Greedy Randomized Adaptive Search

Greedy Randomized Adaptive Search Procedure (GRASP) เป็นฮิวริสติกในกลุ่มของเมตะฮิวริสติก (Meta-Heuristic) สามารถให้คำตอบของปัญหาได้ ดีกว่าฮิวริสติกแบบดั้งเดิม เนื่องจากมีกลไกที่ช่วยการหลบหลีกการติดต่ออยู่ที่ Local Optimal ได้ ดังนั้นจึงเหมาะที่จะนำมาใช้เป็นวิธีการหาคำตอบในการจัดเส้นทางเดินรถขนส่งสินค้า ซึ่งวิธีการ ฮิวริสติก GRASP นี้ได้รับความนิยมนำมาใช้ในการแก้ไขปัญหาการหาค่าที่เหมาะสมเชิงการจัด (Combinatorial Optimization) และประสบความสำเร็จในการนำมาประยุกต์ใช้งานทางด้าน วิศวกรรมอุตสาหการ สามารถให้คำตอบที่ดีโดยใช้เวลาในการคำนวณที่ต่ำ และมีการทำงานที่ไม่ซับซ้อน ดังนั้นจึงเหมาะที่จะนำไปใช้ในการแก้ไขปัญหาในรูปแบบต่างๆ ได้สะดวกรวมถึง การใช้หน่วยความจำที่ไม่สูงมากนักเมื่อเทียบกับฮิวริสติกชนิดอื่นๆ ในกลุ่มของเมตะฮิวริสติก (Blum and Rori, 2003 ; Chaovalitwongse, et al., 2003 ; Lee, 2005) ซึ่งจากคุณสมบัติ ดังกล่าวส่งผลให้วิธีการฮิวริสติก GRASP มีความน่าสนใจในการนำมาประยุกต์ใช้ในกระบวนการ จัดเส้นทางเดินรถขนส่งสินค้าซึ่งสามารถสร้างสมการตามลักษณะของปัญหา เช่น ฟังก์ชั่น วัตถุประสงค์ เงื่อนไขของเวลา และขนาดความจุบรรทุกของรถได้ดังนี้

หาค่าต่ำสุดของ ได้อธิบายไว้ต่อจากด้านล่าง

$$MinZ = \sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=1}^{K} C_{ij} X_{ij}^{k} + \sum_{k=1}^{K} \lambda \left| \sum_{j=0}^{N}\sum_{i=0}^{N} X_{ij}^{k}(q_i - d_i) \right| \qquad (2\text{-}27)$$

ภายใต้เงื่อนไข

$$\sum_{i=0}^{N}\sum_{k=1}^{K} x_{ij}^{k} = 1 \qquad \forall_j \in (1,\ldots\ldots,N) \qquad (2\text{-}28)$$

$$\sum_{j=0}^{N}\sum_{k=1}^{K} x_{ij}^{k} = 1 \qquad \forall_i \in (1,\ldots\ldots,N) \qquad (2\text{-}29)$$

$$\sum_{i=0}^{N} X_{ip}^{k} - \sum_{j=0}^{N} X_{pj}^{k} = 0 \qquad \forall_p \in (1,\ldots\ldots,N), k \in (1,\ldots\ldots,K) \qquad (2\text{-}30)$$

$$\sum_{i=0}^{N} q_i \left(\sum_{j=0}^{N} X_{ij}^{k} \right) \le Q \qquad \forall_k \in (1,\ldots\ldots,K) \qquad (2\text{-}31)$$

$$\sum_{j=1}^{N} X_{0j}^{k} \le 1 \qquad \forall_k \in (1,\ldots\ldots,K) \qquad (2\text{-}32)$$

$$\sum_{i=1}^{N} X_{i0}^{k} \le 1 \qquad \forall_k \in (1,\ldots\ldots,K) \qquad (2\text{-}33)$$

$$\lambda = 3 \qquad \sum\sum X_{ij}^{k}(q_i - d_i) < 0 \quad for \quad all \quad K \qquad (2\text{-}34)$$

$$\lambda = 0.00374 \quad R_k \qquad \sum\sum X_{ij}^{k}(q_i - d_i) > 0 \quad for \quad all \quad K \qquad (2\text{-}35)$$

$$x_{ij}^{k} \in (0,1) \qquad \forall_{ij} \in (1,\ldots\ldots,N), k \in (1,\ldots\ldots,K) \qquad (2\text{-}36)$$

โดยที่

Z = Vehicle routing cost + Penalty cost

K = จำนวนรถทั้งหมด

N = จำนวนโนดทั้งหมด

Q = ขนาดความจุบรรทุกของรถ

R_k = ระยะทางรวมของรถคันที่ K

λ = ค่าใช้จ่ายที่เกิดจากการบรรทุกสินค้าขาดหรือเกินความต้องการ

d_i = ความต้องการสินค้าที่โนด i ซึ่งมีความไม่แน่นอนในแต่ละวันและ $d_0 = 0$

C_{ij} = ค่าใช้จ่ายในการเดินทางระหว่างโนด i และโนด j ซึ่งพิจารณาเฉพาะค่าใช้จ่ายจากระยะทางรวม โดยไม่รวมถึงค่าใช้จ่ายอื่นๆ

q_i = ตัวแปรตัดสินใจว่าปริมาณสินค้าของโนด i ที่บรรทุกบนรถและ $q_0 = 0$

$$X_{ij}^{k} = \begin{cases} 1 & \text{ถ้ารถคันที่ } K \text{ ขนสินค้าระหว่างลูกค้า } i \text{ ไปยังลูกค้า } j \\ 0 & \text{ในกรณีอื่นๆ} \end{cases}$$

สมการที่ (2-27) คือฟังก์ชันวัตถุประสงค์หลักเพื่อต้องการให้ค่าใช้จ่ายในการเดินทางต่ำที่สุด สมการที่ (2-28) และ (2-29) เป็นสมการที่ประกันว่าลูกค้าแต่ละรายรับบริการจากยานพาหนะเพียงคันเดียว สมการที่ (2-30) ประกันว่าเมื่อยานพาหนะเข้ามายังจุดส่งสินค้าแล้วยานพาหนะจะออกจากจุดส่งสินค้านั้น

สมการที่ (2-31) ยานพาหนะขนส่งสินค้าทุกคันสามารถบรรทุกสินค้าได้ไม่เกินข้อจำกัด สมการที่ (2-32) และ (2-33) ประกันว่ายานพาหนะขนส่งแต่ละคันถูกใช้ได้เพียงเส้นทางใดเส้นทางหนึ่งเท่านั้น สมการที่ (2-34) และ (2-35) ค่าใช้จ่ายที่เกิดขึ้นในการขนส่งสินค้าที่ขาดหรือเกินความต้องการ สมการที่ (2-36) แสดงการเชื่อมโยงกันระหว่างโนด I และโนด j โดยเป็นได้สองอย่างคือได้รับการเชื่อมโยงกับไม่ได้รับการเชื่อมโยง ขั้นตอนการจัดเส้นทางเพื่อหาคำตอบด้วยวิธีการฮิวริสติก GRASP มีองค์ประกอบในการทำงานแต่ละรอบแบ่งออกเป็น 2 ระยะคือ ระยะที่หนึ่งการสร้างเส้นทางเบื้องต้น (Construction Phase) และระยะที่สองการปรับปรุงคำตอบจากการสร้างเส้นทางเบื้องต้น (Local Search Phase) โดยคำตอบแต่ละรอบที่ได้จะแตกต่างกันออกไปเนื่องจากมีกระบวนการสุ่มเกิดขึ้นใน ขั้นตอนการสร้างเส้นทาง โดยในการทำงานสองขั้นตอนหลักนี้จะกระทำซ้ำแล้วเปรียบเทียบคำตอบที่ได้ในแต่ละรอบกับคำตอบที่ดีที่สุด (Best Found Solution) หากได้คำตอบที่ดีกว่าก็จะปรับค่าของคำตอบที่ดีที่สุดเป็นค่าของคำตอบที่หาได้ในรอบนั้นๆ จนกว่าจะพบเงื่อนไขการหยุดโดยคำตอบสุดท้ายที่ได้ก็คือเส้นทางที่ถูกเก็บอยู่ในคำตอบที่ดีที่สุด ซึ่งการทำงานของฮิวริสติก GRASP มีขั้นตอนดังนี้

ขั้นตอนที่ 1 ข้อมูลที่นำเข้าประกอบด้วย จำนวนลูกค้า ขนาดความจุบรรทุกของรถ ระยะทาง และปริมาณความต้องการของลูกค้า

ขั้นตอนที่ 2 กำหนดให้ค่าคงที่เท่ากับ 1 เพื่อเริ่มกระบวนการวิ่งวนซ้ำหาคำตอบ

ขั้นตอนที่ 3 สร้างคำตอบเบื้องต้นด้วยกระบวนการการสุ่มเลือกลูกค้าที่อยู่ในบัญชีรายชื่อลูกค้า (Restricted Candidate List: RCL) ที่ยังไม่ถูกจัดเข้ากลุ่มโดยข้อมูลนำเข้าที่เป็นไปตามเงื่อนไข

ขั้นตอนที่ 4 ปรับปรุงคำตอบจากผลลัพธ์ของคำตอบเบื้องต้นด้วยวิธีการฮิวริสติกเช่น การสลับตำแหน่ง และหรือวิธีการย้ายตำแหน่งลูกค้าเป็นต้น

ขั้นตอนที่ 5 ปรับปรุงคำตอบและให้เพิ่มค่าคงที่ (K = K +1) ถ้า K < M (จำนวนรอบของการหาคำตอบ) ให้ไปที่ขั้นตอนที่ 3 มิฉะนั้นแล้วไปที่ขั้นตอนที่ 6

ขั้นตอนที่ 6 คืนค่าคำตอบ

2.5.1 ขั้นตอนการสร้างเส้นทางเบื้องต้น (Construction Phase)

กระบวนการวิ่งวนซ้ำเพื่อสร้างเส้นทาง จะพิจารณาพื้นที่ของคำตอบที่เป็นไปได้ที่ไม่ขัดแย้งกับเงื่อนไข(Feasible Solution) การเลือกเส้นทาง และปรับปรุงเส้นทางของลูกค้าเมื่อพบคำตอบที่ดีในแต่ละรอบการวิ่งวนซ้ำ ซึ่งจะพิจารณาลูกค้าทั้งหมดจากบัญชีรายชื่อลูกค้า (RCL) มีขั้นตอนดังนี้

ขั้นตอนที่ 1 เลือกลูกค้าด้วยวิธีการสุ่มโดยพิจารณาถึงต้นทุนต่างๆ ซึ่งอาจหมายถึงระยะทางของลูกค้าแต่ละรายที่อยู่ในบัญชีรายชื่อลูกค้า

ขั้นตอนที่ 2 ถ้าคำตอบยังไม่สมบูรณ์ทำการสร้างบัญชีรายชื่อลูกค้าใหม่โดยตัดลูกค้าที่ถูกเลือกออกไป

ขั้นตอนที่ 3 สุ่มเลือกลูกค้าที่เหลือจากบัญชีรายชื่อลูกค้าเข้าในเส้นทางจนครบทุกราย

ขั้นตอนที่ 4 คืนค่าคำตอบ

2.5.2 ขั้นตอนการปรับปรุงคำตอบ (Local Search Phase)

ขั้นตอนนี้จะนำเส้นทางเบื้องต้นมาพัฒนาคำตอบให้ดียิ่งขึ้น โดยความหมายของการปรับปรุงคำตอบในระบบการจัดเส้นทางเดินรถขนส่งสินค้าจะหมายถึงการได้ระยะทางรวมที่สั้นที่สุด และมีระยะเวลาในการรอคอยก่อนที่จะส่งสินค้าเพิ่มมากขึ้นโดยการเปรียบเทียบผลการจัดเส้นทางที่ได้ในแต่ละรอบจะพิจารณาจากระยะทางรวมเป็นอันดับแรก ซึ่งก็เป็นวัตถุประสงค์หลักของการวิจัย โดยที่เส้นทางขนส่งที่มีระยะทางรวมที่ต่ำกว่าถือว่าเป็นเส้นทางที่ดีกว่า สำหรับในกรณีที่ระยะทางรวมในการขนส่งเท่ากันอาจจะพิจารณาเปรียบเทียบคุณสมบัติทางด้านอื่นๆ เช่น ระยะเวลาในการรอคอยก่อนส่งสินค้า ถ้าเส้นทางที่มีระยะเวลาในการรอคอยก่อนส่งสินค้านานกว่าจะถือว่าเป็นเส้นทางที่ดีกว่า เพราะจะทำให้มีเวลาในการรอคอยข้อมูลของสินค้าที่อาจจะเข้ามาในระบบได้มากขึ้นซึ่งมีขั้นตอนดังนี้

ขั้นตอนที่ 1 ถ้าผลของคำตอบยังไม่เหมาะสมที่สุดให้ทำขั้นตอนที่ 2

ขั้นตอนที่ 2 ค้นหาคำตอบที่เหมาะสมที่สุดจากเซตของคำตอบเริ่มต้น

ขั้นตอนที่ 3 ได้คำตอบที่เหมาะสมที่สุด

ขั้นตอนที่ 4 คืนค่าคำตอบ

จากที่ผ่านมานักวิจัยหลายท่านที่นำวิธีการฮิวริสติก GRASP มาประยุกต์ใช้งานเพื่อการแก้ไขปัญหาการจัดเส้นทางเดินรถขนส่งสินค้ามาก

งานวิจัยที่เกี่ยวข้อง

เนื่องจากปัญหาการจัดเส้นทางเดินรถ (Vehicle Routing Problem: VRP) ในอดีตที่ผ่านมา ได้รับความสนใจและมีการศึกษากันอย่างกว้างขวาง รวมถึงมีความซับซ้อนกับปัญหาที่พบในสถานะการณ์จริง การพัฒนาวิธีการและแนวทางการแก้ไขปัญหามีหลากหลายโดยมีจุดเริ่มต้นจากบทความของ Dantzig and Ramser ด้วยเหตุที่ปัญหานี้ได้รับความสนใจเป็นอย่างมากเนื่องจากปัญหาการจัดเส้นทางเดินรถนั้นเป็นปัญหาที่พบได้บ่อยในชีวิตประจำวัน และปัญหาการจัดเส้นทางเดินรถเป็นปัญหาที่มีความสนใจในเชิงทฤษฎี และก็ไม่ง่ายในการที่จะหาคำตอบ ซึ่งตัวปัญหาจะถูกแบ่งออกเป็น 2 กลุ่ม คือ กลุ่มของปัญหาที่มีลักษณะเป็นแบบ Deterministic หมายถึงปัญหาที่ต้องทราบข้อมูลที่จำเป็นเบื้องต้นก่อนค่อยเริ่มทำการจัดเส้นทาง ส่วนกลุ่มของปัญหาที่มีลักษณะเป็นแบบ Dynamic (Lasen, 2000) จะเป็นปัญหาที่ไม่ทราบข้อมูลเบื้องต้นก่อนทำการการจัดเส้นทาง แต่จะมีข้อมูลทยอยออกมาในระหว่างที่ทำการจัดเส้นทาง โดยปัญหาพื้นฐานของการจัดเส้นทางเดินรถที่รู้จักและมีการวิจัยกันมากที่สุดคือ ปัญหาการเดินทางของพนักงานขาย (The Traveling Salesman Problem: TSP) ซึ่งมี จุดเริ่มต้นในช่วงทศวรรษที่ 1920 โดยนักคณิตศาสตร์และนักเศรษฐศาสตร์ที่มีชื่อว่า Karl Menger จากนั้นก็ได้รับความนิยมอย่างแพร่หลายโดยนักคณิตศาสตร์ชื่อ Merill Flood ต่อมาในปี 1954 George Dantzig Ray Fulkerson และ Selmer Johnson ได้เสนอวิธีการหาคำตอบในการจัดเส้นทาง ซึ่งสามารถจัดการกับปัญหาการจัดเส้นทางในการเดินทางซึ่งมีเมืองที่ต้องเดินผ่านทั้งหมด 49 เมืองได้ จากนั้นก็มีการศึกษาอย่างต่อเนื่องจนกระทั่งในช่วงปี 2004 Applegate Bixby Chvatal Cook และ Helsgaun สามารถหาเส้นทางที่เหมาะสมในการเดินทางผ่านเมืองทั้งหมด 24,978 เมืองในประเทศสวีเดนได้ สำหรับปัญหาการจัดเส้นทางการเดินรถ (VRP) นั้นถือได้ว่าเป็นปัญหาการเดินทางของพนังานขาย (TSP) จะมียานพาหนะที่ใช้เดินทางมากกว่าหนึ่งคันโดยตัวปัญหาจะถูกแบ่งออกเป็นประเภทย่อยๆ อีกหลายประเภทตามลักษณะเฉพาะของตัวปัญหา ในระยะแรกนั้นจะมีการศึกษาตัวปัญหาที่มีลักษณะเป็นแบบ Deterministic โดยเริ่มจากปัญหาการจัดเส้นทางเดินรถพื้นฐานซึ่งตัวปัญหาจะเป็นการจัดเส้นทางเดินรถให้ผ่านไปยังจุดรับให้ครบทุกจุดด้วยค่าใช้จ่ายในการเดินทางต่ำที่สุด และเมื่อมีข้อจำกัดในเรื่องขนาดความจุของยานพาหนะที่ใช้ขนส่ง จึงถูกเรียกว่า Capacitated VRP (CVRP) (Kara, et al., 2004) กรณีถ้ามีท่าจอดรถมากกว่าหนึ่งแห่งจะถูกจัดเป็นปัญหาเรียกว่า Multiple Depot VRP (MDVRP) ส่วนปัญหาที่มีการไปส่งและรับ คืนสินค้าจะถูกจัดเป็นปัญหาเรียกว่า Vehicle Routing Problem with Pick-up and Delivering

(VRPPD) (Lu and Dessouky, 2004) และถ้ามีข้อจำกัดเรื่องกรอบระยะเวลาในการขนส่งปัญหานี้เรียกว่า VRP with Time Windows (VRPTW) และที่ผ่านมาก็มีผลงานวิจัยนำเสนออีกมากมาย ดังนี้

ตารางผลงานวิจัยที่เกี่ยวกับการจัดเส้นทางเดินรถ

ผู้ดำเนินการวิจัย	ลักษณะปัญหา	วิธีการและผลการวิจัย
Clarke and Wright (1964)	เป็นการจัดเส้นทางการเดินรถที่มีหลายขนาด โดยส่งสินค้าออกจากศูนย์กระจายสินค้าเพียงแห่งเดียว	**ใช้วิธีการฮิวริสติก เพื่อการจัดเรียงลำดับของค่าประหยัดได้ (Saving) และเชื่อมเส้นทางต่างๆ เข้าด้วยกัน ทำให้ทราบจำนวนรถบรรทุกที่ต้องใช้และปริมาณสินค้าแต่ละคัน**
Christofides and Eilon (1969)	นำเสนอวิธีปรับปรุงภายในเส้นทางภายหลังจากได้เส้นทาง	นำเสนอการปรับปรุงเส้นทางด้วยการแลกเปลี่ยนเส้นทางเพื่อให้ได้ระยะทางใหม่ที่เกิดขึ้นมีค่าน้อยกว่าค่าเดิม โดยวิธีการเริ่มต้นจากการสมมติเส้นทางเริ่มต้นขึ้นมาแล้วปรับปรุงจนได้เส้นทางที่ดีที่สุด
Holmes and Parker (1976)	นำเสนอการจัดสรรเส้นทางด้วยการหาค่าการประหยัดในอีกรูปแบบหนึ่ง ซึ่งแตกต่างจาก Clarke Wright	นำเสนอวิธีการแบบ Parallel ในการจัดเส้นทางโดยกำหนดให้โครงข่ายที่ใช้เป็นแบบไม่สมมาตร
Bartholdi, et al. (1983)	นำเสนอการจัดเส้นทางและลำดับการส่งอาหารแก่ผู้สูงอายุในมลรัฐจอร์เจีย ซึ่งมีรถขนส่งประมาณ 4 คันและต้องส่งอาหารให้แก่ผู้สูงอายุจำนวน 200 ราย ในช่วงเวลา 10.00-14.00 น. ให้ครบทุกคน	ได้ใช้เทคนิค Travelling salesman problem โดยกำหนดพิกัด (x, y) เพื่อหาระยะทางที่ห่างกันของแต่ละจุด และใช้เทคนิคการสร้างกราฟที่เรียกว่า Space filling-curve เพื่อศึกษารายละเอียดต่างๆ ของลูกค้า แล้วจึงแบ่งเส้นทางออกเป็นสาย แต่วิธีการดังกล่าวไม่สามารถจะแก้ไขปัญหาได้รวดเร็ว

		เมื่อมีข้อมูลและข้อจำกัดจำนวนมากได้
Bell, et al. (1983)	กล่าวถึงการขนส่งสินค้าประเภทก๊าซและเคมีภัณฑ์ให้กับลูกค้าต่างๆ ซึ่งต้องมีการคาดคะเนระดับของสินค้าของลูกค้าต่างๆ และกำหนดเวลาในการส่ง	แบ่งปัญหาออกเป็น 2 ส่วนคือ - การวิเคราะห์ปริมาณสินค้าที่จะต้องเติมให้กับลูกค้าแต่ละราย - การวิเคราะห์เส้นทางเพื่อให้ต้นทุนในการขนส่งต่ำที่สุด รวมทั้งมีจำนวนการใช้ยวดยานให้น้อยที่สุด Bell, et al. เลือกใช้วิธีการ Set Covering ในการวิเคราะห์ปัญหาเนื่องจากลูกค้าสามารถให้รถมากกว่าหนึ่งคันมาส่งสินค้าและระยะเวลาในการปฏิบัติงานยาวนานกว่า ประมาณ 5 วัน ดังนั้นลูกค้าสามารถเลื่อนส่งสินค้าได้ โดยแบ่งลำดับขั้นตอนในการทำงานแบ่งเป็น 2 ขั้นตอน คือ 1. Route Generator เป็นขั้นตอนของการเลือกเส้นทางที่เป็นไปได้ในแต่ละเส้นทางก่อน โดยมีการระบุลูกค้าต่างๆ พร้อมแสดงต้นทุนและค่าใช้จ่าย 2. Route Optimizer เป็นขั้นตอนของการเลือกเส้นทางที่เหมาะสมเพื่อให้ต้นทุนต่ำสุด โดยเส้นทางที่เลือกต้องสอดคล้องกับความต้องการและเงื่อนไขการวิเคราะห์ปัญหานี้มีความซับซ้อนมาก เพราะมีจำนวนลูกค้ามาก โดยเฉพาะอย่างยิ่งมีการกำหนดเวลาในการส่งสินค้า

Belardo, Duchessi and Seagle (1985)	กล่าวถึงแนวทางในการจัดเส้นทางขนส่งโดยรถบรรทุกไปยังร้านสะดวกซื้อของบริษัท Southland Corporation	ใช้คอมพิวเตอร์ช่วยสนับสนุนการตัดสินใจ (Decision Support System,DSS) แสดงข้อมูลเส้นทางในการเดินรถด้วยภาพกราฟิกรวมทั้งข้อมูลในเชิงโต้ตอบเพื่อให้ผู้เกี่ยวข้องสามารถวิเคราะห์เส้นทางเพื่อให้ได้ข้อมูลลำดับการส่งไปยังลูกค้าต่างๆ
Evans and Norback (1985)	กล่าวถึงการจัดเส้นทางการเดินรถของสินค้าอุปโภคของบริษัท Kraft ซึ่งมีเงื่อนไขของเวลาในการจัดส่งเนื่องจากลักษณะการสั่งสินค้าไม่คงที่ระยะเวลาในการส่งสินค้าน้อยกว่า 1 วัน และมีข้อจำกัดของน้ำหนักและปริมาตรของรถส่งสินค้า และเวลาในการทำงาน	ใช้แนวทางของ DSS แสดงผลลัพธ์ผ่านทางกราฟฟิกและการโต้ตอบและสร้างฐานข้อมูลของน้ำหนักปริมาตรลูกค้า โดยใช้วิธีการจัดเส้นทางที่เรียกว่า Largest angle method และใช้ระยะทางโดยประมาณจากวิธีเส้นตรง (Euclidean Distance) ซึ่งผลลัพธ์จากการจัดเส้นทาง สามารถลดค่าใช้จ่ายประมาณร้อยละ 10.7
Savelbergh (1985)	เสนอวิธีการกระจายสินค้าแบบใหม่โดยพิจารณาทั้งข้อจำกัดของระยะทางในการขนส่ง และข้อจำกัดด้านเวลาในการเดินทาง	แก้ปัญหาการจัดเส้นทางของการเดินรถโดยพิจารณากรอบของเวลาไปพร้อมกันโดยในเบื้องต้นจะสร้างเส้นทางขึ้นมาก่อนหลังจากนั้นจึงปรับปรุงเส้นทางด้วยวิธี k-interchange
Nag, Golden and Assad (1988)	เสนอการจัดเส้นทางโดยมีการใช้รถหลายขนาด และมีเงื่อนไขของเขตการส่งที่ส่งผลต่อการเลือกใช้ประเภทรถ	ใช้วิธีการสร้างเส้นทางจาก First-cut algorithm เพื่อเลือกรถที่เหมาะสมในช่วงแรก หลังจากนั้นจึงใช้เทคนิคการจัดเส้นทาง
Jongkol (1990)	เสนอการจัดเส้นทางเดินรถขนส่งน้ำมันโดยในเทคนิค Matching ของค่าใช้จ่ายซึ่งเป็นเวลาและ	เสนอการจัดเส้นทางด้วยการหาค่าประหยัดจากศูนย์กระจายสินค้าเดียวไปยังสถานีบริการที่กระจาย

	ระยะทาง	อยู่ในกรุงเทพฯ โดยมีขนาดของรถที่แตกต่างกัน โดยแบ่งขั้นตอนเป็น 2 ส่วน คือ 1. การประมาณเส้นทางที่สั้นสุด 2. การจัดกลุ่มลูกค้าที่สามารถส่งสินค้าได้
Klibbua (1990)	กล่าวถึงการแก้ไขปัญหาการจัดเส้นทางในการจัดส่งของโรงงานผลิตอาหารประเภทนม และไอศกรีมในจังหวัดเชียงใหม่ โดยมีเงื่อนไขที่สำคัญคืออายุสินค้าสั้น โดยแบ่งพื้นที่ความรับผิดชอบเป็น 4 พื้นที่เพื่อรองรับกับรถส่งนมและไอศกรีม 4 คัน	งานวิจัยนี้ได้แบ่งเป็น 2 ส่วนคือ ส่วนแรกเป็นการออกแบบการทำงานของคลัง (Warehouse) และขนาดพื้นที่คลังส่วนที่สองเป็นการวางแผนการขนส่งสินค้า ซึ่งกำหนดให้เส้นทางเป็นเส้นทางคงที่ตามความรับผิดชอบ ซึ่งเทคนิคที่ผู้วิจัยใช้ในการจัดเส้นทางคือ 2-opt โดยใช้ระยะทางเป็นฟังก์ชันวัตถุประสงค์ด้วยการสร้างเมตริกซ์ ระยะทางไปยังลูกค้าต่างๆ จากการหาเส้นทางที่สั้นที่สุด (Shortest Path Algorithm)ที่เรียกว่า Dijiktra’s Algorithm
Martin (1998)	เสนอแนวทางของการปรับปรุงประสิทธิภาพของการกระจายสินค้าประเภทเบเกอรี่โดยมีข้อกำหนดคือ ช่วงเวลาในการส่งสินค้า	ใช้วิธีการสร้างเส้นทางในการเดินรถโดยใช้ระยะเวลาเป็นฟังก์ชันวัตถุประสงค์และใช้เทคนิค Nearest-neighbor heuristics โดยหาระยะเวลาในการเดินทางระหว่างลูกค้าจากข้อมูลระยะทางในการเดินทาง และความเร็วเฉลี่ยด้วยการสร้างตารางความสัมพันธ์ระหว่างระยะทางการเดินทางและเวลาเดินทางรถและหาเวลาในการ

		นำสินค้าลงด้วยการสร้างกราฟความสัมพันธ์ของจำนวนสินค้าและเวลาที่ใช้ในการนำสินค้าลงรถ
Lee and Ueng (1999)	เสนอวิธีการจัดเส้นทางของศูนย์กระจายสินค้าเพียงแห่งเดียวด้วยการคำนึงระยะทางในการเดินทางที่สั้นที่สุด และการจัดสรรปริมาณสินค้าให้รถขนส่งอย่างยุติธรรม	นำเสนอการจัดสรรเส้นทางด้วยการหาค่าการประหยัดพร้อมใช้ค่าพารามิเตอร์ปรับแก้ค่าการประหยัดในแบบจำลองเพื่อจัดสรรสินค้า โดยให้ความสำคัญต่อความยุติธรรมในการจัดสรรงานให้พนักงาน
Weigel and Cao 1999)	กล่าวถึงบริษัท Sears Logistics services (SLS) ซึ่งเป็นบริษัทขนส่งสินค้าส่งถึงบ้าน เช่น สินค้าเฟอร์นิเจอร์ และเครื่องใช้ไฟฟ้า โดยต้องการให้ระบบการจัดเส้นทางสามารถ - ส่งสินค้าถึงมือลูกค้าในเวลาที่เหมาะสม - ลดค่าใช้จ่ายในการขนส่งสินค้า - ช่วยให้พนักงานขับรถพึงพอใจกับงาน	ออกแบบระบบสำหรับปัญหาการเดินรถภายใต้ข้อจำกัดของเวลาและความจุของรถ (VRPTW) โดยใช้เวลาซึ่งประกอบด้วยเวลาในการเดินทาง เวลาคอย และเวลาที่ล่าช้า เป็นฟังก์ชันวัตถุประสงค์ โดยใช้ GIS ArcInfo ของ บริษัท ESRI ซึ่งสามารถแสดงรายละเอียดของถนนสายต่างๆ และสร้าง อัลกอริทึมแบบ Cluster first route second.และหา OD Matrix ซึ่งให้ผู้ใช้งานส่งผ่านข้อมูลทางระบบ WAN ทำให้ค่าใช้จ่ายในการขนส่งสินค้าลดลงปีละ 42 ล้านดอลล่าร์

ดังนั้นปัญหาของการจัดเส้นทางเดินรถเป็นปัญหาที่เหมาะสมที่สุดเชิงการจัด ซึ่งมีการศึกษาอย่างแพร่หลายปัญหาในลักษณะต่างๆ อีกหลายรูปแบบ ตัวปัญหามีความซับซ้อนในระดับของเอ็นพีฮาร์ด (Non-polynomial Hard: NP-Hard) ซึ่งมีแนวทางที่ใช้สำหรับการแก้ไขปัญหาอยู่ 2 ประเภทใหญ่ๆ คือ วิธีการหาคำตอบที่ดีที่สุด (Exact Optimization) และการหาคำตอบด้วยวิธีการแบบฮิวริสติก (Heuristic Optimization) ซึ่งมีทั้งฮิวริสติกที่มีการพัฒนาแบบดั้งเดิม และฮิวริ

สติกที่มีการพัฒนาในยุคใหม่ที่เรียกว่า เมตะฮิวริสติก (Meta-Heuristic) ที่สามารถแก้ปัญหาการติดอยู่ใน Local Optimal ได้สำหรับแนวทางการหาคำตอบด้วยวิธีการแบบฮิวริสติกนั้น แม้จะไม่ได้รับประกันว่าจะให้คำตอบที่เหมาะสมที่สุด แต่ก็สามารถให้คำตอบที่น่าพอใจ และใช้เวลาในการคำนวณสมเหตุสมผลจึงทำให้เหมาะที่จะนำมาใช้เป็นแนวทางในการพัฒนาปัญหาการจัดเส้นทางการขนส่งสินค้าที่เหมาะสมในระบบมิลค์รัน เพื่อที่สามารถจัดการกับปัญหาได้ทัน

2.8.3 Procedure

Algorithm ของ Greedy Randomized Adaptive Search Procedure

Algorithm 2.8.1: Pseudocode for the GRASP.

```
Input: α
Output: S_best
1 S_best ← ConstructRandomSolution();
2 while ¬ StopCondition() do
3 |   S_candidate ← GreedyRandomizedConstruction(α);
4 |   S_candidate ← LocalSearch(S_candidate);
5 |   if Cost(S_candidate) < Cost(S_best) then
6 |   |   S_best ← S_candidate;
7 |   end
8 end
9 return S_best;
```

อธิบายขั้นตอนวิธีการทำงานของ GRASP

เป็นฟังก์ชั่นที่เกี่ยวกับการสร้างคำตอบและหาคำตอบที่ใกล้เคียงที่ดีที่สุด โดยใช้ stochastically สำหรับสร้างฟังก์ชั้นการทำงานโดยการสร้างใน RCL

สมมุติให้ ($\alpha \in [0, 1]$) ใช้เพิ่มต้นทุนในแต่ล่ะปัญหาของคำตอบที่ดีที่สุดในปัจจุบัน

Algorithm 2.8.2: Pseudocode the `GreedyRandomizedConstruction` function.

```
Input: α
Output: S_candidate
S_candidate ← ∅;
while S_candidate ≠ ProblemSize do
    Feature_costs ← ∅;
    for Feature_i ∉ S_candidate do
        Feature_costs ←
        CostOfAddingFeatureToSolution(S_candidate, Feature_i);
    end
    RCL ← ∅;
    Fcost_min ← MinCost(Feature_costs);
    Fcost_max ← MaxCost(Feature_costs);
    for F_icost ∈ Feature_costs do
        if F_icost ≤ Fcost_min + α · (Fcost_max − Fcost_min) then
            RCL ← Feature_i;
        end
    end
    S_candidate ← SelectRandomFeature(RCL);
end
return S_candidate;
```

2.8.4 การหาค่า Heuristics

1. α กำหนดให้ค่าของ greediness ของการสร้างขั้นตอนวิธีที่มีค่าใกล้ 0 ซึ่งอาจจเกินค่า Greedy และค่าใกล้ 1 อาจจะเป็นค่าทั่วไป
2. เกณฑ์ที่เป็นทางเลือกโดยให้ α ซึ่ง RCL สามารถที่จะสร้างเป็น n% ของคำตอบที่ใกล้เคียงดีที่สุด และอาจจะเป็นการเลือกในแต่ล่ะรอบการทำงาน
3. เทคนิคที่ถูกออบแบบสำหรับการเรียนรู้ปัญหา เช่นปัญหาการเพิ่มประสิทธิภาพ combinatorial

2.8.5 Code Listing

โค้ดการทำงานของ Greedy Randomized Adaptive Search Procedure สร้างด้วยภาษา Ruby The เป็นการนำอัลกอริทึมที่ใช้กับเมืองเ Berlin52 ซึ่งเป็นปัญหาของการเดินทางของ Salesman Problem (TSP), เดินทางจาก TSPLIB เป็นปัญหาของการสั่งซื้อและเดินทางไปทั่งเมืองซึ่งช่วยลดระยะทางรวมในการเดินทางทั้งหมด เช่น 7542 หน่วย ขั้นตอนการสร้างที่ฉลาดและมีการคำนวณค่าใช้จ่ายในการเดินทาง ดังอัลกอริทึมต่อไปนี้

ตัวอย่างโปรแกรมจากภาษา Ruby

```
def euc_2d(c1, c2)
  Math.sqrt((c1[0] - c2[0])**2.0 + (c1[1] - c2[1])**2.0).round
end
def cost(perm, cities)
  distance =0
  perm.each_with_index do |c1, i|
    c2 = (i==perm.size-1) ? perm[0] : perm[i+1]
    distance += euc_2d(cities[c1], cities[c2])
  end
  return distance
end

def stochastic_two_opt(permutation)
  perm = Array.new(permutation)
  c1, c2 = rand(perm.size), rand(perm.size)
  exclude = [c1]
  exclude << ((c1==0) ? perm.size-1 : c1-1)
  exclude << ((c1==perm.size-1) ? 0 : c1+1)
  c2 = rand(perm.size) while exclude.include?(c2)
  c1, c2 = c2, c1 if c2 < c1
  perm[c1...c2] = perm[c1...c2].reverse
  return perm
end

def local_search(best, cities, max_no_improv)
  count = 0
  begin
    candidate = {:vector=>stochastic_two_opt(best[:vector])}
    candidate[:cost] = cost(candidate[:vector], cities)
    count = (candidate[:cost] < best[:cost]) ? 0 : count+1
    best = candidate if candidate[:cost] < best[:cost]
```

```
  end until count >= max_no_improv
  return best
end

def construct_randomized_greedy_solution(cities, alpha)
  candidate = {}
  candidate[:vector] = [rand(cities.size)]
  allCities = Array.new(cities.size) {|i| i}
  while candidate[:vector].size < cities.size
    candidates = allCities - candidate[:vector]
    costs = Array.new(candidates.size) do |i|
      euc_2d(cities[candidate[:vector].last], cities[i])
    end
    rcl, max, min = [], costs.max, costs.min
    costs.each_with_index do |c,i|
      rcl << candidates[i] if c <= (min + alpha*(max-min))
    end
    candidate[:vector] << rcl[rand(rcl.size)]
  end
  candidate[:cost] = cost(candidate[:vector], cities)
  return candidate
end

def search(cities, max_iter, max_no_improv, alpha)
  best = nil
  max_iter.times do |iter|
    candidate = construct_randomized_greedy_solution(cities, alpha);
    candidate = local_search(candidate, cities, max_no_improv)
    best = candidate if best.nil? or candidate[:cost] < best[:cost]
    puts " > iteration #{(iter+1)}, best=#{best[:cost]}"
  end
  return best
```

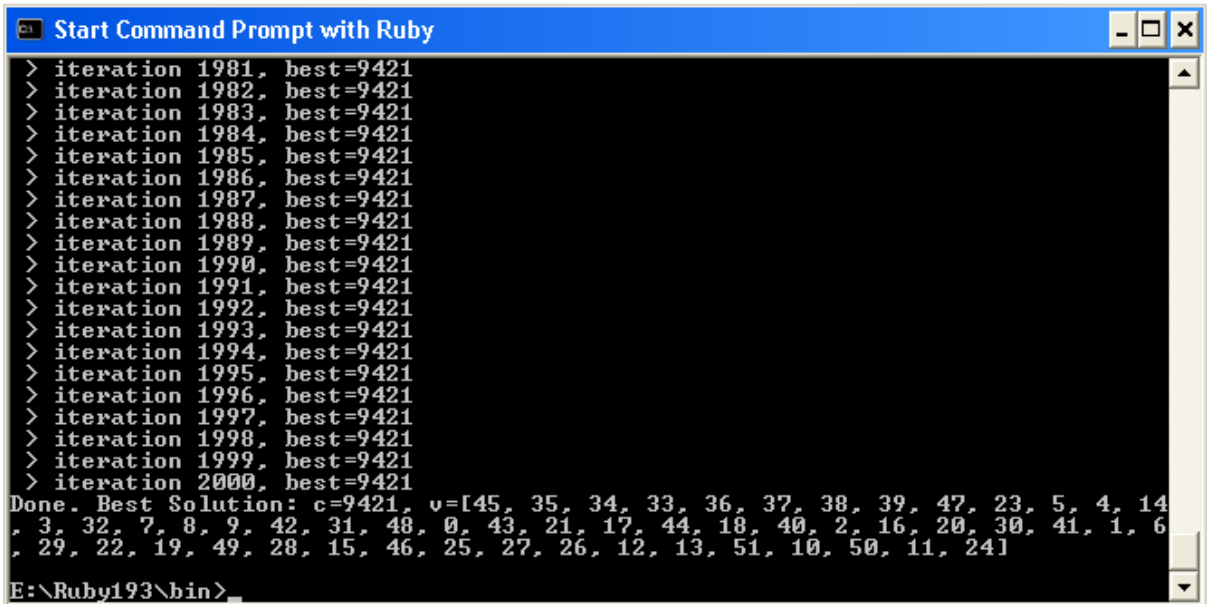
```
end

if __FILE__ == $0
  # problem configuration
  berlin52 = [[565,575],[25,185],[345,750],[945,685],[845,655],
   [880,660],[25,230],[525,1000],[580,1175],[650,1130],[1605,620],
   [1220,580],[1465,200],[1530,5],[845,680],[725,370],[145,665],
   [415,635],[510,875],[560,365],[300,465],[520,585],[480,415],
   [835,625],[975,580],[1215,245],[1320,315],[1250,400],[660,180],
   [410,250],[420,555],[575,665],[1150,1160],[700,580],[685,595],
   [685,610],[770,610],[795,645],[720,635],[760,650],[475,960],
   [95,260],[875,920],[700,500],[555,815],[830,485],[1170,65],
   [830,610],[605,625],[595,360],[1340,725],[1740,245]]

  # algorithm configuration
  max_iter = 2000  #จำนวนรอบการทำงาน
  max_no_improv = 50
  greediness_factor = 0.3

  # execute the algorithm
  best = search(berlin52, max_iter, max_no_improv, greediness_factor)
  puts "Done. Best Solution: c=#{best[:cost]}, v=#{best[:vector].inspect}"
end
```

ผลการรันโปรแกรม


Start Command Prompt with Ruby
> iteration 1981, best=9421
> iteration 1982, best=9421
> iteration 1983, best=9421
> iteration 1984, best=9421
> iteration 1985, best=9421
> iteration 1986, best=9421
> iteration 1987, best=9421
> iteration 1988, best=9421
> iteration 1989, best=9421
> iteration 1990, best=9421
> iteration 1991, best=9421
> iteration 1992, best=9421
> iteration 1993, best=9421
> iteration 1994, best=9421
> iteration 1995, best=9421
> iteration 1996, best=9421
> iteration 1997, best=9421
> iteration 1998, best=9421
> iteration 1999, best=9421
> iteration 2000, best=9421
Done. Best Solution: c=9421, v=[45, 35, 34, 33, 36, 37, 38, 39, 47, 23, 5, 4, 14
, 3, 32, 7, 8, 9, 42, 31, 48, 0, 43, 21, 17, 44, 18, 40, 2, 16, 20, 30, 41, 1, 6
, 29, 22, 19, 49, 28, 15, 46, 25, 27, 26, 12, 13, 51, 10, 50, 11, 24]
E:\Ruby193\bin>

Bibliography

[Bard1996] J. F. Bard and T. A. Feo and S. Holland, "A GRASP for scheduling printed wiring board assembly", I.I.E. Trans., 1996.

[Feo1989] T. A. Feo and M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem", Operations Research Letters, 1989.

[Feo1991] T. A. Feo and K. Venkatraman and J. F. Bard, "A GRASP for a difficult single machine scheduling problem", Computers & Operations Research, 1991.

[Feo1993] T. A. Feo and J. Bard and S. Holland, "A GRASP for scheduling printed wiring board assembly", Technical Report TX 78712-1063, Operations Research Group, Department of Mechanical Engineering, The University of Texas at Austin, 1993.

[Feo1994] T. A. Feo and K. Sarathy and J. McGahan, "A GRASP for single machine scheduling with sequence dependent setup costs and linear delay penalties", Technical Report TX 78712-1063, Operations Research Group, Department of Mechanical Engineering, The University of Texas at Austin, 1994.

[Feo1995] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures", Journal of Global Optimization, 1995.

[Feo1996] T. A. Feo and K. Sarathy and J. McGahan, "A grasp for single machine scheduling with sequence dependent setup costs and linear delay penalties", Computers & Operations Research, 1996.

[Festa2002] P. Festa and M. G. C. Resende, "GRASP: An annotated bibliography", in Essays and Surveys on Metaheuristics, pages 325–367, Kluwer Academic Publishers, 2002.

[Hart1987] J. P. Hart and A. W. Shogan, "Semi–greedy heuristics: An empirical study", Operations Research Letters, 1987.

[Pardalos1995] P. M. Pardalos and L. S. Pitsoulis and M. G. C. Resende, "A parallel GRASP implementation for the quadratic assignment problems", in Parallel Algorithms for Irregularly Structured Problems (Irregular'94), 1995.

[Pitsoulis2002] L. Pitsoulis and M. G. C. Resende, "Greedy randomized adaptive search procedures", in Handbook of Applied Optimization, pages 168–181, Oxford University Press, 2002.

[Prais2000] M. Prais and C. C. Ribeiro, "Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment", INFORMS Journal on Computing, 2000.

[Resende2003] M. G. C. Resende and C. C. Ribeiro, "Greedy randomized adaptive search procedures", in Handbook of Metaheuristics, pages 219–249, Kluwer Academic Publishers, 2003.