การแก้ปัญหาการจัดลำดับเมืองของการเดินทางทางอากาศโดยวิธีการค้นหาเฉพาะที่

ธนบูรณ์ ศรทัตต์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
บัณฑิตวิทยาลัย สาขาเทคโนโลยีวิศวกรรม
สถาบันเทคโนโลยีไทย-ญี่ปุ่น
ปีการศึกษา 2558

LOCAL SEARCH ALGORITHM FOR SEQUENCING CITIES OF AIRLINE CIRCLE TRIP FLIGHT PROBLEM

Thanaboon Saradatta

A Thesis Submitted in Partial Fulfillment of Requirements

for The Degree of Master of Engineering Program in Engineering Technology

Graduate School

Thai-Nichi Institute of Technology

Academic Year 2015

หัวข้อวิทยานิพนธ์	การแก้ปัญหาการจัดลำดับเมืองของการเดินทางทางอากาศโดยวิธีการค้นหาเฉพาะที่
โดย	ธนบูรณ์ ศรทัตต์
สาขาวิชา	เทคโนโลยีวิศวกรรม
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผู้ช่วยศาสตราจารย์ ดร. พิศุทธิ์ พงศ์ชัยฤกษ์

บัณฑิตวิทยาลัย สถาบันเทคโนโลยีไทย-ญี่ปุ่น อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีบัณฑิตวิทยาลัย

(รองศาสตราจารย์ ดร. พิชิต สุขเจริญพงษ์)

วันที่.....เดือน.....พ.ศ.....

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ

(ดร. บุญฑริกา เกษมสันติธรรม)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร. วิภาวดี วงษ์สุวรรณ์)

.....กรรมการ

(ดร. กันติชา กิตติพีรชล)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์

(ผู้ช่วยศาสตราจารย์ ดร. พิศุทธิ์ พงศ์ชัยฤกษ์)

ธนบูรณ์ ศรทัตต์ : การแก้ปัญหาการจัดลำดับเมืองของการเดินทางทางอากาศโดยวิธีการค้นหาเฉพาะที่. อาจารย์ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร. พิศุทธิ์ พงศ์ชัยฤกษ์, 93 หน้า.

งานวิจัยนี้ศึกษาปัญหาการเดินทางของพนักงานขายแบบไม่สมมาตรที่ขึ้นอยู่กับเวลา ที่มีข้อจำกัดในเรื่องของกรอบเวลากำหนดและข้อจำกัดของการลำดับรวมอยู่ด้วย เพื่อที่จะให้ปัญหานั้นมีความใกล้เคียงกับปัญหาที่เกิดขึ้นจริงของการขนส่งทางอากาศ โดยปัญหาที่ทำการศึกษานี้นั้นมีความซับซ้อนมากกว่าปัญหาการเดินทางของพนักงานขายแบบไม่สมมาตรรูปแบบดั้งเดิม เนื่องจากคุณสมบัติของราคาตั๋วเครื่องบิน ข้อจำกัดเรื่องของกรอบเวลากำหนด และข้อจำกัดเรื่องของการลำดับจากการศึกษาเพื่อที่จะแก้ปัญหานี้ งานวิจัยนี้ได้ทำการปรับปรุงวิธี Nearest Neighbor Algorithm เพื่อให้สามารถใช้แก้ปัญหานี้ได้ และได้ทำการพัฒนาวิธีการค้นหาเฉพาะที่โดยใช้การ Swap และการ Insert แบบพิเศษ เพื่อช่วยในการลดโอกาสในการสร้างคำตอบที่เป็นไปไม่ได้ โดยผลจากการทดลองแสดงให้เห็นว่าผลเฉลยที่ดีที่สุดในการรันทั้งหมด 10 ครั้ง ของวิธีการค้นหาเฉพาะที่นั้นจะมีค่าที่ดีกว่าผลเฉลยที่ได้จากวิธี Nearest Neighbor Algorithm

บัณฑิตวิทยาลัย
สาขาวิชา เทคโนโลยีวิศวกรรม
ปีการศึกษา 2558

ลายมือชื่อนักศึกษา .....
ลายมือชื่ออาจารย์ที่ปรึกษา .....

THANABOON SARADATTA : LOCAL SEARCH ALGORITHMS FOR SEQUENCING CITIES OF AIRLINE CIRCLE TRIP FLIGHT PROBLEM. ADVISOR : ASSISTANT PROFESSOR DR. PISUT PONGCHAIRERKS, 93 PP.

This research studies an asymmetric travelling salesman problem with time window constraints and precedence constraints in a real application of air travelling. This studied problem is more complicated than the original asymmetric travelling salesman problem because of the change of ticket prices over time, the time window constraints and the precedence constraints. To solve this problem, this research modifies a nearest neighbor algorithm. This research also improves local search algorithms, which use special swap operator and special insert operator in order to reduce a chance to generate infeasible solutions. The results show that the best found solution value over 10 runs from each local search algorithm is better than the solution value from nearest neighbor algorithm.

Graduate School Student's Signature.....
Field of Study Engineering Technology Advisor's Signature.....
Academic Year 2015

กิตติกรรมประกาศ

วิทยานิพนธ์นี้กระทำสำเร็จลุล่วงไปด้วยดี อันเนื่องมาจากความกรุณาของ ผู้ช่วยศาสตราจารย์ ดร. พิศุทธิ์ พงศ์ชัยฤกษ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ กรุณาสละเวลาอันมีค่าของท่าน มาให้คำแนะนำและให้คำปรึกษาแนวทางและการพัฒนาในการทำวิจัย ตลอดจนสอนขั้นตอนการดำเนินวิจัยตลอดการทำงานวิจัยนี้

ขอขอบพระคุณ ดร. บุณฑริกา เกษมสันติธรรม ที่กรุณาเสียสละเวลาให้คำปรึกษา ข้อแนะนำและความรู้ในเรื่องต่างๆ

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร. วิภาวดี วงษ์สุวรรณ์ ที่กรุณาเสียสละเวลาเป็นคณะกรรมการสอบวิทยานิพนธ์ในครั้งนี้

ขอขอบพระคุณ ดร. กันติชา กิตติพีรชล ที่กรุณาเสียสละเวลาเป็นคณะกรรมการสอบวิทยานิพนธ์ในครั้งนี้

และท้ายที่สุดขอขอบพระคุณ คุณพ่อ คุณแม่ และครอบครัว ที่ให้โอกาสในการศึกษาและขอบคุณเพื่อนๆ อาจารย์ และผู้มีพระคุณทุกท่าน ทั้งผู้ที่กล่าวนามและไม่ได้กล่าวนามใน ณ ที่นี้ผู้เขียนมีความซาบซึ้งในความกรุณาอันดียิ่งจากทุกท่านและขอกราบขอบพระคุณมา ณ โอกาสนี้

ธนบูรณ์ ศรทัตต์

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การจัดการปัญหาด้านการขนส่งในปัจจุบันมีความซับซ้อนมากขึ้นกว่าในอดีตเป็นอย่างมาก โดยเฉพาะการจัดการด้านการขนส่งระหว่างประเทศ ซึ่งในปัจจุบันการขนส่งสินค้าระหว่างประเทศไม่ได้จำกัดเฉพาะบริษัทเอกชนขนาดใหญ่เท่านั้น แต่ยังรวมไปถึงธุรกิจขนาดกลาง และธุรกิจขนาดเล็กหรือ SME อีกด้วย ซึ่งในการขนส่งนั้นก็มีปัจจัยหลายอย่างที่ต้องนำมาพิจารณาเพื่อทำการตัดสินใจว่าควรที่จะเลือกทำการขนส่งด้วยวิธีใดจึงจะเป็นการเหมาะสมที่สุด แต่เนื่องจากเวลานั้นเป็นอุปสรรคที่สำคัญในการขนส่งสินค้าทั้งทางเรือ และทางถนน สำหรับสินค้าบางประเภทแล้วเวลาที่ใช้ในการขนส่งจำเป็นอย่างยิ่งที่จะต้องใช้ให้คุ้มค่าและเกิดประโยชน์สูงสุด การขนส่งสินค้าทางอากาศนั้นมีบทบาทสำคัญในการขนส่งสินค้าที่ต้องการแข่งกับเวลาและลดความเสียหายที่มีสาเหตุจากการขนส่งน้อยที่สุด ด้วยลักษณะเฉพาะตัวที่มีความเร็วสูงเมื่อเทียบกับรูปแบบการขนส่งทุกประเภทสามารถทำระยะทางได้ไกลกว่าการขนส่งทางถนน และยังสามารถทำการขนส่งสินค้าได้หลากหลายประเภท

งานวิจัยชิ้นนี้มีความสนใจในปัญหาการขนส่งสินค้าซึ่งจะต้องมีพนักงานขายหรือพนักงานบริการหลังการขายเป็นผู้ไปส่งสินค้าด้วยตัวเอง เช่น การขนส่งสินค้าที่เป็นเครื่องมือทางการแพทย์ที่มีความจำเป็นต้องมีผู้เชี่ยวชาญไปสอนการติดตั้ง และวิธีใช้อย่างถูกต้อง เป็นต้น ซึ่งต้นทุนในส่วนการขนส่งสินค้านี้เป็นต้นทุนที่บริษัทจำเป็นต้องทำการบริหารจัดการให้ดีที่สุด และการที่จะส่งสินค้าให้กับลูกค้านั้น ในความเป็นจริงแล้วนั้นก็ย่อมต้องมีการทำการนัดหมายกับทางลูกค้า งานวิจัยนี้จึงมีการทำการศึกษาเกี่ยวกับการกำหนดช่วงเวลาที่จะทำการเดินทางเพื่อไปส่งสินค้าด้วย

ลักษณะของปัญหาแบบนี้ยังสามารถนำไปใช้ได้กับบริษัทที่ทำธุรกิจทัวร์ที่จัดตารางการเดินทางให้กับนักท่องเที่ยวที่ต้องการไปในหลายๆ ประเทศในการท่องเที่ยวหนึ่งครั้ง เช่น การทัวร์ยุโรป ซึ่งการไปในหนึ่งครั้งก็ควรที่จะสามารถไปเที่ยวได้ในหลายประเทศ เช่น อังกฤษ ไปต่อฝรั่งเศส แล้วไปอิตาลี เป็นต้น แน่นอนว่าบริษัททัวร์ที่เสนอจำนวนประเทศที่มากกว่าด้วยราคาที่เท่ากัน หรือเสนอประเทศในการเดินทางไปที่เหมือนกันแต่ในราคาที่ประหยัดกว่าจะเป็นบริษัทที่นักท่องเที่ยวเลือก และในความเป็นจริงแล้วนั้นการเดินทางไปยุโรปก็ย่อมจะมีการกำหนดว่าควรจะเดินทางไปประเทศไหนก่อนประเทศในหลัง งานวิจัยนี้จึงมีการทำการศึกษาเกี่ยวกับการกำหนดลำดับของการเดินทางด้วย

โดยเป็นลักษณะของการท่องเที่ยวที่ทำการเริ่มต้นออกเดินทางจากจุดๆ หนึ่ง ไปยังจุดอื่นๆ โดยที่ไปในแต่ละจุดนั้นเพียงครั้งเดียว และเมื่อที่เดินทางไปยังครบทุกจุดแล้วก็จะทำการเดินทางกลับมายังที่จุดเริ่มต้นลักษณะคล้ายกับการเดินวนรอบ โดยมีวัตถุประสงค์เพื่อที่จะทำการประหยัดค่าใช้จ่ายที่ใช้ในการเดินทางนั้นให้ได้มากที่สุด โดยปัญหานี้นั้นจะมีลักษณะคล้ายกับปัญหา Travelling Salesman Problem (TSP) ซึ่งโดยทั่วไปแล้วปัญหา TSP นั้นจะนิยมนำมาใช้กับปัญหาเกี่ยวกับการเดินทางทางบก แต่งานวิจัยนี้จะนำปัญหา TSP นี้มาประยุกต์ใช้เกี่ยวกับการเดินทางทางอากาศ ซึ่งมีลักษณะเป็นเดินทางไปยังเมืองต่างๆ ตามจำนวนของเมืองที่กำหนดให้ โดยเป็นการเดินทางทางอากาศโดยเครื่องบิน ซึ่งการเดินทางโดยเครื่องบินนั้นจะทำให้การเดินทางมีเงื่อนไขที่เพิ่มขึ้นมา โดยเงื่อนไขที่เพิ่มขึ้นมานี้สามารถสรุปได้ดังนี้

- ราคาตั๋วเครื่องบินที่ใช้ในการเดินทางจากประเทศหนึ่งไปยังอีกประเทศหนึ่งในช่วงเวลาเดียวกันที่ถูกเสนอโดยสายการบินที่ต่างกันอาจจะมีราคาที่ต่างกัน

- ราคาตั๋วเครื่องบินที่ใช้ในการเดินทางจากประเทศหนึ่งไปยังอีกประเทศหนึ่งในขาไปอาจจะไม่เท่ากันกับราคาตั๋วที่ใช้ในการเดินทางในทิศทางที่กลับกันหรือในขากลับ ซึ่งข้อนี้จะทำให้ปัญหามีลักษณะเป็นปัญหาที่ไม่สมมาตรกัน หรือเรียกว่า Asymmetric Travelling Salesman Problem (ATSP)

- ราคาตั๋วเครื่องบินที่ใช้ในการเดินทางจากประเทศหนึ่งไปยังอีกประเทศหนึ่งที่ถูกเสนอโดยสายการบินหนึ่งอาจเปลี่ยนแปลงไปตามช่วงเวลา ซึ่งข้อนี้จะทำให้ปัญหามีลักษณะเป็นปัญหาที่ขึ้นกับเวลา

- มีความเป็นไปได้ว่าที่จะไม่มีเที่ยวบินจากประเทศหนึ่งไปยังอีกประเทศหนึ่ง

- มีความเป็นไปได้ว่าที่อาจจะต้องเดินทางไปยังประเทศใดประเทศหนึ่งในช่วงเวลาที่กำหนด ซึ่งข้อนี้จะทำให้ปัญหานี้มีข้อจำกัดในเรื่องของกรอบเวลากำหนด

- มีความเป็นไปได้ว่าที่จะต้องเดินทางไปยังประเทศใดประเทศหนึ่งเป็นลำดับก่อนที่จะเดินทางไปยังอีกประเทศหนึ่ง ซึ่งข้อนี้จะทำให้ปัญหานี้มีข้อจำกัดในเรื่องของการลำดับ

- การเดินทางไปยังแต่ละประเทศนั้นจะถูกเดินทางไปเพียงครั้งเดียว และประเทศที่จะเดินทางไปเป็นประเทศสุดท้ายนั้นก็คือประเทศที่เป็นจุดเริ่มต้นของการออกเดินทาง ซึ่งข้อนี้ก็มีลักษณะที่เหมือนกับเงื่อนไขของปัญหา TSP แบบดั้งเดิม

จากเงื่อนไขที่ได้กล่าวข้างต้นนั้นจะทำให้ปัญหา TSP ที่งานวิจัยนี้ทำการศึกษามีความซับซ้อนมากกว่าปัญหา TSP แบบดั้งเดิม โดยมีการเพิ่มข้อจำกัดต่างๆ ให้กับปัญหาเพื่อที่จะทำให้ปัญหานั้นมีความใกล้เคียงกับปัญหาที่เกิดขึ้นจริงของการเดินทางทางอากาศ และเป้าหมายของการศึกษาปัญหานี้ก็เพื่อศึกษาหาต้นทุนที่ต่ำที่สุดที่ใช้ในการเดินทางทั้งหมด ซึ่งต้นทุนเหล่านี้นั้นจะทำการศึกษาและเก็บข้อมูลจากข้อมูลจริงทั้งหมด และงานวิจัยนี้จะทำการเลือกสายการบินที่เสนอต้นทุนที่ต่ำที่สุดของการ

เดินทางจากประเทศหนึ่งไปยังอีกประเทศหนึ่งจากหลายๆ สายการบินที่เสนอการเดินทางในเส้นทางเดียวกัน และช่วงเวลาเดียวกัน โดยการเดินทางนั้นจะทำการเริ่มต้นเดินทางออกจากประเทศไทยไปยังประเทศต่างๆ และเมื่อเดินทางครบทุกประเทศแล้วก็จะทำการเดินทางกลับมายังประเทศจุดเริ่มต้นนั่นคือประเทศไทย

1.2 วัตถุประสงค์ของการศึกษา

เพื่อศึกษาการจัดลำดับของการเดินทางทางอากาศโดยเครื่องบินไปยังประเทศต่างๆ ตามจำนวนประเทศที่ถูกกำหนด โดยมีการเพิ่มข้อจำกัดต่างๆ เพื่อที่จะให้ปัญหานั้นมีความใกล้เคียงกับปัญหาที่เกิดขึ้นจริง และทำการเก็บข้อมูลต้นทุนของการเดินทางจากข้อมูลจริง โดยวัตถุประสงค์ของการศึกษาเพื่อที่จะหาต้นทุนที่ต่ำที่สุดที่ใช้ในการเดินทางไปยังประเทศต่างๆ และศึกษาเพื่อทำการปรับปรุงและพัฒนาวิธี Nearest Neighbor algorithm และวิธี Local Search algorithm เพื่อให้สามารถนำมาใช้ในการแก้ปัญหาที่ทำการศึกษาได้

1.3 ขอบเขตการศึกษาและวิจัย

1.3.1 ศึกษาประเทศจำนวน 20 ประเทศ

1.3.2 ศึกษาข้อมูลต้นทุนของแต่ละสายการบินใน 20 สัปดาห์

1.3.3 ศึกษาเฉพาะข้อมูลของเที่ยวบินตรงเท่านั้น

1.3.4 ข้อมูลต้นทุนนั้นจะทำการเก็บจากการเดินทางไปยังเมืองที่มีชื่อเสียงต่างๆ ของแต่ละประเทศและเลือกต้นทุนที่ต่ำที่สุด

1.3.5 ทำการศึกษาปัญหาทั้งหมด 6 กรณี โดยแบ่งปัญหาเป็น 2 ชุด

1.3.5.1 ปัญหาชุดที่ 1 ประกอบด้วย กรณีที่ 1 กรณีที่ 2 และ กรณีที่ 3 โดยปัญหาชุดที่ 1 จะทำการศึกษา 15 ประเทศ

1.3.5.2 ปัญหาชุดที่ 2 ประกอบด้วย กรณีที่ 4 กรณีที่ 5 และ กรณีที่ 6 โดยปัญหาชุดที่ 2 จะทำการศึกษา 20 ประเทศ

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 สามารถทำการแก้ปัญหา ATSP ที่มีการเพิ่มข้อจำกัดต่างๆ ได้

1.4.2 งานวิจัยนี้อาจจะเป็นประโยชน์แก่ผู้ที่ต้องการจะจัดการการเดินทางไปยังประเทศต่างๆ หลายๆ ประเทศได้

1.5 สมมติฐานในงานวิจัย

1.5.1 การเดินทางจะเริ่มเดินทางออกจากประเทศไทยและเมื่อเดินทางครบทุกประเทศแล้วก็จะกลับมายังประเทศไทย

1.5.2 ต้นทุนที่ใช้ในการเดินทางไปและกลับระหว่างสองประเทศจะไม่เท่ากัน

1.5.3 ต้นทุนของการเดินทางโดยเครื่องบินนั้นจะเปลี่ยนไปตามช่วงเวลา

1.5.4 ต้นทุนของการเดินทางที่ถูกเสนอโดยสายการบินที่ต่างกันจะมีต้นทุนที่ต่างกัน

1.5.5 ในการเดินทางไปยังประเทศต่างๆ นั้นจะมีข้อจำกัดในเรื่องของกรอบเวลากำหนดและข้อจำกัดในเรื่องของการลำดับรวมอยู่ด้วย

1.6 แผนงานและระยะเวลาในการดำเนินงาน

ตารางที่ 1.1 ระยะเวลาในการเนินงานวิจัย

รายละเอียด	2557	ปี 2558												ปี 2559	
	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.
ศึกษาทฤษฎีและงานวิจัยเกี่ยวข้อง															
กำหนดปัญหาและศึกษาวิธีการแก้ไขปัญหา															
ศึกษาการเขียนโปรแกรม															
เก็บข้อมูลที่ใช้ในงานวิจัย															
วิเคราะห์และสรุปผล															
ทำบทความวิชาการและรูปเล่มงานวิจัย															

บทที่ 2
ปัญหา อัลกอลิทึมและงานวิจัยที่เกี่ยวข้อง

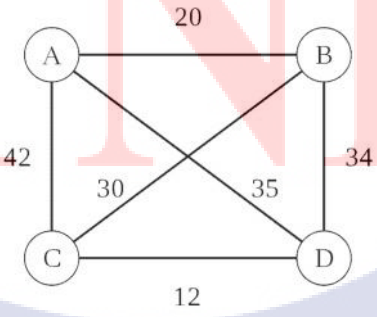
ในบทนี้จะกล่าวถึงปัญหา อัลกอลิทึม และเอกสารงานวิจัยที่เกี่ยวข้องกับการแก้ปัญหาในงานวิจัยนี้

2.1 ปัญหาที่เกี่ยวข้อง

2.1.1 Travelling Salesman Problem (TSP)

ปัญหา Travelling Salesman Problem (TSP) เป็นปัญหาทางคณิตศาสตร์ที่ถูกพัฒนามาตั้งแต่ปี 1800 โดย Sir William Rowan Hamilton และ Thomas Penyngton Kirkman ในปี 1930 ได้มีการศึกษา รูปแบบทั่วไปของ TSP เป็นครั้งแรกโดย Karl Menger โดยปัญหานี้นั้นเป็นปัญหาที่ทำการตัดสินใจหาเส้นทางการเดินทางเมื่อมีเมืองหรือสถานที่ที่ต้องเดินทางไปจำนวน N เมือง หรือ N สถานที่ การเดินทางจะเดินทางจากเมืองใดเมืองหนึ่งในจำนวน N เมือง โดยเส้นทางการเดินทางนั้นๆ จะต้องเดินผ่านทุกเมืองใน N และกลับมาที่เมืองที่ทำการเริ่มต้นในการเดินเหมือนการเดินวนรอบ โดยลักษณะคำตอบที่ได้นั้นอาจมีระยะทางหรือระยะเวลาหรือค่าใช้จ่ายในการเดินทางโดยรวมต่ำที่สุด โดยปัญหา TSP จัดเป็นปัญหา NP-Hard [1] ซึ่งไม่สามารถหาคำตอบที่เหมาะสมที่สุดภายในระยะเวลาการคำนวณที่เหมาะสมได้ โดยเฉพาะเมื่อปัญหามีขนาดใหญ่ อาทิเช่น ถ้าจะหาเส้นทางการเดินทางที่เป็นไปได้ทั้งหมดของ 10 เมือง ซึ่งต้องอาศัยการคิดคำนวณคำตอบที่เป็นไปได้ 3,628,800 ครั้ง

สำหรับปัญหา TSP แบบดั้งเดิมนั้นในบางครั้งเรียกว่า Symmetric TSP ซึ่งจะมีระยะทางระหว่างขาไปและขากลับของระหว่างสองเมืองนั้นเท่ากัน โดยตัวอย่างของปัญหาแสดงไว้ในรูปที่ 2.1



รูปที่ 2.1 ตัวอย่างปัญหา TSP

ปัญหา TSP สามารถเขียนเป็นโมเดลทางคณิตศาสตร์ หรือ Formulation of TSP [2] ได้ดังต่อไปนี้

$$\text{Minimize} \quad z = \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij}, c_{ij} = \infty \text{ for } i = j \tag{2.1}$$

$$\text{Subject to} \quad \sum_{i=0}^{n} x_{ij} = 1,\ j = 0, 1,\ldots,n \tag{2.2}$$

$$\sum_{i=0}^{n} x_{ij} = 1,\ i = 0, 1,\ldots,n \tag{2.3}$$

$$x_{ij} = (0, 1) \text{ for all I, j in } \{0, 1,\ldots,n\} \tag{2.4}$$

$$X = (x_{i,j}) \in S \tag{2.5}$$

โดยที่มีตัวแปรตัดสินใจ x_{ij} ซึ่งแทนการตัดสินใจที่จะเดินทางจากประเทศ i ไปยังประเทศ j และมี

n เป็นจำนวนของปม (Node) หรือ ประเทศ

c_{ij} เป็นระยะทางในการเดินทางจากประเทศ i ไปยังประเทศ j

<u>เทคนิคในการหาคำตอบของปัญหา TSP สามารถแบ่งได้เป็น 3 รูปแบบ</u> ดังนี้

1. Exact Algorithms คือ รูปแบบวิธีการหาคำตอบของปัญหาที่ได้คำตอบที่ดีที่สุดแน่นอน เช่นวิธี Brute Force [3] ซึ่งเป็นการหาทุกคำตอบที่เป็นไปได้ของปัญหาและเลือกผลลัพธ์ที่ดีที่สุด แต่วิธีการแบบนี้ไม่เหมาะกับปัญหา TSP ที่มีขนาดใหญ่ เนื่องจากจะต้องใช้เวลาในการคำนวณที่นานเพื่อให้ได้คำตอบทั้งหมด หรือวิธีการที่ไม่ขัดแย้งกับทุกเงื่อนไขที่กำหนด แต่ใช้เวลานานเมื่อมีเงื่อนไขมาก หรือมีตัวแปรที่เพิ่มขึ้น ซึ่งมีวิธีที่นิยมใช้คือแบบ Branch and Bound [4] โดยวิธีนี้นั้นมีหลักการของการแก้ปัญหา คือ เริ่มต้นด้วยการการหาเส้นทางขึ้นมาเส้นทางหนึ่ง แล้วพยายามหาเส้นทางอื่นขึ้นมาเปรียบเทียบ ถ้าเส้นทางใหม่ที่ได้ดีกว่า ให้นำเส้นทางใหม่ที่ได้เป็นเส้นทางหลักในการเปรียบ เทียบกับเส้นทางอื่นต่อไป แต่ถ้าเส้นทางใหม่ที่สร้างขึ้นเมื่อวัดแล้ว แม้เพียงบางส่วนที่ยาวกว่าเส้นทางหลักก็ให้ยกเลิกเส้นทางนั้นทำเช่นนี้เรื่อยๆ จนกระทั่งได้เส้นทางที่ดีที่สุด แต่อย่างไรก็ดีการแก้ปัญหา TSP ด้วยวิธีนี้ ก็ยังไม่ใช่วิธีที่ดีนักหากขนาดของปัญหานั้นมีขนาดที่ใหญ่ เนื่องจากต้องอาศัยการสร้างคำตอบ ที่เป็นไปได้ทั้งหมด

2. Heuristic และ Approximation Algorithms ซึ่งเป็นรูปแบบวิธีการหาคำตอบตอบของปัญหาที่มีความรวดเร็วกว่าวิธี Exact algorithms โดยเป้าหมายของวิธี Heuristics นี้นั้นคือเป็นวิธีการที่ใช้สำหรับในการหาคำตอบโดยอยู่ภายใต้เวลาที่เหมาะสมที่สามารถหาคำตอบได้ด้วยมือ โดยที่คำตอบที่ได้ออกมานั้นอาจจะไม่ใช่คำตอบที่ดีที่สุดของคำตอบที่สามารถมีได้ในทั้งหมดของปัญหา

หรือคำตอบที่ได้ออกมานั้นอาจจะมีค่าใกล้เคียงกับค่าของคำตอบที่ได้จากวิธี Exact Algorithms แต่สิ่งที่ทำให้วิธี Heuristics นี้นั้นมีประโยชน์ก็คือ การหาคำตอบออกมานั้นไม่จำเป็นที่จะต้องใช้เวลาในการหาคำตอบนานเหมือนวิธี Exact Algorithms โดยตัวอย่างของวิธีที่ใช้ในการหาคำตอบที่มีลักษณะในรูปแบบของวิธี Problem-Based Heuristic และ Approximation Algorithms ที่สร้างขึ้น เพื่อหาคำตอบของปัญหา TSP ได้แก่วิธี Nearest Neighbor Algorithm (NN) [5] โดยวิธีนี่นั้นมีหลักการทำงานคือ ทำการเริ่มต้นโดยการสุ่มจากเมืองใดก็ได้ในเมืองทั้งหมด และจากนั้นก็ทำการเดินทางไปยังเมืองต่อไปโดยการเลือกเมืองที่มีระยะทางใกล้ที่สุด และทำแบบนี้ไปจนเดินทางไปยังครบทุกเมือง แต่บางครั้งคำตอบที่ได้นั้นอาจจะไม่ใช้คำตอบที่ดีที่สุด

3. Metaheuristic โดยวิธีนี้นั้นเป็นวิธีหนึ่งของการหาคำตอบที่ดีที่สุดโดยอาศัยหลักการประมาณ และเป็นวิธีที่ประสบความสำเร็จอย่างมาก เนื่องจากมีความรวดเร็วในการประมวลผลในการแก้ปัญหาที่มีความซับซ้อนสูงๆ โดยตัวอย่างวิธีที่ใช้ในการหาคำตอบที่มีลักษณะในรูปแบบของวิธี Metaheuristics ที่สร้างขึ้นเพื่อหาคำตอบของปัญหา TSP เป็นดังต่อไปนี้

3.1 Ant Colony Optimization Algorithm [6] ซึ่งเลียนแบบพฤติกรรมการหาอาหารของมดโดยค้นการหาเส้นทางที่สั้นที่สุดระหว่างรังกับแหล่งอาหารโดยใช้ฟีโรโมน (Pheromone) ที่มดวางไว้ระหว่างทางเพื่อใช้ในการสื่อสารทางอ้อมกับมดตัวอื่นในฝูง

3.2 Genetic Algorithm [7] โดยวิธีนี้เป็นเทคนิคสำหรับค้นหาผลเฉลย หรือคำตอบโดยประมาณของปัญหา โดยอาศัยหลักการจากทฤษฎีวิวัฒนาการจากชีววิทยา และการคัดเลือกตามธรรมชาติ โดยมีการจำลองโปรแกรมคอมพิวเตอร์ เพื่อแก้ปัญหาค่าเหมาะสมที่สุด โดยมีการแทนค่าคำตอบที่มีอยู่ในลักษณะโครโมโซม แล้วทำการปรับปรุงคำตอบทีละชุดเพื่อที่จะให้ได้คำตอบที่ดีขึ้น

3.3 Memetic Algorithm [8] โดยวิธีนี้นั้นเป็นกระบวนการที่นำขั้นตอนของวิธี Genetic Algorithm มาไฮบริดกับการ Local Search หรือ การค้นหาแบบเฉพาะที่ โดยการค้นหาเฉพาะที่นั้นทำเพื่อช่วยปรับปรุงประสิทธิภาพให้ประชากรดีขึ้น และเพื่อให้ได้ผลลัพธ์ที่ดีขึ้น

2.1.2 Asymmetric Travelling Salesman Problem (ATSP)

Asymmetric TSP หรือ ATSP มีลักษณะของปัญหาที่เหมือนกับ TSP ซึ่งเป็นปัญหาที่ทำการตัดสินใจหาเส้นทางการเดินทางเมื่อมีเมืองหรือสถานที่ที่ต้องเดินทางไปจำนวน N เมือง หรือ N สถานที่ การเดินทางจะเดินทางจากเมืองใดเมืองหนึ่งในจำนวน N เมือง โดยเส้นทางการเดิน ทางนั้นๆ จะต้องเดินผ่านทุกเมืองใน N และกลับมาที่เมืองที่ทำการเริ่มต้นในการเดินเหมือนการเดินวนรอบโดยลักษณะคำตอบที่ได้นั้นอาจมีระยะทางหรือระยะเวลาหรือค่าใช้จ่ายในการเดินทางโดยรวมต่ำที่สุด

เพียงแต่จะมีความแตกต่างกับปัญหาแบบ TSP ตรงที่จะมีระยะทางระหว่างขาไปและขากลับของระหว่างสองเมืองนั้นไม่เท่ากัน โดยตัวอย่างของปัญหา ATSP นั้นสามารถแสดงได้ดังรูปที่ 2.2

		To			
		1	2	3	4
From	1	-	10	6	3
	2	8	-	5	8
	3	7	5	-	2
	4	4	10	2	-

รูปที่ 2.2 ลักษณะปัญหาแบบ ATSP

โดยปัญหา ATSP สามารถเขียนในรูปแบบโมเดลทางคณิตศาสตร์ หรือ Formulation of ATSP [9] ได้ดังต่อไปนี้

$$\text{Minimize} \quad \sum_{i \neq j} c_{ij} x_{ij} \tag{2.6}$$

$$\text{Subject to} \quad \sum_{j=1}^{n} x_{ij} = 1 \quad (i \in V, i \neq j) \tag{2.7}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad (j \in V, j \neq i) \tag{2.8}$$

$$\sum_{i,j \in s} x_{ij} \leq |S| - 1 \quad (S \subset V, 2 \leq |S| \leq n-2) \tag{2.9}$$

$$x_{ij} = 0 \text{ or } 1 \quad (i, j) \in A \tag{2.10}$$

โดยที่มีตัวแปรตัดสินใจ x_{ij} ซึ่งแทนการตัดสินใจที่จะเดินทางจากประเทศ i ไปยังประเทศ j และมี

c_{ij} เป็นระยะทางในการเดินทางจากประเทศ i ไปยังประเทศ j

V เป็นเซตของประเทศ

A เป็นเซตของเส้นเชื่อมระบุทิศทาง

n เป็นจำนวนของประเทศ

S เป็นสับเซตของประเทศที่ไม่รวมจุดต้นกำเนิด

<u>เทคนิคในการหาคำตอบของปัญหา ATSP สามารถแบ่งได้เป็น 3 รูปแบบ</u> ดังนี้

1. Exact Algorithms เป็นวิธีการหาคำตอบที่ได้คำตอบดีที่สุดแน่นอน โดยลักษณะในการหาคำตอบนั้นจะเป็นการหาคำตอบทุกคำตอบที่เป็นไปได้ของปัญหา และทำการเลือกผลลัพธ์ที่ดี

ที่สุด แต่วิธีการในลักษณะนี้นั้นจะไม่เหมาะสมกับปัญหาที่มีขนาดใหญ่ เนื่องจากจะต้องใช้เวลาในการคำนวณที่นานเพื่อให้ได้คำตอบทั้งหมด โดยตัวอย่างวิธี Exact Algorithms ที่นำมาใช้กับ ATSP คือ Branch and Bound [10] โดยงานวิจัยนี้ได้มีการนำเสนอวิธีที่สามารถนำมาแก้ปัญหา ATSP ได้อย่างมีประสิทธิภาพ และสุดท้ายก็ได้มีการนำเอาวิธีที่มีประสิทธิภาพเหล่านี้ มาเพื่อใช้ในการทดสอบการแก้ปัญหาเพื่อที่จะสามารถทำการเปรียบได้ว่าวิธีใดนั้นมีประสิทธิภาพในการแก้ปัญหา ATSP มากกว่ากัน

2. Heuristic และ Approximation Algorithms ซึ่งเป็นรูปแบบวิธีการหาคำตอบตอบของปัญหาที่มีความรวดเร็วกว่าวิธี Exact Algorithms โดยเป้าหมายของวิธี Heuristics นี้นั้นคือเป็นวิธีการที่ใช้สำหรับในการหาคำตอบโดยอยู่ภายใต้เวลาที่เหมาะสมที่สามารถหาคำตอบได้ด้วยมือ โดยที่คำตอบที่ได้ออกมานั้นอาจจะไม่ใช่คำตอบที่ดีที่สุดของคำตอบที่สามารถมีได้ในทั้งหมดของปัญหา หรือคำตอบที่ได้ออกมานั้นอาจจะมีค่าใกล้เคียงกับค่าของคำตอบที่ได้จากวิธี Exact Algorithms โดยตัวอย่างของวิธี Problem-based Heuristic และ Approximation Algorithms ที่นำมาใช้กับ ATSP คือ วิธี Heuristic [11] โดยวิธีที่ใช้แก้ปัญหาในงานวิจัยนี้นั้นมีพื้นฐานมาจากวิธี Randomized Arbitrary Insertion (RAI) โดยหลักการของวิธีนี้จะเริ่มจากการเลือกเส้นทางที่ประกอบด้วยจุดยอดที่กำหนด และจากนั้นก็ทำการเลือกจุดยอดใหม่ที่ไม่ได้อยู่ในเส้นทางเดิม แล้วจากนั้นนำจุดยอดใหม่มาแทรกระหว่างเส้นทางเดิม โดยทำการแทรกเพื่อให้ได้ค่าเป้าหมายที่ต่ำที่สุด โดยการทำเช่นนี้เพื่อที่จะเป็นการสร้างเส้นทางเริ่มต้น และหลังจากนั้นก็จะเป็นขั้นตอนเพื่อที่จะหาคำตอบดีที่สุด โดยที่มีการเลือกการเดินทางจากเมืองหนึ่งไปยังเมืองหนึ่งขึ้นมาอย่างสุ่ม โดยมีการนำจุดยอดบางจุดออกจากเส้นทาง และทำการเลือกจุดยอดจากจุดยอดที่ทำการคัดออกเข้ามาอย่างสุ่ม โดยทำการแทรกจุดไปยังเส้นทางเริ่มต้น และเส้นทางที่สร้างขึ้นมาใหม่ แล้วหลังจากนั้นก็จะทำการเปรียบเทียบว่าคำตอบของเส้นทางไหนดีกว่ากัน

3. Metaheuristics โดยวิธีนี้นั้นเป็นวิธีหนึ่งของการหาคำตอบที่ดีที่สุดโดยอาศัยหลักการประมาณ และเป็นวิธีที่ประสบความสำเร็จอย่างมาก เนื่องจากมีความรวดเร็วในการประมวลผลในการแก้ปัญหาที่มีความซับซ้อนสูงๆ โดยตัวอย่างวิธีที่ใช้ในการหาคำตอบที่มีลักษณะในรูปแบบของวิธี Metaheuristics ที่สร้างขึ้นเพื่อหาคำตอบของปัญหา ATSP เป็นดังต่อไปนี้

3.1 Genetic Algorithm [12] ใช้ในการแก้ปัญหา ATSP โดยในงานวิจัยนี้ได้มีการขยายการค้นหา โดยรวมไปถึงการค้นหาคำตอบที่เป็นไปไม่ได้ โดยแทนที่จะทำการหาเฉพาะคำตอบที่เป็นไปได้ด้วยวิธีการครอสโอเวอร์ แต่กลับทำการค้นหาทั้งคำตอบที่เป็นไปได้และเป็นไปไม่ได้เพื่อที่จะทำให้คุณภาพของคำตอบที่ได้ออกมานั้นดีขึ้น

3.2 Memetic Algorithm [13] ใช้ในการแก้ปัญหา ATSP โดยวิธี Memetic Algorithm นั้นเป็นการทำงานร่วมกันของสองวิธีคือ Genetic Glgorithm และ Local Search โดยการใช้ทั้งสองวิธีนี้ร่วมกันจะทำให้สามารถได้คำตอบที่ดีขึ้น เพราะจะมีการทำ Local Search บนแต่

ละโครโมโซมจะทำให้การค้นหาคำตอบนั้นละเอียดมากขึ้น ซึ่งจะทำให้คำตอบที่ได้ออกมานั้น เป็นคำตอบที่ดีกว่าทำด้วยวิธี Genetic Algorithm อย่างเดียวแน่นอน

2.1.3 Traveling Salesman Problem with Time Windows

ปัญหา Traveling Salesman Problem with Time Windows เป็นปัญหาที่มีลักษณะคล้ายกับปัญหา Traveling Salesman Problem (TSP) โดยที่มีการกำหนดจุดเริ่มต้นของการเดินทางและทำการเดินทางไปยังกลุ่มของจุดต่างๆ ตามจำนวนที่กำหนด และเมื่อทำการเดินทางไปครบทุกจุดแล้วก็จะทำการเดินทางกลับมายังจุดเริ่มต้น โดยที่จะมีการเพิ่มข้อจำกัดเกี่ยวกับกรอบเวลากำหนด (Time Windows) เพิ่มขึ้นมาในปัญหา

Time Windows หรือ Time Window Constraints เป็นข้อจำกัดในเรื่องของกรอบเวลากำหนด ซึ่งนิยมนำมาประยุกต์ใช้ในปัญหาเกี่ยวกับการขนส่งต่างๆ เพื่อให้ปัญหานั้นมีความใกล้เคียงกับการขนส่งที่เกิดขึ้นจริง เนื่องจากการขนส่งสินค้าให้กับทางลูกค้านั้นในความเป็นจริงแล้วก็มักจะถูกกำหนดให้ส่งภายในกรอบเวลาการส่งมอบที่กำหนด โดยมีผู้ศึกษาปัญหาเกี่ยวกับการจัดการเส้นทางการเดินรถ หรือ Vehicle Routing Problem [14] ซึ่งผู้ศึกษานั้นได้มีการเพิ่มข้อจำกัดในเรื่องของ Time Window หรือ กรอบเวลากำหนดเข้าไปด้วย

โดยที่เทคนิคที่นิยมนำมาใช้ในการหาคำตอบของปัญหา TSP with Time Windows ได้แก่ วิธี Heuristic โดยมีผู้ที่ทำการศึกษาวิธี Heuristic แล้วนำมาใช้การแก้ปัญหา TSP with Time Window [15] โดยผู้ที่ศึกษานั้นใช้วิธี Generalized Insertion Heuristic มาเพื่อใช้ในการแก้ปัญหา โดยที่เป้าหมายของการแก้ปัญหานั้นก็เพื่อที่จะศึกษาหาเวลาทั้งหมดที่ใช้ในการเดินทางที่มีค่าน้อยที่สุด

2.1.4 Traveling Salesman Problem with Precedence Constraints

ปัญหา Traveling Salesman Problem with Precedence Constraints เป็นปัญหาที่มีลักษณะคล้ายกับปัญหา Traveling Salesman Problem (TSP) โดยที่มีการกำหนดจุดเริ่มต้นของการเดินทางและทำการเดินทางไปยังกลุ่มของจุดต่างๆ ตามจำนวนที่กำหนด และเมื่อทำการเดินทางไปครบทุกจุดแล้วก็จะทำการเดินทางกลับมายังจุดเริ่มต้น โดยที่จะมีการเพิ่มข้อจำกัดเกี่ยวกับการลำดับ (Precedence Constraints) เพิ่มขึ้นมาในปัญหา

Precedence Constraints [16] หรือที่เรียกว่า Sequential Ordering Problem เป็นข้อจำกัดในเรื่องของการลำดับ ซึ่งข้อจำกัดในเรื่องของการลำดับนี้นั้นมักจะถูกนำมาประยุกต์ใช้กับปัญหา Asymmetric Travelling Salesman Problem (ATSP) ซึ่งในความเป็นจริงของการขนส่งหรือการวางแผนการผลิตนั้นข้อจำกัดในเรื่องของการลำดับนั้นมีการเกิดขึ้นและถูกใช้ขึ้นจริงใน

ชีวิตประจำวัน อย่างเช่นในเรื่องของการขนส่งนั้นก็มักจะมีการกำหนดว่าควรจะไปส่งสินค้าที่ไหนก่อนหรือที่ไหนหลัง

โดยที่เทคนิคที่นิยมนำมาใช้ในการหาคำตอบของปัญหา TSP with Precedence Constraints ได้แก่ วิธี Heuristic โดยมีผู้ที่ทำการศึกษาวิธี Heuristic แล้วนำมาใช้การแก้ปัญหา TSP with Precedence Constraints [17] โดยผู้ที่ศึกษานั้นได้ทำการพัฒนาวิธี Heuristic ขึ้นมาโดยที่มีพื้นฐานจาก Dynamic Programming โดยผู้ที่ศึกษาเรียกวิธี Heuristic นี้ว่า HDP โดยที่เป้าหมายของการศึกษาเพื่อที่จะทำการหาต้นทุนที่มีค่าต่ำที่สุด

2.1.5 Traveling Tourist Problem (TTP)

ปัญหา Traveling Tourist Problem [18] นั้นเป็นปัญหาที่มีความคล้ายคลึงกับปัญหา Traveling Salesman Problem (TSP) โดยที่มีจะการกำหนดจำนวนของเมืองที่จะต้องเดินทางไป และมีการระบุว่าต้องเริ่มเดินทางจากเมืองและสิ้นสุดที่เมืองไหน โดยที่การเดินทางนั้นจะต้องไปให้ครบทุกเมือง โดยเป้าหมายของปัญหานั้นอาจจะเป็นการหาต้นทุนที่ต่ำที่สุดที่ใช้ในการเดินทาง ซึ่งต้นทุนของการเดินทางเหล่านี้นั้นอาจจะมีการเปลี่ยนแปลงค่าไปตามช่วงเวลาได้ แต่ปัญหา TTP นี้นั้นจะมีข้อที่แตกต่างอย่างชัดเจนจากปัญหา TSP คือ ปัญหา TTP นั้นสามารถที่จะกำหนดให้ทำการเดินทางไปในแต่ละเมืองเป็นจำนวนมากกว่า 1 ครั้งได้ ซึ่งจะแตกต่างจากปัญหา TSP โดยที่ปัญหา TSP นั้นจะสามารถเดินทางไปยังแต่ละเมืองได้เพียงแค่ครั้งเดียวเท่านั้น โดยมีผู้ที่สนใจศึกษาปัญหา TTP [19] ซึ่งผู้ที่ทำการศึกษานี้ได้กล่าวว่าสิ่งที่ทำให้ปัญหา TTP ที่เขาได้ทำการศึกษานั้นมีความแตกต่างจากปัญหา TSP ทั่วไปก็คือต้นทุนในการเดินทางที่ทำการศึกษานั้นจะการเปลี่ยนแปลงไปตามช่วงเวลา โดยที่เป้าหมายของผู้ศึกษาก็คือเพื่อศึกษาหาต้นทุนที่ใช้ในการเดินทางที่ต่ำที่สุด

2.1.6 Air Traveling Salesman Problem

ปัญหา Air Traveling Salesman Problem [20] หรือ Air TSP เป็นปัญหาที่มีลักษณะคล้ายกับปัญหา Traveling Salesman Problem (TSP) โดยที่มีการกำหนดจุดเริ่มต้นของการเดินทางและทำการเดินทางไปยังกลุ่มของจุดต่างๆ ตามจำนวนที่กำหนด และเมื่อทำการเดินทางไปครบทุกจุดแล้วก็จะทำการเดินทางกลับมายังจุดเริ่มต้น ซึ่งปัญหา Air TSP นั้นจะมีความแตกต่างจากปัญหา TSP แบบดั้งเดิมตรงที่ปัญหา Air TSP นั้นจะทำการพิจารณาการเดินทางทางอากาศ ส่วนปัญหา TSP แบบดั้งเดิมนั้นจะทำการพิจารณาการเดินทางทางบก ซึ่งในการเดินทางทางอากาศนั้นก็จะมีข้อจำกัดที่อาจจะเพิ่มมากขึ้นกว่าการเดินทางทางบก อย่างเช่นในปัญหา TSP แบบดั้งเดิมแต่ละเมืองก็จะถูกเชื่อมต่อกัน การที่จะทำการเดินทางจากเมือง 1 ไปยังเมือง 2 ก็จะทำการเดินทางออกจากเมือง 1 ไปยังเมือง 2 ด้วยระยะทางที่กำหนดแล้วการเดินทางก็สมบูรณ์ แต่ในปัญหา Air TSP นั้น

แต่ละเมืองนั้นอาจจะไม่ถูกเชื่อมต่อกัน การที่จะเดินทางจากเมือง 1 ไปยังเมือง 2 นั้นอาจจะต้องทำการเดินทางผ่านเมือง 3 มาก่อน ซึ่งตรงจุดนี้อาจจะทำให้ปัญหามีความยากเพิ่มมากขึ้น โดยที่มีผู้ที่สนใจศึกษาปัญหา [21] ที่มีลักษณะคล้ายปัญหา Air TSP โดยปัญหาที่ผู้ที่สนใจทำการศึกษานั้นประกอบไปด้วยการเดินทางออกจากสนามบินจุดเริ่มต้นแล้วเดินทางไปยังจุดภารกิจต่างๆ ที่ถูกกำหนด และเมื่อทำการเดินทางครบทุกจุดแล้วก็จะเดินทางกลับมายังสนามบินจุดเริ่มต้น โดยมีเป้าหมายเพื่อให้ต้นทุนของการดำเนินงานนั้นมีค่าต่ำที่สุด โดยที่ต้นทุนระหว่างจุดต่างๆ นั้นจะมีการเปลี่ยนแปลงไปตามช่วงเวลา

2.1.7 Time Dependent Traveling Salesman Problem

ปัญหา Time Dependent Traveling Salesman Problem เป็นปัญหาที่มีลักษณะคล้ายกับปัญหา Traveling Salesman Problem (TSP) โดยที่มีการกำหนดจุดเริ่มต้นของการเดินทางและทำการเดินทางไปยังกลุ่มของจุดต่างๆ ตามจำนวนที่กำหนด และเมื่อทำการเดินทางไปครบทุกจุดแล้วก็จะทำการเดินทางกลับมายังจุดเริ่มต้น โดยปัญหา Time Dependent TSP นั้นจะมีความแตกต่างจากปัญหา TSP แบบดั้งเดิมตรงที่ต้นทุนที่ใช้ในการดำเนินงานหรือต้นทุนที่ใช้ในการเดินทางนั้นจะมีค่าที่เปลี่ยนแปลงไปตามช่วงเวลา โดยการที่จะหาต้นทุนที่ต่ำที่สุดได้นั้นต้องมีการคำนึงถึงช่วงเวลาที่จะใช้ในการเดินทางด้วย

โดยเทคนิคที่นิยมนำมาใช้ในการหาคำตอบของปัญหา Time Dependent TSP ได้แก่ วิธี Heuristic โดยมีผู้ที่ศึกษาวิธี Heuristic แล้วนำมาใช้การแก้ปัญหา Time Dependent TSP [22] โดยผู้ที่ทำการศึกษานั้นได้ทำการพิจารณาปัญหา Time Dependent TSP ที่มีการเพิ่มข้อจำกัดในเรื่องของกรอบเวลากำหนด และข้อจำกัดในเรื่องของการลำดับ โดยผู้ศึกษานั้นได้ใช้ Dynamic Programming Heuristic มาเพื่อใช้ในการแก้ไขปัญหา

2.1.8 TSP with Time Windows and Precedence Constraints

ปัญหา TSP with Time Windows and Precedence Constraints เป็นปัญหาที่มีลักษณะคล้ายกับปัญหา Traveling Salesman Problem (TSP) โดยที่มีการกำหนดจุดเริ่มต้นของการเดินทางและทำการเดินทางไปยังกลุ่มของจุดต่างๆ ตามจำนวนที่กำหนด และเมื่อทำการเดินทางไปครบทุกจุดแล้วก็จะทำการเดินทางกลับมายังจุดเริ่มต้น โดยที่จะทำการเพิ่มข้อจำกัดกรอบเวลากำหนด (Time Windows) และข้อจำกัดของการลำดับ (Precedence Constraints) เพิ่มเข้าไปในปัญหา TSP ซึ่งการเพิ่มข้อจำกัดทั้ง 2 ข้อจำกัดนี้เข้าไปพร้อมกันในปัญหา TSP นั้นจะทำให้ปัญหา TSP นั้นมีความยากมากขึ้นและมีความซับซ้อนมากยิ่งขึ้น

โดยมีผู้ที่ศึกษาปัญหา TSP with Time Windows and Precedence Constraints โดยเทคนิคที่ผู้ศึกษานำมาใช้ในการแก้ปัญหา TSP with Time Windows and Precedence Constraints [23] ได้แก่ วิธี Metaheuristic โดยที่ผู้ศึกษานั้นได้ทำการใช้วิธี Local Search หรือวิธีการค้นหาเฉพาะที่ ซึ่งเป็นวิธีแบบ Metaheuristic มาเพื่อใช้ในการแก้ปัญหา โดยที่เป้าหมายของการศึกษาก็เพื่อที่จะทำการศึกษาหาต้นทุนรวมที่ต่ำที่สุดที่ใช้ในการเดินทาง ซึ่งต้นทุนของการเดินทางของปัญหาที่ผู้ศึกษาทำการศึกษานั้นจะมีค่าที่เปลี่ยนแปลงไปตามช่วงของเวลา ซึ่งทำให้ปัญหาที่ผู้ศึกษาทำการศึกษานี้เป็นปัญหา TSP ที่ขึ้นอยู่กับเวลา

2.1.9 Vehicle Routing and Scheduling Problem

ปัญหา Vehicle Routing and Scheduling Problem นั้นเป็นปัญหาที่มีลักษณะคล้ายกับปัญหา Traveling Salesman Problem (TSP) โดยที่มีการกำหนดจุดเริ่มต้นของการเดินทางและทำการเดินทางไปยังกลุ่มของจุดต่างๆ ตามจำนวนที่กำหนด โดยจะทำการเดินทางไปยังจุดต่างๆ เพียงครั้งเดียว และเมื่อทำการเดินทางไปครบทุกจุดแล้วก็จะทำการเดินทางกลับมายังจุดเริ่มต้น โดยที่การเดินทางนั้นส่วนใหญ่จะเป็นการเดินทางของรถที่ทำการบรรทุกสินค้า โดยที่ปัญหา Routing and Scheduling Problem นั้นจะมีความแตกต่างกับปัญหา TSP ตรงที่จำนวนรถของการเดินทางนั้นจะมีมากกว่า 1 คัน และความสามารถในการบรรทุกสินค้าหรือปริมาณของการบรรทุกสินค้าของรถแต่ละคันนั้นจะมีอยู่อย่างจำกัด

โดยมีผู้ที่ศึกษาปัญหา Vehicle Routing and Scheduling Problem [24] โดยที่เป้าหมายของการศึกษาปัญหานี้นั้นก็เพื่อที่จะศึกษาหาต้นทุนที่ต่ำที่สุดที่ใช้ในการขนส่ง โดยที่ข้อมูลของปัญหาที่ผู้ศึกษาได้ทำการศึกษานั้น ผู้ศึกษาได้ทำการเก็บข้อมูลจริงจากผู้ผลิตโดยตรงจึงทำให้ปัญหานั้นจะมีเรื่องของข้อจำกัดกรอบเวลากำหนดเพิ่มขึ้นมาด้วยเนื่องจากการส่งสินค้าก็จะมีการกำหนดช่วงเวลาของการขนส่ง โดยที่ผู้ศึกษานั้นได้ใช้เทคนิค Optimization-based algorithm มาเพื่อใช้ในการแก้ปัญหา Vehicle Routing and Scheduling Problem นี้

2.1.10 Vehicle Routing and Scheduling Problem with Time Windows

ปัญหา Vehicle Routing and Scheduling Problem with Time Window Constraints นั้นเป็นปัญหาที่มีลักษณะคล้ายกับปัญหา Vehicle Routing and Scheduling Problem โดยที่เป็นปัญหาเกี่ยวกับเรื่องของการจัดเส้นทางของการขนส่ง โดยการขนส่งนั้นจะออกไปขนส่งยังจุดต่างๆ ตามที่กำหนด และเดินทางไปแต่ละจุดเพียงครั้งเดียว เมื่อเดินทางครบก็จะทำการเดินทางกลับมายังจุดเริ่มต้น โดยที่จำนวนรถของการเดินทางนั้นจะมีมากกว่า 1 คัน และปริมาณของการบรรทุกสินค้าของรถแต่ละคันนั้นจะมีอยู่อย่างจำกัด โดยปัญหา Vehicle Routing and

Scheduling Problem with Time Window Constraints นั้นจะมีการเพิ่มข้อจำกัดกรอบเวลากำหนด (Time Windows) ซึ่งในการขนส่งที่เกิดขึ้นจริงในชีวิตประจำวันนั้นก็จะมีเรื่องของข้อจำกัดกรอบเวลากำหนดเข้ามาเกี่ยวข้องด้วย

โดยมีผู้ที่ศึกษาปัญหา Vehicle Routing and Scheduling Problem with Time Window Constraints [25] โดยผู้ที่ศึกษานั้นได้ทำการใช้เทคนิควิธี Heuristic หลายๆ วิธีเพื่อที่จะใช้ในการแก้ปัญหา และนำผลของวิธี Heuristic เหล่านี้มาเปรียบเทียบกันว่าวิธีไหนสามารถที่จะหาคำตอบได้ดีกว่ากัน โดยที่เป้าหมายของการแก้ปัญหานี้ก็เพื่อต้องการที่จะหาต้นทุนของการขนส่งที่ต่ำที่สุด

2.1.11 Vehicle Routing and Scheduling Problem with Precedence Constraints

ปัญหา Vehicle Routing and Scheduling Problem with Precedence Constraints นั้นเป็นปัญหาที่มีลักษณะคล้ายกับปัญหา Vehicle Routing and Scheduling Problem โดยที่เป็นปัญหาเกี่ยวกับเรื่องของการจัดเส้นทางของการขนส่ง โดยการขนส่งนั้นจะออกไปขนส่งยังจุดต่างๆ ตามที่กำหนด และเดินทางไปแต่ละจุดเพียงครั้งเดียว เมื่อเดินทางครบก็จะทำการเดินทางกลับมายังจุดเริ่มต้น โดยที่จำนวนรถของการเดินทางนั้นจะมีมากกว่า 1 คัน และปริมาณของการบรรทุกสินค้าของรถแต่ละคันนั้นจะมีอยู่อย่างจำกัด โดยปัญหา Vehicle Routing and Scheduling Problem with Precedence Constraints นั้นจะมีการเพิ่มข้อจำกัดของการลำดับ (Precedence Constraints) เข้ามาด้วย ซึ่งการขนส่งในปัจจุบันก็มักจะมีการทำการกำหนดการขนส่งก่อนหลัง

โดยมีผู้ที่ศึกษาปัญหา Vehicle Routing and Scheduling Problem with Precedence Constraints [26] โดยผู้ที่ทำการศึกษานั้นได้ทำการนำเสนอวิธี branch and bound มาเพื่อใช้ในการแก้ปัญหา โดยที่เป้าหมายของการแก้ปัญหานี้ก็เพื่อต้องการที่จะหาต้นทุนของการขนส่งที่ต่ำที่สุด

2.1.12 Traveling Salesman Problem with Pickup and Delivery

ปัญหา Traveling Salesman Problem with Pickup and Delivery นั้นเป็นปัญหาที่มีลักษณะคล้ายกับปัญหา Traveling Salesman Problem (TSP) โดยที่มีการกำหนดจุดเริ่มต้นของการเดินทางและทำการเดินทางไปยังกลุ่มของจุดต่างๆ ตามจำนวนที่กำหนด และเมื่อทำการเดินทางไปครบทุกจุดแล้วก็จะทำการเดินทางกลับมายังจุดเริ่มต้น โดยจะมีการทำการเพิ่มกระบวนการรับของ (Pick up) และกระบวนการส่งของ (Delivery) เข้าไปในการเดินทางด้วย เนื่องจากว่ามีการกำหนดว่าจุดในแต่ละจุดที่จะทำการเดินทางไปนั้นบางจุดก็จะต้องเป็นกระบวนการ

รับของ ส่วนบางจุดก็อาจจะเป็นกระบวนการส่งของ ซึ่งเงื่อนไขเหล่านี้นั้นจะทำให้ปัญหามีความซับซ้อนเพิ่มมากขึ้น

โดยมีผู้ที่ศึกษาปัญหา Traveling Salesman Problem with Pickup and Delivery [27] โดยผู้ศึกษานั้นได้มีการพิจารณาข้อจำกัดในเรื่องของการลำดับ (Precedence Constraints) เข้าไปในปัญหาด้วย และผู้ศึกษานั้นได้ทำการนำเสนอวิธี Heuristic ที่พัฒนาขึ้นมาเพื่อที่จะนำมาใช้ในการแก้ปัญหา

2.2 Local Search

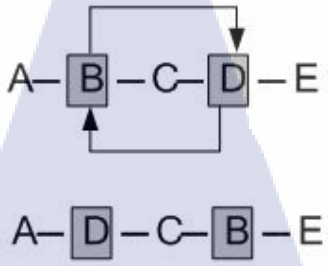
Local Search หรือ การค้นหาเฉพาะที่ เป็นกลวิธีที่ใช้ในการหาคำตอบแบบหนึ่ง จากเซตของผลเฉลยที่มีขนาดใหญ่มากๆ ซึ่งเป็นไปไม่ได้ที่จะตรวจสอบผลเฉลยทั้งหมดเพื่อให้เจอผลเฉลยที่ดีที่สุด โดยจะมีฟังก์ชันต้นทุนสำหรับพิจารณาแต่ละผลเฉลย เพื่อเปลี่ยนแปลงสถานะของผลเฉลยปัจจุบันที่ค้นหา ไปจนกว่าจะเจอผลเฉลยที่เราพอใจ โดยวิธีนี้เป็นวิธีหนึ่งในวิธีการแก้ปัญหาแบบ Metaheuristic ซึ่งวิธี Local Search จะเป็นวิธีที่มีการทำงานซ้ำไปซ้ำมา โดยจะมีการย้ายจากผลเฉลยอันหนึ่ง (S) ไปยังผลเฉลยอีกอันหนึ่ง (S') โดยการย้ายนั้นจะทำตามลักษณะของ Neighborhood Structure ซึ่งขั้นตอนวิธีการค้นหาเฉพาะที่นี้ยังเป็นกลวิธีในการแก้ปัญหาการหาค่าที่เหมาะสมที่สุดเชิงจัดการ (Combinatorial Optimization Problem) ซึ่งเป็นปัญหาเอ็นพีแบบยาก (NP-hard) อีกทั้งยังถูกนำไปประยุกต์ใช้ในหลากหลายวงการ ซึ่งสามารถหาผลเฉลยที่ดีได้สำหรับปัญหาที่พบในทางปฏิบัติ

โดยขั้นตอนหลักๆ ของวิธี Local Search [28] สามารถแสดงได้ดังนี้

ขั้นตอนที่ 1 Initialisation : เป็นขั้นตอนของการเริ่มสร้างผลเฉลย S เพื่อที่จะให้เป็นผลเฉลยเริ่มต้น และก็ทำการคำนวณค่าของ Objective Function F(S) ซึ่งก็คือการคำนวณค่าของต้นทุนที่ใช้ในการเดินทาง ซึ่งการสร้างผลเฉลยเริ่มต้นนั้นจะสามารถสร้างโดยการทำการ random ก็ได้

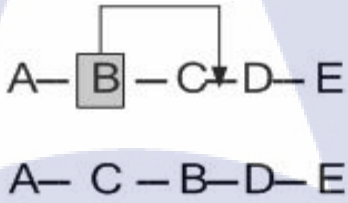
ขั้นตอนที่ 2 Neighbor Generation : เลือกผลเฉลย S' ที่อยู่ใกล้เคียงกับผลเฉลยเริ่มต้น S และทำการคำนวณค่า F(S') โดยการที่จะสร้าง Neighbor S' นั้น จะต้องทำการระบุ Neighborhood Structure ก่อน โดย Neighborhood Structure ที่ได้มีการนำมาใช้บ่อยๆ สามารถแสดงได้ดังนี้

Swap Neighborhood เป็นวิธีการหนึ่งในการปรับปรุงคำตอบเฉพาะที่ซึ่งเป็นการสับเปลี่ยนตำแหน่ง 2 ตำแหน่ง โดยมีหลักการการทำงานดังรูปที่ 2.2



รูปที่ 2.2 หลักการทำงานของ Swap

Insert Neighborhood เป็นการนำสิ่งที่ต้องการนำไปแทรกออกจากตำแหน่งเดิม และนำไปแทรกในตำแหน่งอื่นๆ โดยมีหลักการการทำงานดังรูปที่ 2.3



รูปที่ 2.3 หลักการทำงานของ Insert

ขั้นตอนที่ 3 Acceptance Test : เป็นการทดสอบเพื่อที่จะทำการยอมรับการย้ายจาก S ไปยัง S' หรือไม่ ถ้าการย้ายนั้นถูกยอมรับ S' จะไปแทนที่ S เพื่อเป็นผลเฉลยปัจจุบันแทน แต่ถ้าการย้ายนั้นไม่ถูกยอมรับ S ก็จะยังคงเป็นคำตอบปัจจุบันอยู่ โดยการที่จะยอมรับหรือไม่ยอมรับนั้นส่วนใหญ่แล้วจะทำการดูที่ค่าของ F(S) และ F(S') โดยจะทำการยอมรับเมื่อมีการพัฒนาเกิดขึ้น F(S') < F(S)

ขั้นตอนที่ 4 Termination Test : เป็นการทดสอบว่าวิธีการควรที่จะหยุดหรือยัง ถ้าวิธีการต้องหยุดผลเฉลยที่ได้ออกมาจะเป็นผลเฉลยที่ดีที่สุด แต่ถ้าเป็นอย่างอื่นก็กลับไปยังขั้นตอน Neighbor Generation

โดยมีนักวิจัยได้นำวิธี Local Search มาใช้แก้ปัญหา Traveling Salesman Problem (TSP) หรือ Local Search for Travelling Salesman Problem [29] โดยผู้เขียนได้กล่าวว่า วิธี Metaheuristic อย่างเช่นวิธี Local Search นั้นสามารถที่จะให้ผลลัพธ์ที่ดีสำหรับลักษณะปัญหาแบบ NP-Hard Combinatorial Optimization โดยหนึ่งในปัญหาแบบ NP-hard Combinatorial Optimization

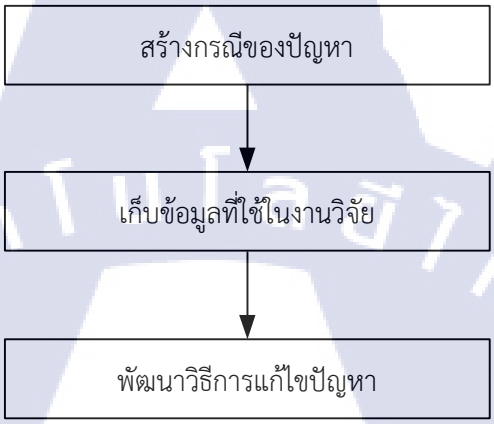
ที่มีชื่อเสียงนั้นก็คือ Travelling Salesman Problem โดยได้มีการนำวิธี Local Search มาใช้แก้ปัญหา Travelling Salesman Problem กันอย่างแพร่หลาย โดย Local Search จะมีวิธีการพัฒนาผลเฉลยเดิมอย่างซ้ำไปซ้ำมา โดยจะมีการทำการค้นหาผลเฉลยที่ดีกว่าในพื้นที่ใกล้เคียง และวิธีนี้จะหยุดทำงานเมื่อไม่สามารถหาคำตอบที่ดีขึ้นได้จากบริเวณพื้นที่ที่ใกล้เคียง หรือมีการค้นหาซ้ำไปซ้ำมาจนครบจำนวนรอบที่กำหนดเอาไว้

และยังมีผู้ที่ศึกษาวิธี Local Search [30] เพื่อนำมาใช้ในการแก้ปัญหา Asymmetric Traveling Salesman Problem (ATSP) โดยที่ปัญหา ATSP นั้นจะมีความแตกต่างกับปัญหา TSP ตรงที่ระยะทางหรือเวลาที่ใช้ในการเดินทางไปและกลับระหว่างสองจุดนั้นจะมีค่าที่ไม่เท่ากัน ซึ่งในส่วนนี้จะทำให้ปัญหา ATSP นั้นจะมีความซับซ้อนและมีความยากมากกว่าปัญหา TSP แบบดั้งเดิม

และมีผู้ที่ศึกษาปัญหา TSP โดยผู้ที่ศึกษานั้นได้ทำการใช้เทคนิค dynamic programming ซึ่งใช้พื้นฐานจากวิธี Local Search [31] มาเพื่อใช้ในการแก้ปัญหา TSP โดยปัญหา TSP ที่ผู้ศึกษาทำการศึกษานั้นมีการทำการตัดสินใจระหว่าง Pick up Route กับ Delivery Route และโดยในส่วนของ Local Search นั้นผู้ศึกษาได้มีการใช้ Swap Neighborhood มาเพื่อใช้ในการแก้ปัญหา

บทที่ 3
วิธีการดำเนินงาน

การดำเนินงานวิจัยนั้นจะทำการดำเนินงานโดยมีขั้นตอนต่างๆ ดังรูปที่ 3.1



รูปที่ 3.1 ขั้นตอนการดำเนินงาน

จากการที่ได้ทำการศึกษาทฤษฎีและงานวิจัยที่ข้องกับปัญหาแล้ว โดยปัญหาที่การศึกษานั้นเป็นรูปแบบหนึ่งของปัญหา Travelling Salesman Problem (TSP) ซึ่งมักจะนิยมนำมาประยุกต์ใช้กับการเดินทางทางบก แต่ในงานวิจัยเล่มนี้จะนำปัญหา TSP นี้มาประยุกต์ใช้กับการเดินทางทางอากาศ ซึ่งเป็นการเดินทางออกจากประเทศไทยไปยังประเทศที่กำหนดรวม 20 ประเทศ และทำการเดินทางทั้งหมดภายใน 20 สัปดาห์ โดยเดินทางไปยังประเทศประเทศเดียวต่อหนึ่งสัปดาห์ และเมื่อเดินทางครบทุกประเทศแล้วก็จะทำการเดินทางกลับมายังประเทศไทย

โดยการเดินทางทางอากาศนั้นจะเป็นการเดินทางด้วยเครื่องบิน ซึ่งการเดินทางโดยเครื่องบินนั้นค่าตั๋วเครื่องบินก็มักจะเปลี่ยนแปลงไปตามช่วงเวลา ทำให้ปัญหานี้เป็นปัญหาการเดินทางที่ขึ้นอยู่กับช่วงเวลา คือต้องเลือกเดินทางไปยังประเทศใดๆ ในช่วงเวลาใดๆ เพื่อจะให้มีค่าใช้จ่ายต่ำที่สุด และคุณสมบัติของราคาตั๋วเครื่องบินนั้นการเดินทางไปและกลับระหว่างสองประเทศใดๆ ต้นทุนที่ใช้ในการเดินทางขาไป และต้นทุนที่ใช้ในการเดินทางขากลับก็มักจะมีค่าที่ไม่เท่ากัน โดยคุณสมบัติข้อนี้จะทำให้ปัญหานี้เป็นปัญหา Asymmetric Travelling Salesman Problem (ATSP) และได้มีการเพิ่มข้อจำกัดในเรื่องของกรอบเวลากำหนด (Time Window Constraints) ซึ่งเป็นข้อจำกัดที่กำหนดว่าต้องเดินทางไปยังประเทศนี้ในสัปดาห์นี้ และข้อจำกัดในเรื่องของการลำดับ (Precedence Constraints)

ซึ่งเป็นข้อจำกัดที่เป็นตัวกำหนดว่าต้องเดินทางไปยังประเทศไหนก่อนแล้วจึงจะเดินทางไปยังประเทศไหนเป็นประเทศต่อไป หรือเป็นการกำหนดว่าต้องเดินทางไปยังประเทศนี้เป็นลำดับก่อนประเทศนี้ และวัตถุประสงค์ของปัญหานี้คือต้องการหาต้นทุนที่ต่ำที่สุดที่ใช้ในการเดินทางไปยังประเทศต่างๆ ตามจำนวนประเทศที่กำหนด ซึ่งจากเงื่อนไขและคุณสมบัติต่างๆ ของปัญหาและข้อจำกัดต่างๆ ที่เพิ่มขึ้นมาของปัญหานั้น จะทำให้ปัญหานี้เป็นปัญหาที่ยังไม่มีผู้ที่สนใจนำมาทำการศึกษา และทำให้ปัญหานี้นั้นมีความใกล้เคียงกับการที่จะทำการเลือกเดินทางในปัจจุบัน จึงทำให้ปัญหานี้เป็นปัญหาที่น่าสนใจที่จะเลือกมาทำการศึกษา

จากการศึกษาทฤษฎีและงานวิจัยต่างๆ นั้น ผู้วิจัยได้ทำการเลือกวิธี Local Search Swap และ Local Search Insert มาเพื่อที่จะใช้ในการแก้ปัญหานี้ โดยวิธี Local Search นี้จะมีการทำงานที่ซ้ำไปซ้ำมา และสามารถที่จะหาผลเฉลยที่ดีให้กับปัญหาได้ จึงเหมาะสมที่จะนำมาใช้ในการแก้ปัญหา และผู้วิจัยนั้นยังได้นำวิธี Nearest Neighbor Algorithm มาเพื่อใช้ในการแก้ปัญหาด้วย และนำผลเฉลยของทั้งสองวิธีนี้นำมาทำการเปรียบเทียบกัน แต่ทั้งนี้ในวิธี Local Search และ วิธี Nearest Neighbor Algorithm จะถูกปรับปรุงวิธีการทำงานของแต่ละวิธีเพื่อที่จะให้เหมาะสมสำหรับที่จะนำมาใช้ในการแก้ปัญหาที่งานวิจัยนี้ได้ทำการศึกษา

3.1 กรณีปัญหาที่สร้างขึ้น

ปัญหาที่ทำการศึกษานั้น ได้ทำการศึกษาประเทศจำนวน 20 ประเทศ โดยข้อมูลของแต่ละประเทศสามารถแสดงได้ในตารางที่ 3.1

ตารางที่ 3.1 ข้อมูลของแต่ละประเทศ

หมายเลขประเทศ	ชื่อประเทศ	ชื่อเมืองที่มีชื่อเสียง	ชื่อสนามบิน
1	ไทย	กรุงเทพ	Suvarnabhumi
2	สิงค์โปร	สิงค์โปร	Changi
3	มาเลเซีย	กัวลาลัมเปอร์	Kuala Lumpur
4	ญี่ปุ่น	โตเกียว	Narita
5	เกาหลีใต้	โซล	Incheon
6	จีน	ปักกิ่ง	Beijing Capital
7	ฮ่องกง	ฮ่องกง	Hong Kong
8	ออสเตรเลีย	ซิดนีย์	Sydney
9	สหรัฐอเมริกา	นิวยอร์ก	JFK

ตารางที่ 3.1 ข้อมูลของแต่ละประเทศ (ต่อ)

หมายเลขประเทศ	ชื่อประเทศ	ชื่อเมืองที่มีชื่อเสียง	ชื่อสนามบิน
10	สหราชอาณาจักร	ลอนดอน	London Heathrow
11	ฝรั่งเศส	ปารีส	Charles de Gaulle
12	อิตาลี	โรม	Leonardo de Vinci
13	เยอรมัน	แฟรงเฟิร์ต	Frankfurt
14	สเปน	มาดริด	Madrid Barajas
15	นิวซีแลนด์	โอ๊คแลนด์	Auckland
16	ไอร์แลนด์	ดับลิน	Dublin
17	แคนาดา	โทรอนโต	Toronto Pearson
18	กรีซ	เอเธนส์	Athens
19	ตุรกี	อิสตันบูล	Istanbul Ataturk
20	สวิตเซอร์แลนด์	ซูริค	Zurich

โดยที่ปัญหานี้จะทำการศึกษาปัญหาทั้งหมดเป็น 6 กรณี โดยข้อมูลที่ใช้ร่วมกันทั้ง 6 กรณี สามารถแสดงได้ดังนี้

1. ทำการเริ่มต้นออกเดินทางจากประเทศไทยและเมื่อเดินทางครบทุกประเทศแล้วก็จะเดินทางกลับมายังประเทศไทย

2. จะทำการเดินทางไปยังหนึ่งประเทศต่อหนึ่งสัปดาห์เท่านั้น

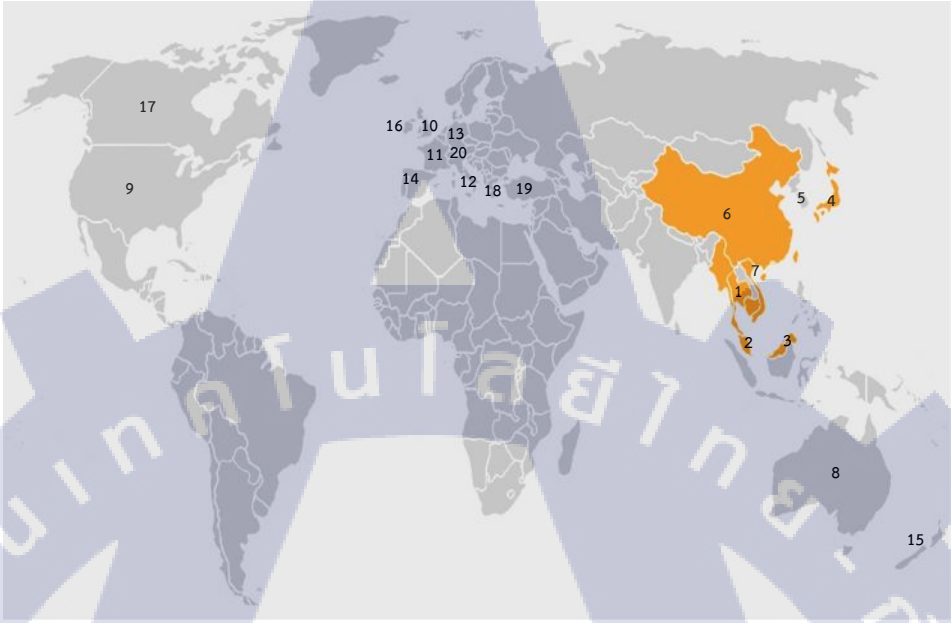
3. กำหนดให้ c_{ijk} นั้นเป็นต้นทุนค่าตั๋วเครื่องบินที่ราคาต่ำที่สุดที่ใช้ในการเดินทางจากประเทศ i ไปยังประเทศ j ในสัปดาห์ที่ k (โดยที่ i และ j และ k = 1,2,...,N และโดยที่ $i \neq j$)

โดยที่ข้อมูลของวันที่ที่ทำการเก็บข้อมูลของราคาตั๋วเครื่องบินที่ทำการเดินทางออกจากประเทศต่างๆ ภายในสัปดาห์ที่ 1 ไปจนถึงสัปดาห์ที่ 20 นั้นสามารถแสดงได้ดังตารางที่ 3.2

ตารางที่ 3.2 ข้อมูลของวันที่ที่ทำการเก็บราคาตั๋วเครื่องบิน

สัปดาห์ที่	วันที่
1	7 มิถุนายน 2558
2	14 มิถุนายน 2558
3	21 มิถุนายน 2558
4	28 มิถุนายน 2558
5	5 กรกฎาคม 2558
6	12 กรกฎาคม 2558
7	19 กรกฎาคม 2558
8	26 กรกฎาคม 2558
9	2 สิงหาคม 2558
10	9 สิงหาคม 2558
11	16 สิงหาคม 2558
12	23 สิงหาคม 2558
13	30 สิงหาคม 2558
14	6 กันยายน 2558
15	13 กันยายน 2558
16	20 กันยายน 2558
17	27 กันยายน 2558
18	4 ตุลาคม 2558
19	11 ตุลาคม 2558
20	18 ตุลาคม 2558

โดยแผนที่แสดงตำแหน่งของแต่ละประเทศสามารถแสดงได้ดังรูปที่ 3.2



รูปที่ 3.2 แผนที่แสดงตำแน่งของแต่ละประเทศ

โดยที่จากปัญหาที่ทำการศึกษาทั้งหมด 6 กรณีนั้นจะถูกทำการแบ่งออกให้เป็นปัญหาทั้งหมด 2 ชุด

โดยปัญหาชุดที่ 1 นั้นจะประกอบไปด้วย ปัญหากรณีที่ 1 ปัญหากรณีที่ 2 และปัญหากรณีที่ 3 โดยปัญหาในชุดที่ 1 นี้จะทำการพิจารณาประเทศทั้งหมด 15 ประเทศ ประกอบด้วยประเทศหมายเลข 1 ไปจนถึงประเทศหมายเลข 15 และปัญหาชุดที่ 1 นี้ จะทำการพิจารณาข้อจำกัดในเรื่องของกรอบเวลากำหนด และข้อจำกัดของการลำดับดังนี้ โดยมี 1 ประเทศที่มีข้อจำกัดในเรื่องของกรอบเวลากำหนด คือ มี 1 ประเทศที่ถูกกำหนดให้ไปอยู่ในสัปดาห์นั้นๆ และมี 1 คู่ของประเทศที่มีข้อจำกัดของการลำดับ คือมีประเทศหนึ่งที่ต้องทำการเดินทางไปก่อนก่อนที่จะทำการเดินทางไปยังอีกประเทศหนึ่งรวม 2 ประเทศแล้วเป็นจำนวน 1 คู่

โดยปัญหาชุดที่ 2 นั้นจะประกอบไปด้วย ปัญหากรณีที่ 4 ปัญหากรณีที่ 5 และปัญหากรณีที่ 6 โดยปัญหาในชุดที่ 2 นี้จะทำการพิจารณาประเทศทั้งหมด 20 ประเทศ ประกอบด้วย ประเทศหมายเลข 1 ไปจนถึงประเทศหมายเลข 20 และปัญหาชุดที่ 2 นี้ จะทำการพิจารณาข้อจำกัดในเรื่องของกรอบเวลากำหนด และข้อจำกัดของการลำดับดังนี้ โดยมี 2 ประเทศที่มีข้อจำกัดในเรื่องของกรอบเวลากำหนด คือมี 2 ประเทศที่ถูกกำหนดให้ไปอยู่ใน 2 สัปดาห์นั้นๆ และมี 2 คู่ของประเทศที่มีข้อจำกัด

ของการลำดับ คือมีประเทศหนึ่งที่ต้องทำการเดินทางไปก่อนก่อนที่จะที่จะทำการเดินทางไปยังอีกประเทศหนึ่งรวม 2 ประเทศเป็น 1 คู่ และมีอีกประเทศหนึ่งที่ต้องทำการเดินทางไปก่อนก่อนที่จะทำการเดินทางไปยังอีกประเทศหนึ่งอีก 2 ประเทศเป็น 1 คู่ รวมทั้งหมดเป็น 2 คู่

โดยเงื่อนไขและข้อจำกัดต่างๆ ของปัญหาชุดที่ 1 และปัญหาชุดที่ 2 นั้น สามารถสรุปได้ดังตารางที่ 3.3

ตารางที่ 3.3 เงื่อนไขและข้อจำกัดของปัญหาชุดที่ 1 และชุดที่ 2

ปัญหาชุดที่ 1	**ปัญหาชุดที่ 2**
ปัญหากรณีที่ 1 ถึง ปัญหากรณีที่ 3	ปัญหากรณีที่ 4 ถึง ปัญหากรณีที่ 6
พิจารณา 15 ประเทศ	พิจารณา 20 ประเทศ
มี 1 ประเทศที่มีข้อจำกัดกรอบเวลากำหนด	มี 2 ประเทศที่มีข้อจำกัดกรอบเวลากำหนด
มี 1 คู่ของประเทศที่มีข้อจำกัดการลำดับ	มี 2 คู่ของประเทศที่มีข้อจำกัดการลำดับ

โดยข้อมูลและรายละเอียดของปัญหากรณีที่ 1 ปัญหากรณีที่ 2 และปัญหากรณีที่ 3 อธิบายได้ดังนี้ โดยปัญหากรณีที่ 1 นั้นจะต้องไปอยู่ในประเทศสิงค์โปรในสัปดาห์ที่ 2 และจะต้องทำการเดินทางไปยังประเทศอิตาลีก่อนที่จะเดินทางไปยังประเทศฝรั่งเศส ปัญหากรณีที่ 2 นั้น จะต้องไปอยู่ในประเทศเกาหลีใต้ในสัปดาห์ที่ 6 และจะต้องทำการเดินทางไปยังประเทศญี่ปุ่นก่อนที่จะเดินทางไปยังประเทศมาเลเซีย ปัญหากรณีที่ 3 นั้น จะต้องไปอยู่ในประเทศญี่ปุ่นในสัปดาห์ที่ 9 และจะต้องทำการเดินทางไปยังประเทศเยอรมันก่อนที่จะเดินทางไปยังประเทศจีน

โดยข้อมูลและรายละเอียดของปัญหากรณีที่ 4 ปัญหากรณีที่ 5 และปัญหากรณีที่ 6 อธิบายได้ดังนี้ โดยปัญหากรณีที่ 4 นั้น จะต้องไปอยู่ในประเทศสหราชอาณาจักรในสัปดาห์ที่ 6 และต้องไปอยู่ในประเทศมาเลเซียในสัปดาห์ที่ 9 และจะต้องทำการเดินทางไปยังประเทศญี่ปุ่นก่อนที่จะเดินทางไปยังประเทศออสเตรเลีย และจะต้องทำการเดินทางไปยังประเทศฝรั่งเศสก่อนที่จะเดินทางไปยังประเทศฮ่องกง ปัญหากรณีที่ 5 นั้น จะต้องไปอยู่ในประเทศเยอรมันในสัปดาห์ที่ 6 และต้องไปอยู่ในประเทศออสเตรเลียในสัปดาห์ที่ 11 และจะต้องทำการเดินทางไปยังประเทศจีนก่อนที่จะเดินทางไปยังประเทศสหรัฐอเมริกา และจะต้องทำการเดินทางไปยังประเทศเกาหลีใต้ก่อนที่จะเดินทางไปยังประเทศมาเลเซีย ปัญหากรณีที่ 6 นั้น จะต้องไปอยู่ในประเทศฝรั่งเศสในสัปดาห์ที่ 8 และต้องไปอยู่ในประเทศเกาหลีใต้ในสัปดาห์ที่ 13 และจะต้องทำการเดินทางไปยังประเทศออสเตรเลียก่อนที่จะเดินทางไปยังประเทศนิวซีแลนด์ และจะต้องทำการเดินทางไปยังประเทศฮ่องกงก่อนที่จะเดินทางไปยังประเทศสหราชอาณาจักร

โดยข้อมูลและรายละเอียดของปัญหาชุดที่ 1 นั้น สามารถสรุปได้ดังตารางที่ 3.4

ตารางที่ 3.4 ข้อมูลและรายละเอียดของปัญหาชุดที่ 1

ปัญหากรณีที่	ข้อจำกัดกรอบเวลากำหนด	ข้อจำกัดการลำดับ
1	อยู่สิงค์โปรในสัปดาห์ที่ 2	ไปอิตาลีก่อนฝรั่งเศส
2	อยู่เกาหลีใต้ในสัปดาห์ที่ 6	ไปญี่ปุ่นก่อนมาเลเซีย
3	อยู่ญี่ปุ่นในสัปดาห์ที่ 9	ไปเยอรมันก่อนจีน

โดยข้อมูลและรายละเอียดของปัญหาชุดที่ 2 นั้น สามารถแสดงได้ดังตารางที่ 3.5

ตารางที่ 3.5 ข้อมูลและรายละเอียดของปัญหาชุดที่ 2

ปัญหากรณีที่	ข้อจำกัดกรอบเวลากำหนด	ข้อจำกัดการลำดับ
4	อยู่สหราชอาณาจักรในสัปดาห์ที่ 6	ไปญี่ปุ่นก่อนออสเตรเลีย
	อยู่มาเลเซียในสัปดาห์ที่ 9	ไปฝรั่งเศสก่อนฮ่องกง
5	อยู่เยอรมันในสัปดาห์ที่ 6	ไปจีนก่อนสหรัฐอเมริกา
	อยู่ออสเตรเลียในสัปดาห์ที่ 11	ไปเกาหลีใต้ก่อนมาเลเซีย
6	อยู่ฝรั่งเศสในสัปดาห์ที่ 8	ไปออสเตรเลียก่อนนิวซีแลนด์
	อยู่เกาหลีใต้ในสัปดาห์ที่ 13	ไปฮ่องกงก่อนสหราชอาณาจักร

3.2 เก็บข้อมูลที่ใช้ในการวิจัย

Skyscanner เป็นเว็บไซต์ค้นหาข้อมูลการเดินทางชั้นนำระดับระดับโลกที่ให้ข้อมูลและเปรียบเทียบตั๋วเครื่องบินนับล้านจากสายการบินจำนวนกว่าพันแห่งแบบออนไลน์ได้ทันที

ทางเว็บไซต์ Skyscanner จะมีตัวเลือกสำหรับค้นหาข้อมูลต่างๆ ที่สามารถปรับเปลี่ยนได้ จึงทำให้สามารถเรียกดูราคาแบบตลอดทั้งเดือนหรือแม้แต่ตลอดปีได้ นอกจากนี้การจองตั๋วเครื่องบินกับสายการบินหรือตัวแทนจำหน่ายทางเว็บไซต์จะไม่คิดค่าธรรมเนียมหรือค่าใช้จ่ายใดๆ

โดยทาง Skyscanner ได้ดำเนินธุรกิจเกี่ยวกับการเดินทางมาเป็นเวลานานกว่า 10 ปี และมีพนักงานกว่า 30 เชื้อชาติในสำนักงาน โดยทุกเดือนจะมีผู้ที่มาเยี่ยมชมเว็บไซต์มากกว่า 60 ล้านคน ซึ่งมาทำการใช้บริการเพื่อค้นหาเที่ยวบินในภาษาต่างๆ มากกว่า 30 ภาษา

โดยหน้าเว็บไซต์ของ Skyscanner สามารถแสดงได้ดังรูปที่ 3.3

รูปที่ 3.3 เว็บไซต์ Skyscanner

โดยข้อมูลต้นทุนนั้นจะเป็นข้อมูลของราคาตั๋วเครื่องบินที่ใช้ในการเดินทางใน 20 ประเทศภายใน 20 สัปดาห์ โดยจะทำการเก็บข้อมูลของราคาตั๋วเครื่องบินในหน่วยเงินบาทไทย เก็บข้อมูลเฉพาะของการบินตรงไปยังประเทศต่างๆ เท่านั้น โดยข้อมูลของราคาตั๋วเครื่องบินนั้นก็จะถูกเสนอโดยหลายๆ สายการบินที่ทำการให้บริการระหว่างประเทศนั้นๆ โดยที่ในงานวิจัยนี้จะทำการเก็บข้อมูลของราคาตั๋วเครื่องบินที่มีราคาต่ำที่สุดที่ใช้ในการเดินทางระหว่างประเทศนั้นๆ และก็มีการทำการเก็บข้อมูลของรายชื่อสายการบินที่ได้เสนอราคาตั๋วที่มีราคาต่ำที่สุดด้วย และโดยข้อมูลต้นทุนเหล่านี้จะถูกเก็บโดยการเดินทางไปยังเมืองที่มีชื่อเสียงต่างๆ ของในแต่ละประเทศ

โดยตัวอย่างของข้อมูลต้นทุนนั้นสามารถแสดงได้ดังรูปที่ 3.4

From	To	Week	Cost (baht)	Airline
Thailand (1) (Suvarnabhumi International Airport)	Singapore (2) (Changi Airport)	1	2616	Tigerair
		2	2301	Tigerair
		3	2190	Tigerair
		4	2190	Tigerair
		5	1990	Tigerair
		6	1990	Tigerair
		7	3182	Jetstar
		8	1990	Tigerair
		9	1990	Tigerair
		10	2082	Jetstar
		11	2430	Tigerair
		12	1990	Tigerair
		13	1990	Tigerair
		14	1990	Tigerair
		15	1990	Tigerair
		16	1990	Tigerair
		17	1990	Tigerair
		18	1990	Tigerair
		19	1990	Tigerair
		20	1990	Tigerair
Thailand (1) (Suvarnabhumi International Airport)	Malaysia (3) (Kuala Lumpur International Airport)	1	4390	Malaysia Airline
		2	4430	Malaysia Airline
		3	4390	Malaysia Airline
		4	4390	Malaysia Airline
		5	4390	Malaysia Airline
		6	4390	Malaysia Airline
		7	4430	Malaysia Airline
		8	4390	Malaysia Airline
		9	4390	Malaysia Airline
		10	4390	Malaysia Airline
		11	4390	Malaysia Airline
		12	4390	Malaysia Airline
		13	4390	Malaysia Airline
		14	4390	Malaysia Airline
		15	4390	Malaysia Airline
		16	4390	Malaysia Airline
		17	4390	Malaysia Airline
		18	4390	Malaysia Airline
		19	4390	Malaysia Airline
		20	4390	Malaysia Airline

รูปที่ 3.4 ตัวอย่างของข้อมูลต้นทุน

โดยที่ข้อมูลของต้นทุนทั้งหมดที่ใช้ในการทดลองนั้นสามารถ Download ได้จาก [32]

3.3 วิธีการแก้ไขปัญหาที่พัฒนาขึ้น

โดยในหัวข้อนี้จะทำการอธิบายความหมายของลำดับของประเทศว่ามีความหมายว่าอย่างไร และนำเสนอวิธีการที่จะนำมาใช้ในการแก้ปัญหา โดยลำดับประเทศจะยกตัวอย่างดังนี้ ถ้าลำดับประเทศ 1-4-2-5-3-1 จะสามารถแปลความหมายได้ว่า เดินทางออกจากประเทศหมายเลข 1 เพื่อเดินทางไปยังประเทศหมายเลข 4 ในสัปดาห์ที่ 1 แล้วเดินทางออกจากประเทศหมายเลข 4 เพื่อเดินทางไปยังประเทศหมายเลข 2 ในสัปดาห์ที่ 2 แล้วเดินทางออกจากประเทศหมายเลข 2 เพื่อเดินทางไปยังประเทศหมายเลข 5 ในสัปดาห์ที่ 3 แล้วเดินทางออกจากประเทศหมายเลข 5 เพื่อเดินทางไปยังประเทศหมายเลข 3 ในสัปดาห์ที่ 4 แล้วเดินทางออกจากประเทศหมายเลข 3 เพื่อเดินทางไปยังประเทศหมายเลข 1 ในสัปดาห์ที่ 5 และจากการที่ได้ศึกษาทฤษฎีและงานวิจัยที่

เกี่ยวข้อง ผู้วิจัยได้ทำการใช้วิธี Local Search Algorithm โดยมีการพัฒนาคำตอบด้วยการ Swap และการ Insert มาเพื่อที่จะใช้ในการแก้ปัญหานี้ โดยการ Swap และ Insert ที่ใช้ในงานวิจัยนี้มีลักษณะพิเศษซึ่งแตกต่างจากการ Swap และ Insert ทั่วไปตรงที่การสุ่มประเทศเพื่อที่จะนำมาใช้ในการ swap หรือ insert นั้นจะถูกทำการสุ่มโดยมีเงื่อนไขเพิ่มขึ้นมา ซึ่งต่างจากวิธีดั้งเดิมจะทำสุ่มโดยไม่มีเงื่อนไข โดยเงื่อนไขในการสุ่มที่เพิ่มขึ้นมานี้ก็เพื่อเป็นการลดโอกาสการที่จะสร้างผลเฉลยที่เป็นไปไม่ได้ หรือไม่สามารถที่จะใช้เป็นคำตอบได้ออกมา จึงเหมาะสมที่จะนำมาใช้ในการแก้ปัญหา นอกเหนือจากนี้งานวิจัยนี้ยังได้ทำการปรับปรุงวิธี Nearest Neighbor Algorithm มาเพื่อการแก้ปัญหา Traveling Salesman ในการเดินทางทางอากาศที่ถูกนำเสนอในงานวิจัยชิ้นนี้ด้วย

Nearest Neighbor Algorithm ที่ถูกปรับปรุงขึ้นมาสำหรับปัญหาการขนส่งทางอากาศนี้เรียกว่าวิธี Modified Nearest Neighbor (MNN) ซึ่งได้อธิบายไว้ในหัวข้อที่ 3.3.1 สำหรับ Local Search Algorithm ที่ใช้การ Swap ในการปรับปรุงคำตอบจะเรียกว่าวิธี LS-SWAP และได้อธิบายไว้ในหัวข้อที่ 3.3.2 ส่วน Local Search Algorithm ที่ใช้การ Insert ในการปรับปรุงคำตอบจะเรียกว่าวิธี LS-INSERT และได้อธิบายไว้ในหัวข้อที่ 3.3.3

3.3.1 วิธี Modified Nearest Neighbor (MNN)

วิธี Nearest Neighbor นั้น เป็นวิธี Heuristic ที่นิยมนำมาใช้ในการแก้ปัญหา Travelling Salesman Problem (TSP) โดยในงานวิจัยนี้จะนำเอาวิธี Nearest Neighbor นี้มาเพื่อใช้ในการเปรียบเทียบกับวิธี Local Search โดยในหัวข้อนี้จะนำเสนอวิธี Nearest Neighbor ที่ถูกปรับปรุงเพื่อที่จะนำมาใช้ในการแก้ปัญหานี้ โดยจะเรียกวิธีนี้ว่า Modified Nearest Neighbor หรือ MNN โดยขั้นตอนของวิธี MNN นี้จะสามารถแสดงได้ดังนี้

ขั้นตอนที่ 1 ทำการกำหนดประเทศเริ่มต้น แล้วให้ประเทศเริ่มต้นเป็นประเทศที่อยู่ ณ ตอนนี้ และให้ k = 1

ขั้นตอนที่ 2 ทำการเลือกประเทศที่จะทำการเดินทางเป็นประเทศต่อไป ตามขั้นตอนดังนี้

ขั้นตอนที่ 2.1 ตรวจสอบว่ามีประเทศไหนที่ต้องไปในสัปดาห์ที่ k เนื่องด้วยข้อจำกัดกรอบเวลากำหนดหรือไม่ ถ้ามี ให้กำหนดประเทศนั้นเป็นประเทศที่จะถูกเดินทางไปเป็นประเทศต่อไป และจากนั้นไปยังขั้นตอนที่ 3 หรือถ้าไม่มีให้ไปขั้นตอนที่ 2.2

ขั้นตอนที่ 2.2 ตรวจสอบว่ามีประเทศไหนที่จะต้องถูกเดินทางไปต่อจากประเทศที่อยู่ ณ ตอนนี้เนื่องด้วยข้อจำกัดของการลำดับหรือไม่ ถ้ามี ให้กำหนดประเทศนั้นเป็นประเทศที่จะถูกเดินทางไปเป็นประเทศต่อไป และจากนั้นไปยังขั้นตอนที่3 หรือถ้าไม่มีให้ไปขั้นตอนที่ 2.3

ขั้นตอนที่ 2.3 ทำการเลือกประเทศที่เดินทางเป็นประเทศต่อไปตามขั้นตอนที่ 2.3.1 ถึง 2.3.4

ขั้นตอนที่ 2.3.1 ให้ L เป็นรายการของประเทศที่มีความเป็นไปได้ที่จะสามารถถูกเดินทางไปในสัปดาห์ที่ k ได้ โดยจะสร้าง L โดยการเพิ่มประเทศที่ยังไม่ถูกเดินทางไปทั้งหมด ที่จะสามารถเดินทางมาจากประเทศที่อยู่ ณ ตอนนี้ได้

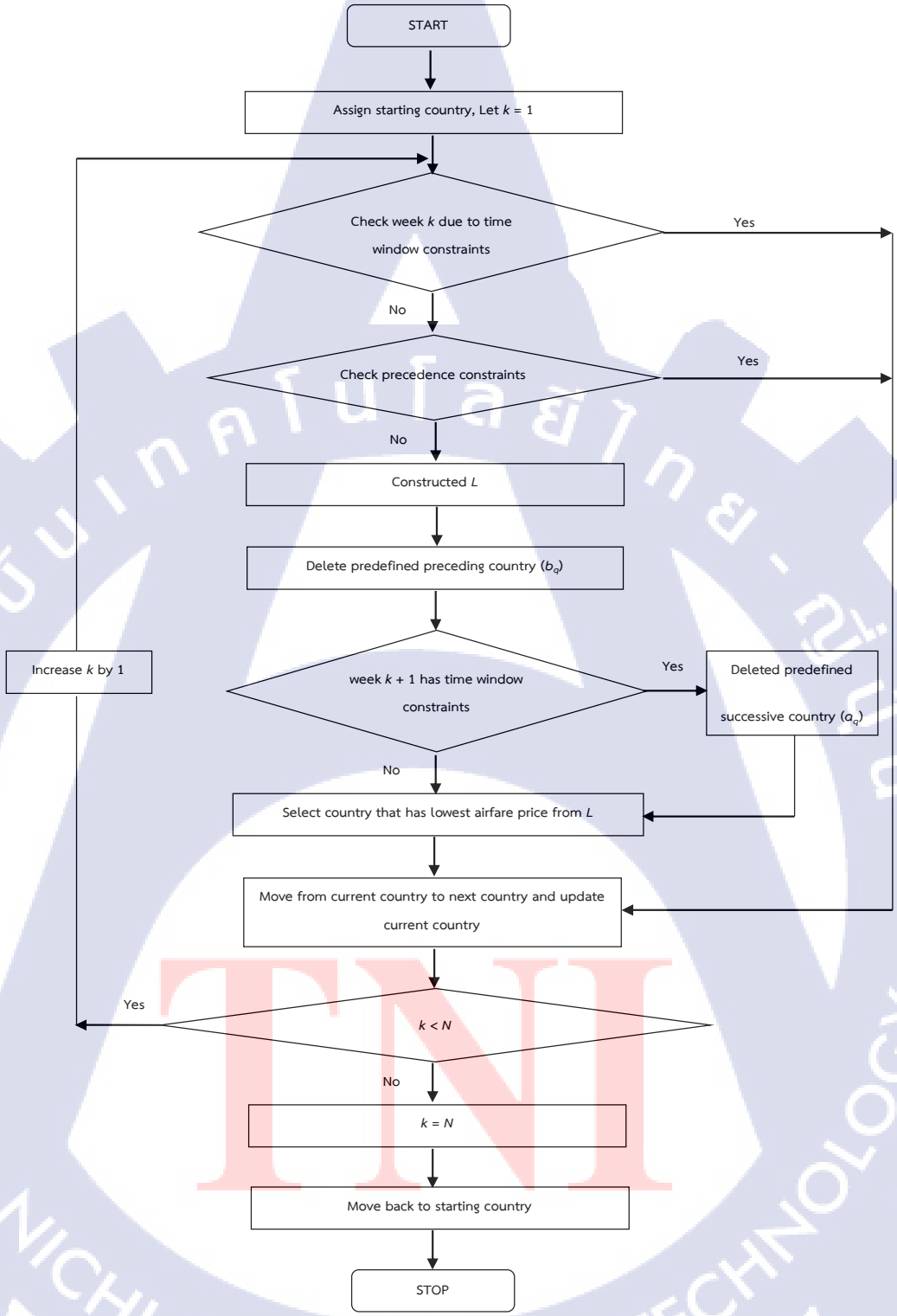
ขั้นตอนที่ 2.3.2 ทำการลบประเทศทุกประเทศที่มีการถูกกำหนดให้มีประเทศอื่นนำหน้าออกจากรายการ L

ขั้นตอนที่ 2.3.3 ถ้ามีประเทศไหนที่จะต้องถูกเดินทางไปในสัปดาห์ที่ k+1 เนื่องด้วยข้อจำกัดกรอบเวลากำหนด ให้ทำการลบประเทศทุกประเทศที่ไม่มีเที่ยวบินเดินทางมายังประเทศนี้ และลบทุกประเทศที่ถูกกำหนดให้มีประเทศต่อหลังออกจากรายการ L

ขั้นตอนที่ 2.3.4 เลือกประเทศที่มีราคาตั๋วเครื่องบินที่ใช้ในการเดิน ทางต่ำที่สุดจากรายการ L เป็นประเทศต่อไป จากนั้นไปยังขั้นตอนที่ 3

ขั้นตอนที่ 3 ทำการเดินทางออกจากประเทศที่อยู่ ณ ตอนนี้ เพื่อไปยังประเทศต่อไปที่ถูกเลือกจากขั้นตอนที่ 2 แล้วทำการปรับปรุงข้อมูลของประเทศที่อยู่ ณ ตอนนี้ให้เป็นประเทศใหม่ ถ้า k นั้นน้อยกว่า N ให้ทำการเพิ่ม k ทีละ 1 และจากนั้นก็ไปทำซ้ำขั้นตอนที่ 2 แต่ถ้า k เท่ากับ N ให้เดินทางกลับไปยังประเทศจุดเริ่มต้น และการจัดลำดับประเทศของการเดินทางก็จะสมบูรณ์

โดยแผนผังการทำงานของวิธี MNN สามารถแสดงได้ดังรูปที่ 3.5



รูปที่ 3.5 แผนผังการทำงานของวิธี MNN

3.3.2 Time Window Constraints

ในข้อนี้จะทำการกำหนดตัวแปรขึ้นมาเพื่อที่จะให้ง่ายต่อการเข้าใจและอธิบายความหมายของข้อจำกัดในเรื่องของกรอบเวลากำหนด โดยให้ w_k แทนประเทศที่ต้องไปอยู่ในสัปดาห์ที่ k (โดยที่ k=1,2,...,N-1 และ $w_k \in \{1,...,N\}$ โดย N แทนจำนวนประเทศทั้งหมด) โดยที่ w_N นั้นไม่ได้รวมอยู่ด้วยเนื่องจากสัปดาห์ที่ N นั้นถูกกำหนดไว้ล่วงหน้าแล้วว่าต้องไปอยู่ในประเทศจุดเริ่มต้น

โดยยกตัวอย่างเช่น $w_1 = 3$ จะหมายความว่าต้องไปอยู่ในประเทศหมายเลข 3 ในสัปดาห์ที่ 1 และอีกตัวอย่างหนึ่งเช่น $w_3 = 4$ จะหมายความว่าต้องไปอยู่ในประเทศหมายเลข 4 ในสัปดาห์ที่ 3

3.3.3 Precedence Constraints

ในข้อนี้จะทำการกำหนดตัวแปรขึ้นมาเพื่อให้ง่ายต่อการเข้าใจและอธิบายความหมายของข้อจำกัดในเรื่องของการลำดับ โดยให้ ประเทศ a_q เป็นประเทศที่ต้องทำการเดินทางไปก่อนที่จะเดินทางไปยังประเทศ b_q โดยที่ q นั้นจะเป็น q ตัวเดียวกัน (โดยที่ a_q และ $b_q \in \{1,...,N\}$ และ q=1,2,...,Q และ $a_q \neq b_q$) โดยที่ Q นั้นจะเป็นตัวเลขของจำนวนคู่ทั้งหมดของประเทศ a_q และประเทศ b_q

โดยยกตัวอย่างเช่น ถ้า Q = 2 และ $a_1 = 3$ และ $b_1 = 5$ และ $a_2 = 4$ และ $b_2 = 2$ หมายความว่า ต้องเดินทางไปยังประเทศหมายเลข 3 ก่อนแล้วจึงเดินทางไปยังประเทศหมายเลข 5 และต้องเดินทางไปยังประเทศหมายเลข 4 ก่อนแล้วจึงเดินทางไปยังประเทศหมายเลข 2

3.3.4 Local Search Swap (LS-SWAP)

วิธี Local Search นั้นเป็นวิธีที่มีการทำงานซ้ำไปซ้ำมา โดยจะมีการย้ายจากผลเฉลยอันหนึ่งไปยังผลเฉลยอีกอันหนึ่ง ซึ่งจะสามารถทำให้หาผลเฉลยที่ดีสำหรับปัญหาได้ โดยวิธี Local Search Swap จะทำงานโดยอาศัยหลักการของการสลับ โดยวิธีที่จะนำเสนอในหัวข้อนี้จะมีความแตกต่างจากวิธีแบบดั้งเดิม โดยการสุ่มเลือกประเทศที่จะนำมาทำการ Swap นั้นจะทำการสุ่มโดยมีเงื่อนไขเพิ่มเติม ซึ่งจะต่างจากวิธีแบบดั้งเดิมตรงที่จะทำการสุ่มโดยไม่มีเงื่อนไข โดยเงื่อนไขเพิ่มเติมนี้จะเป็นการช่วยหลีกเลี่ยงหรือเป็นการช่วยลดการสร้างคำตอบที่เป็นไปไม่ได้ขึ้นมา

โดยวิธี Local Search Swap ในหัวข้อนี้จะทำการเรียกว่า LS-SWAP โดยที่ขั้นตอนของวิธี LS-SWAP นั้นสามารถแสดงได้ดังนี้

ขั้นตอนที่ 1 ถ้าไม่มีเที่ยวบินที่จะเดินทางจากประเทศ i ไปยังประเทศ j ในสัปดาห์ที่ k จะให้ต้นทุน c_{ijk} นั้นมีค่ามากๆ (โดยให้มีค่าเท่ากับ 90 ล้าน โดยที่ i และ j และ k = 1,2,...,N และ $i \neq j$)

ขั้นตอนที่ 2 ทำการสร้าง S_0 โดยวิธีการสุ่ม

ขั้นตอนที่ 3 ให้ Count = 0

ขั้นตอนที่ 4 ทำการสร้าง S_1 ตามขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 4.1 ทำการเลือกตัวเลข $u \in \{2,3,...,N\}$ อย่างสุ่มภายใต้เงื่อนไขต่อไปนี้

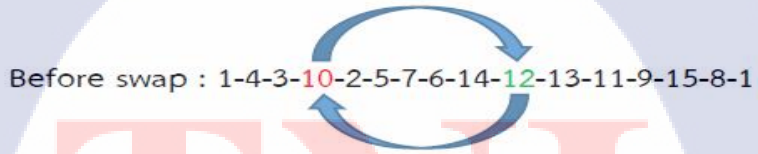
- u ไม่สามารถเป็นหมายเลขใดๆ ของ w_k ได้ โดยที่ k = 1,...,N-1
- u ไม่สามารถเป็นหมายเลขใดๆ ของ b_q ได้ โดยที่ q = 1,...,Q

ขั้นตอนที่ 4.2 ทำการเลือกตัวเลข $v \in \{2,3,...,N\}$ อย่างสุ่มภายใต้เงื่อนไขต่อไปนี้

- v ไม่สามารถเป็นหมายเลขใดๆ ของ w_k ได้ โดยที่ k = 1,...,N-1
- v ไม่สามารถเป็นหมายเลขใดๆ ของ b_q ได้ โดยที่ q = 1,...,Q
- v ต้องไม่เท่ากับ u

ขั้นตอนที่ 4.3 ทำการสร้าง S_1 ตามขั้นตอนต่อไปนี้

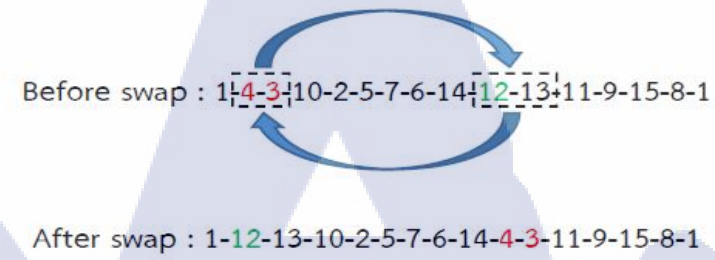
ขั้นตอนที่ 4.3.1 ถ้า u และ v มีค่าไม่เท่ากับหมายเลขใดๆ ของ a_q โดยที่ q = 1,...,Q ทำการสร้าง S_1 โดยการสลับตำแหน่งระหว่างตัวเลข u และตัวเลข v ใน S_0 โดยการทำงานของขั้นตอนนี้สามารถแสดงได้ในรูปที่ 3.6 (ตัวอย่างจากปัญหากรณีที่ 2 โดย u คือ 10 และ v คือ 12) แต่ถ้าไม่เป็นไปตามเงื่อนไขนี้ให้ไปขั้นตอนที่ 4.3.2



รูปที่ 3.6 การ Swap ในขั้นตอนที่ 4.3.1

ขั้นตอนที่ 4.3.2 ถ้าเป็นในกรณีอื่น คือ u หรือ v มีค่าเท่ากับหมายเลข a_q หรือทั้ง u และ v มีค่าเท่ากับหมายเลข a_q (แต่ไม่ใช่ q เดียวกัน) ให้ทำการสลับตำแหน่งระหว่างตัวเลข u และตัวเลข v ใน S_0 ตลอดจนทำการสลับตำแหน่งระหว่างตัวเลขที่อยู่ในตำแหน่งหลังตัวเลข

u กับตัวเลขที่อยู่ในตำแหน่งหลังตัวเลข v ใน S0 โดยการทำงานของขั้นตอนนี้สามารถแสดงได้ในรูปที่ 3.7 (ตัวอย่างจากปัญหากรณีที่ 2 โดย u คือ 4 และ v คือ 12)

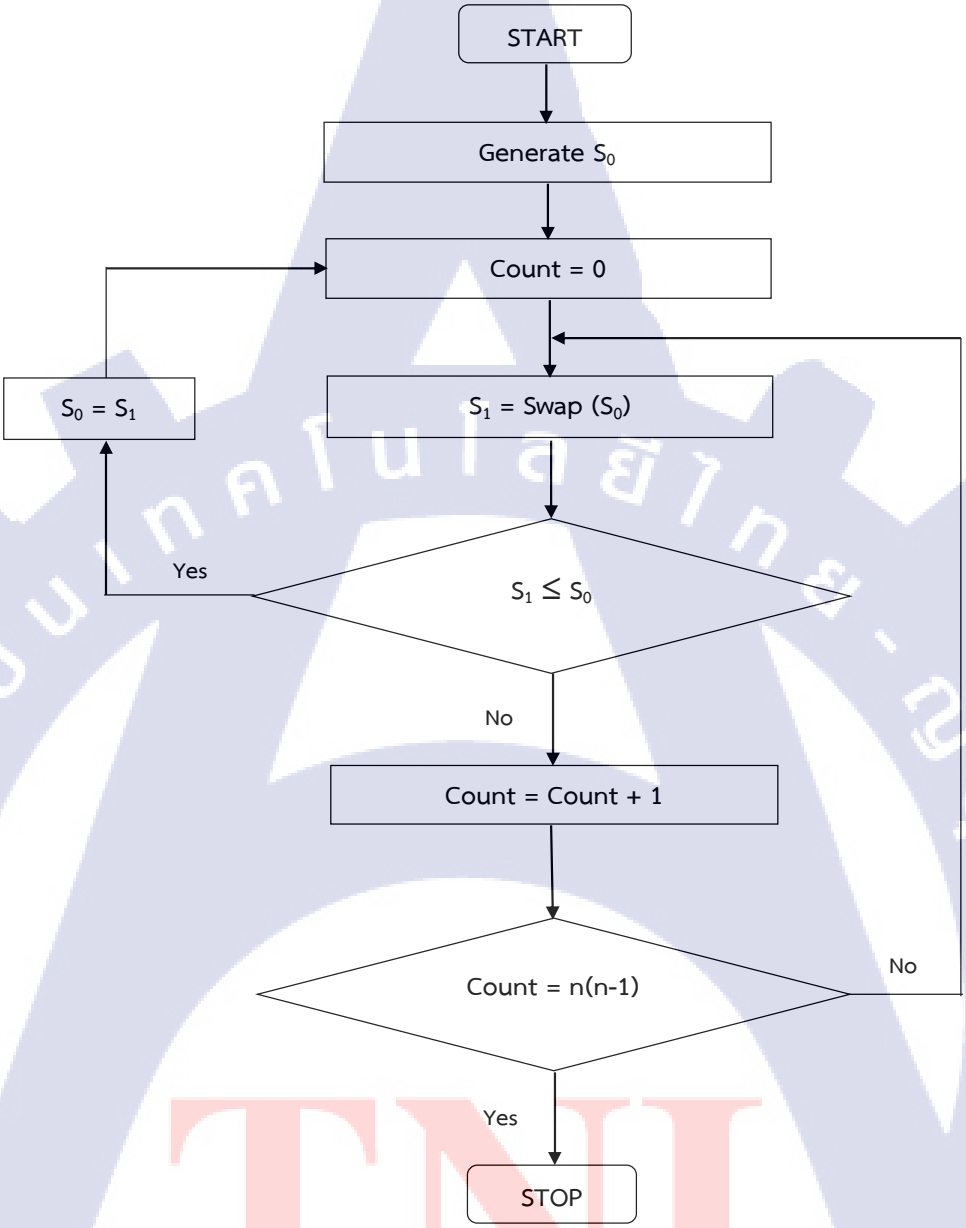


รูปที่ 3.7 การ Swap ในขั้นตอนที่ 4.3.2

ขั้นตอนที่ 5 ถ้าต้นทุนรวมของ S_1 นั้นมีค่าน้อยกว่าหรือเท่ากับต้นทุนรวมของ S_0 จะให้ S_0 เท่ากับ S_1 แล้วไปทำซ้ำขั้นตอนที่ 3 หรือถ้าไม่เป็นแบบนั้นให้ทำการเพิ่ม Count ที่ละ 1 และไปขั้นตอนที่ 6

ขั้นตอนที่ 6 ถ้า Count มีค่าเท่ากับ N(N-1) ให้หยุดการทำงาน หรือถ้าไม่เป็นแบบนั้นให้ไปทำซ้ำจากขั้นตอนที่ 4

โดยแผนผังการทำงานของวิธี LS-SWAP สามารถแสดงได้ดังรูปที่ 3.8



รูปที่ 3.8 แผนผังการทำงานของวิธี LS-SWAP

3.3.5 Local Search Insert (LS-INSERT)

วิธี Local Search นั้นเป็นวิธีที่มีการทำงานซ้ำไปซ้ำมา โดยจะมีการย้ายจากผลเฉลยอันหนึ่งไปยังผลเฉลยอีกอันหนึ่ง ซึ่งจะสามารถทำให้หาผลเฉลยที่ดีสำหรับปัญหาได้ โดยวิธี Local Search Insert จะทำงานโดยอาศัยหลักการของการแทรก โดยวิธีที่จะนำเสนอในหัวข้อนี้จะมีความแตกต่างจากวิธีแบบดั้งเดิม โดยการสุ่มเลือกประเทศที่จะนำมาทำการ Insert นั้นจะทำการสุ่มโดยมีเงื่อนไขเพิ่มเติม ซึ่งจะต่างจากวิธีแบบดั้งเดิมตรงที่จะทำการสุ่มโดยไม่มีเงื่อนไข โดยเงื่อนไขเพิ่มเติมนี้จะเป็นการช่วยหลีกเลี่ยงหรือเป็นการช่วยลดการสร้างคำตอบที่เป็นไปไม่ได้ขึ้นมา

โดยวิธี Local Search Insert ในหัวข้อนี้จะทำการเรียกว่า LS-INSERT โดยที่ขั้นตอนของวิธี LS-INSERT นั้นสามารถแสดงได้ดังนี้

ขั้นตอนที่ 1 จะให้ c_{ijk} มีค่ามากๆ (โดยมีค่าเท่ากับ 90 ล้าน โดยที่ i และ j และ k = 1,2,...,N และ $i \neq j$) ถ้าตรงกับเงื่อนไขหนึ่งข้อหรือมากกว่าหนึ่งข้อของเงื่อนไขเหล่านี้

- ถ้าไม่มีเที่ยวบินที่จะเดินทางจากประเทศ i ไปยังประเทศ j ในสัปดาห์ที่ k
- ถ้า $i = a_q$ และ $j \neq b_q$ โดยที่เป็น q เดียวกัน (โดย q = 1,...,N)
- ถ้า $i \neq a_q$ และ $j = b_q$ โดยที่เป็น q เดียวกัน (โดย q = 1,...,N)

ขั้นตอนที่ 2 ทำการสร้าง S_0 โดยวิธีการสุ่ม

ขั้นตอนที่ 3 ให้ Count = 0

ขั้นตอนที่ 4 ทำการสร้าง S_1 ตามขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 4.1 ทำการเลือกตัวเลข $u \in \{2,3,...,N\}$ อย่างสุ่มภายใต้เงื่อนไขต่อไปนี้

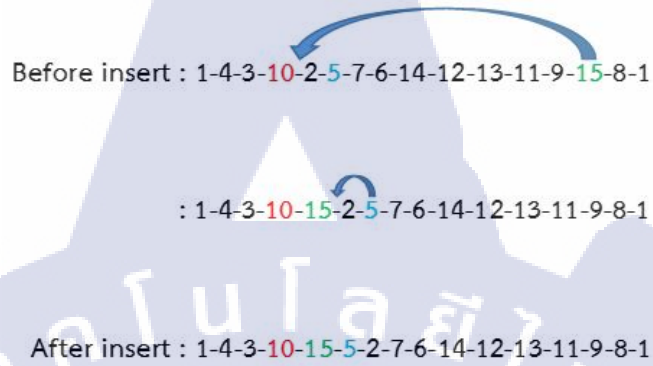
- u ไม่สามารถเป็นหมายเลขใดๆ ของ w_k ได้ โดยที่ k = 1,...,N-1
- u ไม่สามารถเป็นหมายเลขใดๆ ของ b_q ได้ โดยที่ q = 1,...,Q

ขั้นตอนที่ 4.2 ทำการเลือกตัวเลข $v \in \{2,3,...,N\}$ อย่างสุ่มภายใต้เงื่อนไขต่อไปนี้

- v ไม่สามารถเป็นหมายเลขใดๆ ของ a_q ได้ โดยที่ q = 1,...,Q
- v ต้องไม่เท่ากับ u
- ถ้า u เป็นหมายเลข a_q ดังนั้น v จะไม่สามารถเป็นหมายเลข b_q ได้โดยที่เป็น q เดียวกัน โดยที่ q = 1,...,Q

ขั้นตอนที่ 4.3 ทำการสร้าง S_1 โดยย้าย u ออกจากตำแหน่งเดิมใน S_0 แล้วจากนั้นนำ u ไปแทรกไว้ด้านหลัง v ใน S_0 หลังจากที่สร้าง S_1 แล้ว ถ้ามี w_k ไหนที่ไม่ได้อยู่ในตำแหน่งที่ k ตามข้อจำกัดกรอบเวลากำหนดใน S_1 แล้ว S_1 นี้ต้องถูกได้รับการแก้ไข โดยทำการย้าย w_k จากตำแหน่งปัจจุบันและนำไปแทรกยังตำแหน่งที่เป็นตามข้อจำกัดกรอบเวลากำหนดใน S_1 โดยการ

ทำงานของขั้นตอนนี้สามารถแสดงได้ดังรูปที่ 3.9 (ตัวอย่างจากปัญหากรณีที่ 2 โดย น คือ 15 และ v คือ 10)

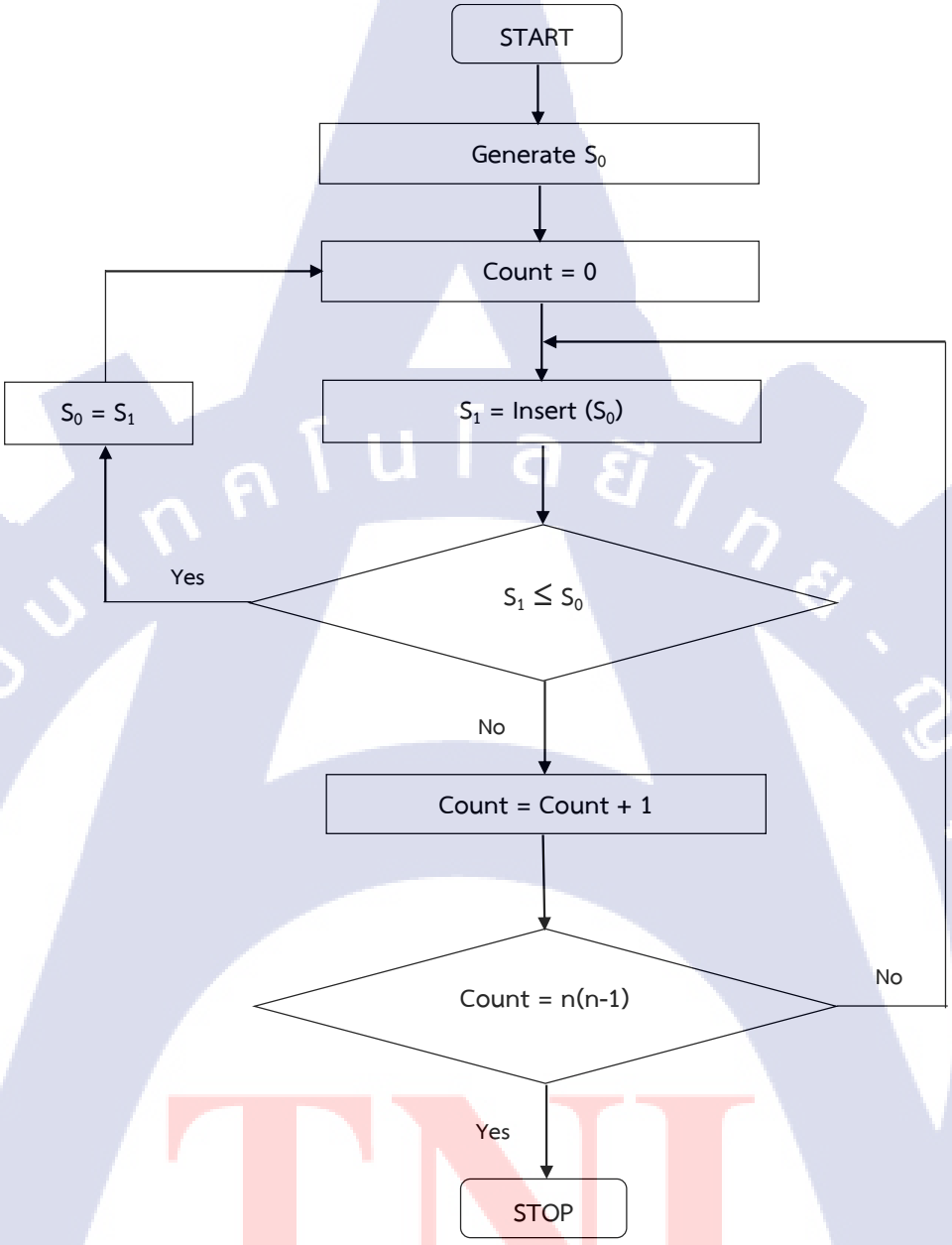


รูปที่ 3.9 การ Insert ในขั้นตอนที่ 4.3

ขั้นตอนที่ 5 ถ้าต้นทุนรวมของ S_1 นั้นมีค่าน้อยกว่าหรือเท่ากับต้นทุนรวมของ S_0 จะให้ S_0 เท่ากับ S_1 แล้วไปทำซ้ำขั้นตอนที่ 3 หรือถ้าไม่เป็นแบบนั้น ให้ทำการเพิ่ม Count ที่ละ 1 และไปขั้นตอนที่ 6

ขั้นตอนที่ 6 ถ้า Count มีค่าเท่ากับ N(N - 1) ให้หยุดการทำงาน หรือถ้าไม่เป็นแบบนั้นให้ไปทำซ้ำจากขั้นตอนที่ 4

โดยแผนผังการทำงานของวิธี LS-INSERT สามารถแสดงได้ดังรูปที่ 3.10



รูปที่ 3.10 แผนผังการทำงานของวิธี LS-INSERT

โดยทั้งวิธี LS-SWAP และ LS-INSERT นั้นจะมี S_0 ที่เป็นผลเฉลยที่เป็นไปได้ (Feasible) แต่อย่างไรก็ตาม S_1 นั้นจะสามารถเป็นผลเฉลยที่เป็นไปไม่ได้ (Infeasible) ได้

โดยที่ S_1 ของวิธี LS-SWAP นั้นจะสามารถเป็นผลเฉลยที่เป็นไปไม่ได้ เนื่องจากเงื่อนไขของการที่ไม่มีเที่ยวบินที่จะเดินทางระหว่างประเทศสองประเทศเท่านั้น ส่วนในวิธี LS-INSERT

นั้น S_1 จะสามารถเป็นผลเฉลยที่เป็นไปไม่ได้ เนื่องจากเงื่อนไขของการที่ไม่มีเที่ยวบินที่จะเดินทางระหว่างประเทศสองประเทศ และเนื่องจากข้อกำหนดในเรื่องของการลำดับ

โดยผลเฉลย S_1 ที่เป็นไปไม่ได้ทุกผลเฉลยนั้น ที่ถูกสร้างโดยทั้งวิธี LS-SWAP และวิธี LS-INSERT นั้นจะทำการให้ค่าต้นทุนที่มีค่ามากๆ ออกมา

บทที่ 4
เงื่อนไขการทดลอง ผลการทดลองและการวิเคราะห์ผลการทดลอง

4.1 เงื่อนไขของการทดลอง

โดยในบทนี้จะนำเสนอผลการทดลองของวิธีที่ถูกพัฒนาขึ้นมาเพื่อใช้แก้ปัญหาทั้ง 3 วิธีซึ่งก็คือวิธี Modified Nearest Neighbor (MNN) วิธี Local Search Swap (LS-SWAP) และวิธี Local Search Insert (LS-INSERT) โดยที่ขั้นตอนการทำงานของ 3 วิธีนี้นั้นได้นำเสนอไปแล้วในบทที่ 3 แล้ว โดยที่เงื่อนไขของการทดลองนั้นสามารถแสดงได้ดังนี้

1. ประสิทธิภาพของวิธี MNN วิธี LS-SWAP และวิธี LS-INSERT จะถูกนำมาเปรียบเทียบในกรณีปัญหา (Problem Instances) จำนวนทั้งหมด 6 กรณี โดยสามารถแบ่งกรณีปัญหาได้เป็น 2 ชุด ตามจำนวนเมืองและจำนวนของข้อจำกัดที่ทำการพิจารณา กรณีปัญหาชุดแรก ได้แก่ กรณีปัญหาที่ 1 ถึงกรณีปัญหาที่ 3 ซึ่งทำการพิจารณาจำนวนเมือง 15 เมือง และพิจารณาเมืองที่มีข้อจำกัดกรอบเวลากำหนดจำนวน 1 เมือง และพิจารณาเมืองที่มีข้อจำกัดของการลำดับจำนวน 1 คู่ สำหรับกรณีปัญหาชุดที่ 2 ได้แก่ กรณีปัญหาที่ 4 ถึงกรณีปัญหาที่ 6 ซึ่งทำการพิจารณาจำนวนเมือง 20 เมือง และพิจารณาเมืองที่มีข้อจำกัดกรอบเวลากำหนด 2 เมือง และพิจารณาเมืองที่มีข้อจำกัดของการลำดับจำนวน 2 คู่ รายละเอียดของกรณีปัญหาทั้งหมดนี้แสดงไว้ในบทที่ 3

2. แม้ว่าวิธี MNN จะคำนวณได้ทั้งจากการโปรแกรมคอมพิวเตอร์และจากการคำนวณด้วยมือ ในวิทยานิพนธ์ฉบับนี้จะคำนวณวิธี MNN ด้วยมือโดยจะเป็นการหาคำตอบเพียงครั้งเดียว เนื่องจากว่าวิธี MNN เป็น Heuristic ที่จะให้ผลลัพธ์เดิมทุกครั้งจากการคำนวณ

3. วิธี LS-SWAP และวิธี LS-INSERT เขียนขึ้นจากโปรแกรมภาษา C# และถูกทำการรันผ่าน Notebook Computer (Intel(R) Core(TM) i5-2430M CPU @ 2.40 GHz with 4 GB of RAM)

4. การรันผลคำตอบของวิธี LS-SWAP และวิธี LS-INSERT นั้นจะทำการรันทั้งหมด 10 รอบ และในแต่ละครั้งของการรันผลคำตอบจะใช้คำตอบเริ่มต้นที่ถูกสร้างมาอย่างสุ่มและแตกต่างกันทั้งหมดของการรัน 10 รอบ

5. การเปรียบเทียบระหว่างวิธีการทั้ง 3 ทำโดยนำคำตอบที่ดีที่สุดของ LS-SWAP และ LS-INSERT ที่ได้จากการรันทั้งหมด 10 รอบมาเปรียบเทียบกับคำตอบที่ได้จาก MNN

4.2 ผลการทดลอง

ในหัวข้อนี้นั้นจะทำการนำเสนอผลการทดลองของวิธี MNN วิธี LS-SWAP และวิธี LS-INSERT ที่ทำการนำมาใช้ในการแก้ปัญหาทั้งหมด 6 กรณี โดยที่ผลการทดลองรวมของวิธี MNN วิธี LS-SWAP และวิธี LS-INSERT นั้นสามารถแสดงได้ดังตารางที่ 4.1 โดยในตารางนี้จะเป็นการแสดงผลของการทดลองของกรณีปัญหาที่ 1 ถึงกรณีปัญหาที่ 6 โดยจะทำการแสดงลำดับการเดินทางที่ดีที่สุดที่ถูกพบจากการแก้ปัญหาด้วยวิธี MNN วิธี LS-SWAP และวิธี LS-INSERT และทำการแสดงต้นทุนรวมที่ต่ำที่สุดของลำดับการเดินทางที่ดีที่สุดที่ถูกพบ และทำการแสดงต้นทุนเฉลี่ยซึ่งได้มาจากการเฉลี่ยต้นทุนที่ทำการรันทั้งหมด 10 รอบของในแต่ละวิธี โดยที่ผลลัพธ์นี้นั้นได้แสดงไว้ในบทความ [23]

ตารางที่ 4.1 ผลการทดลองรวม

กรณีปัญหา	วิธี	ลำดับการเดินทางที่ดีที่สุดที่ถูกพบ	ต้นทุนต่ำสุด	ต้นทุนเฉลี่ย
1	MNN	1-2-3-7-4-5-6-8-15-9-13-12-11-14-10-1	114,616	[
	LS-SWAP	1-2-6-7-4-3-8-15-9-13-14-10-12-11-5-1	111,193	149,509
	LS-INSERT	1-2-7-4-5-6-3-8-15-9-13-10-14-12-11-1	102,280	133,068
2	MNN	1-2-7-4-3-5-6-8-15-9-13-12-14-11-10-1	121,674	[
	LS-SWAP	1-2-7-4-3-5-6-8-15-9-13-10-11-14-12-1	116,309	145,173
	LS-INSERT	1-7-4-3-6-5-2-8-15-9-13-10-11-14-12-1	115,122	160,814
3	MNN	1-2-3-7-5-10-13-6-4-15-8-9-11-14-12-1	173,739	[
	LS-SWAP	1-10-14-12-11-9-13-6-4-5-7-3-8-15-2-1	141,966	179,276
	LS-INSERT	1-10-14-11-12-13-6-5-4-9-15-8-3-2-7-1	133,329	164,028
4	MNN	1-2-4-8-6-10-13-19-3-5-12-18-11-7-15-9-17-16-14-20-1	297,508	[
	LS-SWAP	1-20-12-18-13-10-19-2-3-4-8-15-9-17-16-14-11-7-6-5-1	200,610	236,436
	LS-INSERT	1-13-18-12-14-10-20-19-3-6-5-4-8-15-9-17-16-11-7-2-1	196,620	245,355
5	MNN	1-2-7-4-19-13-12-11-5-3-8-15-17-6-9-16-10-14-18-20-1	224,823	[
	LS-SWAP	1-20-10-19-18-13-11-5-3-2-8-15-4-7-6-9-17-16-14-12-1	194,682	225,415
	LS-INSERT	1-20-12-18-11-13-19-7-4-15-8-2-5-3-6-9-17-16-14-10-1	208,989	236,990
6	MNN	1-2-3-7-10-16-12-11-14-13-18-19-5-4-6-8-15-9-17-20-1	203,281	[
	LS-SWAP	1-2-7-10-20-12-18-11-16-14-13-19-5-4-3-8-15-9-17-6-1	149,065	196,294
	LS-INSERT	1-2-6-8-15-17-9-11-16-14-12-13-5-4-3-7-10-19-18-20-1	182,081	223,917

โดยจากตารางที่ 4.1 จากผลการทดลองรวมนั้นจะสามารถแสดงให้เห็นได้ว่าลำดับของการเดินทางที่ได้จากการแก้ปัญหาโดยวิธี LS-SWAP และวิธี LS-INSERT นั้นจะมีต้นทุนรวมที่ใช้ในการเดินทางที่มีค่าที่ต่ำกว่าลำดับของการเดินทางที่ได้จากการแก้ปัญหาโดยวิธี MNN โดยที่วิธี LS-SWAP

และวิธี LS-INSERT นั้นจะเป็นการทำงานด้วยคอมพิวเตอร์ ส่วนในวิธี MNN นั้นจะเป็นการแก้ปัญหาด้วยมือ โดยที่วิธี LS-SWAP และวิธี LS-INSERT นั้นเป็นวิธีแบบ Local Search ดังรายละเอียดที่ให้ในบทที่ 3 และจากตารางที่ 4.1 จะสามารถเห็นได้ว่าผลการทดลองที่ได้จากวิธี LS-SWAP และผลการทดลองที่ได้จากวิธี LS-INSERT นั้นจะให้ผลการทดลองที่แตกต่างกันในปัญหาแต่ละกรณี โดยในปัญหากรณีที่ 1 นั้นจะสามารถเห็นได้ว่าต้นทุนรวมของลำดับการเดินทางที่ได้จากวิธี LS-INSERT นั้นจะมีค่าที่ต่ำกว่าต้นทุนรวมของลำดับการเดินทางที่ได้จากวิธี LS-SWAP และในปัญหากรณีที่ 2 นั้นจะสามารถเห็นได้ว่าต้นทุนรวมของลำดับการเดินทางที่ได้จากวิธี LS-INSERT นั้นจะมีค่าที่ต่ำกว่าต้นทุนรวมของลำดับการเดินทางที่ได้จากวิธี LS-SWAP และในปัญหากรณีที่ 3 นั้นจะสามารถเห็นได้ว่าต้นทุนรวมของลำดับการเดินทางที่ได้จากวิธี LS-INSERT นั้นจะมีค่าที่ต่ำกว่าต้นทุนรวมของลำดับการเดินทางที่ได้จากวิธี LS-SWAP และในปัญหากรณีที่ 4 นั้นจะสามารถเห็นได้ว่าต้นทุนรวมของลำดับการเดินทางที่ได้จากวิธี LS-INSERT นั้นจะมีค่าที่ต่ำกว่าต้นทุนรวมของลำดับการเดินทางที่ได้จากวิธี LS-SWAP และในปัญหากรณีที่ 5 นั้นจะสามารถเห็นได้ว่าต้นทุนรวมของลำดับการเดินทางที่ได้จากวิธี LS-SWAP นั้นจะมีค่าที่ต่ำกว่าต้นทุนรวมของลำดับการเดินทางที่ได้จากวิธี LS-INSERT และในปัญหากรณีที่ 6 นั้นจะสามารถเห็นได้ว่าต้นทุนรวมของลำดับการเดินทางที่ได้จากวิธี LS-SWAP นั้นจะมีค่าที่ต่ำกว่าต้นทุนรวมของลำดับการเดินทางที่ได้จากวิธี LS-INSERT

และจากตารางผลการทดลองรวมนี้จะสามารถทำการสรุปผลโดยภาพรวมได้ดังนี้ วิธี LS-INSERT นั้นจะเป็นวิธีที่มีความเหมาะสมที่จะนำไปใช้งานกับปัญหาที่มีตัวอย่างแบบง่าย ในขณะที่วิธี LS-SWAP นั้นเป็นวิธีที่มีความเหมาะสมที่จะนำมาใช้งานกับปัญหาที่มีตัวอย่างแบบยาก โดยปัญหาที่มีตัวอย่างแบบง่ายนั้นจะมีลักษณะของปัญหาคือ มีประเทศที่มีข้อจำกัดของการลำดับนั้นไม่เกิน 1 คู่ หมายถึงมีประเทศที่จะต้องทำการเดินทางไปก่อนถึงจะสามารถทำการเดินทางไปยังอีกประเทศได้ในลักษณะนี้จำนวนไม่เกิน 1 คู่ของประเทศ และมีประเทศที่มีข้อจำกัดของกรอบเวลากำหนดนั้นไม่เกิน 1 ประเทศ หมายถึงมีประเทศที่จะต้องทำการไปอยู่ในสัปดาห์นั้นๆ ได้ไม่เกิน 1 ประเทศ ส่วนปัญหาที่มีตัวอย่างแบบยากนั้นจะมีลักษณะของปัญหาคือ มีประเทศที่มีข้อจำกัดของการลำดับนั้นอย่างน้อย 2 คู่ หมายถึงมีประเทศที่จะต้องทำการเดินทางไปก่อนถึงจะสามารถทำการเดินทางไปยังอีกประเทศได้ในลักษณะนี้จำนวนอย่างน้อย 2 คู่ของประเทศ และมีประเทศที่มีข้อจำกัดของกรอบเวลากำหนดนั้นอย่างน้อย 2 ประเทศ หมายถึงมีประเทศที่จะต้องทำการเดินทางไปอยู่ในสัปดาห์นั้นๆ อย่างน้อย 2 ประเทศ โดยถึงแม้ว่าในปัญหาชุดที่ 1 และปัญหาชุดที่ 2 นั้นจะมีความแตกต่างกันในเรื่องของจำนวนประเทศที่ทำการพิจารณา แต่จำนวนของประเทศนั้นอาจจะไม่ได้มีผลกระทบมากนักต่อความยากของปัญหาเมื่อถูกเทียบกับจำนวนของข้อจำกัดในเรื่องของกรอบเวลากำหนด และจำนวนของข้อจำกัดในเรื่องของการลำดับ

โดยผลการทดลองจากตารางที่ 4.1 นั้นเมื่อทำการพิจารณาจากต้นทุนที่ต่ำที่สุดนั้นจะสามารถเห็นได้ว่าวิธี LS-SWAP และวิธี LS-INSERT จะมีค่าที่ดีกว่าต้นทุนที่ได้จากวิธี MNN แต่ถ้าหากพิจารณาจากต้นทุนเฉลี่ยนั้นจะสามารถเห็นได้ว่าวิธี MNN นั้นจะมีค่าต้นทุนที่ต่ำกว่าของวิธี LS-SWAP และวิธี LS-INSERT ดังนั้นจึงมีข้อแนะนำว่าเมื่อจะนำวิธี LS-SWAP และวิธี LS-INSERT นั้นไปประยุกต์ใช้ในการแก้ปัญหาก็ควรที่จะทำการรันคำตอบอย่างน้อย 10 รอบเพื่อที่จะสามารถทำการหาต้นทุนรวมที่มีค่าที่ต่ำกว่าต้นทุนรวมที่ได้จากวิธี MNN ได้ นอกเหนือจากผลลัพธ์ที่ให้ไว้ในตารางที่ 4.1 แล้ว ผลลัพธ์ด้านเวลาในการคำนวณได้ถูกแสดงไว้ในตารางที่ 4.2

ตารางที่ 4.2 เวลาเฉลี่ยที่ใช้ในการรันผล

ปัญหากรณีที่	**วิธี LS-SWAP เวลาเฉลี่ย (วินาที)**	**วิธี LS-INSERT เวลาเฉลี่ย (วินาที)**
1	0.033	0.041
2	0.039	0.043
3	0.036	0.039
4	0.081	0.087
5	0.068	0.067
6	0.071	0.088

วิทยานิพนธ์ฉบับนี้ยังได้ทำการวิเคราะห์ผลการทดลองที่ได้จากตารางที่ 4.1 ในเชิงสถิติ โดยจะแบ่งการวิเคราะห์เป็น การเปรียบเทียบค่าเฉลี่ยประชากรของค่าที่ดีที่สุดจากการรันผล 10 รอบของวิธี LS-SWAP กับวิธี LS-INSERT รวมถึงการเปรียบเทียบกับค่าเฉลี่ยประชากรของคำตอบที่ได้จากวิธี MNN ซึ่งจะทำการแสดงไว้ในหัวข้อที่ 4.2.1.1 และการเปรียบเทียบค่าเฉลี่ยประชากรของค่าเฉลี่ยจากการรันผล 10 รอบของวิธี LS-SWAP กับวิธี LS-INSERT ซึ่งจะทำการแสดงไว้ในหัวข้อที่ 4.2.1.2

4.2.1 วิเคราะห์ผลเชิงสถิติ

โดยในหัวข้อนี้จะทำการวิเคราะห์ผลเชิงสถิติโดยการทำการทดสมมติฐานต่างๆ โดยจะสามารถแสดงได้ดังนี้

4.2.1.1 ทดสอบสมมติฐานโดยพิจารณาจากค่าที่ดีที่สุดจากการรัน 10 รอบ

กรณีที่ 1 H_0 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี MNN ในปัญหาทุกกรณีมีค่าเท่ากับ 0

H_1 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี MNN ในปัญหาทุกกรณีมีค่ามากกว่า 0

กรณีที่ 2 H_0 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-INSERT จากวิธี MNN ในปัญหาทุกกรณีมีค่าเท่ากับ 0

H_1 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-INSERT จากวิธี MNN ในปัญหาทุกกรณีมีค่ามากกว่า 0

กรณีที่ 3 H_0 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาทุกกรณีมีค่าเท่ากับ 0

H_1 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาทุกกรณีมีค่ามากกว่า 0

กรณีที่ 4 H_0 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-INSERT จากวิธี LS-SWAP ในปัญหาชุดที่ 1 มีค่าเท่ากับ 0

H_1 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-INSERT จากวิธี LS-SWAP ในปัญหาชุดที่ 1 มีค่ามากกว่า 0

กรณีที่ 5 H_0 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาชุดที่ 2 มีค่าเท่ากับ 0

H_1 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาชุดที่ 2 มีค่ามากกว่า 0

โดยการทดสอบสมมติฐานทั้ง 5 กรณีนี้จะทำการทดสอบด้วยสถิติ t-test โดยวิธี t-test นี้เป็นเทคนิคการทดสอบสมมติฐานที่นิยมนำมาใช้ในการทดสอบ โดย t-test นี้นั้นจะนำมาใช้ทดสอบความแตกต่างของค่าเฉลี่ย

โดยการทดสอบสมมติฐานนี้จะทำการกำหนดระดับนัยสำคัญที่ 0.2 (~ = 0.2) โดยการทดสอบด้วย t-test นี้จะทำโดยการใช้โปรแกรม Minitab ซึ่งเป็นโปรแกรมที่นิยมนำมาใช้ในการ

ประมวลผลทางสถิติ โดยโปรแกรม Minitab นี้จะให้ค่า P-value ออกมา โดยถ้าค่า P-value ที่ได้ออกมานั้นมีค่าน้อยกว่า ~ (P-value < ~) จะทำการปฏิเสธสมมติฐานหลัก (ปฏิเสธ H_0)

โดยผลการทดสอบสมมติฐานทั้ง 5 กรณีด้วยโปรแกรม Minitab โดยการทดสอบด้วย T-test นั้นสามารถแสดงได้ในตารางที่ 4.2

ตารางที่ 4.3 ผลของการทดสอบสมมติฐานโดยพิจารณาจากค่าที่ดีที่สุดด้วย T-test

	ขนาดของกลุ่มตัวอย่าง	**ค่าเฉลี่ย**	**ค่าเบี่ยงเบนมาตรฐาน**	**t**	**P-value**
เปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี MNN	6	16.40	11.87	3.38	0.010
เปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-INSERT จากวิธี MNN	6	15.13	11.14	3.33	0.010
เปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT	6	1.12	9.89	0.28	0.397
เปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-INSERT จากวิธี LS-SWAP ในปัญหาชุดที่ 1	3	5.03	3.62	2.41	0.069
เปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาชุดที่ 2	3	7.63	10.08	1.31	0.160

จากตารางที่ 4.3 นั้นจะแสดงผลของการทดสอบสมมติฐานด้วย T-test โดยใช้โปรแกรม Minitab ด้วยระดับนัยสำคัญ 0.2 (~ = 0.2) โดยสามารถอธิบายผลของการทดสอบสมมติฐานได้ต่อไปนี้

ผลของการทดสอบสมมติฐานในกรณีที่ 1 นั้นจะสามารถเห็นได้ว่า P-value นั้นมีค่าเท่ากับ 0.01 ซึ่งมีค่าน้อยกว่าระดับนัยสำคัญคือ 0.2 (P-value < ~) ดังนั้นจะทำการปฏิเสธสมมติฐานหลัก (ปฏิเสธ H_0) โดยผลที่ได้คือ ค่าเฉลี่ยของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี MNN ในปัญหาทุกกรณีมีค่ามากกว่า 0 โดยจะสามารถสรุปได้ว่าโดยเฉลี่ยแล้ววิธี LS-SWAP นั้นสามารถทำงานได้ดีกว่าวิธี MNN ด้วยระดับนัยสำคัญ 0.2

ผลของการทดสอบสมมติฐานในกรณีที่ 2 นั้นจะสามารถเห็นได้ว่า P-value นั้นมีค่าเท่ากับ 0.01 ซึ่งมีค่าน้อยกว่าระดับนัยสำคัญคือ 0.2 (P-value < ~) ดังนั้นจะทำการปฏิเสธสมมติฐานหลัก (ปฏิเสธ H_0) โดยผลที่ได้คือ ค่าเฉลี่ยของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-INSRT จากวิธี MNN ในปัญหาทุกกรณีมีค่ามากกว่า 0 โดยจะสามารถสรุปได้ว่าโดยเฉลี่ยแล้ววิธี LS-INSERT นั้นสามารถทำงานได้ดีกว่าวิธี MNN ด้วยระดับนัยสำคัญ 0.2

ผลของการทดสอบสมมติฐานในกรณีที่ 3 นั้นจะสามารถเห็นได้ว่า P-value นั้นมีค่าเท่ากับ 0.397 ซึ่งมีค่ามากกว่าระดับนัยสำคัญคือ 0.2 (P-value > ~) ดังนั้นจะทำการยอมรับสมมติฐานหลัก (ยอมรับ H_0) โดยผลที่ได้คือ ค่าเฉลี่ยของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาทุกกรณีมีค่าเท่ากับ 0 จึงทำให้สรุปของกรณีนี้ได้ว่า ไม่มีหลักฐานเพียงพอที่จะสามารถสนับสนุนได้ว่าโดยเฉลี่ยแล้ววิธี LS-SWAP นั้นสามารถทำงานได้ดีกว่าวิธี LS-INSERT ด้วยระดับนัยสำคัญ 0.2

ผลของการทดสอบสมมติฐานในกรณีที่ 4 นั้นจะสามารถเห็นได้ว่า P-value นั้นมีค่าเท่ากับ 0.069 ซึ่งมีค่าน้อยกว่าระดับนัยสำคัญคือ 0.2 (P-value < ~) ดังนั้นจะทำการปฏิเสธสมมติฐานหลัก (ปฏิเสธ H_0) โดยผลที่ได้คือ ค่าเฉลี่ยของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-INSERT จากวิธี LS-SWAP ในปัญหาชุดที่ 1 มีค่ามากกว่า 0 โดยจะสามารถสรุปได้ว่าโดยเฉลี่ยแล้ววิธี LS-INSERT นั้นสามารถทำงานได้ดีกว่าวิธี LS-SWAP ในปัญหาชุดที่ 1 ด้วยระดับนัยสำคัญ 0.2

ผลของการทดสอบสมมติฐานในกรณีที่ 5 นั้นจะสามารถเห็นได้ว่า P-value นั้นมีค่าเท่ากับ 0.16 ซึ่งมีค่าน้อยกว่าระดับนัยสำคัญคือ 0.2 (P-value < ~) ดังนั้นจะทำการปฏิเสธสมมติฐานหลัก (ปฏิเสธ H_0) โดยผลที่ได้คือ ค่าเฉลี่ยของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาชุดที่ 2 มีค่ามากกว่า 0 โดยจะสามารถสรุปได้ว่าโดยเฉลี่ยแล้ววิธี LS-SWAP นั้นสามารถทำงานได้ดีกว่าวิธี LS-INSERT ในปัญหาชุดที่ 2 ด้วยระดับนัยสำคัญ 0.2

4.2.1.2 ทดสอบสมมติฐานโดยพิจารณาจากค่าเฉลี่ย

กรณีที่ 1 H_0 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาทุกกรณีมีค่าเท่ากับ 0

H_1 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาทุกกรณีมีค่ามากกว่า 0

กรณีที่ 2 H_0 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-INSERT จากวิธี LS-SWAP ในปัญหาชุดที่ 1 มีค่าเท่ากับ 0

H_1 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-INSERT จากวิธี LS-SWAP ในปัญหาชุดที่ 1 มีค่ามากกว่า 0

กรณีที่ 3 H_0 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาชุดที่ 2 มีค่าเท่ากับ 0

H_1 : ค่าเฉลี่ยประชากรของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาชุดที่ 2 มีค่ามากกว่า 0

โดยการทดสอบสมมติฐานทั้ง 3 กรณีนี้จะทำการทดสอบด้วยสถิติ T-test โดยวิธี T-test นี้เป็นเทคนิคการทดสอบสมมติฐานที่นิยมนำมาใช้ในการทดสอบ โดย T-test นี้นั้นจะนำมาใช้ทดสอบความแตกต่างของค่าเฉลี่ย

โดยการทดสอบสมมติฐานนี้จะทำการกำหนดระดับนัยสำคัญที่ 0.2 ($\sim$ = 0.2) โดยการทดสอบด้วย T-test นี้จะทำโดยการใช้โปรแกรม Minitab ซึ่งเป็นโปรแกรมที่นิยมนำมาใช้ในการประมวลผลทางสถิติ โดยโปรแกรม Minitab นี้จะให้ค่า P-value ออกมา โดยถ้าค่า P-value ที่ได้ออกมานั้นมีค่าน้อยกว่า $\sim$ (P-value < $\sim$) จะทำการปฏิเสธสมมติฐานหลัก (ปฏิเสธ H_0)

โดยผลการทดสอบสมมติฐานทั้ง 3 กรณีด้วยโปรแกรม Minitab โดยการทดสอบด้วย T-test นั้นสามารถแสดงได้ในตารางที่ 4.4

ตารางที่ 4.4 ผลของการทดสอบสมมติฐานโดยพิจารณาจากค่าเฉลี่ยด้วย T-test

	ขนาดของกลุ่มตัวอย่าง	**ค่าเฉลี่ย**	**ค่าเบี่ยงเบนมาตรฐาน**	**t**	**P-value**
เปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT	6	1.47	10.10	0.36	0.368
เปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-INSERT จากวิธี LS-SWAP ในปัญหาชุดที่ 1	3	2.87	11.90	0.42	0.358
เปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาชุดที่ 2	3	6.93	4.69	2.56	0.062

ผลของการทดสอบสมมติฐานในกรณีที่ 1 นั้นจะสามารถเห็นได้ว่า P-value นั้นมีค่าเท่ากับ 0.368 ซึ่งมีค่ามากกว่าระดับนัยสำคัญคือ 0.2 (P-value > ~) ดังนั้นจะทำการยอมรับสมมติฐานหลัก (ยอมรับ H_0) โดยผลที่ได้คือ ค่าเฉลี่ยของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาทุกกรณีมีค่าเท่ากับ 0 ซึ่งจากที่ทำการทดสอบสมมติฐานโดยที่พิจารณาจากค่าที่ดีที่สุดนั้นยังไม่สามารถสรุปได้ว่าวิธีไหนสามารถทำงานได้ดีกว่ากัน และเมื่อทดสอบสมมติฐานโดยที่พิจารณาจากค่าเฉลี่ยแล้วนั้นจะสามารถสรุปได้ว่า ไม่มีหลักฐานเพียงพอที่จะสามารถสนับสนุนได้ว่าโดยเฉลี่ยแล้ววิธี LS-SWAP นั้นสามารถทำงานได้ดีกว่าวิธี LS-INSERT ในปัญหาทุกกรณีด้วยระดับนัยสำคัญ 0.2

ผลของการทดสอบสมมติฐานในกรณีที่ 2 นั้นจะสามารถเห็นได้ว่า P-value นั้นมีค่าเท่ากับ 0.358 ซึ่งมีค่ามากกว่าระดับนัยสำคัญคือ 0.2 (P-value > ~) ดังนั้นจะทำการยอมรับสมมติฐานหลัก (ยอมรับ H_0) โดยผลที่ได้คือ ค่าเฉลี่ยของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-INSERT จากวิธี LS-SWAP ในปัญหาชุดที่ 1 มีค่าเท่ากับ 0 โดยที่ผลของการทดสอบสมมติฐานนี้จะขัดแย้งกับผลการทดสอบสมมติฐานโดยพิจารณาจากค่าที่ดีที่สุด ซึ่งอาจจะเป็นเพราะว่าค่าเฉลี่ยที่ได้จากวิธี LS-INSERT ในปัญหาชุดที่ 1 นั้นไม่ได้มีค่าที่ดีกว่าค่าเฉลี่ยที่ได้จากวิธี LS-SWAP ทุกค่า

ผลของการทดสอบสมมติฐานในกรณีที่ 3 นั้นจะสามารถเห็นได้ว่า P-value นั้นมีค่าเท่ากับ 0.062 ซึ่งมีค่าน้อยกว่าระดับนัยสำคัญคือ 0.2 (P-value < ~) ดังนั้นจะทำการปฏิเสธสมมติฐานหลัก (ปฏิเสธ H_0) โดยผลที่ได้คือ ค่าเฉลี่ยของเปอร์เซ็นต์ของผลเฉลยที่ดีขึ้นของวิธี LS-SWAP จากวิธี LS-INSERT ในปัญหาชุดที่ 2 มีค่ามากกว่า 0 โดยจะสามารถสรุปได้ว่าโดยเฉลี่ยแล้ววิธี LS-SWAP นั้นสามารถทำงานได้ดีกว่าวิธี LS-INSERT ในปัญหาชุดที่ 2 ด้วยระดับนัยสำคัญ 0.2 ซึ่งจะเหมือนกับผลที่ได้จากการทดสอบสมมติฐานโดยพิจารณาจากค่าที่ดีที่สุด

โดยที่การทดสอบสมมติฐานโดยพิจารณาจากค่าเฉลี่ยของวิธี LS-SWAP เทียบกับวิธี MNN และวิธี LS-INSERT เทียบกับวิธี MNN นั้นจะไม่ได้นำมาแสดงในที่นี้ เนื่องจากว่าผลการทดสอบที่ได้จากการพิจารณาจากค่าเฉลี่ยนั้นบ่งบอกว่าวิธี LS-SWAP และวิธี LS-INSERT นั้นไม่สามารถที่จะทำงานได้ดีกว่าวิธี MNN แต่อย่างไรก็ตามจากการรันผลของวิธี LS-SWAP และวิธี LS-INSERT ซึ่งทำการรันผลทั้งหมด 10 ครั้งนั้นจะสามารถให้ผลเฉลยที่ดีกว่าผลเฉลยของวิธี MNN ได้ในปัญหาทุกกรณี

บทที่ 5
สรุปผลการทดลอง

5.1 สรุปผลการวิจัย

- งานวิจัยนี้ทำการศึกษาปัญหา Asymmetric Travelling Salesman Problem (ATSP) ที่มีการเพิ่มข้อจำกัดในเรื่องของกรอบเวลากำหนด (Time Window Constraints) และข้อจำกัดในเรื่องของการลำดับ (Precedence Constraints) โดยตัวปัญหานั้นเป็นการศึกษาของการเดินทางทางอากาศ โดยมีเป้าหมายเพื่อศึกษาหาต้นทุนที่ใช้ในการเดินทางที่ต่ำที่สุด

- งานวิจัยนี้ได้นำเสนอวิธีที่จะนำมาใช้ในการแก้ไขปัญหาคือ วิธี MNN วิธี LS-SWAP และวิธี LS-INSERT โดยที่วิธี MNN นั้นเป็นวิธีที่ถูกปรับปรุงจากวิธี Nearest Neighbor

- วิธี LS-SWAP และวิธี LS-INSERT นั้นจะแตกต่างจากวิธีดั้งเดิมตรงที่การสุ่มประเทศเพื่อที่จะนำมาใช้ในการ Swap หรือ Insert นั้นจะถูกทำการสุ่มโดยมีเงื่อนไขเพิ่มขึ้นมา ซึ่งต่างจากวิธีดั้งเดิมจะทำสุ่มโดยไม่มีเงื่อนไข โดยเงื่อนไขในการสุ่มที่เพิ่มขึ้นมานี้ก็เพื่อเป็นการลดโอกาสการที่จะสร้างผลเฉลยที่เป็นไปไม่ได้ หรือไม่สามารถที่จะใช้เป็นคำตอบได้ออกมา

- โดยจากการที่ได้ทำการรันโปรแกรมและวิเคราะห์ผลการทดลองโดยการทดสอบสมมติฐานนั้น วิธี LS-SWAP และวิธี LS-INSERT สามารถที่จะทำงานได้ดีกว่าวิธี MNN ด้วยระดับนัยสำคัญ 0.2

- โดยที่วิธี LS-INSERT นั้นจะเหมาะสมกับปัญหาในกรณีง่าย คือ มีประเทศที่มีข้อจำกัดกรอบเวลากำหนดไม่เกิน 1 ประเทศ และมีคู่ของประเทศที่มีข้อจำกัดของการลำดับไม่เกิน 1 คู่ มากกว่าวิธี LS-SWAP ด้วยระดับนัยสำคัญ 0.2 โดย P-value มีค่าเท่ากับ 0.069

- โดยที่วิธี LS-SWAP นั้นจะเหมาะสมกับปัญหาในกรณียาก คือ มีประเทศอย่างน้อย 2 ประเทศที่มีข้อจำกัดกรอบเวลากำหนด และมีจำนวนคู่ของประเทศอย่างน้อย 2 คู่ที่มีข้อจำกัดของการลำดับ มากกว่าวิธี LS-INSERT ด้วยระดับนัยสำคัญ 0.2 โดย P-value มีค่าเท่ากับ 0.160

- เนื่องจากว่าหากทำการพิจารณาต้นทุนเฉลี่ยนั้นจะสามารถเห็นได้ว่าวิธี MNN นั้นจะมีค่าต้นทุนที่ต่ำกว่าของวิธี LS-SWAP และวิธี LS-INSERT ดังนั้นจึงมีข้อแนะนำว่าเมื่อจะนำวิธี LS-SWAP และวิธี LS-INSERT นั้นไปประยุกต์ใช้ในการแก้ปัญหาก็ควรที่จะทำการรันคำตอบอย่างน้อย 10 รอบ เพื่อที่จะสามารถทำการหาต้นทุนรวมที่มีค่าที่ต่ำกว่าต้นทุนรวมที่ได้จากวิธี MNN ได้

5.2 ข้อเสนอแนะ

จากงานวิจัยฉบับนี้ได้มีการปรับปรุงคำตอบด้วยวิธี Local Search ในรูปแบบการแทรกเมือง หรือ การ Insert และรูปแบบการสลับเมือง หรือการ Swap โดยถ้าหากมีผู้ที่สนใจศึกษาปัญหาในลักษณะนี้และสามารถทำการหาวิธีการที่ใช้ในการแทรกเมืองหรือวิธีการที่ใช้ในการสลับเมืองที่มีความแตกต่างออกไปจากที่ทำการนำเสนอในงานวิจัยฉบับนี้ ก็อาจจะทำให้สามารถหาต้นทุนรวมที่ใช้ในการเดินทางที่มีค่าที่ต่ำกว่านี้ได้ และถ้าหากผู้ที่สนใจศึกษาปัญหาในลักษณะนี้นั้นสามารถที่จะทำการหา Neighborhood Structure ที่มีความแตกต่างออกไปจากที่ทำการนำเสนอในงานวิจัยฉบับนี้และสามารถนำมาประยุกต์ใช้ได้ ก็อาจจะจะทำให้สามารถหาลำดับของการเดินทางที่มีต้นทุนรวมที่ใช้ในการเดินทางที่มีค่าต่ำกว่าที่หาได้จากวิธีที่ได้นำเสนอในงานวิจัยนี้ได้

บรรณานุกรม

บรรณานุกรม

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-Completeness.* New York : Macmillan Higher Education, 1979.

[2] S. Saad et al., "Solving standard traveling salesman problem and multiple traveling salesman problem by using branch and bound," *20th National Sysposium on Mathematical Sciences,* SKSM20, Putrajaya, Malaysia, December 22, 2012, pp. 1406-1411.

[3] H. Calgor, *"The brute force algorithm,"* [Online]. Available : http://www.ctl.ua.edu/math103/hamilton/HCalgor.htm#the Brute force algorithm. [Accessed : August 1, 2013].

[4] M. Brusno and S. Stahl, *Branch and Bound Applications in Combinatorial Data Analysis,* USA : Springer, 2005.

[5] D. Rosenkrantz et al., "An analysis of several heuristics for the traveling salesman problem," *SIAM Journal on Computing,* vol. 6, no.3, pp.563-581, May 1977.

[6] Y. Li and S. Gong, "Dynamic ant colony optimization for TSP," *International Journal of Advance Manufacturing Technology,* vol. 22, no.2, pp.528-533, July 2003.

[7] H. Ahmed, "Genetic algorithm for traveling salesman problem using sequential constructive crossover operator," *International Journal of Biometrics & Bioinformatics,* vol. 3, no.6, pp.96-105, August 2010.

[8] K. Ghoseiri and H. Sarhadi, "A memetic algorithm for symmetric traveling salesman problem." *International Journal of Management Science and Engineering Management,* vol. 3, no.4, pp.275-283, February 2008.

[9] G. B. Dantzig et al., "Solution of a large-scale traveling salesman problem," *Operations Research,* vol. 2, no.1, pp.393-410, August 1954.

[10] R. Roberti and P. Toth, "Model and algorithms for asymmetric travelling salesman problem : an experiment comparison," E*URO Journal on Transportation and Logistics,* vol. 1, no.2, pp.113-133, May 2012.

[11] J. Brest and J. Zeronik, "A heuristic for asymmetric traveling salesman problem," 6th Metaheuristics International Conference, MIC2005, Vienna, Austria, August 22, 2010, pp. 145-150.

[12] J. Majumdar and A. K. Bhunia, "Genetic algorithm for asymmetric traveling salesman problem with imprecise travel times," *Journal of Computational and Applied Mathematics,* vol. 235, no.9, pp.3063-3078, August 2011.

[13] M. Shaikh and M. Panchal, "Solving asymmetric travelling salesman problem using memetric algorithm," *International Journal of Emerging Technology and Advanced Engineering,* vol. 2, no.11, pp.634-639, November 2012.

[14] O. Jaruphat and P. Chaovalitwongse, "Heuristic for open vehicle routing problem to reduce transportation cost," *Engineering Journal,* vol. 4, no.3, pp.58-72, May 2013.

[15] M. Gendreau et al., "A generalized insertion heuristic for the traveling salesman problem with time windows," *Operations Research,* vol. 46, no.3, pp.330-335, June 1998.

[16] N. Ascheuer et al., "A branch & cut algorithm for the asymmetric travelling salesman problem with precedence constraints," *International Journal of Computational Optimization and Application,* vol. 17, no.1, pp.61-84, January 1999.

[17] L. Bianco et al., "Exact and heuristic procedures for the traveling salesman problem with precedence constraints, based on dynamic programming," *Information Systems and Operational Research,* vol. 32, no.1, pp.19-31, February 1994.

[18] Dohop, *"The traveling tourist problem (TTP),"* [Online]. Available : http://www.dohop.com/hackathon/winners/ttp.html. [Accessed : February 7, 2016].

[19] J. Touyz, *"Solving the travelling tourist problem,"* [Online]. Available : http://www.joshdoesstats.com/blog/2013/12/22/solving-travelling-tourist-problem-using-evolutionary-markov-chain-monte-carlo/. [Accessed : February 8, 2016].

[20] Openflights, *"The air-traveling salesman,"* [Online]. Available : http://sites.google.com/site/travellingcudasalesman/data. [Accessed : February 9, 2016].

[21] F. Furini, *"The time dependent traveling salesman planning problem in controlled airspace,"* [Online]. Available : http://www.optimization-online.org/DB_HTML/2015/06/4971.html. [Accessed : February 9, 2016].

[22] C. Malandraki and R. B. Dial, "A restriced dynamic programming heuristic algorithm for the time dependent traveling salesman problem," *European Journal of Operational Research,* vol. 90, no.1, pp.45-55, April 1996.

[23] T. Saradatta and P. Pongchairerks, "A time-dependent ATSP with time window and precedence constraints in air travel," *Jurnal Teknologi,* in press.

[24] S. Acharya, "Vehicle routing and scheduling problems with time window constraint-optimization based models," *International Journal of Mathematical Sciences & Applications,* vol. 3, no.1, pp.167-179, January 2013.

[25] M. M. Solomon, "Algorithm for the vehicle routing and scheduling problems with time windows constraints," *Operations Research,* vol. 35, no.2, pp.254-265, March 1987.

[26] G. A. Korsah et al., *"Optimal vehicle routing and scheduling with precedence constraints and location choice,"* [Online]. Available : http://www.ri.cmu.edu/publication_view.html?pub_id=6778. [Accessed : April 26, 2016].

[27] J. Renaud et al., "A heuristic for the pickup and delivery traveling salesman problem," *Computers & Operations Research,* vol. 27, no.9, pp.905-916, August 2000.

[28] H. H. Hoos and T. Stutzle, *Stochastic Local Search : Foundations and Application.* San Fransico : Morgan Kaufmann, 2005.

[29] O. Mersmann et al., "Local search and the traveling salesman problem : A feature-based characterization of problem hardness," *6th International Conference on Learning and Intelligent Optimization,* LION6, Paris, France, January 16, 2012, pp. 115-129.

[30] P. C. Kanellakis and C. H. Papadimitriou, "Local search for the asymmetric traveling salesman problem," *Operations Research,* vol. 28, no.5, pp.1086-1099, October 1980.

[31] S. Urrutia et al., "A dynamic programming based local search approach for the double traveling salesman problem with multiple stacks," International *Transactions in Operational Research,* vol. 22, no.1, pp.61-75, January 2015.

[32] T. Saradatta and P. Pongchairerks, *"Instances for time-dependent ATSP with time window and precedence constraints in air travel,"* [Online]. Available : https://drive.google.com/folderview?id=0B2XqS3TSsvP7UFFEalE0UlpWbHc&usp=sharing. [Accessed: November 20, 2015].

ภาคผนวก

ภาคผนวก ก.

Pseudo Code โปรแกรม C# ของวิธี LS-SWAP ในปัญหาชุดที่ 1

Pseudo Code โปรแกรม C# ของวิธี LS-SWAP ในปัญหาชุดที่ 1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Swap
{
    class Program
    {
        static void Main()
        {
                Console.Write("seed number : ");
                int seed = int.Parse(Console.ReadLine());
                Console.WriteLine();
                Console.Write("Number of cities: ");
                int num = int.Parse(Console.ReadLine());
                int[, ,] cost = new int[num + 1, num + 1, num + 1];
                for (int i = 1; i <= num; i++)
                {
                    for (int j = 1; j <= num; j++)
                    {
                        for (int k = 1; k <= num; k++)
                        {
                            if (i != j)
                            {
                                Console.Write("Travelling cost from cities " + i + " to " + j + " on week " + k + " is: ");
                                cost[i, j, k] = int.Parse(Console.ReadLine());
                            }
                        }
                    }
                }
                    Console.WriteLine();
                    Console.WriteLine("Random Number");
                    Console.WriteLine("-------------");
                    Console.WriteLine("City\t\tRandomNumber");
                    Random rnd = new Random(seed);
                    double[] rand = new double[num + 1];
```

```
for (int i = 2; i <= num; i++)
{
    if (i != 2)
    {
        rand[i] = rnd.NextDouble();
    }
    if (rand[12] == rand[i])
    {
        rand[11] = rand[i] + 0.000001;
    }
}
for (int i = 2; i <= num; i++)
{
    if (i != 2)
    {
        Console.Write(i);
        Console.Write("\t:");
        Console.WriteLine("\t" + rand[i]);
    }
}
Console.WriteLine();
Console.WriteLine("Sorting cities by random number (min to max)");
Console.WriteLine("----------------------------------------");
int[] solution0 = new int[num + 2];
double min = 10;
int minindex = 0;
Console.WriteLine("City(S0)");
for (int j = 2; j <= num; j++)
{
    if (j != 2)
    {
        min = 10;
        for (int i = 2; i <= num; i++)
        {
            if (i != 2)
            {
                if (rand[i] < min)
                {
                    min = rand[i];
                }
            }
```

```
        }
        for (int i = 2; i <= num; i++)
        {
            if (i != 2)
            {
                if (rand[i] == min)
                {
                    solution0[1] = 1;
                    solution0[2] = 2;
                    solution0[j] = i;
                    minindex = i;
                }
            }
        }
        rand[minindex] = 10;
    }
}
solution0[16] = 1;
for (int i = 1; i <= num + 1; i++)
{
    Console.Write(solution0[i] + " ");
}
Console.WriteLine();
Console.WriteLine();
Console.WriteLine("Total Cost(S0)");
Console.WriteLine("--------------");
int Cost0 = 0;
int Totalcost0 = 0;
for (int i = 1; i <= num; i++)
{
    Cost0 = cost[solution0[i], solution0[i + 1], i];
    Totalcost0 = Totalcost0 + Cost0;
}
Console.Write(Totalcost0);
int Count = 0;
Random random = new Random(seed);
do // ***
{
    Console.WriteLine();
    Console.WriteLine();
    Console.WriteLine("Random solution for swap");
```

```
Console.WriteLine("------------------------");
int a = random.Next(3, num + 1);
int b = random.Next(3, num + 1);
if (a == 2)
{
    do
    {
        a = random.Next(3, num + 1);
    } while (a == 2);
}
if (b == 2)
{
    do
    {
        b = random.Next(3, num + 1);
    } while (b == 2);
}
if (a == b)
{
    do
    {
        b = random.Next(3, num + 1);
        if (b == 2)
        {
            do
            {
                b = random.Next(3, num + 1);
            } while (b == 2);
        }
    } while (b == a);
}
if (a == 11)
{
    do
    {
        a = random.Next(3, num + 1);
    } while (a == 11);
}
if (b == 11)
{
    do
```

```
    {
        b = random.Next(3, num + 1);
    } while (b == 11);
}
if (a == b)
{
    do
    {
        b = random.Next(3, num + 1);

        if (b == 11)
        {
            do
            {
                b = random.Next(3, num + 1);
            } while (b == 11);
        }
    } while (b == a);
}
Console.WriteLine(a + "," + b);
Console.WriteLine();
Console.WriteLine("Solution after swap(S1)");
Console.WriteLine("----------------------");
int[] solution1 = new int[num + 2];
Array.Copy(solution0, solution1, num + 2);
int x = 0;
int y = 0;
if (a == 12)
{
    for (int i = 1; i <= num; i++)
    {
        if (solution0[i] == a)
        {
            x = i;
        }
        if (solution0[i] == b)
        {
            y = i;
        }
    }
    if (x > y)
```

```
    {
        solution1[y] = a;
        solution1[y + 1] = 11;
        solution1[x + 1] = b;
        for (int j = y; j <= x - 2; j++)
        {
            solution1[j + 2] = solution0[j + 1];
        }
    }
    if (y > x)
    {
        solution1[x] = b;
        solution1[y - 1] = a;
        solution1[y] = 11;
        for (int k = x; k <= y - 3; k++)
        {
            solution1[k + 1] = solution0[k + 2];
        }
    }
    if (y + 1 == x)
    {
        solution1[y] = a;
        solution1[x] = 11;
        solution1[x + 1] = b;
    }
    if (x + 2 == y)
    {
        solution1[x] = b;
        solution1[y] = 11;
        solution1[y - 1] = a;
    }
}
if (b == 12)
{
    for (int i = 1; i <= num; i++)
    {
        if (solution0[i] == b)
        {
            x = i;
        }
        if (solution0[i] == a)
```

```
        {
            y = i;
        }
    }
    if (x > y)
    {
        solution1[y] = b;
        solution1[y + 1] = 11;
        solution1[x + 1] = a;
        for (int j = y; j <= x - 2; j++)
        {
            solution1[j + 2] = solution0[j + 1];
        }
    }
    if (y > x)
    {
        solution1[x] = a;
        solution1[y - 1] = b;
        solution1[y] = 11;
        for (int k = x; k <= y - 3; k++)
        {
            solution1[k + 1] = solution0[k + 2];
        }
    }
    if (y + 1 == x)
    {
        solution1[y] = b;
        solution1[x] = 11;
        solution1[x + 1] = a;
    }
    if (x + 2 == y)
    {
        solution1[x] = a;
        solution1[y] = 11;
        solution1[y - 1] = b;
    }
}
if (a != 12) // Swap
{
    if (b != 12)
    {
```

```
            for (int i = 1; i <= num; i++)
            {
                if (solution0[i] == a)
                {
                    solution1[i] = b;
                }
            }
            for (int i = 1; i <= num; i++)
            {
                if (solution0[i] == b)
                {
                    solution1[i] = a;
                }
            }
        }
    }
    for (int i = 1; i <= num + 1; i++)
    {
        Console.Write(solution1[i] + " ");
    }
    Console.WriteLine();
    Console.WriteLine();
    Console.WriteLine("Total Cost(S1)");
    Console.WriteLine("--------------");
    int Cost1 = 0;
    int Totalcost1 = 0;
    for (int i = 1; i <= num; i++)
    {
        Cost1 = cost[solution1[i], solution1[i + 1], i];
        Totalcost1 = Totalcost1 + Cost1;
    }
    Console.Write(Totalcost1);
    if (Totalcost1 <= Totalcost0)
    {
        solution0 = solution1;
        Totalcost0 = Totalcost1;
        Count = 0;
    }
    else
    {
        Count = Count + 1;
```

```
            }
            Console.WriteLine();
            Console.Write("(" + Count + ")");
        } while (Count < num * (num - 1));
        Console.WriteLine();
        Console.Read();
}}}
```

ภาคผนวก ข.

Pseudo Code โปรแกรม C# ของวิธี LS-INSERT ในปัญหาชุดที่ 1

Pseudo Code โปรแกรม C# ของวิธี LS-INSERT ในปัญหาชุดที่ 1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Insert
{
    class Program
    {
        static void Main()
        {
            Console.Write("seed number : ");
            int seed = int.Parse(Console.ReadLine());
            Console.WriteLine();
            Console.Write("Number of cities: ");
            int num = int.Parse(Console.ReadLine());
            int[, ,] cost = new int[num + 1, num + 1, num + 1];
            for (int i = 1; i <= num; i++)
            {
                for (int j = 1; j <= num; j++)
                {
                    for (int k = 1; k <= num; k++)
                    {
                        if (i != j)
                        {
                            Console.Write("Travelling cost from cities " + i + " to " + j + " on week " + k + " is: ");
                            cost[i, j, k] = int.Parse(Console.ReadLine());
                        }
                    }
                }
            }
            for (int t = 1; t <= 200; t++)
            {
                Console.WriteLine();
                Console.WriteLine("Random Number");
                Console.WriteLine("-------------");
                Console.WriteLine("City\t\tRandomNumber");
                Random rnd = new Random(seed);
                double[] rand = new double[num + 1];
```

```
for (int i = 2; i <= num; i++)
{
    if (i != 2)
    {
        rand[i] = rnd.NextDouble();
    }
    if (rand[12] == rand[i])
    {
        rand[11] = rand[i] + 0.000001;
    }
}
for (int i = 2; i <= num; i++)
{
    if (i != 2)
    {
        Console.Write(i);
        Console.Write("\t:");
        Console.WriteLine("\t" + rand[i]);
    }
}
Console.WriteLine();
Console.WriteLine("Sorting cities by random number (min to max)");
Console.WriteLine("-----------------------------------------");
int[] solution0 = new int[num + 2];
double min = 10;
int minindex = 0;
Console.WriteLine("City(S0)");
for (int j = 2; j <= num; j++)
{
    if (j != 2)
    {
        min = 10;
        for (int i = 2; i <= num; i++)
        {
            if (i != 2)
            {
                if (rand[i] < min)
                {
                    min = rand[i];
                }
            }
```

```
        }
        for (int i = 2; i <= num; i++)
        {
            if (i != 2)
            {
                if (rand[i] == min)
                {
                    solution0[1] = 1;
                    solution0[2] = 2;
                    solution0[j] = i;
                    minindex = i;
                }
            }
        }
        rand[minindex] = 10;
    }

}
solution0[16] = 1;
for (int i = 1; i <= num + 1; i++)
{
    Console.Write(solution0[i] + " ");
}
Console.WriteLine();
Console.WriteLine();
Console.WriteLine("Total Cost(S0)");
Console.WriteLine("--------------");
int Cost0 = 0;
int Totalcost0 = 0;
for (int i = 1; i <= num; i++)
{
    Cost0 = cost[solution0[i], solution0[i + 1], i];
    Totalcost0 = Totalcost0 + Cost0;
}
Console.Write(Totalcost0);
int Count = 0;
int loop = 0;
Random random = new Random(seed);
do // ***
{
    Console.WriteLine();
```

```
Console.WriteLine();
Console.WriteLine("Random solution for insert");
Console.WriteLine("-------------------------");
int a = random.Next(2, num + 1);
int b = random.Next(2, num + 1);
if (a == 11)
{
    do
    {
        a = random.Next(2, num + 1);
        if (a == 2)
        {
            do
            {
                a = random.Next(2, num + 1);
            } while (a == 2);
        }
    } while (a == 11);
}
if (a == 2)
{
    do
    {
        a = random.Next(2, num + 1);
        if (a == 11)
        {
            do
            {
                a = random.Next(2, num + 1);
            } while (a == 11);
        }
    } while (a == 2);
}
if (b == 12)
{
    do
    {
        b = random.Next(2, num + 1);
    } while (b == 12);
}
if (a == b)
```

```
{
    do
    {
        b = random.Next(2, num + 1);
        if (b == 12)
        {
            do
            {
                b = random.Next(2, num + 1);
            } while (b == 12);
        }
    } while (b == a);
}
if (a == 12)
{
    if (b == 11)
    {
        do
        {
            b = random.Next(2, num + 1);

            if (b == 12)
            {
                do
                {
                    b = random.Next(2, num + 1);
                } while (b == 12);
            }
        } while (b == 11);
    }
}
Console.WriteLine(a + "," + b);
Console.WriteLine();
Console.WriteLine("Solution after insert(S1)");
Console.WriteLine("----------------------");
int[] solution1 = new int[num + 2];
Array.Copy(solution0, solution1, num + 2);
int x = 0;
int y = 0;
for (int i = 1; i <= num; i++)
{
```

```
    if (solution0[i] == a)
    {
        x = i;
    }
    if (solution0[i] == b)
    {
        y = i;
    }
}

if (a != 12)
{
    if (x > y)
    {
        solution1[y] = b;
        solution1[y + 1] = a;
        for (int j = y; j <= x - 2; j++)
        {
            solution1[j + 2] = solution0[j + 1];
        }
    }
    if (y > x)
    {
        solution1[y] = a;
        solution1[y - 1] = b;
        for (int k = x; k <= y - 2; k++)
        {
            solution1[k] = solution0[k + 1];
        }
    }
    if (x + 1 == y)
    {
        solution1[y] = a;
        solution1[x] = b;
    }
    if (y + 1 == x)
    {
        solution1[y] = a;
        solution1[x] = b;
    }
}
```

```
if (a == 12)
{
    if (x > y)
    {
        solution1[y] = b;
        solution1[y + 1] = a;
        solution1[y + 2] = 11;
        for (int i = y; i <= x - 2; i++)
        {
            solution1[i + 3] = solution0[i + 1];
        }
    }
    if (y > x)
    {
        solution1[y] = 11;
        solution1[y - 1] = a;
        solution1[y - 2] = b;
        for (int i = x; i <= y - 3; i++)
        {
            solution1[i] = solution0[i + 2];
        }
    }
    if (y + 1 == x)
    {
        solution1[y] = b;
        solution1[x] = a;
    }
    if (x + 2 == y)
    {
        solution1[y] = 11;
        solution1[y - 1] = a;
        solution1[x] = b;
    }
}
for (int i = 1; i <= num + 1; i++)
{
    Console.Write(solution1[i] + " ");
}
Console.WriteLine();
Console.WriteLine();
Console.WriteLine("Total Cost(S1)");
```

```
            Console.WriteLine("----------");
            int Cost1 = 0;
            int Totalcost1 = 0;
            for (int i = 1; i <= num; i++) // FIX HERE
            {
                Cost1 = cost[solution1[i], solution1[i + 1], i];
                Totalcost1 = Totalcost1 + Cost1;
            }
            Console.Write(Totalcost1);
            if (Totalcost1 < Totalcost0)
            {
                solution0 = solution1;
                Totalcost0 = Totalcost1;
                Count = 0;
                if (Count == 0)
                {
                    loop = loop + 1;
                }
            }
            else
            {
                Count = Count + 1;
            }
            Console.WriteLine();
            Console.Write("(" + Count + ")");
            Console.Write("[" + loop + "]");
        } while (Count < num * (num - 1)); // ***
        Console.WriteLine();
        Console.Read();
    }
  }
 }
}
```

ภาคผนวก ค.

ผลการทดลองของวิธี LS-SWAP และวิธี LS-INSERT ในปัญหา 6 กรณี

ตารางที่ ค.1 ผลการทดลองของปัญหากรณีที่ 1 โดยวิธี LS-SWAP

รันครั้งที่	เวลา (วินาที)	ลำดับการเดินทาง	ต้นทุนรวม (บาท)
1	0.046	1-2-3-8-15-9-13-10-14-12-11-6-7-4-5-1	117,918
2	0.032	1-2-3-10-13-9-15-8-6-14-12-11-5-4-7-1	176,670
3	0.025	1-2-12-11-14-9-15-8-5-4-3-10-13-6-7-1	207,632
4	0.030	1-2-4-3-8-15-9-13-7-5-6-10-14-12-11-1	158,073
5	0.029	1-2-7-4-9-15-8-6-5-3-10-14-12-11-13-1	139,186
6	0.021	1-2-6-8-15-9-5-4-3-10-12-11-14-13-7-1	159,758
7	0.032	1-2-6-13-10-14-12-11-5-4-3-8-15-9-7-1	136,421
8	0.022	1-2-10-14-12-11-13-3-8-9-15-4-5-6-7-1	159,794
9	0.052	1-2-8-15-4-9-13-10-14-12-11-5-6-3-7-1	128,449
10	0.037	1-2-6-7-4-3-8-15-9-13-14-10-12-11-5-1	111,193

ตารางที่ ค.2 ผลการทดลองของปัญหากรณีที่ 1 โดยวิธี LS-INSERT

รันครั้งที่	เวลา (วินาที)	ลำดับการเดินทาง	ต้นทุนรวม (บาท)
1	0.036	1-2-8-15-4-5-3-10-13-14-12-11-9-6-7-1	130,888
2	0.038	1-2-7-15-8-6-5-12-11-14-10-3-4-9-13-1	196,678
3	0.041	1-2-7-3-8-15-4-5-6-10-14-12-11-9-13-1	143,550
4	0.054	1-2-6-5-4-15-8-3-10-13-14-12-11-9-7-1	131,352
5	0.033	1-2-3-10-13-14-12-11-9-15-8-6-5-4-7-1	109,776
6	0.03	1-2-8-15-9-6-5-4-7-3-13-12-11-14-10-1	152,372
7	0.051	1-2-7-5-4-9-15-8-3-10-12-11-14-13-6-1	133,289
8	0.035	1-2-7-4-5-6-3-10-13-14-12-11-9-15-8-1	113,401
9	0.042	1-2-3-4-15-8-9-13-10-14-12-11-5-6-7-1	117,089
10	0.044	1-2-7-4-5-6-3-8-15-9-13-10-14-12-11-1	102,280

ตารางที่ ค.3 ผลการทดลองของปัญหากรณีที่ 2 โดยวิธี LS-SWAP

รันครั้งที่	เวลา (วินาที)	ลำดับการเดินทาง	ต้นทุนรวม (บาท)
1	0.047	1-2-7-4-3-5-6-8-15-9-13-10-11-14-12-1	116,309
2	0.035	1-6-10-14-12-5-4-3-2-8-15-9-11-13-7-1	148,700
3	0.034	1-8-15-9-11-5-4-3-10-14-12-13-6-7-2-1	146,976
4	0.036	1-2-7-4-3-5-6-10-13-14-12-11-9-15-8-1	127,577
5	0.039	1-12-2-7-6-5-4-3-10-13-14-11-9-15-8-1	159,364
6	0.036	1-10-14-12-11-5-6-9-13-7-4-3-8-15-2-1	188,505
7	0.041	1-2-8-15-9-5-7-4-3-10-11-14-12-13-6-1	155,053
8	0.037	1-10-14-12-11-5-4-3-7-2-8-15-9-13-6-1	136,349
9	0.039	1-10-14-12-11-5-4-3-8-15-9-13-6-7-2-1	134,835
10	0.042	1-10-2-7-6-5-4-3-8-15-9-13-11-14-12-1	138,062

ตารางที่ ค.4 ผลการทดลองของปัญหากรณีที่ 2 โดยวิธี LS-INSERT

รันครั้งที่	เวลา (วินาที)	ลำดับการเดินทาง	ต้นทุนรวม (บาท)
1	0.052	1-7-4-3-6-5-2-8-15-9-13-10-11-14-12-1	115,122
2	0.038	1-7-9-15-8-5-6-4-3-10-11-14-13-12-2-1	162,193
3	0.039	1-8-15-9-6-5-7-2-4-3-10-13-11-14-12-1	152,400
4	0.055	1-8-15-9-11-5-4-3-10-14-12-13-6-7-2-1	146,976
5	0.044	1-11-9-14-12-5-4-3-10-13-6-7-15-8-2-1	217,821
6	0.037	1-2-4-3-6-5-12-14-10-13-11-9-15-8-7-1	136,306
7	0.055	1-8-15-2-6-5-4-3-7-9-13-10-11-14-12-1	149,347
8	0.050	1-4-3-15-9-5-7-10-11-14-12-13-6-8-2-1	214,212
9	0.031	1-12-9-15-8-5-7-2-4-3-10-11-14-13-6-1	198,638
10	0.033	1-7-4-3-6-5-2-8-15-9-13-10-11-14-12-1	115,122

ตารางที่ ค.5 ผลการทดลองของปัญหากรณีที่ 3 โดยวิธี LS-SWAP

รันครั้งที่	เวลา (วินาที)	ลำดับการเดินทาง	ต้นทุนรวม (บาท)
1	0.044	1-2-8-15-9-13-6-5-4-3-10-11-14-12-7-1	153,419
2	0.031	1-13-6-7-15-8-3-5-4-9-11-10-14-12-2-1	174,690
3	0.038	1-10-14-12-11-9-13-6-4-5-7-3-8-15-2-1	141,966
4	0.031	1-2-10-9-11-14-12-5-4-15-8-3-13-6-7-1	201,186
5	0.041	1-2-8-15-7-10-12-5-4-9-11-14-13-6-3-1	191,504
6	0.046	1-11-9-15-8-2-3-5-4-10-14-12-13-6-7-1	162,119
7	0.026	1-12-14-11-9-13-6-5-4-15-7-10-3-8-2-1	222,031
8	0.030	1-3-10-12-7-2-8-15-4-9-11-14-13-6-5-1	171,252
9	0.034	1-7-3-2-8-15-9-5-4-10-11-14-12-13-6-1	163,622
10	0.043	1-8-3-5-12-11-9-15-4-10-14-13-6-7-2-1	210,967

ตารางที่ 6 ผลการทดลองของปัญหากรณีที่ 3 โดยวิธี LS-INSERT

รันครั้งที่	เวลา (วินาที)	ลำดับการเดินทาง	ต้นทุนรวม (บาท)
1	0.035	1-7-9-15-8-2-3-5-4-10-11-14-12-13-6-1	157,804
2	0.056	1-10-14-12-13-6-7-5-4-3-11-9-15-8-2-1	142,760
3	0.032	1-10-14-12-11-5-7-2-4-15-8-9-13-6-3-1	165,509
4	0.034	1-8-15-9-10-14-11-5-4-7-3-2-12-13-6-1	188,112
5	0.039	1-11-14-12-2-8-3-5-4-7-15-9-10-13-6-1	167,831
6	0.049	1-2-7-3-10-14-11-5-4-15-8-9-12-13-6-1	164,606
7	0.038	1-10-14-11-12-13-6-5-4-9-15-8-3-2-7-1	133,329
8	0.032	1-10-14-13-6-3-2-7-4-5-12-11-9-15-8-1	169,272
9	0.033	1-13-6-7-15-8-3-5-4-9-11-10-14-12-2-1	174,690
10	0.038	1-8-15-9-13-6-7-3-4-5-12-11-14-10-2-1	176,368

ตารางที่ ค.7 ผลการทดลองของปัญหากรณีที่ 4 โดยวิธี LS-SWAP

ครั้งที่	เวลา	ลำดับการเดินทาง	ต้นทุนรวม
1	0.085	1-6-5-19-18-10-14-13-3-2-4-8-15-9-17-16-12-20-11-7-1	213,069
2	0.071	1-20-12-14-13-10-18-19-3-6-5-2-4-8-15-9-17-16-11-7-1	213,453
3	0.064	1-4-8-9-16-10-17-15-3-19-13-12-18-20-14-11-7-2-6-5-1	275,477
4	0.105	1-5-4-8-6-10-20-19-3-15-9-17-16-14-12-18-13-11-7-2-1	241,484
5	0.053	1-5-19-14-16-10-17-6-3-4-8-15-9-13-12-18-20-11-7-2-1	213,402
6	0.076	1-20-12-14-16-10-18-19-3-4-8-2-6-13-11-7-15-9-17-5-1	255,096
7	0.075	1-5-4-8-6-10-20-2-3-15-9-17-16-13-19-18-12-14-11-7-1	245,537
8	0.086	1-20-19-18-13-10-11-7-3-6-5-2-4-8-15-9-17-16-14-12-1	215,741
9	0.090	1-20-12-18-13-10-19-2-3-4-8-15-9-17-16-14-11-7-6-5-1	200,610
10	0.097	1-11-7-15-17-10-20-2-3-5-4-8-9-16-14-12-18-19-13-6-1	290,495

ตารางที่ 8 ผลการทดลองของปัญหากรณีที่ 4 โดยวิธี LS-INSERT

ครั้งที่	เวลา	ลำดับการเดินทาง	ต้นทุนรวม
1	0.083	1-20-16-11-7-10-6-5-3-13-14-12-18-19-9-17-15-4-8-2-1	307,390
2	0.075	1-12-18-20-16-10-14-13-3-4-8-15-9-17-6-5-19-11-7-2-1	221,146
3	0.152	1-12-18-20-19-10-14-13-3-6-5-4-8-15-9-17-16-11-7-2-1	205,330
4	0.107	1-12-19-14-16-10-11-7-3-6-5-4-8-15-17-9-13-18-20-2-1	221,869
5	0.096	1-13-18-12-14-10-20-19-3-6-5-4-8-15-9-17-16-11-7-2-1	196,620
6	0.054	1-19-18-20-16-10-17-6-3-13-12-14-11-7-5-4-8-9-15-2-1	273,181
7	0.114	1-6-12-13-16-10-18-19-3-5-4-8-15-9-17-20-14-11-7-2-1	224,317
8	0.066	1-4-8-15-5-10-19-6-3-13-18-20-12-14-16-17-9-11-7-2-1	284,091
9	0.063	1-4-8-2-20-10-13-5-3-15-9-17-16-14-12-18-19-11-7-6-1	267,630
10	0.068	1-13-14-19-18-10-11-7-3-5-4-8-15-2-6-9-17-16-20-12-1	251,976

ตารางที่ ค.9 ผลการทดลองของปัญหากรณีที่ 5 โดยวิธี LS-SWAP

ครั้งที่	เวลา	ลำดับการเดินทาง	ต้นทุนรวม
1	0.064	1-10-14-12-18-13-16-17-7-15-8-6-9-11-20-19-4-5-3-2-1	248,821
2	0.084	1-10-14-11-12-13-18-19-4-15-8-7-5-3-6-9-17-16-20-2-1	213,929
3	0.056	1-6-9-11-10-13-19-4-5-3-8-15-17-16-14-12-18-20-2-7-1	202,882
4	0.056	1-7-4-6-9-13-14-16-17-15-8-2-10-20-11-12-18-19-5-3-1	226,078
5	0.058	1-20-10-19-18-13-11-5-3-2-8-15-4-7-6-9-17-16-14-12-1	194,682
6	0.055	1-2-7-6-9-13-18-19-4-15-8-5-3-10-14-12-16-11-17-20-1	267,443
7	0.131	1-20-11-14-10-13-18-19-4-15-8-2-7-5-3-6-9-17-16-12-1	211,544
8	0.057	1-7-5-3-19-13-10-16-17-15-8-4-6-9-11-14-12-18-20-2-1	232,823
9	0.056	1-6-9-16-10-13-12-20-7-2-8-4-19-18-14-11-17-15-5-3-1	254,392
10	0.066	1-20-10-14-16-13-18-19-17-15-8-2-4-5-3-7-6-9-11-12-1	201,553

ตารางที่ ค.10 ผลการทดลองของปัญหากรณีที่ 5 โดยวิธี LS-INSERT

ครั้งที่	เวลา	ลำดับการเดินทาง	ต้นทุนรวม
1	0.072	1-4-5-3-10-13-14-19-17-15-8-2-7-6-9-11-16-20-18-12-1	222,397
2	0.062	1-20-12-18-11-13-19-7-4-15-8-2-5-3-6-9-17-16-14-10-1	208,989
3	0.062	1-11-19-18-12-13-6-9-17-15-8-2-7-4-5-3-10-14-16-20-1	222,507
4	0.061	1-19-18-20-14-13-6-9-17-15-8-2-7-4-5-3-10-16-11-12-1	217,541
5	0.062	1-11-14-19-18-13-7-5-3-2-8-15-4-6-9-17-16-20-10-12-1	218,687
6	0.060	1-5-3-7-20-13-12-16-17-15-8-2-4-6-9-11-14-19-18-10-1	261,733
7	0.102	1-2-5-3-19-13-10-16-17-15-8-7-4-6-9-11-20-18-14-12-1	239,657
8	0.062	1-7-5-3-19-13-14-20-17-15-8-4-11-18-12-2-6-9-16-10-1	311,012
9	0.061	1-10-14-11-18-13-12-20-4-15-8-2-7-6-9-17-16-19-5-3-1	227,105
10	0.070	1-5-3-19-18-13-11-2-4-15-8-7-6-9-17-16-14-12-20-10-1	240,269

ตารางที่ ค.11 ผลการทดลองของปัญหากรณีที่ 6 โดยวิธี LS-SWAP

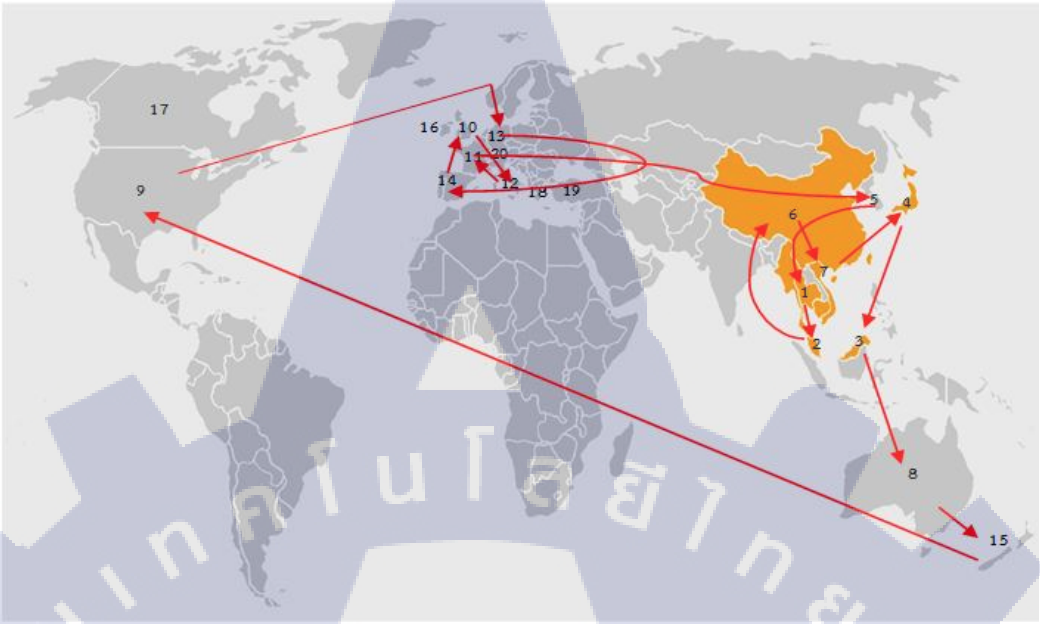
ครั้งที่	เวลา	ลำดับการเดินทาง	ต้นทุนรวม
1	0.074	1-7-10-14-13-19-18-11-20-12-16-9-5-8-15-17-6-4-3-2-1	218,850
2	0.088	1-2-7-10-20-12-18-11-16-14-13-19-5-4-3-8-15-9-17-6-1	149,065
3	0.069	1-2-8-15-4-9-17-11-14-19-3-6-5-7-10-16-13-18-20-12-1	191,991
4	0.097	1-19-17-9-16-20-18-11-2-8-15-4-5-6-3-7-10-13-14-12-1	222,844
5	0.096	1-8-15-17-9-13-19-11-20-2-3-6-5-4-7-10-16-14-18-12-1	173,569
6	0.058	1-8-15-9-17-16-14-11-20-2-3-4-5-7-10-12-18-19-13-6-1	189,342
7	0.065	1-8-15-9-17-16-14-11-13-19-6-4-5-3-2-7-10-18-20-12-1	185,691
8	0.056	1-2-7-10-20-12-18-11-14-16-9-17-5-19-13-6-8-15-4-3-1	206,635
9	0.061	1-7-10-16-20-12-18-11-14-19-2-4-5-8-15-17-9-13-6-3-1	211,971
10	0.055	1-8-15-9-16-14-12-11-20-17-6-4-5-3-2-7-10-18-19-13-1	212,979

ตารางที่ ค.12 ผลการทดลองของปัญหากรณีที่ 6 โดยวิธี LS-INSERT

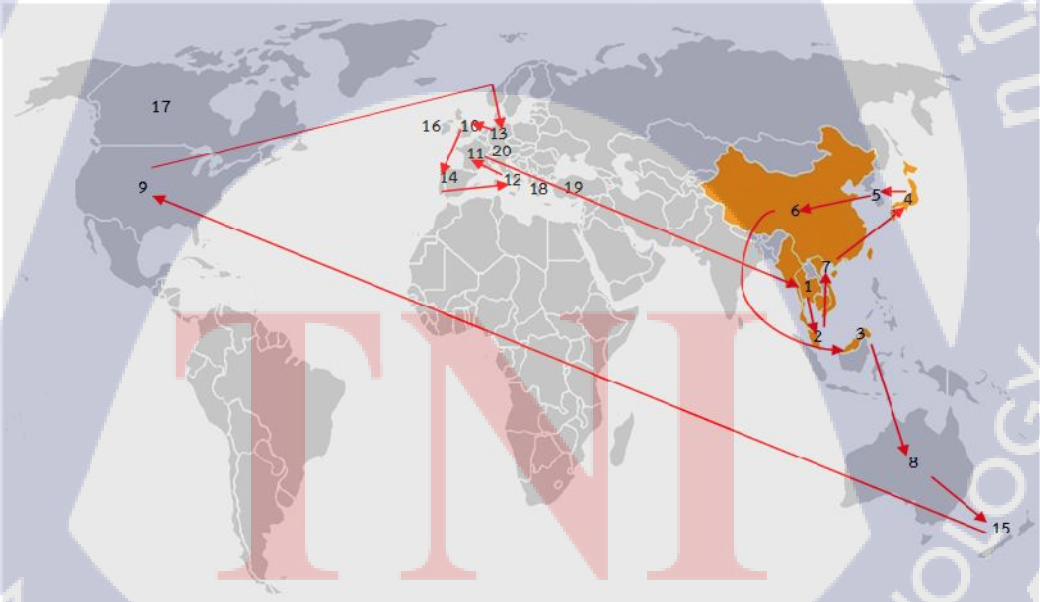
ครั้งที่	เวลา	ลำดับการเดินทาง	ต้นทุนรวม
1	0.116	1-7-10-18-13-16-14-11-9-17-6-4-5-3-8-15-2-19-20-12-1	190,878
2	0.079	1-6-8-15-2-4-3-11-12-13-9-17-5-7-10-19-18-14-16-20-1	248,927
3	0.073	1-2-6-8-15-17-9-11-16-14-12-13-5-4-3-7-10-19-18-20-1	182,081
4	0.117	1-6-17-9-16-13-19-11-18-20-2-4-5-3-8-15-7-10-14-12-1	214,377
5	0.089	1-8-15-17-13-19-18-11-9-6-3-2-5-4-7-10-20-16-14-12-1	228,501
6	0.070	1-3-19-18-20-16-13-11-2-8-15-4-5-7-10-12-14-9-17-6-1	225,346
7	0.108	1-8-15-4-7-10-20-11-16-14-18-12-5-6-3-2-19-17-9-13-1	226,895
8	0.063	1-7-10-20-19-18-12-11-16-14-13-6-5-3-8-15-17-9-4-2-1	198,810
9	0.077	1-19-18-14-12-16-9-11-6-7-10-13-5-4-3-8-15-17-20-2-1	267,923
10	0.084	1-13-16-14-19-7-10-11-17-6-2-4-5-3-8-15-9-12-18-20-1	255,431

ภาคผนวก ง.

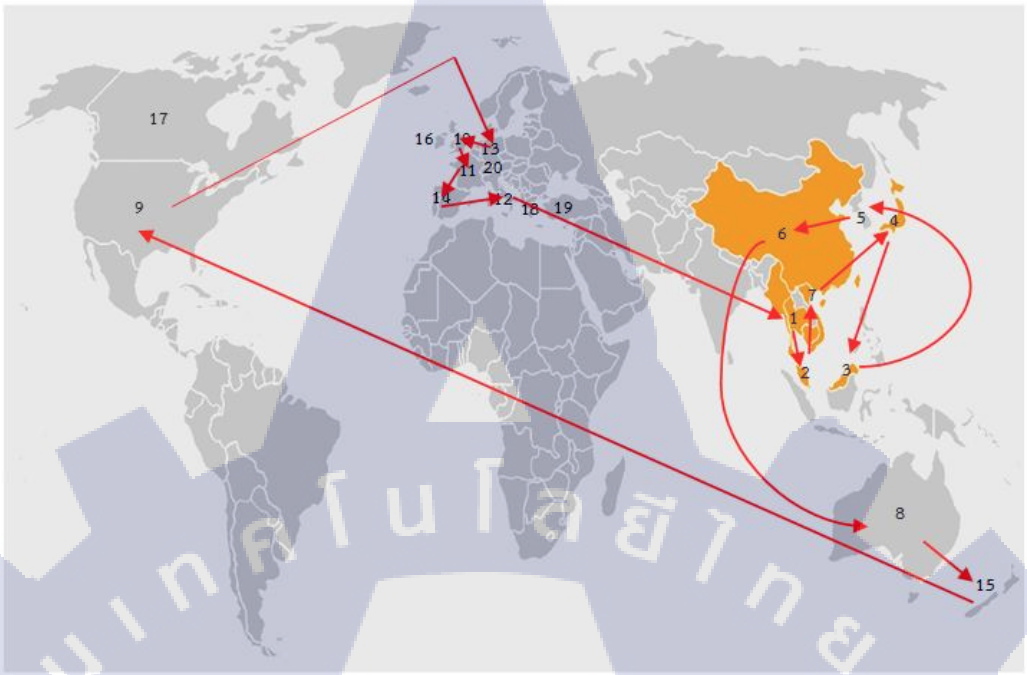
ลำดับการเดินทางที่มีต้นทุนต่ำที่สุดโดยวิธี LS-SWAP และวิธี LS-INSERT



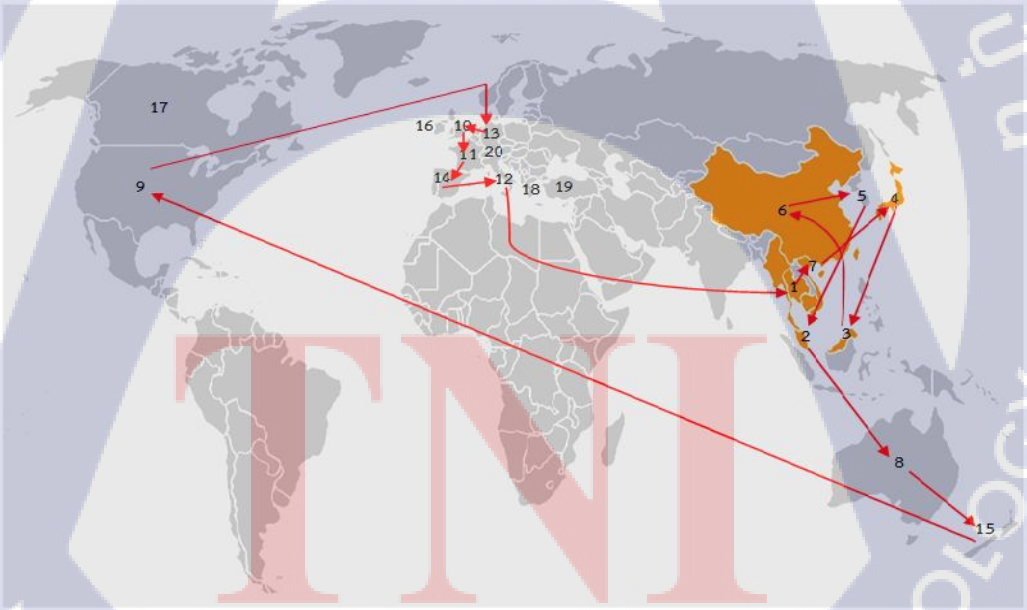
รูปที่ ง.1 ลำดับการเดินทางที่มีต้นทุนต่ำที่สุดในปัญหากรณีที่ 1 โดยวิธี LS-SWAP



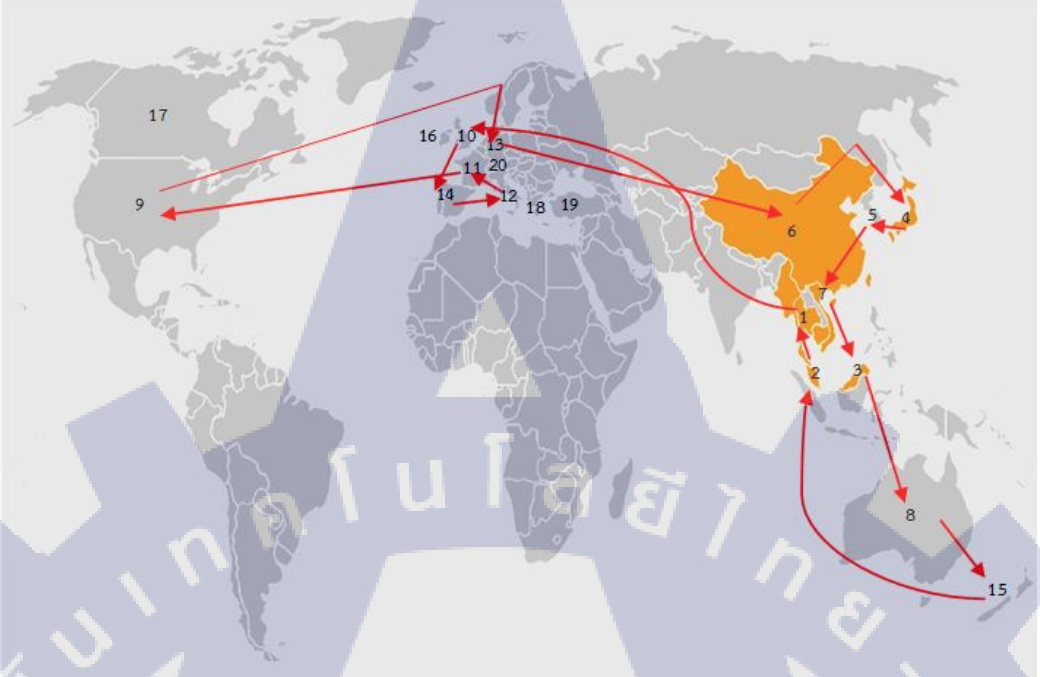
รูปที่ ง.2 ลำดับการเดินทางที่มีต้นทุนต่ำที่สุดในปัญหากรณีที่ 1 โดยวิธี LS-INSERT



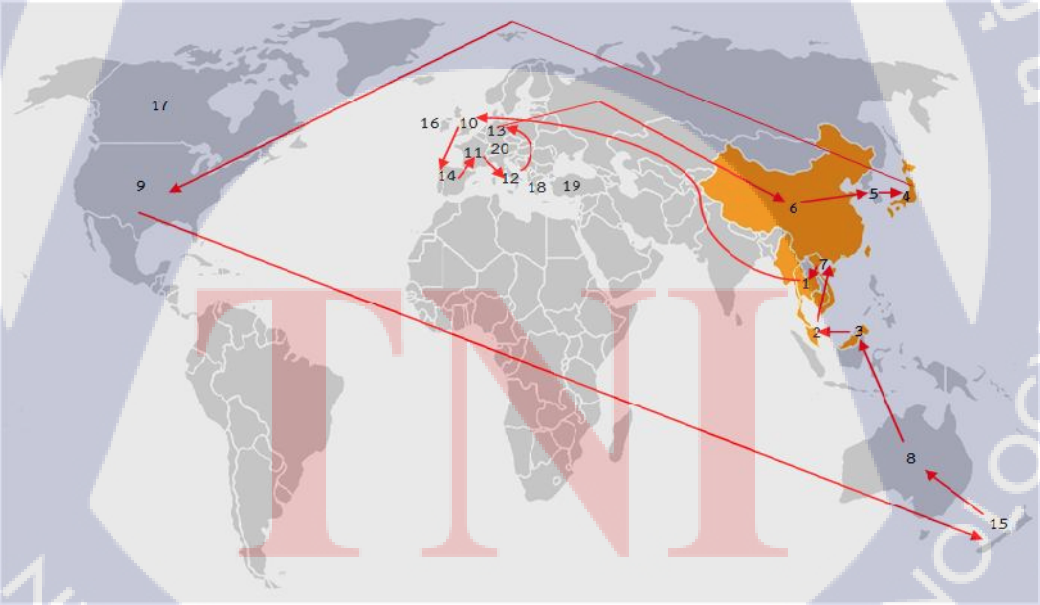
รูปที่ ง.3 ลำดับการเดินทางที่มีต้นทุนต่ำที่สุดในปัญหากรณีที่ 2 โดยวิธี LS-SWAP



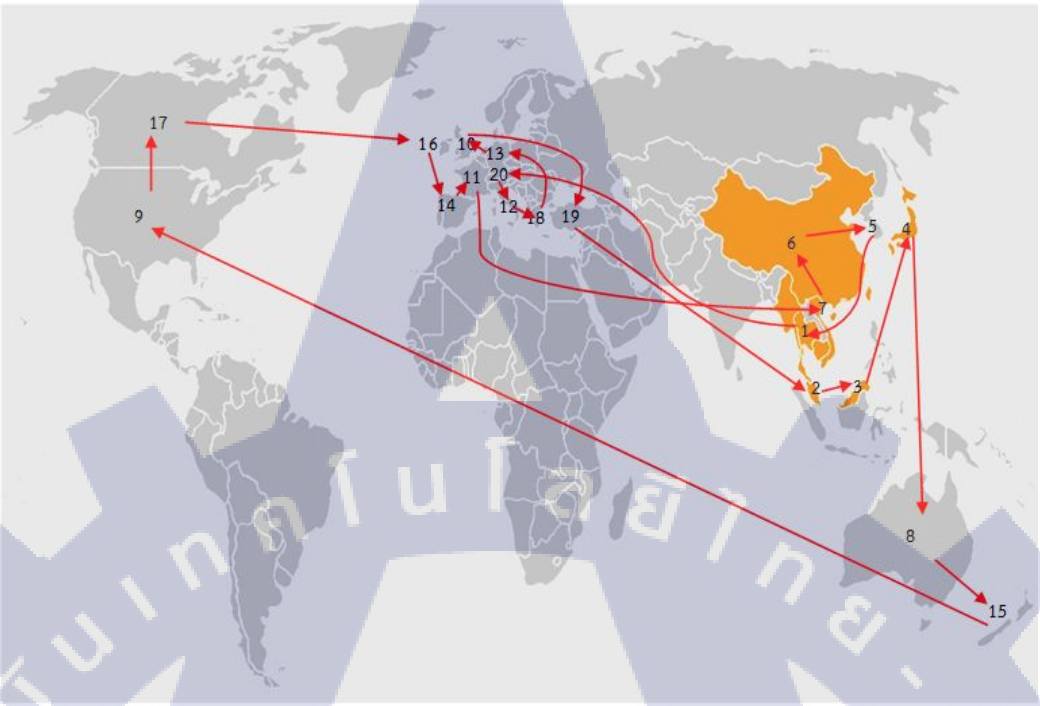
รูปที่ ง.4 ลำดับของการเดินทางที่มีต้นทุนต่ำที่สุดในปัญหากรณีที่ 2 โดยวิธี LS-INSERT



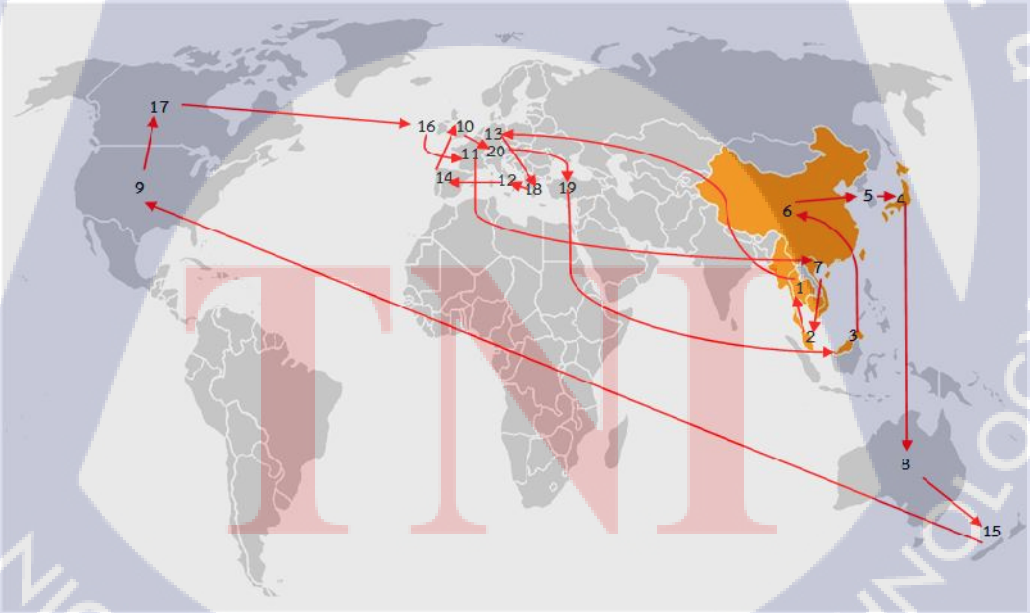
รูปที่ ง.5 ลำดับของการเดินทางที่มีต้นทุนต่ำที่สุดในปัญหากรณีที่ 3 โดยวิธี LS-SWAP



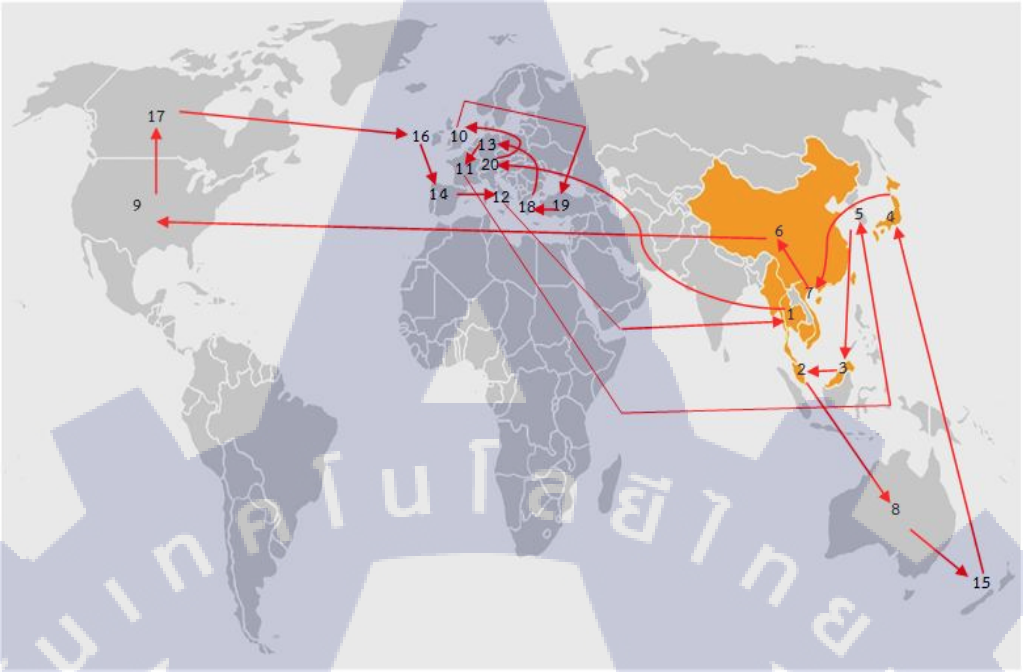
รูปที่ ง.6 ลำดับของการเดินทางที่มีต้นทุนต่ำที่สุดในปัญหากรณีที่ 3 โดยวิธี LS-INSERT



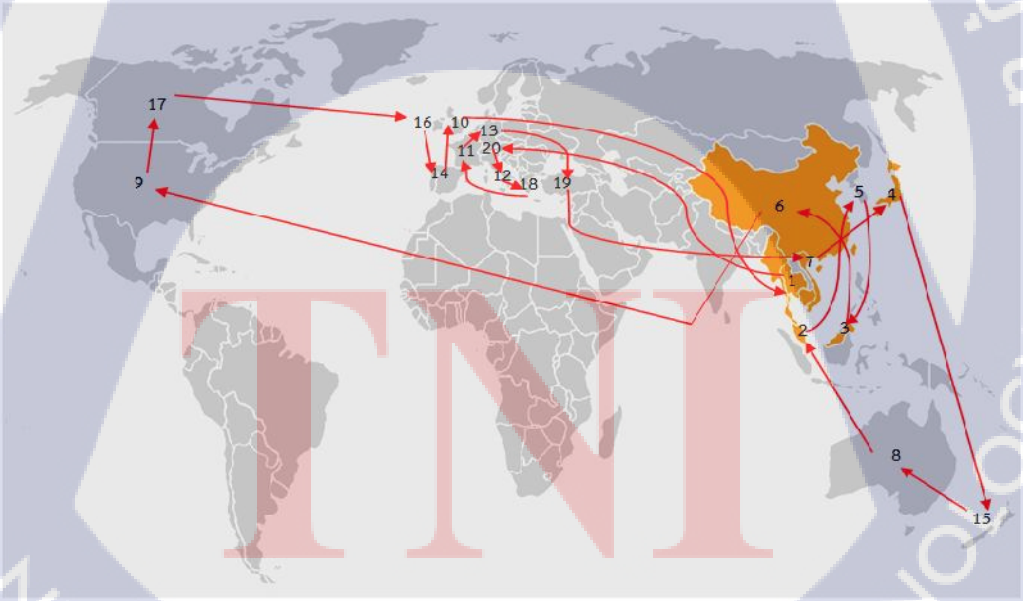
รูปที่ ง.7 ลำดับของการเดินทางที่มีต้นทุนต่ำที่สุดในปัญหากรณีที่ 4 โดยวิธี LS-SWAP



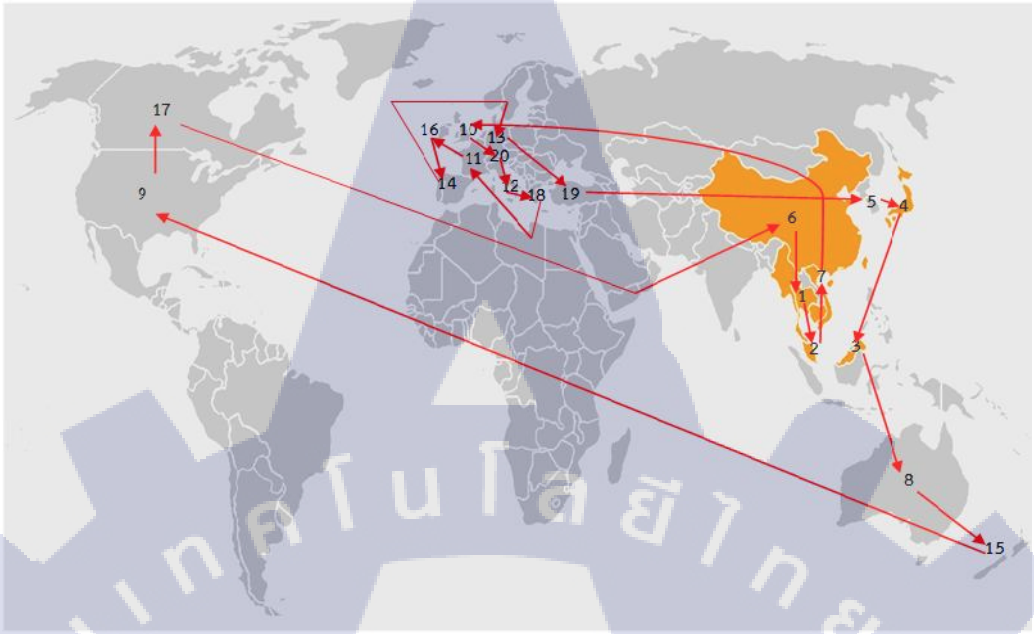
รูปที่ ง.8 ลำดับของการเดินทางที่มีต้นทุนต่ำที่สุดในปัญหากรณีที่ 4 โดยวิธี LS-INSERT



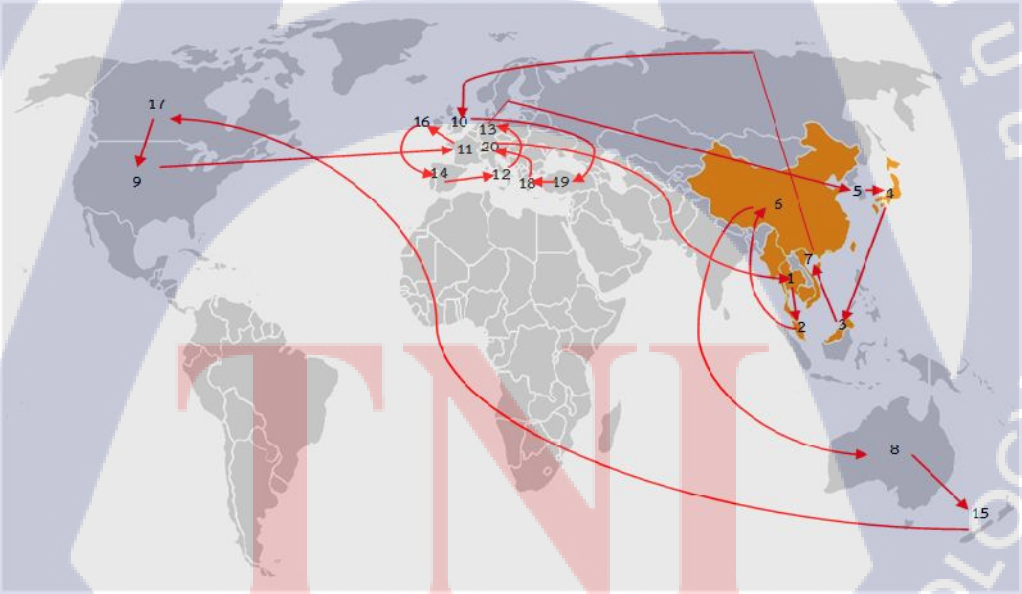
รูปที่ ง.9 ลำดับของการเดินทางที่มีต้นทุนต่ำที่สุดในปัญหากรณีที่ 5 โดยวิธี LS-SWAP



รูปที่ ง.10 ลำดับของการเดินทางที่มีต้นทุนต่ำที่สุดในปัญหากรณีที่ 5 โดยวิธี LS-INSERT



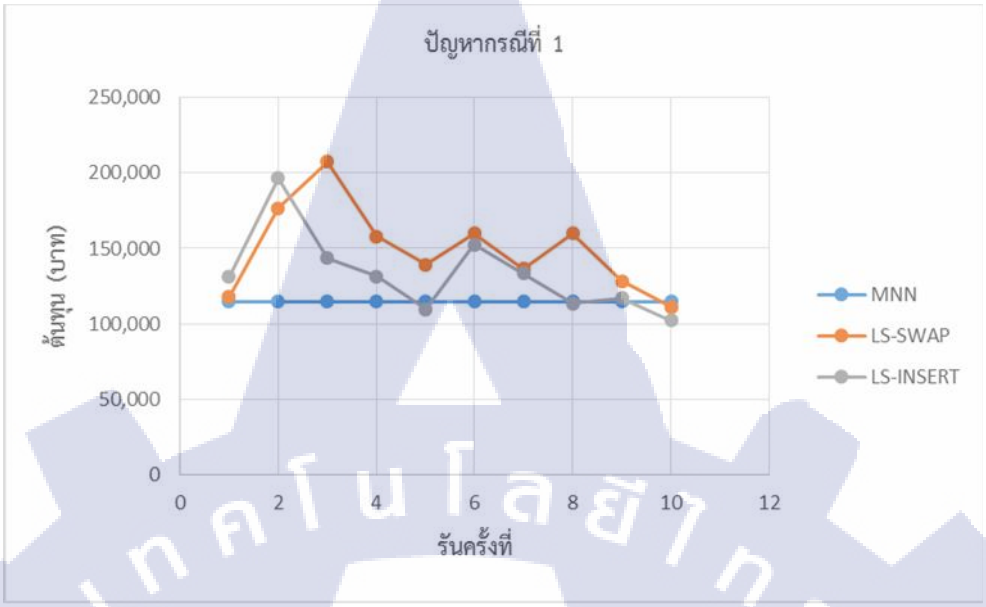
รูปที่ ง.11 ลำดับของการเดินทางที่มีต้นทุนต่ำที่สุดในปัญหากรณีที่ 6 โดยวิธี LS-SWAP



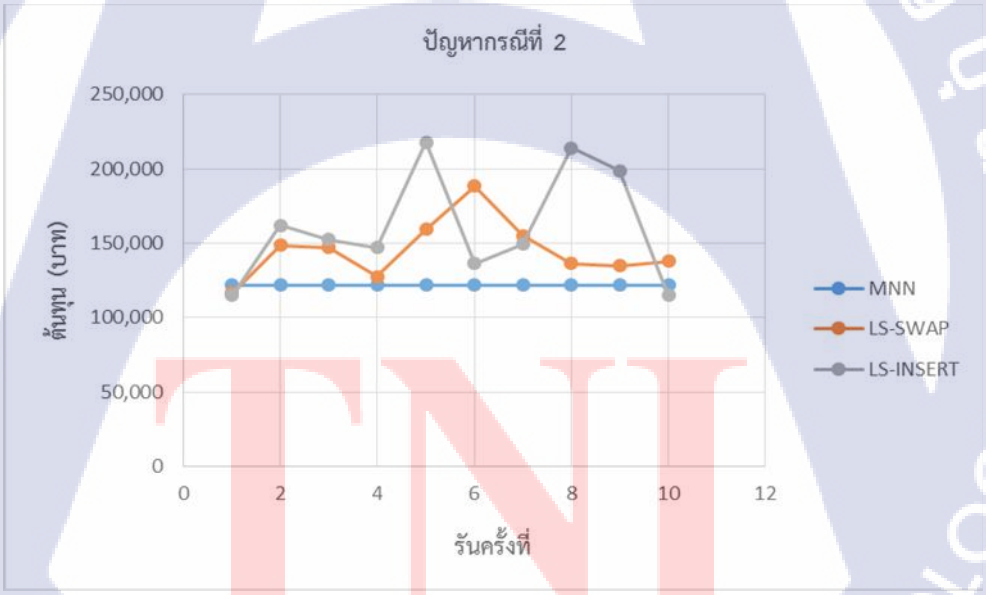
รูปที่ ง.12 ลำดับของการเดินทางที่มีต้นทุนต่ำที่สุดในปัญหากรณีที่ 6 โดยวิธี LS-INSERT

ภาคผนวก จ.

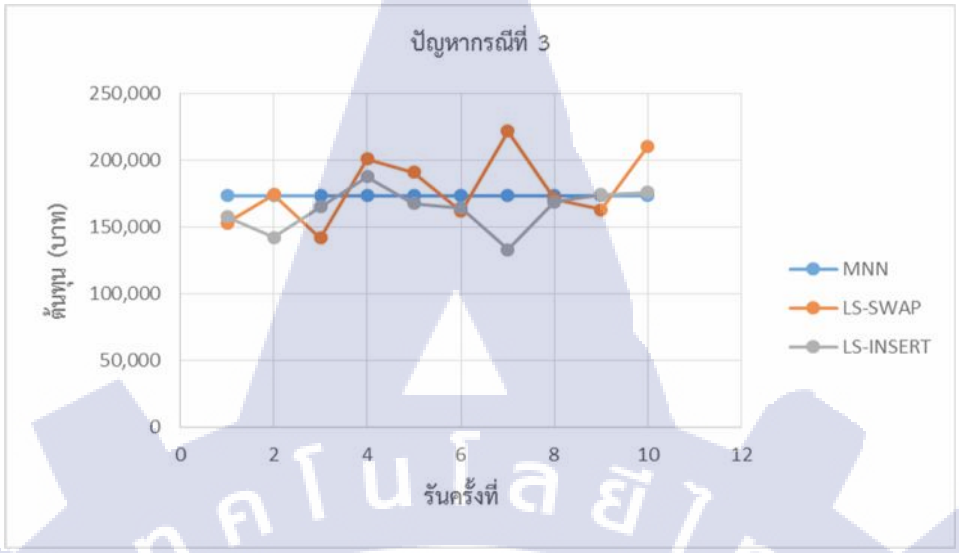
แผนภาพการกระจายค่าคำตอบในปัญหา 6 กรณี



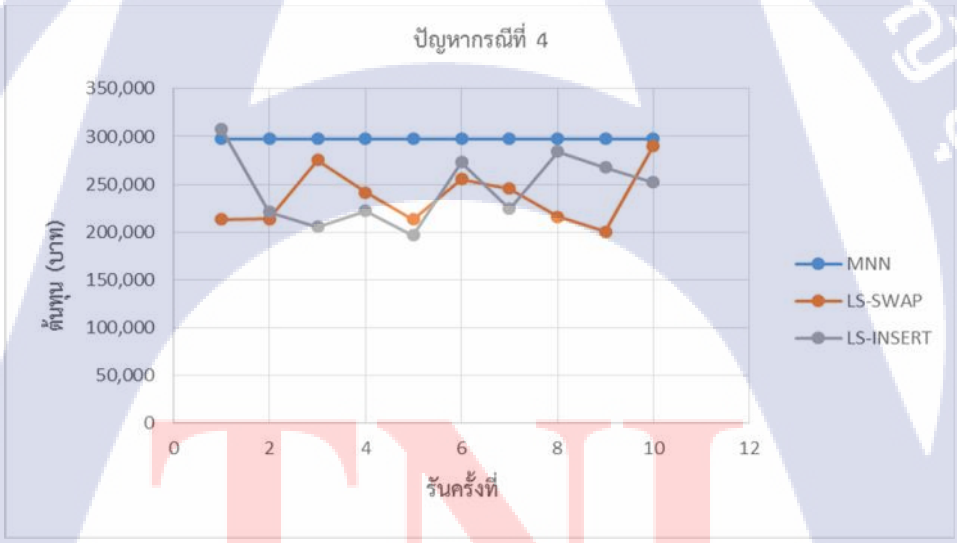
รูปที่ จ.1 แผนภาพการกระจายค่าคำตอบของปัญหากรณีที่ 1



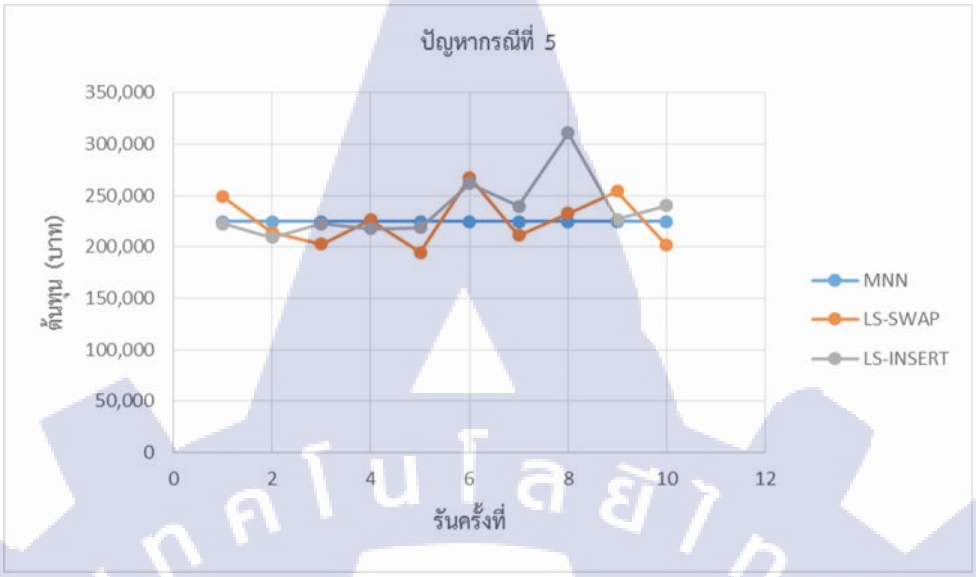
รูปที่ จ.2 แผนภาพการกระจายค่าคำตอบของปัญหากรณีที่ 2



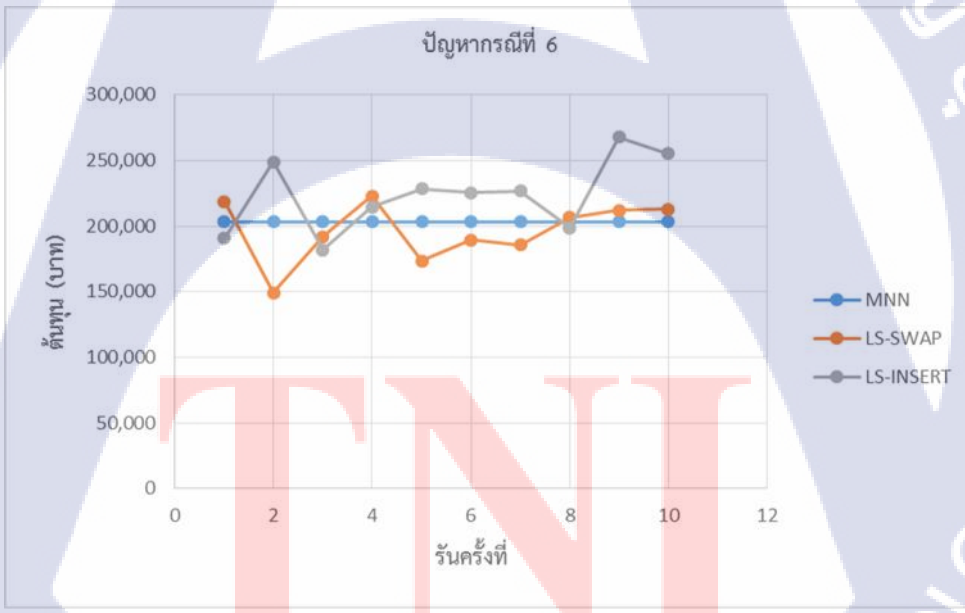
รูปที่ จ.3 แผนภาพการกระจายค่าคำตอบของปัญหากรณีที่ 3



รูปที่ จ.4 แผนภาพการกระจายค่าคำตอบของปัญหากรณีที่ 4



รูปที่ จ.5 แผนภาพการกระจายค่าคำตอบของปัญหากรณีที่ 5



รูปที่ จ.6 แผนภาพการกระจายค่าคำตอบของปัญหากรณีที่ 6