูกค้า x แล้วกลับมาที่คลังสินค้า การประหยัด $S(x,y)$ เป็นระยะทางที่ถูกคำนวณค่าการประหยัด ซึ่งถ้าเส้นทางเริ่มจากคลังสินค้า $\rightarrow$ ลูกค้า x $\rightarrow$ คลังสินค้า และ จากคลังสินค้า $\rightarrow$ ลูกค้า y $\rightarrow$ คลังสินค้า จะถูกรวมเป็นเส้นทางเดียวกันคือ จากคลังสินค้า $\rightarrow$ ลูกค้า x $\rightarrow$ ลูกค้า y $\rightarrow$ คลังสินค้า การประหยัดสามารถถูกคำนวณได้จากสูตรดังต่อไปนี้

$$S(x,y) = dist(depot,x) + dist(depot,y) - dist(x,y)$$

ตัวอย่างเช่น เราสามารถหาค่า $S(1,2) = 12 + 8 - 9 = 11$ ซึ่งเมตริกซ์การประหยัดถูกแสดงในตารางที่ 4 ขั้นตอนต่อไป เราจะใช้เมตริกซ์แบบประหยัดแจกจ่ายลูกค้าไปตามเส้นทางต่างๆ

ตารางที่ 4 เมตริกซ์แบบประหยัด

	ลูกค้า 1	ลูกค้า 2	ลูกค้า 3	ลูกค้า 4
ลูกค้า 1	0			
ลูกค้า 2	11	0		
ลูกค้า 3	21	15	0	
ลูกค้า 4	18	15	28	0

เมื่อทำการแจกจ่ายลูกค้าไปตามเส้นทางต่างๆ จะต้องพยายามประหยัดระยะทางให้ได้มากที่สุด กระบวนการทำซ้ำถูกใช้ในการแจกจ่ายนี้ เริ่มแรกสุด แต่ละลูกค้าถูกแจกจ่ายในเส้นทางของตัวเอง เส้นทางสองเส้นทางจะถูกรวมไปเป็นเส้นทางเดียว ถ้าจำนวนของสินค้าทั้งหมดทั้งสองเส้นทางไม่เกินความจุของยานพาหนะ ในแต่ละการวนซ้ำจะทำการรวมเส้นทางต่างๆ ที่มีการประหยัดที่สูงที่สุดไปไว้ในเส้นทางเดียวกัน กระบวนการนี้ถูกทำซ้ำจนกระทั่งไม่สามารถรวมเส้นทางได้อีก ในตัวอย่างนี้ เราให้ความจุของยานพาหนะเป็น 150 และตารางที่ 5 แสดงความต้องการสินค้าของแต่ละลูกค้า

ตารางที่ 5 ความต้องการสินค้าของแต่ละลูกค้า

ลูกค้าคนที่	จำนวนสินค้า
1	48
2	36
3	43
4	92

ขั้นตอนแรกสุด ค่าประหยัดที่มากที่สุดของ 28 เกิดจากการรวมเส้นทางที่ 3 และเส้นทางที่ 4 เส้นทางนี้สามารถรวมกันได้เนื่องจากจำนวนสินค้าทั้งหมดคือ 43 + 92 = 135 ยังต่ำกว่าความจุ

ของยานพาหนะ ดังนั้น ลูกค้าทั้งสองถูกรวมไปอยู่เส้นทางเดียวกันดังตารางที่ 6 และค่าประหยัด 28 ถูกกำจัดจากการพิจารณาต่อไป

ตารางที่ 6 เมตริกซ์แบบประหยัดที่ผ่านการปรับปรุงครั้งที่ 1

	เส้นทาง	ลูกค้า 1	ลูกค้า 2	ลูกค้า 3	ลูกค้า 4
ลูกค้า 1	1	0			
ลูกค้า 2	2	11	0		
ลูกค้า 3	3	21	15	0	
ลูกค้า 4	3	18	15	28	0

ค่าประหยัดที่มากที่สุดถัดไปคือ 21 ซึ่งจะรวมเส้นทางที่ 3 ของลูกค้าที่ 3 กับเส้นทางที่ 1 ของลูกค้าที่ 1 จำนวนสินค้าทั้งหมดในเส้นทางรวมคือ 48 + 43 + 92 = 183 ซึ่งเกินความจุของยานพาหนะ ดังนั้น เส้นทางทั้งสองเส้นทางไม่สามารถรวมกันได้ ค่าประหยัดที่มากที่สุดถัดไปคือ 18 ซึ่งจะรวมเส้นทางที่ 3 ของลูกค้าที่ 4 กับเส้นทางที่ 1 ของลูกค้าที่ 1 ซึ่งได้แสดงให้เห็นจากค่าประหยัด 21 แล้วว่าจำนวนสินค้าของเส้นทางรวมเกินความจุของรถ ดังนั้น เส้นทางทั้งสองเส้นทางไม่สามารถรวมกันได้ ค่าประหยัดที่มากที่สุดถัดไปคือ 15 ซึ่งจะรวมเส้นทางที่ 2 ของลูกค้าที่ 2 กับเส้นทางที่ 3 ของลูกค้าที่ 3 จำนวนสินค้าทั้งหมดในเส้นทางรวมคือ 36 + 43 + 92 = 171 ซึ่งเกินความจุของยานพาหนะ ดังนั้น เส้นทางทั้งสองเส้นทางไม่สามารถรวมกันได้ ค่าประหยัดที่มากที่สุดถัดไปคือ 15 ซึ่งจะรวมเส้นทางที่ 2 ของลูกค้าที่ 2 กับเส้นทางที่ 3 ของลูกค้าที่ 4 เราได้แสดงให้เห็นจากค่าประหยัดข้างต้นแล้วว่า เส้นทางทั้งสองไม่สามารถรวมกันได้เช่นกัน ค่าประหยัดที่มากที่สุดสุดท้ายคือ 11 ซึ่งจะรวมเส้นทางที่ 1 ของลูกค้าที่ 1 กับเส้นทางที่ 2 ของลูกค้าที่ 2 จำนวนสินค้าทั้งหมดในเส้นทางคือ 48 + 36 = 84 ซึ่งต่ำกว่าความจุของยานพาหนะ ดังนั้น การรวมเส้นทางนี้ถูกยอมรับ และถูกแสดงดังตารางที่ 7

ตารางที่ 7 เมตริกซ์แบบประหยัดที่ผ่านการปรับปรุงครั้งที่ 2

	เส้นทาง	ลูกค้า 1	ลูกค้า 2	ลูกค้า 3	ลูกค้า 4
ลูกค้า 1	1	0			
ลูกค้า 2	1	11	0		
ลูกค้า 3	3	21	15	0	
ลูกค้า 4	3	18	15	28	0

ผลลัพธ์สุดท้าย เราสามาถจัดเส้นทางได้เป็นสองเส้นทาง คือ $\{1,2\}$ และ $\{3,4\}$ ขณะที่แต่ละเส้นทางจะถูกแจกจ่ายให้กับยานพาหนะคันเดียว

เมื่อผ่านอัลกอริทึมแบบประหยัดแล้ว จึงทำการจัดลำดับลูกค้าในแต่ละเส้นทางโดยใช้วิธี Arbitrary Insertion ในการจัดลำดับมีขั้นตอนดังต่อไปนี้

2.1. สร้างเส้นทางเริ่มต้น เลือกโหนดเริ่มต้นของเส้นทางขึ้นมาหนึ่งโหนด

2.2. สุ่มเลือกโหนดที่ไม่ได้อยู่ในเส้นทาง

2.3. แทรกโหนดที่สุ่มขึ้นมาในตำแหน่งต่างๆของเส้นทาง โดยพิจารณาการแทรกลูกค้าในตำแหน่งที่ทำให้ค่าระยะทางรวมในการแทรกเพิ่มน้อยที่สุด

2.4. ทำซ้ำในขั้นตอนที่ 2 และ ขั้นตอนที่ 3 ต่อไปเรื่อยๆ จนกระทั่งลูกค้าที่เหลือถูกรวมอยู่ในเส้นทางทั้งหมด

เพื่อให้สามารถเข้าใจวิธี Arbitrary Insertion ได้ง่ายขึ้น จึงเสนอตัวอย่างในการคำนวณดังต่อไปนี้ กำหนดให้เส้นทางนี้มีลูกค้าทั้งหมด 4 คน โดยระยะทางระหว่างคลังสินค้าไปยังลูกค้าและระยะทางระหว่างลูกค้าไปยังลูกค้าถูกแสดงในตารางที่ 3 เริ่มจากการสร้างเส้นทางเริ่มต้น เลือกโหนดเริ่มต้นของเส้นทางขึ้นมาหนึ่งโหนด ซึ่งแสดงในภาพที่ 8

ภาพที่ 8 การสร้างเส้นทางเริ่มต้นของวิธี Arbitrary Insertion

จากนั้นจึงทำการสุ่มเลือกโหนดที่ไม่ได้อยู่ในเส้นทางขึ้นมา 1 โหนด ซึ่งโหนดที่ยังไม่ได้อยู่ในเส้นทางคือ ลูกค้าคนที่ 1, 2 และ 4 สมมติสุ่มได้ลูกค้าคนที่ 1 จึงทำการแทรกลูกค้าคนที่ 1 เข้าไปยังตำแหน่งต่างๆของเส้นทางแล้ว จึงทำการคำนวณระยะทางรวมได้ดังแสดงในภาพที่ 9

ระยะทางรวม = 17 + 8 + 12 = 37 หน่วย

ระยะทางรวม = 12 + 8 + 17 = 37 หน่วย

ภาพที่ 9 ขั้นตอนการแทรกตำแหน่งลูกค้าของวิธี Arbitrary Insertion ครั้งที่ 1

จะเห็นได้ว่าระยะทางของทั้งสองเส้นทางมีค่าเท่ากัน จะทำการเลือกเส้นทางไหนก็ได้ จึงทำการเลือกเส้นทางที่เดินทางจากคลังสินค้าไปยังลูกค้าคนที่ 3 ก่อน แล้วเดินทางต่อไปยังลูกค้าคนที่ 1 จากนั้นจึงเดินทางกลับมายังคลังสินค้าเดิม เป็นเส้นทางเริ่มต้นในขั้นตอนถัดไป

จากนั้นจึงทำการสุ่มเลือกลูกค้าที่ยังไม่ได้อยู่ในเส้นทางคนต่อไปขึ้นมา ซึ่งจะเหลือลูกค้าคนที่ 2 และลูกค้าคนที่ 4 สมมติว่าลูกค้าที่ถูกสุ่มได้คือลูกค้าคนที่ 2 จะสามารถแทรกลูกค้าคนที่ 2 ลงไปในเส้นทาง ดังแสดงในภาพที่ 10

ระยะทางรวม = 8 + 10 + 8 + 12 = 38 หน่วย

ระยะทางรวม = 17 + 10 + 9 + 12 = 48 หน่วย

ภาพที่ 10 ขั้นตอนการแทรกตำแหน่งลูกค้าของวิธี Arbitrary Insertion ครั้งที่ 2

ระยะทางรวม = 17 + 8 + 9 + 8 = 42 หน่วย

ภาพที่ 10 (ต่อ)

ผลรวมของระยะทางเมื่อทำการส่งลูกค้าทั้งสามคนแล้ว จะเห็นได้ว่าระยะทางรวมของเส้นทางที่ทำการส่งสินค้าจากคลังสินค้า → ลูกค้าคนที่ 2 → ลูกค้าคนที่ 3 → ลูกค้าคนที่ 1 แล้วกลับไปคลังสินค้า มีค่าน้อยที่สุดจึงเลือกเส้นทางนี้เป็นเส้นทางเริ่มต้นทำการแทรกในขั้นตอนต่อไป ซึ่งเหลือเพียงลูกค้าคนที่ 4 ที่ยังไม่ได้อยู่ในเส้นทาง จึงทำการแทรกลูกค้าคนที่ 4 ในตำแหน่งต่างๆ ดังนี้

ระยะทางรวม = 15 + 8 + 10 + 8 + 12 = 53 หน่วย

ระยะทางรวม = 8 + 8 + 4 + 8 + 12 = 40 หน่วย

ระยะทางรวม = 8 + 10 + 4 + 9 + 12 = 43 หน่วย

ระยะทางรวม = 8 + 10 + 8 + 9 + 15 = 50 หน่วย

ภาพที่ 11 ขั้นตอนการแทรกตำแหน่งลูกค้าของวิธี Arbitrary Insertion ครั้งที่ 3

3. การปรับปรุงผลเฉลยโดยใช้อัลกอริทึมเชิงพันธุกรรม

อัลกอริทึมเชิงพันธุกรรม (Genegic algorithm) ที่ใช้แก้ปัญหาในงานวิจัยจะทำการกำหนดค่าสำหรับองค์ประกอบต่างๆ ดังนี้

3.1 สร้างโครโมโซมจากผลเฉลยเบื้องต้น เพื่อใช้เป็นต้นแบบของโครโมโซมพ่อ และโครโมโซมแม่ที่มีความหลากหลายมากขึ้น ซึ่งยีนส์แต่ละตัวบ่งบอกถึงลักษณะดังนี้คือ มาจากคลังสินค้าที่เท่าไหร่ (iDepot), เส้นทางไหน (iRoute), และเป็นลูกค้าคนที่เท่าไหร่ของเส้นทางนั้น (iPos) โดยกำหนดให้เซตของยีนส์แต่ละตัวเป็นดังนี้ (iDepot, iRout, iPos) ยกตัวอย่างเช่น โครโมโซมชุดหนึ่งจะประกอบด้วย {(1, 1, 2), (1, 1, 1), (1, 2, 1), (1, 3, 1)} จากนั้นจึงทำการสุ่มยีนส์ขึ้นมาหนึ่งตัว เพื่อแทนที่ยีนส์ตัวใดตัวหนึ่งที่สุ่มขึ้นมาจากโครโมโซม

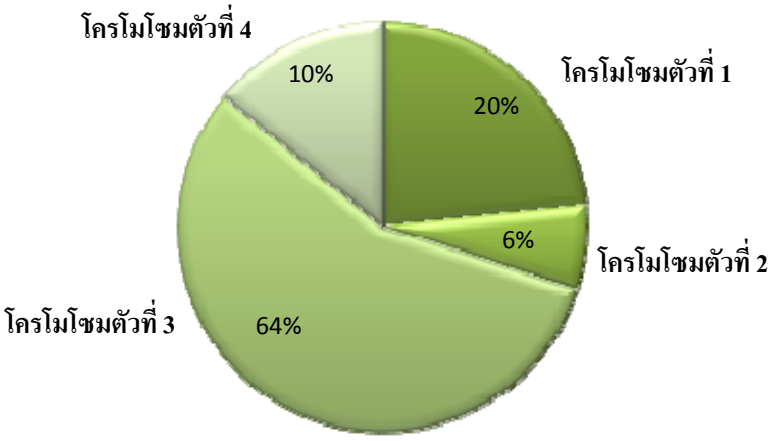
3.2 สร้างโครโมโซมพ่อและโครโมโซมแม่ โดยการสุ่มโครโมโซมจากผลเฉลยเบื้องต้นขึ้นมาหนึ่งโครโมโซม จากนั้นจึงทำการสุ่มยีนส์ในโครโมโซมดังกล่าวขึ้นมา 2 ตัว เพื่อสลับตำแหน่งกัน ยกตัวอย่างเช่น โครโมโซมตั้งต้นคือ {(1, 1, 2), (1, 1, 1), (1, 2, 1), (1, 1, 3)} จากนั้นทำการสุ่มยีนส์ขึ้นมาดังนี้ (1, 1, 1), (1, 2, 1) นำยีนส์ที่สุ่มได้มาสลับตำแหน่งกันได้โครโมโซมใหม่คือ {(1, 1, 2), (1, 2, 1), (1, 1, 1), (1, 1, 3)}

3.3 ในงานวิจัยนี้จะใช้วิธีวงล้อสัดส่วนค่าความเหมาะสม (Roulette wheel selection) ในการคัดเลือกสายพันธุ์ เนื่องจากวิธีนี้จะนำโครโมโซมทุกตัวของประชากรรุ่นนั้นมาเขียนเป็นช่องซึ่งแต่ละช่องมีขนาดตามสัดส่วนของค่าความเหมาะสม ดังนั้นโครโมโซมที่มีค่าความเหมาะสมมากกว่าจึงมีโอกาสถูกเลือกมากกว่า ซึ่งมีวิธีการดังนี้

3.3.1 คำนวณค่าความน่าจะเป็นในการคัดเลือก (Probability of selection: P_s) ซึ่งแสดงในตารางที่ 8 เพื่อนำไปสร้างวงล้อสัดส่วนค่าความเหมาะสมดังแสดงในภาพที่ 12

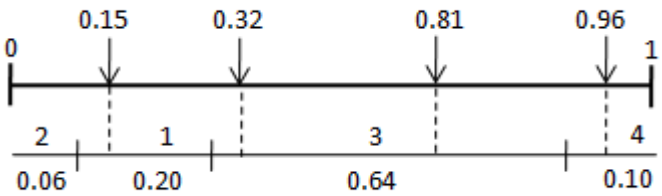
ตารางที่ 8 การคำนวณค่าความน่าจะเป็นในการคัดเลือก

โครโมโซม	ยีนส์	ค่าความเหมาะสม (f)	$\sum f \div f$	P_s
1	(1, 1, 2), (1, 1, 1), (1, 2, 1), (1, 1, 3)	169	6.86	0.20
2	(1, 1, 2), (1, 2, 1), (1, 1, 1), (1, 1, 3)	576	2.01	0.06
3	(1, 1, 1), (1, 1, 2), (1, 1, 3), (1, 2, 1)	54	21.48	0.64
4	(1, 1, 3), (1, 2, 1), (1, 1, 1), (1, 1, 2)	361	3.21	0.10
		$\sum f = 1160$	$\sum = 33.56$	$\sum = 1$



ภาพที่ 12 วงล้อสัดส่วนค่าความเหมาะสมในการเลือกโครโมโซม

3.3.2 สร้างตัวเลขสุ่มที่มีค่าอยู่ระหว่าง 0 ถึง 1 ที่อิสระต่อกัน เท่ากับขนาดของประชากร (Population size) ที่ต้องการเก็บค่า สมมติให้ขนาดของประชากรเท่ากับ 4 ค่า และค่าที่สุ่มได้เป็นดังนี้ 0.81, 0.32, 0.15, 0.96 ตามลำดับ ดังแสดงในภาพที่ 13



ภาพที่ 13 การสุ่มเลือกโครโมโซมจากวงล้อสัดส่วนค่าความเหมาะสม

3.3.3 หลังจากการ reproduction เราจะได้ประชากรที่มีโครโมโซม 4 ตัวใหม่ ในที่นี้มีโครโมโซมตัวที่ 3 ถูกเลือกขึ้นมา 2 ที และโครโมโซมตัวที่ 2 ไม่ได้ถูกเลือกซึ่งแสดงในตารางที่ 9

ตารางที่ 9 โครโมโซมที่ได้ชุดใหม่ภายหลังจากการ Reproduction

โครโมโซมเก่า	โครโมโซมใหม่	ยีนส์	ค่าความเหมาะสม: f (หน่วย)
3	1	54231	54
1	2	12345	169
4	3	31524	361
3	4	54231	54

3.4 ในการผสมยีนส์โดยการข้ามสายพันธุ์ (Crossover) นั้นมีลักษณะหลายๆ แบบด้วยกัน การข้ายสายพันธุ์จะเกิดขึ้นโดยการสับเปลี่ยนค่าของยีนส์ระหว่างโครโมโซม 2 โครโมโซมของโครโมโซมพ่อและแม่ ทำให้ได้ประชากรรุ่นลูก (Offspring population) 2 โครโมโซม และในการควบคุมการข้ามสายพันธุ์จะนำค่าความน่าจะเป็นของการข้ามสายพันธุ์ (Crossover probability: P_c) ในงานวิจัยนี้จะให้ค่า $P_c = 0.8$ และวิธีการข้ามสายพันธุ์มีขั้นตอนดังต่อไปนี้

3.4.1 สุ่มเลือกโครโมโซมขึ้นมา 2 ชุด เพื่อเป็นโครโมโซมพ่อและโครโมโซมแม่

โครโมโซมพ่อ

ลูกค้าคนที่	1	2	3	4	5
	(1, 2, 3)	(1, 5, 3)	(2, 5, 3)	(2, 1, 4)	(1, 4, 2)

โครโมโซมแม่

ลูกค้าคนที่	1	2	3	4	5
	(2, 1, 4)	(1, 4, 2)	(1, 2, 3)	(2, 5, 3)	(1, 5, 3)

ภาพที่ 14 ตัวอย่างของโครโมโซมพ่อและโครโมโซมแม่ในกระบวนการข้ามสายพันธุ์

3.4.2 สุ่มเลือกตำแหน่งยีนส์เพื่อทำการสลับที่จากโครโมโซมพ่อและโครโมโซมแม่ โดยที่ จำนานการสุ่มเท่ากับ = [[จำนวนลูกค้า / 4]]

โครโมโซมพ่อ

ลูกค้าคนที่	1	2	3	4	5
	(1, 2, 3)	(1, 5, 3)	(2, 5, 3)	(2, 1, 4)	(1, 4, 2)

โครโมโซมแม่

ลูกค้าคนที่	1	2	3	4	5
	(2, 1, 4)	(1, 4, 2)	(1, 2, 3)	(2, 5, 3)	(1, 5, 3)

ภาพที่ 15 ลักษณะการสุ่มตำแหน่งของยีนส์ในกระบวนการข้ามสายพันธุ์

3.4.3 การสลับตำแหน่งยีนส์ในโครโมโซมของลูกคนที่ 1 ทำได้โดยเก็บค่ายีนส์ของโครโมโซมพ่อที่ไม่ตรงกับค่ายีนส์ที่ถูกสุ่มได้ในโครโมโซมแม่ไว้ในตำแหน่งเดิมแสดงในภาพที่ 16

นำยีนส์ที่สุ่มได้จากโครโมโซมแม่มาจัดเก็บในตำแหน่งที่ว่างตามลำดับของโครโมโซมแม่ และในโครโมโซมของลูกตัวที่ 2 ก็ทำในลักษณะเดียวกัน ดังแสดงในภาพที่ 17

โครโมโซมลูก 1

ลูกค้าคนที่	1	2	3	4	5
		(1, 5, 3)		(2, 1, 4)	

โครโมโซมลูก 2

ลูกค้าคนที่	1	2	3	4	5
		(1, 4, 2)		(2, 5, 3)	

ภาพที่ 16 การเก็บค่าของการสลับตำแหน่งยีนส์ของกระบวนการข้ามสายพันธุ์

โครโมโซมลูก 1

ลูกค้าคนที่	1	2	3	4	5
	(1, 4, 2)	(1, 5, 3)	(1, 2, 3)	(2, 1, 4)	(2, 5, 3)

โครโมโซมลูก 2

ลูกค้าคนที่	1	2	3	4	5
	(1, 2, 3)	(1, 4, 2)	(1, 5, 3)	(2, 5, 3)	(2, 1, 4)

ภาพที่ 17 การเรียงลำดับของยีนส์ที่ถูกสุ่มเลือกในกระบวนการข้ามสายพันธุ์

3.5 การผ่าเหล่า (Mutation) เป็นกระบวนการที่เกิดขึ้นหลังกระบวนการข้ามสายพันธุ์ นั่นหมายความว่าได้โครโมโซมลูกที่เกิดจากการผสมระหว่างโครโมโซมพ่อและแม่แล้ว จึงนำโครโมโซมลูกมาดำเนินการผ่าเหล่า ในการผ่าเหล่านั้นจะทำให้เกิดการเปลี่ยนแปลงและมีลักษณะใหม่ๆ ขึ้น จึงทำให้ได้ผลลัพธ์ในลักษณะที่แตกต่างออกไปจากเดิม ในงานวิจัยนี้จะใช้วิธี 2-SWAP ในกระบวนการผ่าเหล่า โดยจะทำการสุ่มเลือกตำแหน่งขึ้นมา 2 จุด แล้วทำการสลับตำแหน่งของยีนส์ในระหว่างตำแหน่งที่ถูกสุ่มแบบย้อนกลับ (Inverse) แสดงในภาพที่ 18 และ 19 การควบคุมการผ่าเหล่าใช้ความน่าจะเป็นของการผ่าเหล่า (Mutation probability: P_m) เท่ากับ 0.01

ลูกค้าคนที่	1	2	3	4	5	6
	(1, 2, 3)	(1, 5, 3)	(2, 5, 3)	(2, 1, 4)	(1, 4, 2)	(2, 5, 7)

↓ ↓

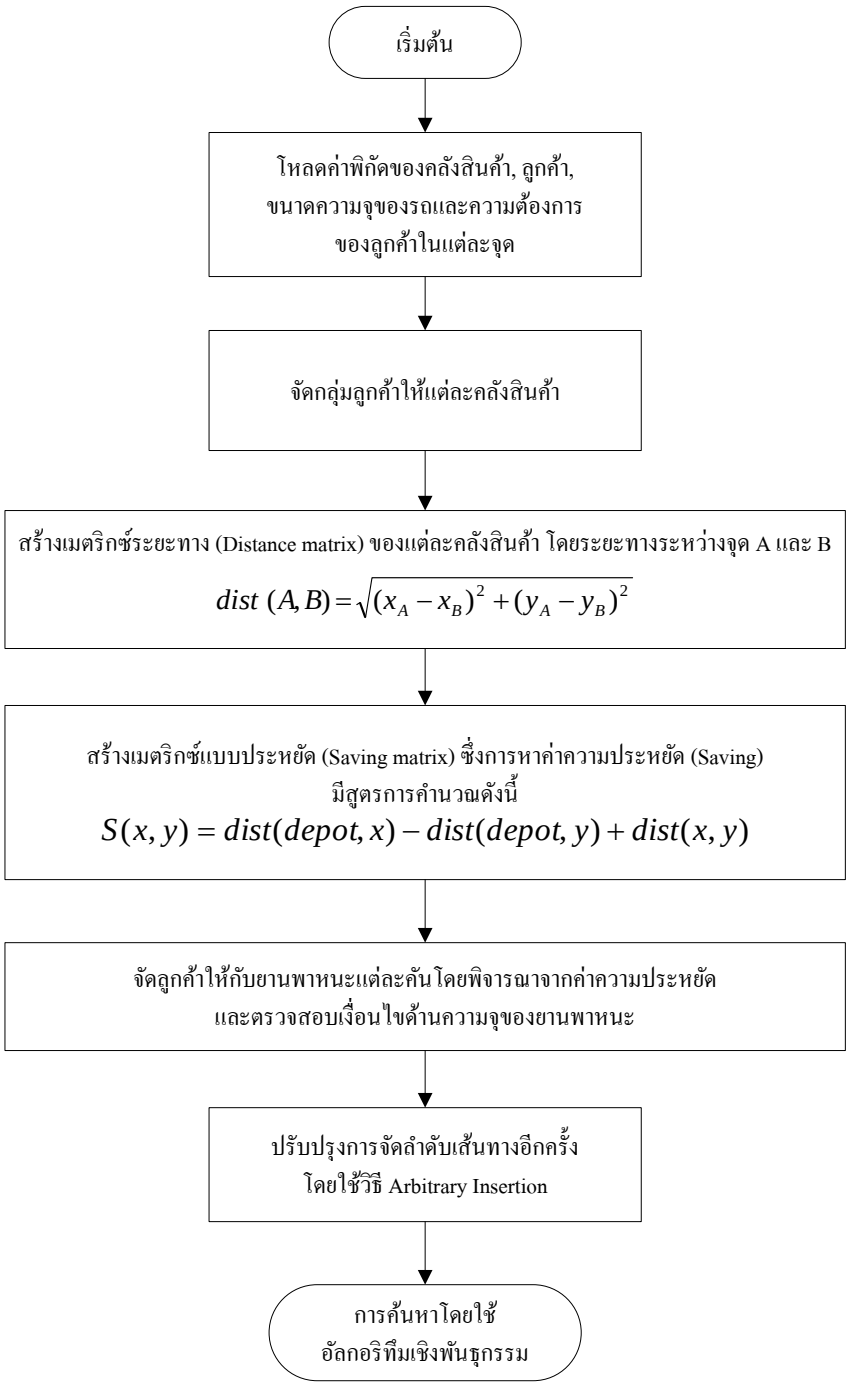
ลูกค้าคนที่	1	2	3	4	5	6
	(1, 2, 3)	(1, 5, 3)	(2, 5, 3)	(2, 1, 4)	(1, 4, 2)	(2, 5, 7)

ภาพที่ 18 การสุ่มตำแหน่งในกระบวนการผ่าเหล่า

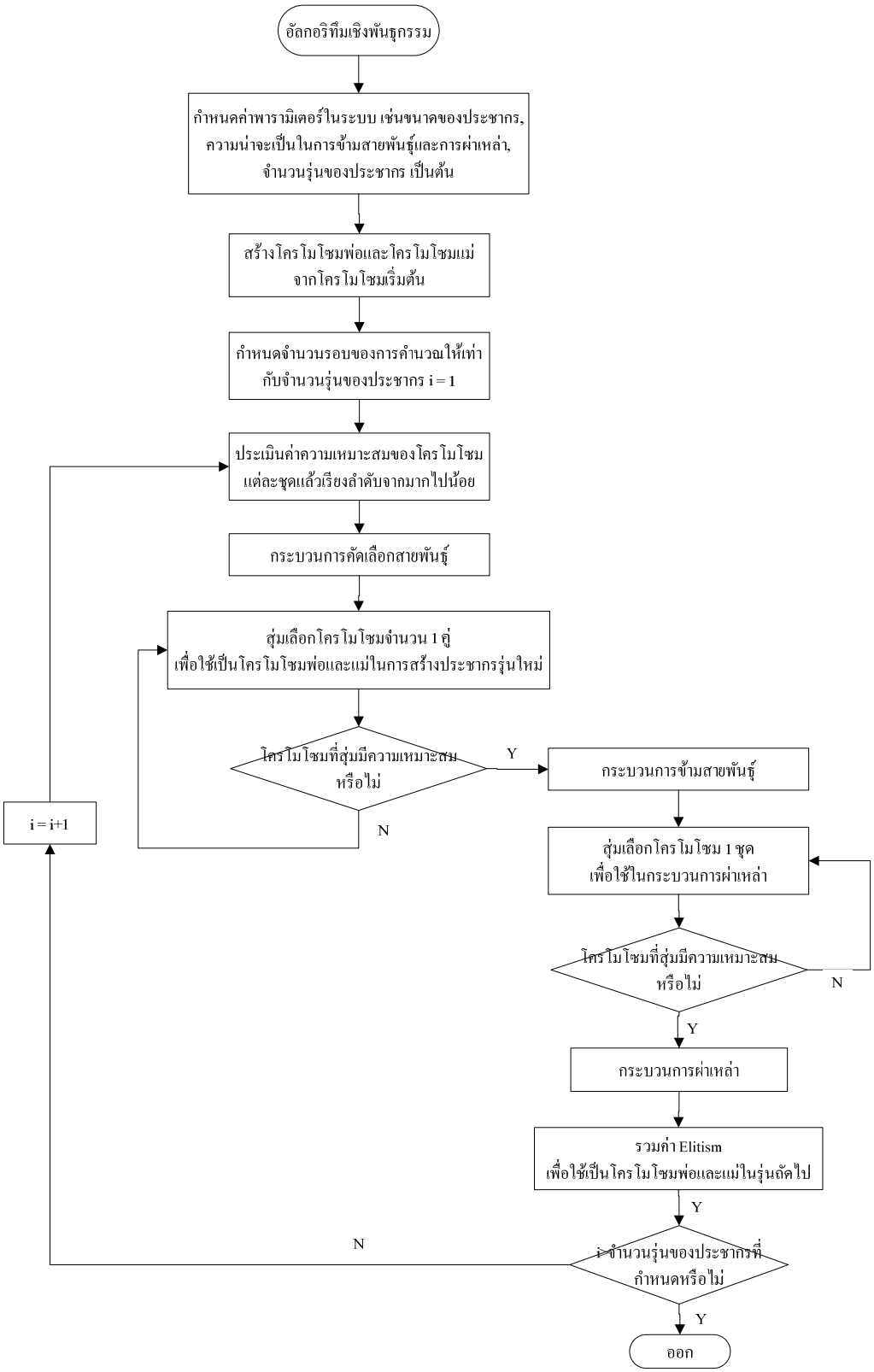
ลูกค้าคนที่	1	2	3	4	5	6
	(1, 2, 3)	(1, 2, 4)	(2, 1, 4)	(2, 5, 3)	(1, 5, 3)	(2, 5, 7)

ภาพที่ 19 การสลับตำแหน่งในกระบวนการผ่าเหล่า

ในแต่ละขั้นตอนของอัลกอริทึมเชิงพันธุกรรม โครโมโซมใหม่จะถูกยอมรับก็ต่อเมื่อจำนวนสินค้าในแต่ละขั้นตอนไม่เกินความจุของยานพาหนะ ขั้นตอนการหาผลเฉลยเบื้องต้นและการปรับปรุงผลเฉลยโดยการใช้อัลกอริทึมเชิงพันธุกรรมสามารถแสดงเป็นแผนผังดังภาพที่ 20 และภาพที่ 21



ภาพที่ 20 แผนผังขั้นตอนการหาผลเฉลยเบื้องต้น



ภาพที่ 21 แผนผังขั้นตอนการปรับปรุงผลเฉลยโดยใช้อัลกอริทึมเชิงพันธุกรรม

ผลและวิจารณ์

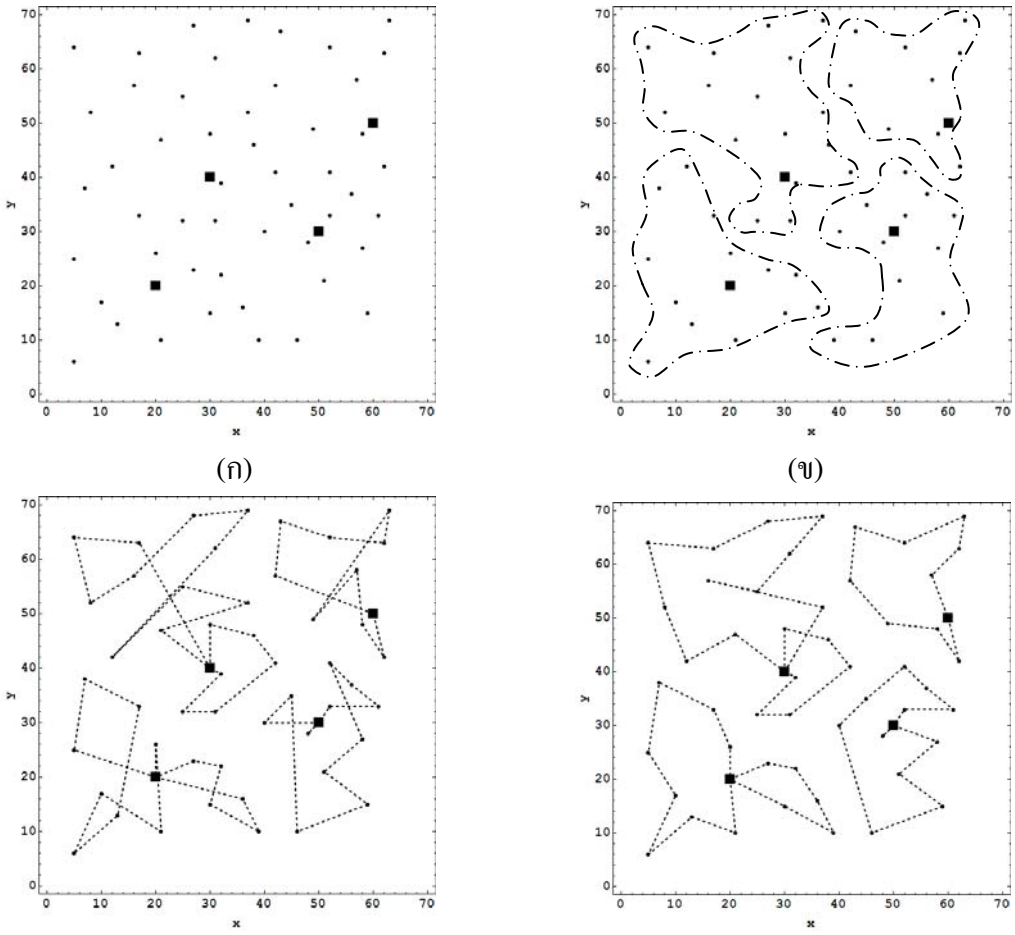
งานวิจัยนี้ได้นำเสนอการจัดเส้นทางยานพาหนะโดยใช้หลักการทำงานของอัลกอริทึมเชิงพันธุกรรม ข้อมูลที่นำมาศึกษาจะแบ่งออกเป็น 13 ปัญหา ดังแสดงในตารางที่ 10

ตารางที่ 10 ลักษณะของปัญหาที่ใช้ในการทำวิจัย

ปัญหาที่	จำนวนคลังสินค้า	จำนวนลูกค้า	ความจุรถ
1	4	50	80
2	4	50	160
3	5	75	140
4	2	100	100
5	2	100	200
6	3	100	100
7	4	100	100
8	3	100	200
9	4	100	200
10	5	100	200
11	3	75	100
12	4	75	100
13	5	75	100

ชุดข้อมูลพิกัดของลูกค้าและคลังสินค้าที่ใช้ในงานวิจัยนี้ได้ถูกแสดงไว้ในภาคผนวก ข ซึ่งในปัญหาที่ 1 ถึงปัญหาที่ 13 นี้มีความหลากหลายทางด้านความจุของยานพาหนะ จำนวนคลังสินค้าและจำนวนลูกค้า ซึ่งความหลากหลายนี้จะนำพาซึ่งการศึกษาในเรื่องต่างๆ เช่นในเรื่องของเวลาในการประมวลผล การลดลงของระยะทางทั้งหมดในระบบ ตัวอย่างเช่น การเปรียบเทียบเวลาที่ใช้ในการประมวลผลหรือระยะทางทั้งหมดของระบบ ของปัญหาที่ 6 และปัญหาที่ 7 เมื่อเปลี่ยนจำนวนคลังสินค้า

การคำนวณทั้งหมดในงานวิจัยนี้ กำหนดความแม่นยำในระดับ Double precision และผลการคำนวณจะปัดเศษในทศนิยมตำแหน่งที่สอง ในตารางที่ 11 แสดงการเปรียบเทียบผลเฉลยด้านระยะทางและเวลาเมื่อผ่านอัลกอริทึมแบบประหยัด, วิธี Arbitrary Insertion และอัลกอริทึมเชิงพันธุกรรม



ภาพที่ 22 แผนภาพแสดงขั้นตอนการปรับปรุงผลเฉลยของการจัดเส้นทางยานพาหนะ

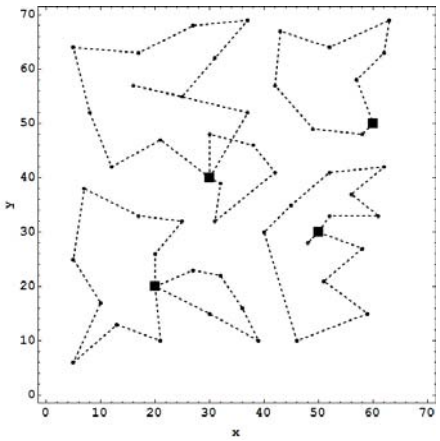
(ก) ตำแหน่งของลูกค้าและคลังสินค้าก่อนถูกจัดกลุ่ม

(ข) การจัดกลุ่มของลูกค้าในแต่ละคลังสินค้า

(ค) การจัดเรียงเส้นทางของลูกค้าหลังจากผ่านอัลกอริทึมแบบประหยัด

(ง) การจัดเรียงเส้นทางของลูกค้าหลังจากผ่าน Arbitrary Insertion

(จ) การจัดเรียงเส้นทางของลูกค้าหลังจากผ่านอัลกอริทึมเชิงพันธุกรรม



(จ)

ภาพที่ 22 (ต่อ)

จากภาพที่ 22 แสดงให้เห็นถึงขั้นตอนการปรับปรุงผลเฉลยของการจัดเส้นทางยานพาหนะในปัญหาที่ 2 ซึ่งมีคลังสินค้าทั้งหมด 4 แห่ง จำนวนลูกค้า 50 ราย และความจุของยานพาหนะเท่ากับ 160 หน่วย โดยภาพที่ 22 (ก) แสดงให้เห็นถึงตำแหน่งของลูกค้าและคลังสินค้าก่อนถูกจัดกลุ่ม จากนั้นจึงนำชุดข้อมูลมาคำนวณหาผลเฉลยเบื้องต้นการจัดกลุ่มของลูกค้าในแต่ละคลังสินค้า โดยพิจารณาจากระยะทางการกระจัดของลูกค้าว่าใกล้กับคลังสินค้าไหนมากที่สุดก็จะจัดให้ลูกค้าอยู่ในกลุ่มของคลังสินค้านั้นๆ ดังแสดงในภาพที่ 22 (ข) จากนั้นจึงทำการจัดเส้นทางเบื้องต้นของลูกค้าให้กับรถในแต่ละคลังสินค้าโดยใช้อัลกอริทึมแบบประหยัด (Saving Algorithm) จะสังเกตได้ว่าในแต่ละคลังสินค้าจะสร้างเส้นทางย่อยขึ้น แต่ลูกค้าในแต่ละเส้นทางย่อยนั้นยังไม่ถูกจัดเรียงลำดับการส่ง ดังแสดงในภาพที่ 22 (ค) และขั้นตอนสุดท้ายของการหาผลเฉลยเบื้องต้นจะทำการจัดเรียงลำดับลูกค้าในแต่ละเส้นทางโดยใช้วิธี Arbitrary Insertion แสดงให้เห็นถึงเส้นทางที่ถูกจัดเรียงเป็นระเบียบมากขึ้น ดังแสดงในภาพที่ 22 (ง) จากนั้นจึงทำการปรับปรุงผลเฉลยโดยใช้อัลกอริทึมเชิงพันธุกรรม จะเห็นได้ว่าลูกค้าบางรายจะถูกย้ายไปยังสินคลังอื่น เพื่อให้ได้เส้นทางที่มีความประหยัดมากขึ้น ดังแสดงในภาพที่ 22 (จ)

ค่าที่ได้จากการผ่านอัลกอริทึมแบบประหยัดเป็นค่าที่ยังไม่ผ่านการจัดเรียงลูกค้าในแต่ละเส้นทาง อย่างไรก็ตามเมื่อผ่านกระบวนการ Arbitrary Insertion แล้วระยะทางในระบบจะลดลงอย่างเห็นได้ชัด หลังจากนั้นเส้นทางจะถูกปรับปรุงโดยอัลกอริทึมเชิงพันธุกรรมอีกครั้งหนึ่ง ค่าที่ได้จะลดลงจากวิธี Arbitrary Insertion ในระดับหนึ่ง ค่าที่ได้จากอัลกอริทึมเชิงพันธุกรรมในแต่ละครั้งจะได้ค่าที่ไม่เท่ากันขึ้นอยู่กับการสร้างชุดโครโมโซมต้นแบบ ดังนั้นค่าที่นำเสนอในตารางที่ 11

เป็นค่าที่ดีที่สุดที่สามารถคำนวณได้จากโครโมโซมต้นแบบที่ถูกสุ่มขึ้นมา แต่อย่างไรก็ตามอาจจะมีค่าที่ดีกว่านี้ได้ถ้ามีโครโมโซมต้นแบบที่มีความเหมาะสมมากกว่า

ตารางที่ 11 สรุปผลการคำนวณระยะทางและเวลาเมื่อผ่านอัลกอริทึมแบบประหยัด, Arbitrary Insertion และอัลกอริทึมเชิงพันธุกรรม

ปัญหา	วิธีการแบบประหยัด		Arbitrary Insertion		อัลกอริทึมเชิงพันธุกรรม	
	ระยะทาง (หน่วย)	เวลา (วินาที)	ระยะทาง (หน่วย)	เวลา (วินาที)	ระยะทาง (หน่วย)	เวลา (วินาที)[1]
1	703.27	1.00	661.86	1.00	590.43	5.28
2	646.14	1.00	530.69	1.00	486.69	4.16
3	888.30	1.00	709.45	1.00	662.45	4.54
4	1406.68	8.00	1114.66	1.00	1058.33	29.83
5	1418.68	9.00	919.90	1.00	792.90	9.93
6	1174.47	5.00	943.00	1.00	897.01	23.45
7	1163.23	3.00	962.94	1.00	911.94	20.55

หมายเหตุ [1] เวลาเฉลี่ยต่อรุ่นของประชากร

ตารางที่ 11 (ต่อ)

ปัญหา	อัลกอริทึมแบบประหยัด		Arbitrary Insertion		อัลกอริทึมเชิงพันธุกรรม	
	ระยะทาง (หน่วย)	เวลา (วินาที)	ระยะทาง (หน่วย)	เวลา (วินาที)	ระยะทาง (หน่วย)	เวลา (วินาที)[1]
8	1209.13	2.00	837.62	0.00	785.45	7.04
9	1072.42	1.00	822.12	0.00	779.22	6.83
10	1154.60	1.00	810.64	0.00	767.93	6.72
11	1005.92	1.00	874.17	0.00	837.15	6.19
12	1016.66	1.00	866.12	0.00	812.22	5.77
13	923.69	0.00	770.27	0.00	723.13	5.08

หมายเหตุ [1] เวลาเฉลี่ยต่อรุ่นของประชากร

ตารางที่ 12 แสดงจำนวนโหนดเฉลี่ยต่อเส้นทาง

ปัญหาที่	จำนวนโหนดเฉลี่ยต่อเส้นทาง
1	5.85
2	9.14
3	7.77
4	7.52
5	11.09
6	8.25
7	7.88
8	13.11
9	14.5
10	14.6
11	6.17
12	6.69
13	7.00

ลักษณะของปัญหาทั้ง 13 ปัญหานี้ เราสามารถแบ่งได้ออกเป็นสองประเภทคือ จำนวนคลังสินค้าต่างกัน และความจุของยานพาหนะต่างกัน ในการพิจารณาผลที่มีต่อเวลาที่ใช้ในการประมวลผลและระยะทางของปัญหาที่มีจำนวนคลังสินค้าต่างกัน ปัญหาที่ {4, 6, 7}, {5, 8, 9, 10} และ {11, 12, 13} ถูกใช้ในการศึกษาครั้งนี้ จะสังเกตเห็นได้ว่าเวลาที่ใช้ในการคำนวณของปัญหาที่ 7 ในเซตของปัญหาที่ {4, 6, 7} ซึ่งมีจำนวนคลังสินค้ามากที่สุด มีค่าน้อยกว่าปัญหาที่เหลือทั้งสอง เนื่องจากการเพิ่มขึ้นของคลังสินค้า ทำให้การกระจายตัวของเส้นทางมีมากขึ้นตามไปด้วย ซึ่งมีผลทำให้เวลาในการประมวลผลมีค่าน้อยลง และเมื่อพิจารณาถึงระยะทางทั้งหมดของระบบแล้ว การเพิ่มของจำนวนคลังสินค้าจะทำให้ระยะทางรวมของระบบลดลงตามไปด้วย แต่จะต้องคำนึงถึงปัจจัยด้านจำนวนโหนดเฉลี่ยในแต่ละเส้นทางควบคู่กันไป ซึ่งจำนวนโหนดเฉลี่ยต่อเส้นทางจะถูกแสดงในตารางที่ 12 กล่าวคือ เมื่อจำนวนโหนดเฉลี่ยในเส้นทางมีมากขึ้น จะส่งผลให้ระยะทางในระบบลดลง

ในการพิจารณาผลที่มีต่อเวลาที่ใช้ในการคำนวณและระยะทางของปัญหาที่มีจำนวนความจุของยานพาหนะที่ต่างกัน ซึ่งจะใช้ปัญหาที่ {1, 2}, {4, 5}, {6, 8}, {7, 9} และ {3, 13} ในการศึกษาครั้งนี้ สามารถสังเกตได้ว่า ปัญหาที่มีความจุรถมากกว่า จะใช้เวลาในการประมวลผลกับระยะทางรวมของระบบน้อยกว่าของปัญหาที่มีความจุรถน้อย เนื่องจากเมื่อความจุรถเพิ่มมากขึ้น จำนวนลูกค้าในแต่ละเส้นทางจะมากขึ้น ทำให้จำนวนเส้นทางของทั้งระบบลดลง เวลาที่ใช้คำนวณในวิธี Arbitrary Insertion และอัลกอริทึมเชิงพันธุกรรมจึงลดลงตามไปด้วย การจัดระบบที่กระทัดรัดนี้มีผลต่อระยะทางของระบบที่ลดลง

สรุปและข้อเสนอแนะ

สรุป

ปัญหาการจัดเส้นทางยานพาหนะที่มีคลังสินค้าหลายแห่งเป็นปัญหาที่มีปัจจัยเกี่ยวข้องหลายประการ เช่น จำนวนยานพาหนะที่ใช้ในการขนส่งสินค้า ขีดจำกัดด้านความจุของยานพาหนะ จำนวนคลังสินค้า การจำกัดระยะทางในแต่ละเส้นทาง ความต้องการในการขนส่ง เป็นต้น งานวิจัยนี้จึงจัดทำขึ้นเพื่อทำการวิเคราะห์ปัญหาการจัดเส้นทางยานพาหนะ โดยมีวัตถุประสงค์เพื่อหาระยะทางที่ใช้ในการขนส่งสินค้าน้อยที่สุด เนื่องจากปัญหาที่ทำการวิจัยเป็นปัญหาที่ไม่สามารถแก้ปัญหาได้ภายในระยะเวลารวดเร็วเมื่อปัญหามีขนาดใหญ่ ดังนั้นการหาผลเฉลยที่ดีที่สุดจึงไม่สามารถทำได้ ในงานวิจัยนี้ได้นำเสนอวิธีการแก้ปัญหาโดยใช้อัลกอริทึมแบบประหยัดในการจัดกลุ่มให้ลูกค้าและจัดเรียงลำดับลูกค้าในแต่ละเส้นทางโดยใช้วิธี Arbitrary Insertion ในการหาผลเฉลยเบื้องต้น จากนั้นจึงนำหลักการของอัลกอริทึมเชิงพันธุกรรมมาประยุกต์ใช้ในการปรับปรุงผลเฉลย โดยใช้โปรแกรม Mathematica เป็นเครื่องมือในการประมวลผล จากการทดลองปรากฏว่าค่าที่ผ่านกระบวนการ Arbitrary Insertion แล้วระยะทางในระบบจะลดลงอย่างเห็นได้ชัด จากนั้นเส้นทางจะถูกปรับปรุงโดยอัลกอริทึมเชิงพันธุกรรมอีกครั้งหนึ่ง ค่าที่ได้จะลดลงจากวิธี Arbitrary Insertion ในระดับหนึ่ง เนื่องจากค่าที่ใช้การคำนวณในอัลกอริทึมเชิงพันธุกรรมนี้ถูกจำกัดโครงสร้างไว้ ดังนั้นค่าเหล่านี้จะไม่มีความหลากหลายตามหลักวิวัฒนาการ ค่าที่ได้จากอัลกอริทึมเชิงพันธุกรรมในแต่ละครั้งจะได้ค่าที่ไม่เท่ากันขึ้นอยู่กับการสร้างชุดโครโมโซมต้นแบบที่ทำการสุ่มขึ้นมา

ข้อเสนอแนะ

จากตารางเวลาข้างต้นจะเห็นได้ว่า เวลาที่ใช้ในการคำนวณมีค่ามากเกินกว่าที่จะนำไปใช้ในระดับอุตสาหกรรมได้ ทั้งนี้เนื่องจาก โปรแกรม Mathematica เป็นโปรแกรมที่ใช้หน่วยความจำสำรอง (Ram) ของเครื่องคอมพิวเตอร์เป็นจำนวนมาก อย่างไรก็ตาม เหตุที่เราเลือกที่จะใช้โปรแกรม Mathematica แทนภาษาอื่นๆ เช่น ภาษาซีหรือภาษาเบสิค ก็เนื่องมาจากว่า ระบบการคิดของโปรแกรม Mathematica ที่มองทุกๆอย่างเป็นเซต ซึ่งสามารถนำไปประยุกต์ใช้กับอัลกอริทึมแบบประหยัดและอัลกอริทึมเชิงพันธุกรรมได้ง่ายขึ้นกว่าโปรแกรมภาษาอื่นๆ

ดังนั้น ในการแก้ไขโปรแกรมเพื่อใช้ในระดับอุตสาหกรรมต่อไป เราอาจจะใช้โปรแกรม Visual Basic แทน Mathematica เพื่อความรวดเร็วในการคำนวณ และเราอาจจะหาวิธีการอื่นที่มาแทนอัลกอริทึมแบบประหยัดเพิ่มความหลากหลายของโครโมโซมเริ่มต้นก่อนเข้าสู่กระบวนการเชิงพันธุกรรม วิธีที่ถูกเลือกใหม่นั้นควรจะลดขั้นตอนของการสร้างโครโมโซมต้นแบบเพราะว่าอัลกอริทึมเชิงพันธุกรรม จะถูกกระทำทุกๆหนึ่งรอบของแต่ละโครโมโซมต้นแบบ

เอกสารและสิ่งอ้างอิง

Bodin, L. and B. Golden. 1981. Classification in vehicle routing and scheduling. **Network** 11: 97-108.

_____, _____, A. Assad and M. Ball. 1983. Routing and Scheduling of Vehicles and Crews: The State of the Art. **Comput. & Ops Res.** 10(2): 67-211.

Christofides, N. and S. Eilon. 1969. An Algorithm for the Vehicle-Dispatching Problem. **Operational Research Quarterly**. 20: 309-318.

Clark, G. and J.W. Wright. 1964. Scheduling of vehicle from a central depot to a number of delivery points. **Operations Res.** 12: 568-581.

Gillett, B. and L. Miller. 1974. A heuristic algorithm for vehicle dispatch problem. **Operations Res.** 22: 340-349.

Golden, B., A. Assad., L. Levy., and F. Gheysens. 1984. The fleet size and mix vehicle routing problem. **Comput. & Ops Res** 11(1): 49-66.

_____, T. Magnanti and H. Nguyen. 1977. Implementing vehicle routing algorithms. **Network** 7: 113-148.

Holland, J.H. 1975. **Adaptation in Natural and Artificial Systems**. University of Michigan Press, Ann Arbor.

Holmes, R.A. and R.G. Packer. 1976. A Vehicle scheduling procedure based upon saving and a solution perturbation scheme. **Operations Res. Quart.** 27(1): 83-92.

Landau, R.H., M.J. Paez and C.C. Bordeianu. 2008. **A Survey of Computational Physics**. 2nd ed. Princeton University Press, New Jersey.

Renaud, J. and F. F. Boctor. 1996. An Improved petal heuristic for the vehicle routing problem. **Journal of the Operational Research** 47: 329-336.

_____, _____ and G. Laporte. 1995. A Tabu Search Heuristic for the Multi-Depot Vehicle Routing Problem. **Computers Ops. Res.** 23: 229-235.

Tan, KC., L.H. Lee and K. Ou. 2001. Artificial intelligence heuristics in solving vehicle routing problems with time window constraints. **Engineering Applications of Artificial Intelligence** 14(6): 825–837.

Zhu Qili, K. 1999. **Heuristic Methods For Vehicle Routing Problem With Time Windows.** B.S. Thesis, National University of Singapore.

ภาคผนวก

ภาคผนวก ก

รายละเอียดโปรแกรม Mathematica

1. หัวข้อในโปรแกรมถูกจัดเรียงลำดับดังนี้

1. กำหนดข้อมูลเริ่มต้นและตัวแปรต่างๆ ของระบบ (Initialize the dataset and parameters)
2. รวบรวมโมดูลต่างๆ ที่ต้องใช้ภายในตัวโปรแกรม (A collection of modules) แสดงในตารางผนวกที่ ก

ตารางผนวกที่ ก1 โมดูลที่ใช้ในโปรแกรม Mathematica

โมดูล	หน้าที่
keepinform	เก็บข้อมูลจำนวนคลังสินค้า, จำนวนเส้นทางในแต่ละคลังสินค้าและจำนวนลูกค้าในแต่ละเส้นทางของแต่ละระบบ
numOrders	คำนวณจำนวนความต้องการทั้งหมดในแต่ละเส้นทาง
distRoute	คำนวณระยะทางในแต่ละเส้นทาง
distTotal	คำนวณระยะทางทั้งหมดในระบบ
seqcust	จัดเรียงลำดับลูกค้าในแต่ละเส้นทางโดยใช้ Nearest insert algorithm
genchromosome	สร้างโครโมโซมโดยจัดเรียงลูกค้าจากคนที่หนึ่งจนถึงคนสุดท้าย และสร้างเซตของ{iDepot, iRoute, iPos}
genrouteAllDepot	แปลงโครโมโซมเป็นเซตที่มีลักษณะเดียวกับทุกเส้นทางในแต่ละคลังสินค้า
checkOrderQ	ใช้ตรวจสอบจำนวนสินค้าในแต่ละเส้นทางว่าเกินความจุของยานพาหนะ คำตอบจะอยู่ในรูปแบบของบูลีน (Boolean) เท่านั้น
lstfitness	คำนวณค่าความเหมาะสมของทุกๆ โครโมโซม
roulettewheel	คัดเลือกโครโมโซมพ่อและแม่โดยใช้วิธีวงล้อสัดส่วนค่าความเหมาะสม
crossover	ทำการข้ามสายพันธุ์ของโครโมโซมพ่อและแม่แต่ละคู่
mutation	ทำการผ่าเหล่าโดยวิธี 2-Swap ในการสลับค่ายีนส์ในโครโมโซม

3. สร้างเมตริกซ์ระยะทาง (Build distance matrix)

4. การจัดลูกค้าให้แต่ละคลังสินค้าที่ใกล้ที่สุด (The nearest neighborhood method)

5. สร้างเมตริกซ์แบบประหยัดของแต่ละคลังสินค้า (Initialize the savings matrix of each depot)

6. แจกจ่ายลูกค้าไปแต่ละเส้นทางของคลังสินค้า (Savings matrix method)

7. ประยุกต์ใช้ nearest insert algorithm กับทุกๆเส้นทางในระบบ เพื่อที่จะหาการจัดเรียงที่ได้ ระยะทางน้อยที่สุดของแต่ละเส้นทาง (Nearest insert method)

8. ประยุกต์ใช้อัลกอริทึมเชิงพันธุกรรมกับแต่ละระบบ (Genetic algorithm module)

9. สร้างโครโมโซมต้นแบบของโครโมโซมพ่อและแม่ (Generate ancestors and apply genetic algorithm) ผลที่ได้คือ ค่าที่ดีที่สุดของทุกเส้นทางในแต่ละคลังสินค้าหลังจากผ่านอัลกอริทึมเชิงพันธุกรรม

- สร้างบรรพบุรุษจากทุกๆเส้นทางของแต่ละคลังสินค้า (generate ancestors from routeAllDepot) เพื่อเพิ่มความหลากหลายก่อนผ่านอัลกอริทึมเชิงพันธุกรรม

- ใช้อัลกอรึทึมเชิงพันธุกรรมกับโครโมโซมบรรพบุรุษทุกตัว (apply genetic algorithm to each ancestor)

10. สรุปการจัดเส้นทางก่อนผ่าน Nearest insert method, อัลกอริทึมเชิงพันธุกรรม และหลังผ่านอัลกอริทึมเชิงพันธุกรรม พร้อมทั้งคำนวณระยะทางทั้งหมดของแต่ละระบบ

2. รหัสคำสั่งที่ใช้ในโปรแกรม Mathemetica สำหรับสร้างแบบจำลอง

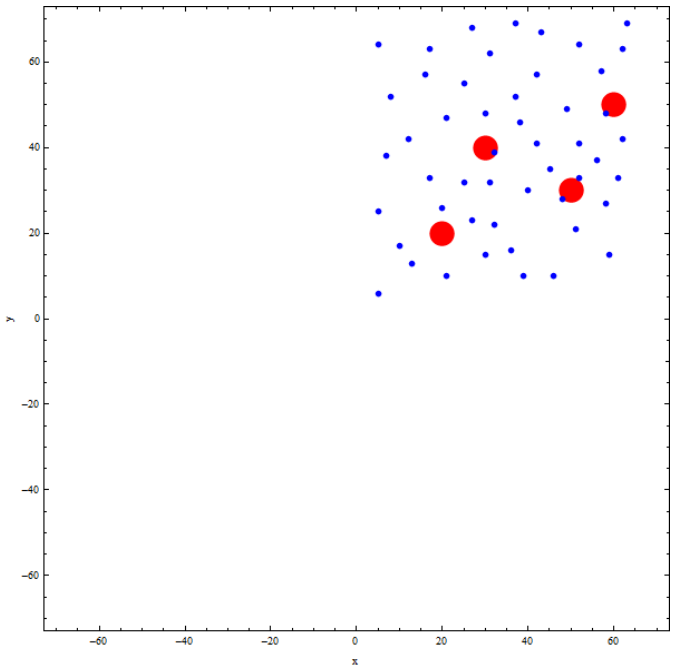
```
starttime=AbsoluteTime[DateString[]];
```

(1) Initialize the dataset and parameters

```
(*------------------------------------------------------------------------*)

(* list of coordinates of depots: lstDepot *)
lstDepot={{20,20},{30,40}, {50, 30},{60,50}};

(* list of coordinates of customers: lstCust *)
lstCust={{37,52},{49,49},{52,64},{20,26},{40,30},{21,47},{17,63},{31,62},{52
,33},{51,21},{42,41},{31,32},{5,25},{12,42},{36,16},{52,41},{27,23},{17,33},
{13,13},{57,58},{62,42},{42,57},{16,57},{8,52},{7,38},{27,68},{30,48},{43,67
},{58,48},{58,27},{37,69},{38,46},{46,10},{61,33},{62,63},{63,69},{32,22},
```

```
{445,35},{59,15},{5,6},{10,17},{21,10},{5,64},{30,15},{39,10},{32,39},{25,32
},{25,55},{48,28},{56,37}};

(* list of orders of customers: lstOrder *)

(* number of values in the list MUST be equal to number of customers *)

(* values are only POSITIVE INTERGER number *)
lstOrder={7,30,16,9,21,15,19,23,11,5,19,29,23,21,10,15,3,41,9,28,8,8,16,10,2
8,7,15,14,6,19,11,12,23,26,17,6,9,15,14,7,27,13,11,16,10,5,25,17,18,10};

(* capacity of a truck *)
capacity=80;

(* number of ancestors before genetic algorithm *)
numancestor=10;

(* number of changes of each ancestor from initial ancestor *)
numchange=3;

(* GENETIC ALGORITHM PARAMETERS *)

(* number of generation - should bigger than 1000 *)
numgen=150;

(* initial population size of each generation *)
popsize=200;

(* number of best chromosomes to be kept *)
elitism=2;

(* number of offsprings in each generation *)
numchilds=100;

(* number of mutation *)
nummutate=80;

(* crossover probability *)
Pc=0.8;

(* mutation probability *)
Pm=0.01;
(*--------------------------------------------------------------------------*)
```

```
SeedRandom[];

(* find number of depots: nDepot *)
nDepot=Length[lstDepot];
Print["number of depots = ",nDepot]

(* find number of customers: nCust *)
nCust=Length[lstCust];
Print["number of customers = ",nCust]

(* find number of customers ordering *)
Print["number of customers ordering = ", Length[lstOrder]]
MatrixForm[lstOrder];

(* find the maximum absolute value in lstDepot and lstCust *)
len=Max[Join[Abs[lstDepot],Abs[lstCust]]];

(* plot positions of depots *)
plotDepot=ListPlot[lstDepot,PlotStyle→{RGBColor[1,0,0],PointSize[0.04`]},Dis
playFunction→Identity];

(* plot positions of customers *)
plotCust=ListPlot[lstCust,PlotStyle→{RGBColor[0,0,1],PointSize[0.01`]},Displ
ayFunction→Identity];

(* plot positions of all *)
Show[plotDepot,plotCust,PlotRange→{{-len-1,len+1},{-len-
1,len+1}},Axes→False,Frame→True,FrameLabel→{"x","y"},ImageSize→600,AspectRat
io→Automatic,DisplayFunction→$DisplayFunction]


number of depots =  4

number of customers =  50

number of customers ordering =  50
```



(2) A collection of modules

```
(* collect all modules used in this code *)
```

Keep information of a number of routes and customers of each route of each depot

```
(* keep information of a number of routes and customers of each route of
each depot: keepinform *)

(* routeAll = list collecting all routes of all depots, such as
routeAllDepot and routeAllDepotNearest *)

keepinform[routeAll_]:=Module[{inform,informin,numroutes,numcusts,k,i},
  inform={};
  For[k=1,k<=nDepot,k++,
   informin={};
   numroutes=Length[routeAll[[k]]];
   For[i=1,i<=numroutes,i++,
    numcusts=Length[routeAll[[k]][[i]]];
    informin=Join[informin,{numcusts}];
    ];
   inform=Join[inform,{informin}];
   ];
  inform
  ]
```

Compute a number of orders in a route

```
(* numOrders is used to compute the number of orders in a route:numOrders *)

numOrders[route_,lstOfOrder_]:=Module[{numroute,orderroute,sumorder},
  numroute=Length[route];
  orderroute=0;

  (* accumulate orders of each customer in the route *)
  For[sumorder=1,sumorder□numroute,sumorder++,
   orderroute+=lstOfOrder[[route[[sumorder]]]];
   ];

  orderroute
  ]
```

Compute distance of a route (DC - Custs - DC)

```
(* distRoute is used to compute total distance of a route, starting from
depot to customers to depot: distRoute *)
(* kDepot = depot # *)
(* route  = route used to compute the distance *)

distRoute[kDepot_,route_]:=Module[{oneroute,nsum,sumdst,i,ipos,fpos},

(* join depot and customers in one route *)
oneroute=Join[{kDepot},route+nDepot,{kDepot}];

  (* the number of connections in oneroute *)
  nsum=Length[oneroute]-1;

  (* start summing distance *)
  sumdst=0;
  For[i=1,i≤nsum,i++,
   ipos=oneroute[[i]];
   fpos=oneroute[[i+1]];
   sumdst+=dstMat[[ipos]][[fpos]];
   ];

  sumdst
  ]
```

Compute total distance of the system (all routes)

```
(* distTotal is used to compute total distance in the system. it accumulates
all distance of every route: distTotal *)
(* routeAll = list collecting all routes of all depots, such as
routeAllDepot and routeAllDepotNearest *)

distTotal[routeAll_]:=Module[{k,thisdepotroute,accudist,numroutes,i},
  accudist=0;

  (* accumulate distance of all routes in all depots *)
  For[k=1,k≤nDepot,k++,
   thisdepotroute=routeAll[[k]];
   numroutes=Length[thisdepotroute];
   For[i=1,i≤numroutes,i++,
    accudist+=distRoute[k,thisdepotroute[[i]]];
    ];
   ];

  accudist
  ]
```

Sequence customers in a route by using nearest insert algorithm

```
(* using nearestinsert algorithm to sequence customers in a route:seqcust *)
(* kDepot = depot # *)
(* route  = route used to sequence customers *)

seqcust[kDepot_,route_]:=Module[{seqroute,seqroutetemp1,seqroutetemp2,i,j,th
iscust,gapseqroute,keepmindist,caldist},

  seqroute={};

  For[i=1,i≤Length[route],i++,

   thiscust=route[[i]];
   gapseqroute=Length[seqroute]+1;
   keepmindist=distRoute[kDepot,Insert[seqroute,thiscust,1]];

   (* try inserting a customer into each gap and compute distance *)
   For[j=1,j≤gapseqroute,j++,
    seqroutetemp1=Insert[seqroute,thiscust,j];
```

```
    caldist=distRoute[kDepot,seqroutetemp1];

    (* accept a route of minimum distance *)
    If[caldist≤keepmindist,
     keepmindist=caldist;
     seqroutetemp2=seqroutetemp1;
     ,
     (* EXIT IF *)
     ];
    ];

   seqroute=seqroutetemp2;
   ];

  seqroute
  ]
```

Generate chromosome (a list of position in depot and route of each customer) from routeAllDepot*

```
(* sequence position of customers from the first to the last customer:
genchromosome *)
(* lstrouteAllDepot = a list, structured as same as routeAllDepot *)

genchromosome[lstrouteAllDepot_]:=Module[{chromo,i},
  chromo={};
  Table[chromo=Join[chromo,Position[lstrouteAllDepot,i]],{i,1,nCust}];
  chromo
  ]
```

Generate routeAllDepot* from a chromosome

```
(* generate a list,such as routeAllDepot,from chromosome:genrouteAllDepot *)
(* lstchromo = a list structured as same as genchromosome *)
(* inform = information of system, calculated from MODULE keepinform *)

genrouteAllDepot[lstchromo_,inform_]:=Module[{lstrouteAllDepot,k,i,j,lstrout
eindepot,lstcustinroute,pos},
  lstrouteAllDepot={};

  (* scan depot *)
  For[k=1,k≤nDepot,k++,
```

```
   lstrouteindepot={};

   (* scan route in the depot *)
   For[i=1,i≤Length[inform[[k]]],i++,
    lstcustinroute={};

    (* scan customer in the route *)
    For[j=1,j≤inform[[k]][[i]],j++,
     pos=Position[lstchromo,{k,i,j}];
     lstcustinroute=Join[lstcustinroute,pos];
     ];

    lstrouteindepot=Join[lstrouteindepot,{Flatten[lstcustinroute]}];
    ];

   lstrouteAllDepot=Join[lstrouteAllDepot,{lstrouteindepot}];
   ];

  lstrouteAllDepot
  ]
```

Check whether the number of orders in each route exceed the capacity or not

```
(* checkOrderQ is to check if the number of orders in each route exceed the
capacity or not: checkOrderQ *)
(* this module return TRUE if not exceed, but FALSE if exceed *)
(* lstchromo = a list structured as same as genchromosome *)
(* inform = information of system, calculated from MODULE keepinform *)

checkOrderQ[lstchromo_,inform_]:=Module[{lstRouteAllDepot,lstRouteAllDepotTe
mp,lstQ},
  lstRouteAllDepot=genrouteAllDepot[lstchromo,inform];
  lstRouteAllDepotTemp=Flatten[lstRouteAllDepot,1];
  lstQ=Map[(TrueQ[numOrders[#,lstOrder]≤capacity])&,lstRouteAllDepotTemp];
  If[MemberQ[lstQ,False],False,True]
  ]
```

Compute a list of fitnesses of all chromosomes

```
(* calcualte fitness values of all chromosome: lstfitness *)
(* in this study, a fitness value is total distance of a system *)
(* lstchromoall = a list of all chromosomes, as same as lstchromosome *)
```

```
lstfitness[lstchromoall_,inform_]:=Module[
lstchromoall];
  dist=Map[(distTotal[#])&,routeAll]
  ]
```

Roulette wheel module

```
(* roulette wheel method is used to select good chromosomes: roulettewheel*)
(* this module returns a list of all good chromosomes (parents) *)
(* lstoldchromoall = a list of all chromosomes, as same as lstchromosome *)

roulettewheel[lstoldchromoall_,inform_]:=Module[{fitness,totalfitness,ratiof
itness,totalratiofitness,probselect,lst,lstelitism,
   intervalprob,lstrnd,lstnamechromo,selected,i,j,lstparents},

  (* build a list of fitnesses of all chromosomes *)
  fitness=lstfitness[lstoldchromoall,inform];

  (* sum all fitnesses *)
  totalfitness=Total[fitness];

  (* calculate totalfitness/fitness of each chromosome *)
  ratiofitness=totalfitness/fitness;

  (* sum all ratiofitnesses *)
  totalratiofitness=Total[ratiofitness];

  (* calculate probability of selection of each chromosome *)
  probselect=ratiofitness/totalratiofitness;

  (* keep two chromosomes of two most probselect *)
  lst=Sort[probselect,Greater];

lstelitism=Join[Flatten[Position[probselect,lst[[1]]]],Flatten[Position[prob
select,lst[[2]]]]];

  (* build a list of interval of probability *)
  intervalprob=Accumulate[probselect];

  (* buil a list of random numbers *)
  lstrnd=RandomReal[1,Length[intervalprob]];

  (* selection *)
```

```
  selected=Select[lstrnd,#≤intervalprob[[1]]&];
  lstnamechromo=Table[1,{j,1,Length[selected]}];
  For[i=2,i≤Length[intervalprob],i++,
   selected=Select[lstrnd,intervalprob[[i-1]]<#≤ intervalprob[[i]]&];
   lstnamechromo=Join[lstnamechromo,Table[i,{j,1,Length[selected]}]];
   ];

  lstnamechromo=Join[lstelitism,lstnamechromo];
  lstnamechromo=RandomSample[lstnamechromo];
  lstparents=Map[(lstoldchromoall[[#]])&,lstnamechromo]
  ]
```

Crossover

```
(* crossover a pair of parents. the output is two childs: crossover *)
(* father & mother - each is chromosome *)
(* inform = information of system, calculated from MODULE keepinform *)
(* method: start with father and mother - {
   OverscriptBox["A", "_"] ,
   OverscriptBox["B", "_"] ,C,
   OverscriptBox["D", "_"] ,E} && {C,
   OverscriptBox["B", "_"] ,
   OverscriptBox["A", "_"] ,
   OverscriptBox["E", "_"] ,D} - ones with bar are chosen *)
(* A,B,C,D,and E are positions of customers in a system *)
(* so child1 - {
   OverscriptBox["A", "_"] ,
   OverscriptBox["B", "_"] ,C,D,
   OverscriptBox["E", "_"] } and child2 - {C,
   OverscriptBox["A", "_"] ,
   OverscriptBox["B", "_"] ,E,
   OverscriptBox["D", "_"] } *)
(* child1 = crossover[father,mother][[1]] and child2 =
crossover[father,mother][[2]] *)

crossover[father_,mother_,inform_]:=Module[{numrand,checkorder,switchedmothe
r,switchedfather,unswitchedmother,unswitchedfather,lst,lstposswitchedmother,
lstposswitchedfather,lstposunswitchedmother,lstposunswitchedfather,unchanged
,changed,trans,child1,child2,lstchilds,nloop},

  (* a number of randomization *)
  numrand=Floor[nCust/4];
```

```
nloop=0;
checkorder=True;

(* return child1 if accepting constraint *)
While[checkorder,

 (* give labels of positions of customers in mother chromosome, as will be
 switched *)
 switchedmother={};
 While[Length[switchedmother]<numrand,
  switchedmother=Union[switchedmother,{RandomInteger[{1,nCust}]}];
  ];

 (* give labels of positions of customers in mother chromosome, as will
 not be switched *)
 lst=Range[nCust];
 unswitchedmother=Delete[lst,Partition[switchedmother,1]];

 (* lists of positions of switched and unswitched genes of mother *)
 lstposswitchedmother=Map[(mother[[#]])&,switchedmother];
 lstposunswitchedmother=Map[(mother[[#]])&,unswitchedmother];

 (* give labels of positions of customers in father chromosome,
 uncrossovered and crossovered by mother *)
 unchanged=Flatten[Map[(Position[father,#])&,lstposunswitchedmother]];
 changed=Delete[lst,Partition[unchanged,1]];

 (* generate child1 *)
 trans=MapThread[(#1→#2)&,{changed,lstposswitchedmother}];
 child1=ReplacePart[father,trans];

 (* check if order in each route is less than the capacity ornot. if NO,
 do all over again *)
 If[checkOrderQ[child1,inform],checkorder=False,checkorder=True];

 (* break while loop, prevent infinite loop *)
 nloop++;
 If[nloop>100,
  child1=father;
  Break[];
  ,
  ];
 ];
```

```
nloop=0;
checkorder=True;

(* return child2 if accepting constraint *)
While[checkorder,

 (* give labels of positions of customers in father chromosome, as will be
 switched *)
 switchedfather={};
 While[Length[switchedfather]<numrand,
  switchedfather=Union[switchedfather,{RandomInteger[{1,nCust}]}];
  ];

 (* give labels of positions of customers in father chromosome, as will
 not be switched *)
 lst=Range[nCust];
 unswitchedfather=Delete[lst,Partition[switchedfather,1]];

 (* lists of positions of switched and unswitched genes of father *)
 lstposswitchedfather=Map[(father[[#]])&,switchedfather];
 lstposunswitchedfather=Map[(father[[#]])&,unswitchedfather];

 (* give labels of positions of customers in mother chromosome,
 uncrossovered and crossovered by father *)
 unchanged=Flatten[Map[(Position[mother,#])&,lstposunswitchedfather]];
 changed=Delete[lst,Partition[unchanged,1]];

 (* generate child2 *)
 trans=MapThread[(#1→#2)&,{changed,lstposswitchedfather}];
 child2=ReplacePart[mother,trans];

 (* check if order in each route is less than the capacity or not. if NO,
 do all over again *)
 If[checkOrderQ[child2,inform],checkorder=False,checkorder=True];

 (* break while loop, prevent infinite loop *)
 nloop++;
 If[nloop>100,
  child2=mother;
  Break[];
  ,
  ];
 ];
```

```
  lstchilds={child1,child2}
  ]
```

Mutation

```
(* use 2-swap method to mutate genes in a child, output is a new chromosome:
mutation *)
(* lstchromo - a chromosome *)
(* inform = information of system, calculated from MODULE keepinform *)

mutation[lstchromo_,inform_]:=Module[{len,checkorder,int1,int2,lstfront,lstc
enter,lstback,trialchromo},
  len=Length[lstchromo];

  (* return a new chromosome if accepting constrint *)
  checkorder=True;
  While[checkorder,

   (* random two integer numbers *)
   int1=RandomInteger[{1,len}];
   int2=RandomInteger[{1,len}];
   While[int1==int2,int2=RandomInteger[{1,len}];];

   (* build a new chromosome *)
   If[int1<int2,
    lstfront=Take[lstchromo,int1-1];
    lstcenter=Reverse[Take[lstchromo,{int1,int2}]];
    lstback=Take[lstchromo,-(len-int2)];
    ,
    lstfront=Take[lstchromo,int2-1];
    lstcenter=Reverse[Take[lstchromo,{int2,int1}]];
    lstback=Take[lstchromo,-(len-int1)];
    ];

   trialchromo=Join[lstfront,lstcenter,lstback];

   (* check if order in each route is less than the capacity or not. if NO,
   do all over again *)
   If[checkOrderQ[trialchromo,inform],checkorder=False,checkorder=True];
   ];

  trialchromo
  ]
```

(3) Build distance matrix

```
(* build distance matrix: dstMat *)
U=Join[lstDepot,lstCust];
dstMat=Table[Table[

    (* calculate distance between two vertices *)
    dx=U[[i]][[1]]-U[[j]][[1]];
    dy=U[[i]][[2]]-U[[j]][[2]];
    dst=1. Sqrt[dx*dx+dy*dy]
    , {j, 1, nDepot+nCust}], {i, 1, nDepot+nCust}];
MatrixForm[dstMat];
```

(4) The nearest neighbourhood method

```
(* distance between all customers and each depot *)
distcusttodepot=Table[dstMat[[j]][[i+nDepot]],{i,1,nCust},{j,1,nDepot}];

(* minimum distance of customers to a depot *)
mindistcusttodepot=Table[Min[distcusttodepot[[i]]],{i,1,nCust}];

(* iDepot of above *)
nearDepot=Table[

   (* the nearest depot of each customer *)
   numdepot=Position[distcusttodepot[[i]],mindistcusttodepot[[i]]];

   (* sometimes the distance from customer to many depots is equal, so
   choose the first depot *)
   First[Flatten[numdepot]]

   ,{i,1,nCust}];

(* assign each customer to the nearest depot *)
(* lstCustinDepot keeps lists of customers in each depot *)
lstCustinDepot=Table[Flatten[Position[nearDepot,j]],{j,1,nDepot}];

(*Print["list of customers in each depot = ",MatrixForm[lstCustinDepot]]*)
```

(5) Initialize the savings matrix of each depot

```
(* inisavmat keeps the initial savings matrix of each depot: inisavmat *)
inisavmat={};
For[k=1,k□nDepot,k++,

 (* number of customers in each depot *)
 numCustinDepot=Length[lstCustinDepot[[k]]];

 (* temporary initial savings matrix of each depot *)
 tmpinisavmat=Table[

   (* at the same vertices, the elements in the savings matrices are all
   zero *)
   (* otherwise, s(x,y) = dist(dc,x) + dist(dc,y) - dist(x,y) *)
   If[i□j,

    eleTable=0,
    eleFirst=lstCustinDepot[[k]][[i]]+nDepot;
    eleLast=lstCustinDepot[[k]][[j]]+nDepot;
    eleTable=dstMat[[k]][[eleFirst]]+dstMat[[k]][[eleLast]]-
dstMat[[eleFirst]][[eleLast]]]
   ,{i,1,numCustinDepot},{j,1,numCustinDepot}
   ];

 (* join the temporary savings matrix with inisavmat *)
 inisavmat=Join[inisavmat,{tmpinisavmat}];
 ]

(* print the initial savings matrices *)
(*
For[k=1,k□nDepot,k++,
Print["Depot# = ", k, "    the initial savings matrix = ",
MatrixForm[inisavmat[[k]]]]
]
*)
```

(6) Savings matrix method

```
(* routeAllDepot keeps every route of all depot: routeAllDepot *)
routeAllDepot={};
For[k=1,k≤nDepot,k++,

  (* number of customers in each depot: numCustinkDepot *)
  numCustinkDepot=Length[inisavmat[[k]]];

  (* list of customers in this depot: lstCustinkDepot *)
  lstCustinkDepot=lstCustinDepot[[k]];

  (* arrange elements in the savings matrix of each depot from max to min:
  arrele *)

arrele=Flatten[Table[inisavmat[[k]][[i]][[j]],{i,2,numCustinkDepot},{j,1,i-
1}]];
  arrele=Reverse[Sort[arrele]];

  (* number of elements in arrele: numarrele *)
  numarrele=Length[arrele];

  (* create list of orders of each depot: lstOrderkDepot *)
  lstOrderkDepot={};
  For[i=1,i≤numCustinkDepot,i++,

   (* ith-customer *)
   ithcust=lstCustinkDepot[[i]];
   lstOrderkDepot=Join[lstOrderkDepot,{lstOrder[[ithcust]]}];
   ];

  (* keep all routes in this depot: routekDepot *)
  routekDepot={};
  For[l=1,l≤numarrele,l++,

   (* choose each element from arrele *)
   chosenarrele=arrele[[l]];

   (* a pair of iTemp and jTemp from inisavmat with an element chosen by
   chosenarrele *)
   iTempjTemp=First[Position[inisavmat[[k]],chosenarrele]];
   iTemp=iTempjTemp[[1]];
   jTemp=iTempjTemp[[2]];
```

```
(* NOTE - saving matrix algorithm!!!
   check whether iTemp is in routekDepot
   YES (1) - check whether jTemp is in routekDepot
        YES (1.1) - check whether iTemp and jTemp are in the same route or
                    not
             YES (1.1a)  - exit IF
             NO  (1.1b)  - try combining both routes and check whether #
                           of orders exceed the limit
                  YES (1.1.1) - exit IF
                  NO  (1.1.2) - delete those two routes and join the
                                combined route
        NO  (1.2) - try attaching jTemp to the list of iTemp and check
                    whether # of orders exceed the limit
             YES (1.2.1) - build another new list for jTemp and exit IF
             NO  (1.2.2) - exit IF
   NO  (2) - check whether iTemp is in routekDepot
        YES (2.1) - try attaching iTemp to the list of jTemp and check
                    whether # of orders exceed the limit
             YES (2.1.1) - build another new list for iTemp and exit IF
             NO  (2.1.2) - exit IF
        NO  (2.2) - build new list for iTemp, try attaching jTemp to the
                    list of iTemp, and check whether # of orders exceed
                    the limit
             YES (2.2.1) - build another new list for jTemp and exit IF
             NO  (2.2.2) - exit IF
 *)

(* prepare MODULE often used in saving matrix method *)
(* 0- nthrow - used to find nth-row of any iTemp or jTemp in routekDepot
 *)
(* ijTemp = iTemp or jTemp *)
nthrow[ijTemp_,routeofkDepot_]:=Module[{posnthrow},
  posnthrow=First[First[Position[routeofkDepot,ijTemp]]]
  ];

(* 1- numorder - used to compute # of orders of each route *)
(* route = a list of route *)

numorder[route_,lstofOrderkDepot_]:=Module[{numroute,orderroute,sumorder},
  numroute=Length[route];
  orderroute=0;
  For[sumorder=1,sumorder≤numroute,sumorder++,
   orderroute+=lstofOrderkDepot[[route[[sumorder]]]];
```

```
   ];
  orderroute
  ];

(* check whether iTemp is in routekDepot *)
If[MemberQ[Flatten[routekDepot],iTemp],

 (* YES - 1 *)
 (* check whether jTemp is in routekDepot *)
 If[MemberQ[Flatten[routekDepot],jTemp],

   (* YES - 1.1 *)
   (* try combining both routes *)
   rowofiTemp=nthrow[iTemp,routekDepot];
   rowofjTemp=nthrow[jTemp,routekDepot];

   (* check whether iTemp and jTemp are in the same route or not *)
   If[rowofiTemp==rowofjTemp,
    (* YES - 1.1a *)
    (* EXIT *)
    ,
    (* NO - 1.1b *)
    routeofiTemp=routekDepot[[rowofiTemp]];
    routeofjTemp=routekDepot[[rowofjTemp]];
    routeofiTempjTemp=Join[routeofiTemp,routeofjTemp];

    (* check whether # of orders exceed the limit *)
    If[numorder[routeofiTempjTemp,lstOrderkDepot]>capacity,
     (* YES - 1.1.1 *)
     (* EXIT *)
     ,
     (* NO - 1.1.2 *)
     (* delete routes of iTemp and jTemp and then join the combined route
     in routekDepot *)
     routekDepot=Delete[routekDepot,rowofiTemp];
     rowofjTemp=nthrow[jTemp,routekDepot];
     routekDepot=Delete[routekDepot,rowofjTemp];
     routekDepot=Join[routekDepot,{routeofiTempjTemp}];
     ];
    ];
   ,
   (* NO - 1.2 *)
   (* try attaching jTemp to the list of iTemp *)
```

```
  rowofiTemp=nthrow[iTemp,routekDepot];
  routeofiTemp=routekDepot[[rowofiTemp]];
  routeofiTemp=Join[routeofiTemp,{jTemp}];

  (* check whether # of orders exceed the limit *)
  If[numorder[routeofiTemp,lstOrderkDepot]>capacity,

   (* YES - 1.2.1 *)
   routekDepot=Join[routekDepot,{{jTemp}}];
   ,
   (* NO - 1.2.2 *)
   routekDepot=Delete[routekDepot,rowofiTemp];
   routekDepot=Join[routekDepot,{routeofiTemp}];
   ];
  ];
,
(* NO - 2 *)
(* check whether jTemp is in routekDepot *)
If[MemberQ[Flatten[routekDepot],jTemp],

  (* YES  - 2.1 *)
  (* try attaching iTemp to the list of jTemp *)
  rowofjTemp=nthrow[jTemp,routekDepot];
  routeofjTemp=routekDepot[[rowofjTemp]];
  routeofjTemp=Join[routeofjTemp,{iTemp}];

  (* check whether # of orders exceed the limit *)
  If[numorder[routeofjTemp,lstOrderkDepot]>capacity,

   (* YES - 2.1.1 *)
   routekDepot=Join[routekDepot,{{iTemp}}];
   ,
   (* NO - 2.1.2 *)
   routekDepot=Delete[routekDepot,rowofjTemp];
   routekDepot=Join[routekDepot,{routeofjTemp}];
   ];
  ,
  (* NO - 2.2 *)
  (* build new list for iTemp and try attaching jTemp to the list of
  iTemp *)
  routeofiTemp={iTemp,jTemp};

  (* check whether # of orders exceed the limit *)
```

```
      If[numorder[routeofiTemp,lstOrderkDepot]>capacity,

       (* YES - 2.2.1 *)
       routekDepot=Join[routekDepot,{{iTemp}},{{jTemp}}];
       ,
       (* NO - 2.2.2 *)
       routekDepot=Join[routekDepot,{{iTemp,jTemp}}];
       ];
      ];
    ];
   ]; (* reach i = numCustinkDepot *)

  (* change labels of customers, used in FOR LOOP k, to their real labels *)
  lenroutekDepot=Length[routekDepot];
  routekDepotlbl={};
  For[i=1,i≤lenroutekDepot,i++,
   subroute=Map[(lstCustinkDepot[[#]])&,routekDepot[[i]]];
   routekDepotlbl=Join[routekDepotlbl,{subroute}];
   ];

  (* routeAllDepot keeps all tracks in the same list *)
  routeAllDepot=Join[routeAllDepot,{routekDepotlbl}];

  ];(* reach k = nDepot *)
```

(7) Nearest insert method

```
(* apply nearest insert algorithm (MODULE seqcust) with routeAllDepot and
obtain: routeAllDepotNearest *)
routeAllDepotNearest={};
For[k=1,k≤nDepot,k++,
 subroute=Map[(seqcust[k,#])&,routeAllDepot[[k]]];
 routeAllDepotNearest=Join[routeAllDepotNearest,{subroute}];
 ]
```

(8) Genetic algorithm module

```
(* apply MODULE geneticalgorithm with each list of routeAllDepot:
geneticalgorithm *)
(* lstrouteAllDepot = a list, structured as same as routeAllDepot *)
```

```
geneticalgorithm[lstrouteAllDepot_]:=Module[{inform,iniChromosome,lstChromos
ome,numchromo,rndcust1,rndcust2,posrndcust1,posrndcust2,trialChromosome,lstN
ewChromosome,childs,nloop,numlstNewChromo,rndnum,intrnd1,intrnd2,father,moth
er,twochilds,newoffspring,del,lstfit,lstfitsort,elimilstfit,poselimilstfit,p
osfinal,finalchromo,routeAllDepotFinal,n},

  (* generate a set of chromosomes before passing them into the kernel:
  lstChromosome *)

  inform=keepinform[lstrouteAllDepot];
  iniChromosome=genchromosome[lstrouteAllDepot];
  lstChromosome={iniChromosome};

  numchromo=Length[lstChromosome];
  While[numchromo<popsize,

   (* random a chromosome from lstChromosome *)
   iniChromosome=lstChromosome[[RandomInteger[{1,numchromo}]]];

   (* random two different customers and switch positions of two customers*)
   rndcust1=RandomInteger[{1,nCust}];
   rndcust2=RandomInteger[{1,nCust}];
   posrndcust1=iniChromosome[[rndcust1]];
   posrndcust2=iniChromosome[[rndcust2]];
   While[(rndcust1==rndcust2) ||(posrndcust1==posrndcust2),
    rndcust1=RandomInteger[{1,nCust}];
    rndcust2=RandomInteger[{1,nCust}];
    posrndcust1=iniChromosome[[rndcust1]];
    posrndcust2=iniChromosome[[rndcust2]];
    ];

trialChromosome=ReplacePart[iniChromosome,{rndcust1→posrndcust2,rndcust2→pos
rndcust1}];

   (* accept trialChromosome if (1) no trialChromosome in lstChromosome and
   (2) # order of each route is less than the capacity *)

If[((!MemberQ[lstChromosome,trialChromosome])&&checkOrderQ[trialChromosome,i
nform]),
    lstChromosome=Join[lstChromosome,{trialChromosome}];
    ,
    ];
```

```
 numchromo=Length[lstChromosome];
 ];

(* this is the main kernel of genetic algorithm            *)
(* - selection by roulette wheel                           *)
(* - random two parents with a random number accepting Pc  *)
(* - crossover                                             *)
(* - random an offspring with a random number accepting Pm *)
(* - mutate the offspring                                  *)
(* - combine sets of parents and offsprings                *)
(* - eliminate chromosomes with high fitness               *)

Print["    number of generation = ", numgen];
For[n=1,n≤numgen,n++,
 Print["    nth-generation = ", n,"/",numgen];

 Print["                          ...selection..."];

 (* selection by roulette wheel *)
 lstNewChromosome=roulettewheel[lstChromosome,inform];

 Print["                          ...crossover..."];

 (* crossover until # of childs = numchild *)
 childs={};
 nloop=0;
 While[Length[childs]<numchilds,

  (* random two different parents with a random number accepting Pc *)
  numlstNewChromo=Length[lstNewChromosome];
  rndnum=1;
  intrnd1=1;
  intrnd2=1;
  father=lstNewChromosome[[intrnd1]];
  mother=lstNewChromosome[[intrnd2]];
  While[(rndnum>Pc || intrnd1==intrnd2||father==mother),
   intrnd1=RandomInteger[{1,numlstNewChromo}];
   intrnd2=RandomInteger[{1,numlstNewChromo}];
   father=lstNewChromosome[[intrnd1]];
   mother=lstNewChromosome[[intrnd2]];
   rndnum=RandomReal[];
   ];
```

```
 twochilds=crossover[father,mother,inform];
 childs=Union[childs,twochilds];

 (* break while loop, prevent infinite loop *)
 nloop++;
 If[nloop≥100,
  Break[];
  ,
  ];
 ];

Print["                              ...mutation...."];

(* let nummutate-offsprings have mutation *)
For[i=1,i≤nummutate,i++,

 (* random an offspring with a random number accepting Pm *)
 rndnum=1;
 While[rndnum>Pm,
  intrnd1=RandomInteger[{1,Length[childs]}];
  rndnum=RandomReal[];
  ];

 (* mutate the offspring *)
 newoffspring=mutation[childs[[intrnd1]],inform];
 If[!MemberQ[childs,newoffspring],
  childs=ReplacePart[childs,intrnd1→newoffspring];
  ,
  ];
 ];

(* combine parents and childs' chromosomes *)
lstChromosome=Union[lstChromosome,childs];

(* eliminate chromosomes with high fitnesses *)
del=Length[lstChromosome]-popsize;
lstfit=lstfitness[lstChromosome,inform];
lstfitsort=Sort[lstfit,Greater];
elimilstfit=Take[lstfitsort,del];
poselimilstfit=Flatten[Map[(Position[lstfit,#])&,elimilstfit],1];
lstChromosome=Delete[lstChromosome,poselimilstfit];
];
```

```
  (* choose a chromosome with the smallest fitness *)
  lstfit=lstfitness[lstChromosome,inform];
  posfinal=Flatten[Position[lstfit,Min[lstfit]]];
  finalchromo=Flatten[lstChromosome[[posfinal]],1];
  routeAllDepotFinal=genrouteAllDepot[finalchromo,inform]
  ]
```

(9) Generate ancestors and apply genetic algorithm

Generate ancestors from routeAllDepot

```
(* generate ancestors from routeAllDepot *)

lstancestor={routeAllDepotNearest};

While[Length[lstancestor]<numancestor,
 ancestor=routeAllDepot;

 countchange=0;
 inform=keepinform[ancestor];
 ancestortmp=ancestor;
 While[countchange<numchange,

  (* random two different depots *)
  rndDepotdel=RandomInteger[{1,Length[inform]}];
  rndDepotin=RandomInteger[{1,Length[inform]}];
  While[rndDepotdel==rndDepotin,
   rndDepotdel=RandomInteger[{1,Length[inform]}];
   rndDepotin=RandomInteger[{1,Length[inform]}];
   ];

  (* random two routes in two depots *)
  rndRoutedel=RandomInteger[{1,Length[inform[[rndDepotdel]]]}];
  rndRoutein=RandomInteger[{1,Length[inform[[rndDepotin]]]}];

  (* random two customers in two routes *)
  rndCustdel=RandomInteger[{1,inform[[rndDepotdel]][[rndRoutedel]]}];
  rndCustin=RandomInteger[{1,inform[[rndDepotin]][[rndRoutein]]+1}];

  tempcust=ancestor[[rndDepotdel]][[rndRoutedel]][[rndCustdel]];
  trialancestor=Insert[ancestor,tempcust,{rndDepotin,rndRoutein,rndCustin}];
  trialancestor=Delete[trialancestor,{rndDepotdel,rndRoutedel,rndCustdel}];
```

```
  trialancestor=DeleteCases[trialancestor,{},Infinity];

  (* accept or decline newancestor, depending on constraint *)
  inform=keepinform[trialancestor];
  If[checkOrderQ[genchromosome[trialancestor],inform],
   countchange++;
   ancestor=trialancestor;
   ancestortmp=trialancestor;
   inform=keepinform[trialancestor];
   ,
   ancestor=ancestortmp;
   inform=keepinform[ancestor];
   ];

  ];

 lstancestor=Union[lstancestor,{ancestor}];
 ]
```

Apply genetic algorithm to each ancestor

```
(* apply genetic algorithm to each ancestor *)
len=Length[lstancestor];
Print["number of ancestors = ", len];
Print[""];

possibleRouteAllDepot={};
For[npos=1,npos≤len,npos++,
 Print["nth-ancestor = ", npos,"/",len];

possibleRouteAllDepot=Union[possibleRouteAllDepot,{geneticalgorithm[lstances
tor[[npos]]]}];
 ]

(* search for the best chromosome (the most minimum routeAllDepot) *)
possiblemindist=Map[(distTotal[#])&,possibleRouteAllDepot];
mindist=Min[possiblemindist];
pos=Flatten[Position[possiblemindist,mindist]];
routeAllDepotFinal=Flatten[possibleRouteAllDepot[[pos]],1];

possiblemindist
```

(10) Show result

```
(* print list of routes of all depots *)

Print["--------solution before passing nearest insert method--------"];
Print[""];
For[k=1,k≤nDepot,k++,
 Print["Depot# = ", k, "    all routes = ", MatrixForm[routeAllDepot[[k]]]];
 ]
Print[""];
Print["Total distance = ", distTotal[routeAllDepot]];
Print[""];
Print[""];

Print["--------solution after passing nearest insert method but before
genetic algorithm--------"];
Print[""];
For[k=1,k≤nDepot,k++,
 Print["Depot# = ", k, "    all routes = ",
MatrixForm[routeAllDepotNearest[[k]]]];
 ]
Print[""];
Print["Total distance = ", distTotal[routeAllDepotNearest]];
Print[""];
Print[""];

Print["--------solution after passing genetic algorithm--------"];
Print[""];
For[k=1,k≤nDepot,k++,
 Print["Depot# = ", k, "    all routes = ",
MatrixForm[routeAllDepotFinal[[k]]]];
 ]
Print[""];
Print["Total distance = ", distTotal[routeAllDepotFinal]];
Print[""];
Print[""];

endtime=AbsoluteTime[DateString[]];

(* Collapsed time used in program *)
Print[" total elapsed time =  ", (endtime-starttime)/60.," minutes"];
```

ภาคผนวก ข

ชุดข้อมูลที่ใช้ในการทำวิจัย

1. พิกัดของคลังสินค้าในแต่ละปัญหาของการทำวิจัย

ตารางผนวกที่ ข1 ชุดข้อมูลพิกัดของคลังสินค้า

ปัญหา	พิกัดของคลังสินค้า
1, 2	(20,20), (30,40), (50,30), (60,50)
3, 10, 13	(40,40), (50,22), (55,55), (25,45), (20,20)
4	(35,20), (35,50)
5	(15,35), (55,35)
6, 8 ,11	(15,20), (50,20), (35,55)
7, 9, 12	(15,35), (55,35), (35,20), (35,50)

2. พิกัดและความต้องการของลูกค้า

ตารางผนวกที่ ข2 ชุดข้อมูลพิกัดและความต้องการของลูกค้าของปัญหาที่มีลูกค้า 50 คน

ลูกค้าคนที่	พิกัด		ความต้องการ	ลูกค้าคนที่	พิกัด		ความต้องการ
	x	y			x	y	
1	37	52	7	12	31	32	29
2	49	49	30	13	5	25	23
3	52	64	16	14	12	42	21
4	20	26	9	15	36	16	10
5	40	30	21	16	52	41	15
6	21	47	15	17	27	23	3
7	17	63	19	18	17	33	41
8	31	62	23	19	13	13	9
9	52	33	11	20	57	58	28
10	51	21	5	21	62	42	8
11	42	41	19	22	42	57	8

ตารางผนวกที่ ข2 (ต่อ)

ลูกค้าคนที่	พิกัด		ความต้องการ	ลูกค้าคนที่	พิกัด		ความต้องการ
	x	y			x	y	
23	16	57	16	37	32	22	9
24	8	52	10	38	45	35	15
25	7	38	28	39	59	15	14
26	27	68	7	40	5	6	7
27	30	48	15	41	10	17	27
28	43	67	14	42	21	10	13
29	58	48	6	43	5	64	11
30	58	27	19	44	30	15	16
31	37	69	11	45	39	10	10
32	38	46	12	46	32	39	5
33	46	10	23	47	25	32	25
34	61	33	26	48	25	55	17
35	62	63	17	49	48	28	18
36	63	69	6	50	56	37	10

ตารางผนวกที่ ข3 ชุดข้อมูลพิกัดและความต้องการของลูกค้าของปัญหาที่มีลูกค้า 75 คน

ลูกค้าคนที่	พิกัด		ความต้องการ	ลูกค้าคนที่	พิกัด		ความต้องการ
	x	y			x	y	
1	22	22	18	25	17	64	14
2	36	26	26	26	41	46	18
3	21	45	11	27	55	34	17
4	45	35	30	28	35	16	29
5	55	20	21	29	52	26	13
6	33	34	19	30	43	26	22
7	50	50	15	31	31	76	25
8	55	45	16	32	22	53	28
9	26	59	29	33	26	29	27
10	40	66	26	34	50	40	19
11	55	65	37	35	55	50	10
12	35	51	16	36	54	10	12
13	62	35	12	37	60	15	14
14	62	57	31	38	47	66	24
15	62	24	8	39	30	60	16
16	21	36	19	40	30	50	33
17	33	44	20	41	12	17	15
18	9	56	13	42	15	14	11
19	62	48	15	43	16	19	18
20	66	14	22	44	21	48	17
21	44	13	28	45	50	30	21
22	26	13	12	46	51	42	27
23	11	28	6	47	50	15	19
24	7	43	27	48	48	21	20

ตารางผนวกที่ ข3 (ต่อ)

ลูกค้าคนที่	พิกัด		ความต้องการ	ลูกค้าคนที่	พิกัด		ความต้องการ
	x	y			x	y	
49	12	38	5	63	20	30	11
50	15	56	22	64	15	5	28
51	29	39	12	65	50	70	9
52	54	38	19	66	57	72	37
53	55	57	22	67	45	42	30
54	67	41	16	68	38	33	10
55	10	70	7	69	50	4	8
56	6	25	26	70	66	8	11
57	65	27	14	71	59	9	3
58	40	60	21	72	35	60	1
59	70	64	24	73	27	24	6
60	64	4	13	74	40	20	10
61	36	6	15	75	40	37	20
62	30	20	18				

ตารางผนวกที่ ข4 ชุดข้อมูลพิกัดและความต้องการของลูกค้าของปัญหาที่มีลูกค้า 100 คน

ลูกค้าคนที่	พิกัด		ความต้องการ	ลูกค้าคนที่	พิกัด		ความต้องการ
	x	y			x	y	
1	41	49	10	25	65	20	6
2	35	17	7	26	45	30	17
3	55	45	13	27	35	40	16
4	55	20	19	28	41	37	16
5	15	30	26	29	64	42	9
6	25	30	3	30	40	60	21
7	20	50	5	31	31	52	27
8	10	43	9	32	35	69	23
9	55	60	16	33	53	52	11
10	30	60	16	34	65	55	14
11	20	65	12	35	63	65	8
12	50	35	19	36	2	60	5
13	30	25	23	37	20	20	8
14	15	10	20	38	5	5	16
15	30	5	8	39	60	12	31
16	10	20	19	40	40	25	9
17	5	30	2	41	42	7	5
18	20	40	12	42	24	12	5
19	15	60	17	43	23	3	7
20	45	65	9	44	11	14	18
21	45	20	11	45	6	38	16
22	45	10	18	46	2	48	1
23	55	5	29	47	8	56	27
24	65	35	3	48	13	52	36

ตารางผนวกที่ ข4 (ต่อ)

ลูกค้าคนที่	พิกัด		ความต้องการ	ลูกค้าคนที่	พิกัด		ความต้องการ
	x	y			x	y	
49	6	68	30	73	44	17	9
50	47	47	13	74	46	13	8
51	49	58	10	75	49	11	18
52	27	43	9	76	49	42	13
53	37	31	14	77	53	43	14
54	57	29	18	78	61	52	7
55	63	23	2	79	57	48	23
56	53	12	6	80	56	37	6
57	32	12	7	81	55	54	26
58	36	26	18	82	15	47	16
59	21	24	28	83	14	37	11
60	17	34	3	84	11	31	7
61	12	24	13	85	16	22	41
62	24	58	19	86	4	18	35
63	27	69	10	87	28	18	26
64	15	77	9	88	26	52	9
65	62	77	20	89	26	35	15
66	49	73	25	90	31	67	3
67	67	5	25	91	15	19	1
68	56	39	36	92	22	22	2
69	37	47	6	93	18	24	22
70	37	56	5	94	26	27	27
71	57	68	15	95	25	24	20
72	47	16	25	96	22	27	11

ตารางผนวกที่ ข4 (ต่อ)

ลูกค้าคนที่	พิกัด		ความต้องการ
	x	y	
97	25	21	12
98	19	21	10
99	20	26	9
100	18	18	17

ประวัติการศึกษา และการทำงาน

ชื่อ – นามสกุล	นางสาวอุบลรัตน์ เธียรธนาคม
วัน เดือน ปี ที่เกิด	8 พฤศจิกายน พ.ศ. 2524
สถานที่เกิด	จ. พะเยา
ประวัติการศึกษา	ปริญญาตรีวิทยาศาสตร์ สาขาสถิติ มหาวิทยาลัยเกษตรศาสตร์
ตำแหน่งหน้าที่การงานปัจจุบัน	
สถานที่ทำงานปัจจุบัน	
ผลงานดีเด่นและรางวัลทางวิชาการ	
ทุนการศึกษาที่ได้รับ	