บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ปัญหาการรับและส่งสินค้า (The Pickup and Delivery Problem, PDP) เป็นปัญหารูปแบบหนึ่งของปัญหาการจัดเส้นทางเดินรถ (The Vehicle Routing Problem, VRP) ซึ่งจัดเป็นปัญหาการหาคำตอบที่เหมาะสมที่สุดเชิงการจัด (Combinatorial Optimization) และมีความซับซ้อนของการคำนวณในระดับเอ็นพีแบบยาก (NP-Hard) การค้นหาคำตอบเพื่อให้ได้คำตอบที่เหมาะสมที่สุด (Optimal Solution) จะใช้เวลานานในการค้นหาคำตอบ เมื่อขนาดของปัญหาใหญ่ขึ้น เวลาที่ใช้ในการหาคำตอบจะเพิ่มขึ้นอย่างเอ็กโพเนนเชียล (Blum and Roli, 2003) ทำให้ใช้เวลานานเกินไป ไม่เหมาะสมสำหรับปัญหาขนาดใหญ่ ดังนั้นจึงจำเป็นต้องอาศัยวิธีการหาคำตอบแบบฮิวริสติกในการแก้ปัญหา

เนื้อหาในบทนี้ ได้ทำการแบ่งเนื้อหาออกเป็น 5 ส่วน ได้แก่ รูปแบบของปัญหาการจัดเส้นทางเดินรถและงานวิจัยที่เกี่ยวข้อง การแทนรูปแบบปัญหาการจัดเส้นทางเดินรถ ขั้นตอนวิธีของไดสตรา วิธีการหาคำตอบ และบทสรุป

2.1 รูปแบบของปัญหาการจัดเส้นทางเดินรถและงานวิจัยที่เกี่ยวข้อง

ปัญหาการจัดเส้นทางเป็นปัญหาที่ได้รับการศึกษาอย่างมากในช่วงสามถึงสี่ทศวรรษที่ผ่านมา และเป็นงานวิจัยด้านการวิจัยดำเนินงาน (Operations Research) ซึ่งแตกแขนงรูปแบบของปัญหาไว้หลากหลายรูปแบบ โดยปัญหาการจัดเส้นทางที่มีการศึกษาและรู้จักกันอย่างกว้างขวางที่สุด คือ The Traveling Salesman Problem (TSP) ซึ่งเป็นรูปแบบของปัญหาการเดินทางของพนักงานขาย 1 คน ที่มีเงื่อนไขว่า พนักงานขายต้องการเดินทางไปขายของยังเมืองต่าง ๆ ทุกเมือง ซึ่งไปได้เพียงเมืองละ 1 ครั้ง แล้วต้องกลับมายังเมืองเดิมโดยเส้นทางที่เดินทางนี้ต้องมีค่าใช้จ่ายรวมในการเดินทางต่ำที่สุด

ปัญหาที่มีรูปแบบคล้ายกับปัญหา TSP คือ ปัญหาการจัดเส้นทางเดินรถ (The Vehicle Routing Problem, VRP) ซึ่งรูปแบบของปัญหาคือ การจัดเส้นทางให้รถ m คัน ซึ่งต้องเดินทางออกจากคลังสินค้า เพื่อไปส่งสินค้าให้กับลูกค้าในที่ต่างๆ รายละ 1 ครั้ง แล้วกลับมาที่คลังสินค้า (Depot) โดยที่ลูกค้าแต่ละรายมีความต้องการสินค้า (Demand) ในปริมาณที่แตกต่างกัน

นอกจากนี้รถแต่ละคันสามารถบรรทุกสินค้าได้ไม่เกินความจุของรถ โดยมีวัตถุประสงค์ของการจัดเส้นทางเพื่อให้ต้นทุนการขนส่งต่ำที่สุด ปัญหา VRP สามารถแบ่งออกได้เป็น 2 ประเภท ได้แก่ ประเภทที่หนึ่ง ปัญหาการจัดเส้นทางเดินรถที่มีความแน่นอน (Deterministic VRP) นั่นคือ ข้อมูลที่ต้องใช้ในการจัดเส้นทางนั้นมีค่าแน่นอน ไม่เปลี่ยนแปลง และทราบล่วงหน้าก่อนทำการจัดเส้นทาง เช่นลูกค้าที่ต้องการให้ไปส่งสินค้า ปริมาณของสินค้าที่ลูกค้าแต่ละรายต้องการ และต้นทุนการขนส่งระหว่างสถานที่ 2 สถานที่ใดๆ ประเภทที่สอง ปัญหาการจัดเส้นทางเดินรถที่ไม่มีความแน่นอน (Stochastic VRP, SVRP) ข้อมูลที่ใช้ในการจัดเส้นทางมีค่าไม่แน่นอน อาจเปลี่ยนแปลงได้ ซึ่งจะมีค่าความน่าจะเป็นเข้ามาเกี่ยวข้อง (Gendreau et al., 1996) เช่นในบางครั้งขณะที่รถกำลังออกไปส่งสินค้าให้กับลูกค้า อาจจะมีลูกค้ารายใหม่ต้องการสินค้ากะทันหันเกิดขึ้นได้ ทำให้รถต้องไปส่งสินค้าให้กับลูกค้ารายนี้ด้วย ปริมาณสินค้าที่ลูกค้าต้องการอาจเปลี่ยนแปลงได้ หรือระยะเวลาที่ใช้ในการเดินทางมีค่าไม่เท่ากันอันเนื่องมาจากการปัญหาการจราจร ซึ่งเป็นปัญหาจริงที่เกิดขึ้นในการขนส่งสินค้าในปัจจุบัน

ปัญหา VRP ได้รับการศึกษาและวิจัยมาเป็นระยะเวลานาน โดยปัญหาที่เก่าแก่ที่สุดคือปัญหา Classical VRP ซึ่งเป็นปัญหาการจัดเส้นทางเดินรถสำหรับส่งสินค้า โดยที่รถบรรทุกแต่ละคันมีความจุแตกต่างกันได้ Clarke and Wright (1964) ได้เสนอ Saving Algorithm สำหรับใช้แก้ปัญหาและได้รับความนิยมอย่างแพร่หลาย เนื่องจากสามารถแก้ปัญหาได้อย่างรวดเร็ว เหมาะกับปัญหาขนาดใหญ่ ซึ่งคำตอบที่ได้จะให้ค่าใกล้เคียงกับค่าที่เหมาะสมที่สุด นอกจากนี้ Gillett and Miller (1971) ยังได้ทำการเสนอ Sweep Algorithm เพื่อใช้แก้ปัญหา VRP เช่นกัน การแก้ปัญหาโดยใช้ Sweep Algorithm นั้นจะใช้การกวาดมุมเพื่อกำหนดลูกค้าให้กับเส้นทาง โดยที่เส้นทางการขนส่งสินค้าจะถูกกำหนดโดยลำดับของลูกค้าที่พบเมื่อทำการกวาด สำหรับปัญหา VRP ที่รถทุกคันที่ใช้ในการขนส่งสินค้ามีความจุเท่ากัน จะเรียกว่า The Capacitated VRP (CVRP)

ปัญหาอีกรูปแบบหนึ่งของปัญหา VRP ที่มีเวลาเข้ามาเกี่ยวข้อง คือ The Vehicle Routing Problem with Time Windows (VRPTW) ลูกค้าแต่ละรายจะมีการกำหนดช่วงเวลาที่สามารถรับบริการได้เอาไว้ ดังนั้นการจัดส่งสินค้าให้กับลูกค้านั้น ต้องจัดเส้นทางเพื่อกำหนดลำดับการส่งสินค้าให้กับลูกค้าโดยที่ลูกค้าต้องได้รับบริการทุกราย ถ้าหากรถไปถึงลูกค้าก่อนที่ลูกค้าจะสามารถรับบริการได้ รถต้องคอยจนกว่าลูกค้าจะเริ่มได้รับบริการ แต่รถไม่สามารถไปถึงลูกค้าหลังจากเวลาที่ลูกค้าสามารถรับบริการได้ Solomon (1987) ได้เสนอขั้นตอนวิธีเชิงสร้าง (Constructive Algorithm) เพื่อใช้แก้ปัญหา VRPTW ไว้ด้วยกันหลายแบบ โดยขั้นตอนวิธีที่ได้รับความนิยมอย่างแพร่หลายคือ Insertion Heuristic I1 เนื่องจากเป็นขั้นตอนวิธีที่มีประสิทธิภาพมาก

ในการแก้ปัญหา Fisher et al. (1997) ได้เสนอขั้นตอนวิธี 2 วิธีสำหรับแก้ปัญหา VRPTW ซึ่งทั้ง 2 ขั้นตอนวิธีนั้นเป็นขั้นตอนวิธีเหมาะสมที่สุด (Optimization Algorithms) ได้แก่ The Lagrangian Relaxation/Variable Splitting Approach และ The K-tree Approach นอกจากนี้ Fisher et al. ยังได้ทำการสร้างแบบจำลองทางคณิตศาสตร์ของปัญหาให้อยู่ในรูปแบบของการโปรแกรมเชิงจำนวนเต็มแบบผสมไว้ด้วย

ปัญหาการจัดเส้นทางเดินรถบางประเภทนั้น นอกจากจะต้องไปส่งสินค้าให้กับลูกค้าแล้ว ยังต้องเก็บสินค้าจากลูกค้ากลับเข้าคลังสินค้าด้วย ตัวอย่างเช่น การขนส่งเครื่องดื่มที่บรรจุในขวดแก้ว ซึ่งจะเห็นว่า นอกจากรถต้องส่งขวดเครื่องดื่มให้กับลูกค้าแล้ว ยังต้องเก็บขวดเปล่ากลับมายังคลังสินค้าด้วย ปัญหาในลักษณะนี้เป็นปัญหาการรับและส่งสินค้ารูปแบบหนึ่ง เรียกว่า The Vehicle Routing Problem with Pickup and Delivery (VRPPD) โดยพิจารณาสำหรับสินค้าที่ต้องไปรับและส่งนั้นเป็นสินค้าประเภทเดียวกัน ซึ่งเป็นการส่งสินค้าจากคลังสินค้าไปสู่ลูกค้า และรับสินค้าจากลูกค้ากลับสู่คลังสินค้า Nagy and Salhi (2005) ได้แบ่งปัญหานี้ ออกเป็น 3 แบบ แบบแรกคือ Delivery-First, Pickup-Second VRPPD โดยลูกค้าที่ต้องไปส่งสินค้าให้ เรียกว่า Linehauls ส่วนลูกค้าที่ต้องไปเก็บสินค้าคืน เรียกว่า Backhauls ลักษณะของปัญหาแบบนี้คือ จะทำการส่งสินค้าให้กับ Linehauls ทั้งหมดก่อน จากนั้นจึงจะไปเก็บสินค้ากลับคืนจาก Backhauls ปัญหาแบบนี้เกิดขึ้นเนื่องจากมีความความยุ่งยากในการจัดเรียงสินค้า เราสามารถเรียกปัญหานี้อีกชื่อหนึ่งได้ว่า The Vehicle Routing Problem with Backhauls (VRPB) แบบที่สองคือ Mixed Pickups and Deliveries รูปแบบของปัญหาคือ Linehaul และ Backhaul สามารถรับหรือส่งสินค้าในลำดับใดของเส้นทางก็ได้ แบบสุดท้ายคือ Simultaneous Pickups and Deliveries นั่นคือลูกค้ารายหนึ่งสามารถเป็นได้ทั้ง Linehaul และ Backhaul นั่นคือ มีทั้งการรับและส่งสินค้าให้กับลูกค้ารายเดียวกัน นอกจากนี้ Nagy and Salhi ยังได้เสนอฮิวริสติกสำหรับใช้แก้ปัญหา โดยฮิวริสติกนี้จะแบ่งออกเป็น 2 ส่วน ส่วนแรกคือฮิวริสติกสำหรับทำการสร้างเส้นทาง ส่วนที่สองคือการปรับปรุงเส้นทาง ซึ่งได้เสนอวิธีการปรับปรุงเส้นทางไว้หลากหลายรูปแบบ และได้แนะนำลำดับของวิธีการปรับปรุงเส้นทางที่จะทำให้ได้คำตอบที่มีคุณภาพไว้

ปัญหาการจัดเส้นทางเดินรถที่เกี่ยวข้องกับการรับและส่งสินค้า โดยที่ต้องรับสินค้าจากที่หนึ่งเพื่อนำไปส่งยังอีกที่หนึ่ง เรียกว่า The Pickup and Delivery Problem (PDP) จุดที่ต้องไปรับสินค้าเรียกว่าต้นทาง จุดที่ต้องไปส่งสินค้าเรียกว่าปลายทาง โดยที่ต้นทางและปลายทางของสินค้าทุกชิ้นนั้น เป็นจุดที่อยู่ต่างที่กันทั้งหมด ปัญหาหนึ่งที่จัดได้ว่าเป็นปัญหาการรับและส่งสินค้าคือ The Dial-A-Ride Problem (DARP) ปัญหา DARP คือปัญหาการจัดเส้นทางเพื่อรับและ

ส่งสินค้า โดยที่ตัวสินค้านั้นมีโหลดเท่ากับหนึ่ง โดยทั่วไปแล้วปัญหา DARP จะใช้เป็นแบบจำลองสำหรับปัญหาการรับและส่งคนจากที่หนึ่งไปยังอีกที่หนึ่ง โดยมีเงื่อนไขเฉพาะคือ เวลาที่ใช้ในการขับขี่ (Ride Time) เพื่อนำแต่ละคนที่รับจากต้นทางไปส่งยังปลายทางนั้น ต้องไม่เกินเวลาขับขี่สูงสุดที่กำหนดไว้ Cordeau and Laporte (2003) ได้ทำการวิจัยและเสนอแก้ปัญหา DARP โดยใช้ Tabu Search ผู้วิจัยได้แนะนำถึงวิธีการสร้างคำตอบข้างเคียง การสร้างคำตอบเริ่มต้น และวิธีการประเมินคำตอบข้างเคียงเอาไว้ Psaraftis (1983) ได้เสนอขั้นตอนวิธีเหมาะสมที่สุดในการแก้ปัญหา DARP โดยใช้การโปรแกรมพลวัต (Dynamic Programming) ซึ่งจะทำให้ได้คำตอบที่เหมาะสมที่สุด พร้อมทั้งได้อธิบายถึงลักษณะของปัญหาอาจทำให้เกิดคำตอบที่เป็นไปไม่ได้ (Infeasible Solution) Xiang et al. (2005) ได้นำเสนอฮิวริสติกเพื่อใช้สำหรับแก้ปัญหา DARP ซึ่งมีความรวดเร็วเหมาะสำหรับปัญหาที่มีขนาดใหญ่และมีความซับซ้อน ซึ่งฮิวริสติกที่ใช้แก้ปัญหานั้น จะทำการปรับปรุงเส้นทางโดยใช้ยุทธวิธีการหาคำตอบที่เป็น Local Optima ก่อน เมื่อพบแล้วจะใช้ยุทธวิธี Diversification เพื่อหาคำตอบที่เป็น Global Optima ต่อไป

นักวิจัยหลายๆ ท่าน ได้เขียนบทความแนะนำปัญหา PDP อย่างกว้างๆ ซึ่งเป็นประโยชน์ต่อผู้ที่เพิ่งจะเริ่มต้นทำการศึกษา Savelsbergh and Sol (1995) ได้ทำการอธิบายลักษณะปัญหา PDP เช่น ลักษณะของการขนส่ง กรอบเวลา ฟังก์ชันวัตถุประสงค์ พร้อมทั้งสำรวจปัญหา PDP ในรูปแบบต่างๆ เช่น ปัญหาการรับและส่งสินค้าโดยใช้รถหนึ่งคันโดยไม่มีกรอบเวลา ปัญหาการรับและส่งสินค้าโดยใช้รถหนึ่งคันโดยมีกรอบเวลา ปัญหาการรับและส่งสินค้าโดยรถหลายคันโดยไม่มีกรอบเวลา และปัญหาการรับและส่งสินค้าโดยใช้รถหลายคันโดยมีกรอบเวลา รวมถึงสำรวจวิธีการแก้ปัญหาโดยใช้ทั้งขั้นตอนวิธีเหมาะสมที่สุดและขั้นตอนวิธีฮิวริสติก Xu et al. (2003) ได้เสนอการแก้ปัญหา PDP โดยสร้างรูปแบบปัญหาให้สอดคล้องกับปัญหาที่เกิดขึ้นจริงในทางปฏิบัติ พร้อมทั้งเสนอวิธีการแก้ปัญหาโดยใช้ Column Generation

ปัญหา PDP ได้รับการศึกษาอย่างแพร่หลายและแตกแขนงปัญหาออกไปได้อย่างหลากหลาย Lu and Dessouky (2004) ได้ทำการศึกษาปัญหา The Multiple Vehicle Pickup and Delivery Problem (MVPDP) และแก้ปัญหานี้โดยใช้ขั้นตอนวิธีเหมาะสมที่สุดโดยจะทำการสร้างแบบจำลองทางคณิตศาสตร์ให้อยู่ในรูป 0-1 Integer Programming ก่อน จากนั้นจึงใช้ Branch-and-Cut Algorithm เพื่อหาคำตอบที่เหมาะสมที่สุด

สำหรับปัญหา The Pickup and Delivery Problem with Time Windows (PDPTW) เป็นปัญหา PDP ที่มีเวลาเข้ามาเกี่ยวข้อง Nanry and Barnes (2000) เสนอฮิวริสติกสำหรับแก้ปัญหาโดยใช้

Reactive Tabu Search และได้เสนอโครงสร้างการย้ายคำตอบไปยังคำตอบข้างเคียงโดยใช้ Single Paired Insertion (SPI) Swapping Pairs Between Routes (SBR) และ Within Route Insertion (WRI) พร้อมทั้งแนะนำโครงสร้างข้อมูล (Data Structure) ที่ใช้สำหรับสร้าง Tabu List วิธีการกำหนด Tabu Attributes และการควบคุม Tabu Length วัตถุประสงค์สำหรับการแก้ปัญหาคือหาระยะเวลาที่ใช้ในการเดินทางที่น้อยที่สุด Lau and Liang (2002) ได้อธิบายขั้นตอนวิธีที่จะใช้ในการย้ายคำตอบไปยังคำตอบข้างเคียงซึ่งเป็นการขยายความเพิ่มจากงานวิจัยของ Nanry and Barnes ซึ่งบอกเพียงโครงสร้างที่ใช้ในการย้ายเท่านั้น วัตถุประสงค์สำหรับการแก้ปัญหานั้นเป็นเช่นเดียวกับวัตถุประสงค์ของ Nanry and Barnes คือหาระยะเวลาที่ใช้ในการเดินทางที่น้อยที่สุด Li and Lim (2001) เสนอเมตาฮิวริสติกสำหรับแก้ปัญหา PDPTW โดยใช้ Tabu Search ร่วมกับ Simulated Annealing การแก้ปัญหามีวัตถุประสงค์หลายวัตถุประสงค์โดยเรียงตามความสำคัญคือ 1.เพื่อหาจำนวนรถที่ใช้น้อยที่สุด 2.เพื่อใช้ระยะทางให้น้อยที่สุด 3. เพื่อใช้เวลาในการเดินทางน้อยที่สุด และ 4.เพื่อใช้เวลาคอยให้น้อยที่สุด

ในบางครั้งปัญหาการจัดเส้นทางเดินรถ ข้อมูลของปัญหาบางอย่างอาจไม่มีความแน่นอน เรียกปัญหาลักษณะนี้ว่า Stochastic VRP ข้อมูลของปัญหาที่อาจไม่มีความแน่นอนเกิดขึ้น ได้แก่ ลูกค้ามีความไม่แน่นอน ปริมาณสินค้าที่ลูกค้าต้องการไม่แน่นอน หรือเวลาที่ใช้ในการเดินทางไม่แน่นอน

The Vehicle Routing Problem with Stochastic Customers (VRPSC) คือปัญหาการจัดเส้นทางเดินรถโดยที่ตัวลูกค้าไม่มีความแน่นอน ตัวอย่างสถานการณ์ของปัญหาในลักษณะนี้ ได้แก่ บริษัทแห่งหนึ่งต้องการทำการสร้างเส้นทางสำหรับขนส่งสินค้าให้กับลูกค้า n ราย โดยที่เส้นทางที่สร้างนั้นต้องมีต้นทุนการขนส่งต่ำที่สุด อย่างไรก็ตามเส้นทางที่สร้างขึ้นนั้น จะถูกนำไปใช้ในช่วงระยะเวลาหนึ่งซึ่งมากกว่าหนึ่งวัน ในแต่ละวันนั้นลูกค้าบางรายอาจจะไม่ได้ต้องการสินค้าจากทางบริษัท ทำให้ลูกค้าที่ทางบริษัทต้องไปส่งสินค้าในแต่ละวันนั้นมีความหลากหลาย ยิ่งกว่านั้นบริษัทไม่สามารถที่จะทำการสร้างเส้นทางใหม่ที่จะไปส่งสินค้าให้กับลูกค้าได้ทุกวัน อันเนื่องมาจากค่าใช้จ่ายในการคำนวณสร้างเส้นทางนั้นแพง ประกอบกับทางบริษัทไม่สามารถทราบข้อมูลล่วงหน้าได้ว่าลูกค้ารายใดต้องการให้ไปส่งสินค้าวันไหนบ้าง บริษัทจึงต้องการออกแบบเส้นทางที่สร้างขึ้นมาเพียงหนึ่งครั้งสำหรับใช้ในช่วงระยะเวลาหนึ่ง ซึ่งถึงแม้ว่าในแต่ละวันลูกค้าที่บริษัทต้องไปส่งสินค้าให้นั้นจะแตกต่างกัน แต่เส้นทางที่สร้างขึ้นนั้น ก็ยังคงทำให้เกิดค่าใช้จ่ายต่ำที่สุด สำหรับปัญหานี้ ลูกค้าแต่ละรายจะถูกกำหนดให้มีอยู่ในปัญหาด้วยค่าความน่าจะเป็นค่าหนึ่ง เมื่อแทนปัญหาให้อยู่ในรูปแบบของกราฟแล้ว ลูกค้าจะถูกแทนด้วยจุด

ยอด (Vertex) ซึ่งจุดยอดที่มีอยู่เสมอตลอดช่วงระยะเวลาที่สนใจ จะมีค่าความน่าจะเป็น $p_i = 1$ เรียกจุดยอดนี้ว่า *black vertices* ส่วนจุดยอดที่ $0 < p_i < 1$ เรียกว่า *white vertices* สำหรับปัญหา SVRP ที่เกิดความไม่แน่นอนของปริมาณสินค้าที่ลูกค้าต้องการ เรียกว่า The Vehicle Routing Problem with Stochastic Demands (VRPSD) ส่วนปัญหา SVRP ที่เกิดความไม่แน่นอนของระยะเวลาเดินทาง เรียกว่า The Vehicle Routing Problem with Travel Times (Gendreau et al. 1996; Jaillet 1988)

สำหรับงานวิจัยนี้ จะทำการวิจัยปัญหาการรับและส่งสินค้าอีกรูปแบบหนึ่งซึ่งมีรูปแบบที่แตกต่างไปจากปัญหาการรับและส่งสินค้าจากที่ได้เคยทำการวิจัยกันมาก่อนหน้านี้ ปัญหาการรับและส่งสินค้าในงานวิจัยนี้เป็นการพิจารณาการรับและส่งสินค้าโดยที่สินค้าจะถูกรับที่ต้นทางและต้องถูกนำไปส่งถึงปลายทางให้ทันภายในระยะเวลารับประกัน ลักษณะการรับและส่งสินค้าของปัญหานี้คือ โหนดแต่ละโหนดมีสินค้าจำนวนมากที่ต้องทำการขนส่งไปยังปลายทาง สินค้าแต่ละชิ้นมีเวลาพร้อมส่งต่างกัน ดังนั้นสินค้าที่ออกจากโหนดเดียวกันจึงอาจไม่สามารถถูกขนส่งได้ภายในรอบเดียวกันทั้งหมด เพราะอาจทำให้สินค้าบางชิ้นไม่สามารถถูกส่งถึงปลายทางภายในระยะเวลารับประกัน ด้วยเหตุนี้โหนดแต่ละโหนดจึงอาจถูกเยี่ยมมากกว่า 1 ครั้ง เพื่อทำการรับและส่งสินค้าที่เหลืออยู่จนครบทั้งหมด

งานวิจัยนี้นำเสนอวิธีการจัดเส้นทางไปยังโหนดต่าง ๆ เพื่อไปรับหรือส่งสินค้าแต่ละชิ้น โดยมีวัตถุประสงค์หลักเพื่อให้จำนวนพาหนะที่ใช้ในการขนส่งน้อยที่สุด และวัตถุประสงค์รองคือระยะทางในการเดินทางรวมน้อยที่สุด ซึ่งเส้นทางการขนส่งสินค้านั้น จะระบุถึงลำดับของโหนดที่พาหนะต้องเดินทางไป รวมถึงลำดับการรับและส่งสินค้าที่เกิดขึ้น ณ แต่ละโหนด เวลาเริ่มต้นการขนส่งของพาหนะ เวลาที่พาหนะเดินทางถึงแต่ละโหนด เวลาที่พาหนะเริ่มให้บริการรับหรือส่งสินค้าแต่ละชิ้น รวมถึงเวลาเสร็จสิ้นการขนส่งของพาหนะ

2.2 การแทนรูปแบบปัญหาการจัดเส้นทางเดินรถด้วยกราฟ

2.2.1 การจัดเส้นทางเดินรถสำหรับส่งสินค้า

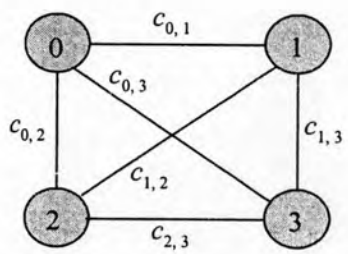
ปัญหาการจัดเส้นทางเดินรถสำหรับส่งสินค้านั้น สามารถทำให้เข้าใจและวิเคราะห์ปัญหาให้ง่ายขึ้น โดยการแทนรูปแบบของปัญหาให้อยู่ในรูปของกราฟแบบบริบูรณ์ (Complete Graph) ซึ่งกราฟดังกล่าว จะประกอบไปด้วย 3 ส่วน คือ โหนด (Node) ด้านไม่มี

ทิศทาง (Edge) และน้ำหนักบนด้าน (Weight) เราสามารถเขียนแทนเป็นสัญลักษณ์ได้ว่า กราฟ $G=(N_0,E)$ โดยที่ $N_0=\{0\}\cup N$ ซึ่ง $N=\{1,\ldots,n\}$

สำหรับปัญหาการจัดเส้นทางเดินรถ จุดยอด 0 ใช้แทนคลังสินค้าหรือที่จอดรถ และจุดยอด $i\in N$ ใช้แทนจุดที่มีการส่งสินค้า สำหรับแต่ละด้าน $(i,j)\in E$ โดยที่ $i,j\in N_0$ จะมีค่าใช้จ่ายในการเดินทาง c_{ij} เกิดขึ้น กำหนดให้ค่าใช้จ่ายในการเดินทาง c_{ij} แปรผันตรงกับระยะทางระหว่างจุดยอด i ไปจุดยอด j โดยที่ค่าใช้จ่ายในการเดินทางมีลักษณะสมมาตร นั่นคือ c_{ij} มีค่าเท่ากับ c_{ji} นอกจากนี้ค่าใช้จ่ายในการเดินทางจะต้องสอดคล้องกับกฎ *Triangle Inequality* ดังนี้

$$c_{ik}+c_{kj}\ \geq\ c_{ij} \qquad \text{for all } i,j,k\in N_0$$

ตัวอย่างของกราฟ แสดงดังรูปที่ 2.1



รูปที่ 2.1 กราฟแทนปัญหาการส่งสินค้า

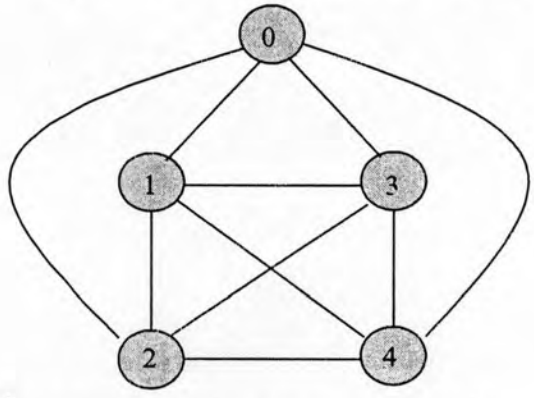
2.2.2 การจัดเส้นทางเดินรถสำหรับรับและส่งสินค้า

ลักษณะของปัญหาการรับและส่งสินค้าจากงานวิจัยที่ผ่านมานั้น คือการขนส่งสินค้า n ชิ้น โดยสินค้าแต่ละชิ้นจะมีต้นทางและปลายทางของสินค้า ต้นทางคือจุดที่ต้องไปรับสินค้า ส่วนปลายทางคือจุดที่ต้องนำสินค้าไปส่ง โดยที่ต้นทางและปลายทางของสินค้าแต่ละชิ้นจะอยู่ต่างที่กันทั้งหมด รถที่ทำการขนส่งสินค้าจะต้องไปรับสินค้าที่ต้นทาง แล้วนำไปส่งยังปลายทางของสินค้านั้น ปัญหาการรับและส่งสินค้านี้สามารถทำให้เข้าใจและวิเคราะห์ปัญหาให้ง่ายขึ้น โดยการแทนรูปแบบของปัญหาให้อยู่ในรูปของกราฟแบบบริบูรณ์ (Complete Graph) $G=(N_0,A)$ โดยที่ $N_0=\{0\}\cup N$ ซึ่ง $N=\{1,\ldots,2n\}$ โดยที่จุดยอด 0 แทนที่จอดรถ จุดยอด i ใช้แทนต้นทางของสินค้าชิ้นที่ i จุดยอด $i+n$ ใช้

แทนปลายทางของสินค้าชิ้นที่ i โดยที่ $i \in \{1,2,\ldots,n\}$ ดังนั้น กราฟ G จะประกอบไปด้วยจุดยอดทั้งสิ้น $2n + 1$ จุด สำหรับเซตของด้าน A ให้ $A = \{(i,j): i,j \in N_0, i \neq j\}$ โดยที่แต่ละด้านจะมีค่าใช้จ่ายในการเดินทาง c_{ij} เกิดขึ้น กำหนดให้ค่าใช้จ่ายในการเดินทาง c_{ij} แปรผันกับระยะทางระหว่างจุดยอด i ไปจุดยอด j และค่าใช้จ่ายในการเดินทางมีลักษณะสมมาตร นั่นคือ c_{ij} มีค่าเท่ากับ c_{ji} นอกจากนี้ค่าใช้จ่ายในการเดินทางจะต้องสอดคล้องกับกฎ *Triangle Inequality* ดังนี้

$$c_{ik} + c_{kj} \geq c_{ij} \qquad \text{for all } i,j,k \in N_0$$

ตัวอย่างกราฟที่ใช้แทนปัญหาการรับและส่งสินค้า เป็นดังรูปที่ 2.2



รูปที่ 2.2 กราฟแทนปัญหาการรับและส่งสินค้า

2.3 ขั้นตอนวิธีของไดสตรา (Dijkstra's Algorithm)

ขั้นตอนวิธีของไดสตราเป็นขั้นตอนวิธีที่รู้จักกันอย่างดีสำหรับปัญหาที่เกี่ยวข้องกับกราฟ ซึ่งใช้สำหรับหาระยะทาง (Distance) และทางเดินที่สั้นที่สุด (Shortest Path) จากโหนดหนึ่งไปยังโหนดอื่น ๆ ทั้งหมดในกราฟ

กำหนดให้ $G = (N,E)$ เป็นกราฟที่มีน้ำหนัก (Weighted Graph G) โดยที่น้ำหนักบนแต่ละด้านมีค่าเป็นบวกและต้องการหาทางเดินที่สั้นที่สุดเส้นทางหนึ่งจากโหนด s ไป z ขั้นตอนวิธีของไดสตราเป็นการกำหนดค่าประจำให้โหนด (label) ให้L(v) แทนค่าประจำโหนด v ณ ขั้นใด ๆ โหนดบางโหนดได้ค่าประจำโหนดชั่วคราวและโหนดที่เหลือจะได้ค่าประจำโหนดถาวร ให้ $\mathcal{P}$ แทนเซตของโหนดที่ได้ค่าประจำถาวร ถ้า L(v) เป็นความยาวของทางเดินที่สั้นที่สุดเส้นหนึ่งจาก

โหนด s ไป v เมื่อเริ่มต้นให้ทุกโหนดมีค่าประจำโหนดชั่วคราว แต่ละการวนซ้ำกันของขั้นตอนวิธีนี้ จะเปลี่ยนสภาพของค่าประจำโหนดค่าหนึ่งจากชั่วคราวเป็นถาวร ดังนั้นเราอาจจบการคำนวณได้เมื่อ z ได้รับค่าประจำโหนดถาวร ณ จุดนี้ $L(z)$ จะเป็นความยาวของทางเดินที่สั้นที่สุดเส้นหนึ่งจาก s ไป z (วนิดา เหมะกุล, 2535: 140)

ขั้นตอนวิธีของไดสตรา

ต้องการหาทางเดินที่สั้นที่สุดจากโหนด a ไปยังโหนดอื่น ๆ ในกราฟที่มีน้ำหนัก ให้น้ำหนักของด้าน (i , j) คือ $w(i,j) > 0$ และค่าประจำโหนด x คือ $L(x)$

ให้น้ำหนักของด้านที่เชื่อมระหว่างโหนด u กับ v แทนด้วย $w(u,v)$ และถ้าไม่มีด้านเชื่อมระหว่างโหนด u กับ v ให้ $w(u, v) = \infty$

ขั้น 1 (เริ่มต้น)

ก. ให้ $L(s) = 0$ และ s ไม่มี predecessor (-)

ข. ให้ $\mathcal{P} = \{s\}$

ขั้น 2 (ให้ค่าประจำโหนด)

ก. แต่ละโหนด v ที่ไม่อยู่ใน $\mathcal{P}$ ให้ $L(v) = w(s, v)$ (อาจชั่วคราว) และให้ predecessor ของโหนด v คือโหนด s (อาจชั่วคราว)

ข. ทำการให้ค่าประจำโหนดถาวรโดยเลือกโหนด u ที่มีค่า L(u) ต่ำสุดในบรรดาโหนดที่ไม่อยู่ใน $\mathcal{P}$ ใส่ u ใน $\mathcal{P}$

ขั้น 3 (ขยาย $\mathcal{P}$ และปรับค่าประจำโหนดใหม่)

repeat

ขั้น 3.1 (ปรับค่าประจำโหนดชั่วคราว)

สำหรับแต่ละโหนด x ที่ไม่อยู่ใน $\mathcal{P}$ แต่อยู่ติดกับโหนด u ให้

$$L(x) = \min\{L(x) , L(u) + w(u, x)\}$$

ถ้าค่าประจำโหนด x ถูกเปลี่ยน ให้โหนด u เป็น predecessor ใหม่ของโหนด x (อาจชั่วคราว)

ขั้น 3.2 (ทำการให้ค่าประจำโหนดถาวร)

เลือกโหนด u ที่มีค่า L(u) ต่ำสุดในบรรดาโหนดที่ไม่อยู่ใน $\mathcal{P}$ ใส่ u ใน $\mathcal{P}$

until $\mathcal{P}$ ได้โหนดครบทุกโหนดของกราฟ G

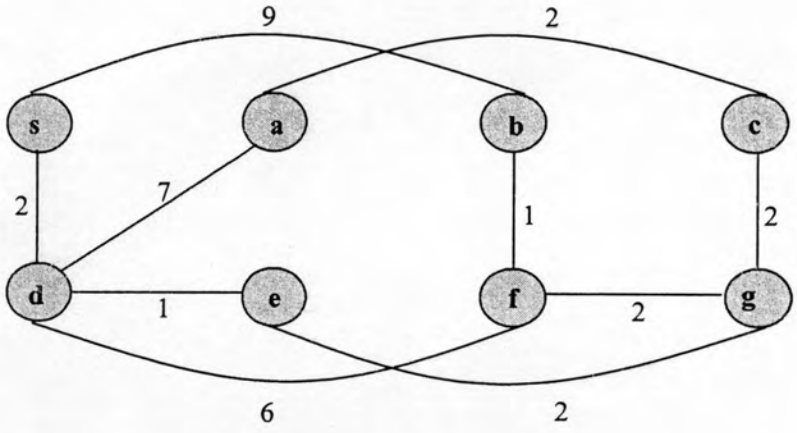
ขั้น 4 (หาระยะทางและทางเดินที่สั้นที่สุด)

สำหรับแต่ละโหนด $y \neq s$

ถ้า $L(y)$ คือ ∞ แสดงว่าไม่มีเส้นทางทางเดินจากโหนด s ไปโหนด y และไม่มีทางเดินที่สั้นที่สุดจากโหนด s ไปโหนด y

มิฉะนั้น เราสามารถหาทางเดินที่สั้นที่สุดจากโหนด s ไปโหนด y โดยใช้อันดับย้อนกลับจากโหนด y นำด้วย predecessor ของโหนด y predecessor ของ predecessor ของโหนด y ไปเรื่อย ๆ จนถึงโหนด s

ตัวอย่างของการคำนวณหาระยะทางและทางเดินที่สั้นที่สุดโดยใช้ขั้นตอนวิธีของไดสตราเป็นดังรูปที่ 2.3 และตารางที่ 2.1



รูปที่ 2.3 ตัวอย่างกราฟสำหรับหาระยะทางและทางเดินที่สั้นที่สุด

จากรูปที่ 2.3 โหนดทั้งสิ้น 8 โหนด ได้แก่ โหนด s a b c d e f และ g เราสามารถหาระยะทางและทางเดินที่สั้นที่สุดจากโหนด s ไปยังโหนดอื่น ๆ ของกราฟ โดยใช้ขั้นตอนวิธีของไดสตราแสดงวิธีคำนวณดังตารางที่ 2.1

ตารางที่ 2.1 วิธีการคำนวณหาระยะทางและทางเดินที่สั้นที่สุดโดยใช้ขั้นตอนวิธีของไดสตรา สำหรับกราฟในรูปที่ 2.3

โหนด / ขั้น/รอบ	s	a	b	c	d	e	f	g	$\mathcal{P}$
ขั้น 1	0 (-)								s
ขั้น 2		∞ (s)	9 (s)	∞ (s)	2 (s)	∞ (s)	∞ (s)	∞ (s)	d
ขั้น 3 รอบที่ 1		9 (d)	9 (s)	∞ (s)		3 (d)	8 (d)	∞ (s)	e
ขั้น 3 รอบที่ 2		9 (d)	9 (s)	∞ (s)			8 (d)	5 (e)	g
ขั้น 3 รอบที่ 3		9 (d)	9 (s)	7 (g)			7 (g)		c
ขั้น 3 รอบที่ 4		9 (d)	9 (s)				7 (g)		f
ขั้น 3 รอบที่ 5		9 (d)	8 (f)						b
ขั้น 3 รอบที่ 6		9 (d)							a

จากตารางที่ 2.1 สรุประยะทางและทางเดินที่สั้นที่สุดจากโหนด s ไปยังโหนดอื่น ๆ ของกราฟตามวิธีในขั้น 4 ได้ดังนี้

- ทางเดินที่สั้นที่สุดระหว่างโหนด s กับโหนด a คือ s, d, a โดยมีระยะทางคือ 9 หน่วย
- ทางเดินที่สั้นที่สุดระหว่างโหนด s กับโหนด b คือ s, d, e, g, f, b โดยมีระยะทางคือ 8 หน่วย
- ทางเดินที่สั้นที่สุดระหว่างโหนด s กับโหนด c คือ s, d, e, g, c โดยมีระยะทางคือ 7 หน่วย
- ทางเดินที่สั้นที่สุดระหว่างโหนด s กับโหนด d คือ s, d โดยมีระยะทางคือ 2 หน่วย
- ทางเดินที่สั้นที่สุดระหว่างโหนด s กับโหนด e คือ s, d, e โดยมีระยะทางคือ 3 หน่วย
- ทางเดินที่สั้นที่สุดระหว่างโหนด s กับโหนด f คือ s, d, e, g, f โดยมีระยะทางคือ 7 หน่วย
- ทางเดินที่สั้นที่สุดระหว่างโหนด s กับโหนด g คือ s, d, e, g โดยมีระยะทางคือ 5 หน่วย

2.4 วิธีการหาคำตอบ

ขั้นตอนวิธี (Algorithm) ที่ใช้ในการหาคำตอบสำหรับปัญหาที่เหมาะสมที่สุดเชิงการจัด (Combinatorial Optimization) แบ่งออกได้เป็น 2 ประเภท ได้แก่ ขั้นตอนวิธีเหมาะสมที่สุด (Optimization Algorithms) และ ขั้นตอนวิธีฮิวริสติก (Heuristic Algorithms) โดยความแตกต่างของทั้ง 2 ขั้นตอนวิธีนี้จะอยู่ที่ผลลัพธ์ของคำตอบ สำหรับการหาคำตอบโดยใช้ขั้นตอนวิธีเหมาะสมที่สุดนั้น รับประกันได้ว่าคำตอบที่ได้นั้นให้ค่าที่เหมาะสมที่สุด (Optimal Solution) แต่ใช้เวลานานในการหาคำตอบ ซึ่งเมื่อขนาดของปัญหาใหญ่ขึ้น ก็จะใช้เวลาในการหาคำตอบเพิ่มขึ้นอย่างเอ็กโพเนนเชียล ส่วนคำตอบที่ได้จากขั้นตอนวิธีฮิวริสติกนั้น ไม่รับประกันว่าคำตอบที่ได้นั้นจะให้ค่าที่เหมาะสมที่สุด แต่ใช้เวลารวดเร็วในการหาคำตอบ โดยถ้าขั้นตอนวิธีฮิวริสติกที่ใช้ในการหาคำตอบนั้นเป็นขั้นตอนวิธีที่มีคุณภาพ ก็จะสามารถทำให้คำตอบที่ได้นั้นให้ค่าที่ใกล้เคียงกับค่าที่เหมาะสมที่สุด หรืออาจให้ค่าที่เหมาะสมที่สุดก็เป็นได้ (Blum and Roli, 2003; Minic 1998)

2.4.1 ขั้นตอนวิธีเหมาะสมที่สุด

ขั้นตอนวิธีเหมาะสมที่สุด (Optimization Algorithms) ที่ใช้สำหรับแก้ปัญหาที่เหมาะสมที่สุดเชิงการจัด (Combinatorial Optimization) นั้นมีด้วยกันหลากหลายวิธี โดยวิธีที่ได้รับความนิยมอย่างแพร่หลาย ได้แก่ ขั้นตอนวิธีการแตกกิ่งและจำกัดเขต และการโปรแกรมพลวัต

2.4.2 ขั้นตอนวิธีฮิวริสติก

เนื่องจากขั้นตอนวิธีเหมาะสมที่สุดนั้นใช้เวลาในการค้นหาคำตอบนาน จึงไม่เป็นที่นิยมสำหรับปัญหาที่มีขนาดใหญ่ จึงได้มีการศึกษาขั้นตอนวิธีฮิวริสติกซึ่งเป็นขั้นตอนวิธีที่มีความรวดเร็วในการหาคำตอบ และได้รับการวิจัยอย่างแพร่หลาย ขั้นตอนวิธีฮิวริสติกสามารถแบ่งออกเป็น 2 ประเภท คือ ขั้นตอนวิธีฮิวริสติกดั้งเดิม (Classic Heuristic Algorithms) และ ขั้นตอนวิธีฮิวริสติกสมัยใหม่ (Modern Heuristic Algorithms) (Minic, 1998)

2.4.2.1 ขั้นตอนวิธีฮิวริสติกดั้งเดิม

ขั้นตอนวิธีฮิวริสติกดั้งเดิม (Classic Heuristic Algorithms) เป็นขั้นตอนวิธีฮิวริสติกที่ใช้สำหรับแก้ปัญหาในช่วงยุคเริ่มต้นที่มีการใช้วิธีฮิวริสติกในการหาคำตอบ ซึ่งฮิวริสติกที่ใช้แก้ปัญหาในยุคแรก มี 2 ประเภทคือ ขั้นตอนวิธีเชิงสร้าง (Constructive Algorithms) และขั้นตอนวิธีเชิงปรับปรุง (Improvement Algorithms)

2.4.2.1.1 ขั้นตอนวิธีเชิงสร้าง

ขั้นตอนวิธีเชิงสร้าง (Constructive Algorithms) จะเริ่มต้นโดยยังไม่มีคำตอบใดๆ แล้วค่อยๆ สร้างส่วนของคำตอบขึ้นเรื่อย ๆ โดยการเติมส่วนของคำตอบเข้าไปเรื่อย ๆ จนกระทั่งทุกส่วนของคำตอบถูกเติมลงไปจนได้เป็นคำตอบที่สมบูรณ์ขึ้นมา 1 คำตอบ (ปารเมศ ชุติมา, 2546)

2.4.2.1.2 ขั้นตอนวิธีเชิงปรับปรุง

ขั้นตอนวิธีเชิงปรับปรุง (Improvement Algorithms) เริ่มต้นจากคำตอบซึ่งสมบูรณ์อยู่แล้ว ซึ่งอาจได้มาจากการสุ่ม หลังจากนั้นพยายามหาทางปรับปรุงคำตอบโดยการสลับสับเปลี่ยนตำแหน่งของคำตอบ จนกว่าจะได้คำตอบที่น่าพอใจ

ขั้นตอนวิธีเชิงปรับปรุงเกือบทั้งหมด ที่ใช้สำหรับสร้างเส้นทางการขนส่งนั้น คือ Local Search Algorithms (Minic, 1998)

Local Search Algorithms คือขั้นตอนวิธีเชิงปรับปรุงอย่างหนึ่ง ซึ่งมีลักษณะของการหาคำตอบคือ เริ่มต้นจากคำตอบเริ่มต้น (Initial Solution) ซึ่งอาจถูกสร้างขึ้นมาโดยการสุ่ม หรือสร้างโดยขั้นตอนวิธีใดวิธีหนึ่งก็ได้ แล้วทำการสร้างเซตของคำตอบขึ้นมาใหม่จากคำตอบปัจจุบัน (Current Solution) โดยผ่านฟังก์ชันฟังก์ชันหนึ่ง เรียกฟังก์ชันนี้ว่า Neighborhood Structure (N) และเรียกคำตอบที่ถูกสร้างขึ้นมาใหม่จากคำตอบปัจจุบันว่า คำตอบข้างเคียง (Neighbor Solution) จากนั้น จะทำการเลือกคำตอบที่ดีที่สุดในคำตอบข้างเคียงทั้งหมดและดีกว่าคำตอบปัจจุบัน มาเป็นคำตอบปัจจุบันตัวต่อไป ทำซ้ำเช่นนี้เรื่อยไปจนกระทั่งไม่มีการปรับปรุงคำตอบจึงหยุดการค้นหาคำตอบ และจะได้

ว่าคำตอบปัจจุบันตัวล่าสุดคือคำตอบของปัญหา ซึ่งคำตอบที่ได้จาก Local Search Algorithms นั้น อาจเป็นเพียงแค่ Local Optima เท่านั้น (Blum and Roli, 2003)

การค้นหาคำตอบโดยวิธี Local Search สามารถอธิบายได้ด้วยขั้นตอนวิธีดังรูปที่ 2.4

```
Procedure Local Search
begin
    s ← GenerateInitialSolution()
    repeat
        s ← Improve(N(s))
    until no improvement is possible
end
```

รูปที่ 2.4 Algorithm : Local Search

2.4.2.2 ขั้นตอนวิธีฮิวริสติกสมัยใหม่

เนื่องจากประสิทธิภาพของขั้นตอนวิธีฮิวริสติกแบบดั้งเดิมนั้น ยังไม่ดีพอที่จะทำให้คำตอบที่ได้นั้นเป็นคำตอบที่ดีที่สุด (Optimal Solution) จึงมีการศึกษาและพัฒนาฮิวริสติกเรื่อยมา จนถึงช่วงเริ่มต้นยุค 1980 ได้มีผู้คิดค้นฮิวริสติกรูปแบบใหม่ขึ้น ซึ่งมีประสิทธิภาพสูงในการค้นหาคำตอบ เราเรียกฮิวริสติกรูปแบบใหม่นี้ว่า Metaheuristics แนวความคิดของการค้นหาคำตอบด้วย Metaheuristics คือต้องการจะหลบหนีจาก Local Optima โดยยอมที่จะเคลื่อนย้ายไปที่คำตอบที่แย่กว่า เพื่อหวังที่จะได้เจอคำตอบที่เป็น Global Optima (Blum and Roli, 2003; Minic 1998) Modern Heuristics หรือ Metaheuristics มีด้วยกันหลายประเภทได้แก่

2.4.2.2.1 Simulated Annealing

Simulated Annealing ถือได้ว่าเป็น Metaheuristics ที่เก่าแก่ที่สุด และมียุทธวิธีในการค้นหาคำตอบโดยการหลีบเลี่ยงจาก Local Optima โดยยอมให้มี

การย้ายคำตอบไปสู่คำตอบที่แย่กว่าคำตอบปัจจุบัน การค้นหาคำตอบโดยวิธี Simulated Annealing นั้น เป็นการค้นหาคำตอบโดยเลียนแบบจากพฤติกรรมของกลไกการเผาโลหะให้อ่อนตัว ซึ่งคิดค้นโดย Metropolis เริ่มต้นจะให้ความร้อนแก่โลหะที่จุดหลอมเหลวของโลหะนั้น จากนั้นค่อยๆ ลดอุณหภูมิให้โลหะค่อยๆ เย็นตัวลงและกลายเป็นของแข็งเหมือนเดิม ซึ่งโครงสร้างของโลหะนั้นจะแข็งแรงมากน้อยเพียงใด ก็ขึ้นอยู่กับวิธีการลดอุณหภูมิที่ใช้เผาโลหะนั้น ซึ่งถ้าหากลดอุณหภูมิเร็วเกินไป จะทำให้โลหะเปราะและไม่แข็งแรงได้

วิธีการหาคำตอบของ Simulated Annealing นั้น เป็นการค้นหาคำตอบโดยทำการวนซ้ำเพื่อปรับปรุงคำตอบไปเรื่อยๆ โดยใช้หลักความน่าจะเป็น (Probabilistic) ในการยอมรับหรือปฏิเสธคำตอบข้างเคียง (Neighbor Solution) ซึ่งสามารถอธิบายได้ด้วยขั้นตอนวิธีดังรูปที่ 2.5

Procedure Simulated Annealing (for minimization problem)
begin
  $k \leftarrow 0$
  initialize T
  $s \leftarrow$ GenerateInitialSolution()
  repeat
    repeat
      $s' \leftarrow$ Pickup($N(s)$)
      if ($f(s') < f(s)$) **then**
        $s \leftarrow s'$
      else if $random(0,1) < \exp\left(-\frac{f(s')-f(s)}{T}\right)$
        $s \leftarrow s'$
    until (Termination-Condition)
      $T \leftarrow g(T,k)$
      $k \leftarrow k+1$
  until (Halting-Criterion)
end

รูปที่ 2.5 Algorithm : Simulated Annealing

สำหรับการค้นหาคำตอบโดยวิธี Simulated Annealing นั้น เริ่มต้นที่คำตอบเริ่มต้น (Initial Solution) ซึ่งอาจได้จากการสุ่มหรือสร้างขึ้นมาจากขั้นตอนวิธีเชิงปรับปรุงก็ได้ ซึ่งคำตอบเริ่มต้นนี้จะเป็นคำตอบปัจจุบัน s (Current Solution) จากนั้นทำการกำหนดอุณหภูมิเริ่มต้นเพื่อใช้หลอมเหลวที่อุณหภูมิ T ในแต่ละรอบของการวนซ้ำ จะทำการสร้างคำตอบข้างเคียง $N(s)$ จากคำตอบปัจจุบัน s และเลือกคำตอบข้างเคียง $s' \in N(s)$ ขึ้นมา 1 คำตอบเพื่อที่จะใช้เป็นคำตอบปัจจุบันตัวต่อไป โดยถ้าหากคำตอบข้างเคียงที่เลือกมานั้นเป็นคำตอบที่ดีกว่าคำตอบปัจจุบัน ก็จะทำการกำหนดให้คำตอบข้างเคียงตัวนี้เป็นคำตอบปัจจุบันตัวต่อไป แต่ถ้าหากคำตอบข้างเคียงที่เลือกมานั้นเป็นคำตอบที่แย่กว่าคำตอบปัจจุบัน ก็จะมีการยอมรับหรือปฏิเสธคำตอบข้างเคียงตัวนี้ให้เป็นคำตอบปัจจุบันตัวต่อไปด้วยความน่าจะเป็น โดยจะทำการสุ่มตัวเลขจากการกระจายแบบยูนิฟอร์ม (0,1) ขึ้นมาตัวหนึ่ง แล้วทำการเปรียบเทียบกับค่าความน่าจะเป็นที่คำนวณได้จากฟังก์ชันการกระจายความน่าจะเป็นของโบลซ์แมน (Boltzmann Distribution)

$$P(T,s,s') = \exp\left(-\frac{f(s')-f(s)}{T}\right)$$

ถ้าหากค่าที่สุ่มได้มีค่าน้อยกว่าค่าความน่าจะเป็นที่คำนวณได้จากฟังก์ชันการกระจายความน่าจะเป็นของโบลซ์แมน ก็จะยอมรับคำตอบข้างเคียงที่แย่ตัวนั้นเป็นคำตอบปัจจุบันตัวต่อไป ด้วยวิธีการเช่นนี้จะช่วยทำให้การหาคำตอบหลุดจาก Local Optima แล้วอาจจะเจอคำตอบที่เป็น Global Optima ได้ เมื่อทำการหาคำตอบไปได้ระยะหนึ่งจนถึงเกณฑ์ที่กำหนดไว้สำหรับการลดอุณหภูมิ (Termination-Condition) แล้ว ก็จะทำการลดอุณหภูมิลงมาระดับหนึ่ง โดยใช้ฟังก์ชันควบคุมการเย็นตัว (Cooling Schedule) $g(T, k)$ ซึ่งฟังก์ชันควบคุมการเย็นตัวที่นิยมใช้กันคือ $T_{k+1} = \alpha T_k$ โดยที่ $\alpha \in (0,1)$ จากนั้นก็จะทำการหาคำตอบต่อไปจนกว่าจะถึงเกณฑ์ในการหยุดหาคำตอบ (Halting-Criterion) (Blum and Roli, 2003)

2.4.2.2.2 Tabu Search

Tabu Search (TS) เป็น Metaheuristics ที่ใช้สำหรับหาคำตอบที่เหมาะสมที่สุดเชิงการจัด ซึ่งถูกคิดค้นโดย Glover หลักการของ TS จะมีการใช้ Short-Term Memory เพื่อช่วยให้การค้นหาคำตอบหลุดจาก Local Optima และเป็นการป้องกันการเกิดการวนซ้ำ Short-Term Memory สามารถถูกสร้างให้อยู่ในรูปของ Tabu List ซึ่งเป็น List สำหรับเก็บคำตอบที่เคยเป็นคำตอบในรอบก่อนๆ เพื่อที่ว่าการย้ายคำตอบไปสู่คำตอบจุดต่อไป จะต้องไม่ย้ายไปสู่คำตอบที่อยู่ใน Tabu List นี้ นั่นคือเป็นคำตอบต้องห้าม คำตอบข้างเคียงของคำตอบปัจจุบันจะต้องไม่ใช่คำตอบที่อยู่ใน Tabu List ซึ่งเรียกเซตของคำตอบข้างเคียงนี้ว่า Allowed Set ในแต่ละรอบของการหาคำตอบคำตอบที่ดีที่สุดที่อยู่ใน Allowed Set จะถูกเลือกไปเป็นคำตอบปัจจุบันตัวต่อไป สำหรับในส่วนของ Tabu List นั้น จะมีการกำหนดความยาวของ List ในการเก็บคำตอบเอาไว้ เรียกว่า Tabu Tenure โดยคำตอบที่ถูกห้ามไว้ใน Tabu List นั้น มิได้ถูกห้ามตลอดไป แต่จะถูกห้ามเอาไว้ระยะหนึ่งโดยขึ้นกับความยาวของ List ที่เก็บคำตอบต้องห้าม ซึ่งวิธีการเก็บคำตอบต้องห้ามนั้น จะใช้หลัก FIFO ดังนั้นคำตอบที่ถูกห้ามอยู่ใน Tabu List นั้น จะถูกห้ามเอาไว้เป็นจำนวนเท่ากับความยาวของ Tabu Tenure หลังจากนั้น จึงหลุดออกจาก Tabu List และสามารถใช้เป็นคำตอบปัจจุบันต่อไปได้ สังเกตว่าถ้าหาก Tabu List มี Tabu Tenure ขนาดเล็ก จะทำให้พื้นที่ในการค้นหาคำตอบอยู่ในช่วงแคบๆ เท่านั้น แต่ถ้ามี Tabu Tenure ขนาดใหญ่ ก็จะทำให้พื้นที่การค้นหาคำตอบกว้างขึ้น ดังนั้นขนาดของ Tabu Tenure จึงมีความสำคัญต่อการค้นหาคำตอบ สำหรับวิธีการเก็บคำตอบเอาไว้ใน Tabu List นั้น ถ้าหากเก็บคำตอบทั้งคำตอบเอาไว้ใน Tabu List จะทำให้ Tabu List นั้นไม่มีประสิทธิภาพ เนื่องจากใช้หน่วยความจำมากเกินไปทำให้การค้นหาช้า จึงนิยมเก็บเฉพาะบางส่วนของคำตอบเท่านั้น เรียกส่วนของคำตอบที่เก็บไว้ใน Tabu List ว่า Attributes ซึ่งทำให้มีประสิทธิภาพมากขึ้น แต่ยังเกิดข้อเสียตรงที่ว่า Attributes หนึ่งอาจเป็นตัวแทนของคำตอบได้มากกว่า 1 คำตอบก็ได้ ดังนั้น อาจทำให้คำตอบที่ดีบางตัวอาจถูกละเลยข้ามไป ไม่ถูกสร้างให้อยู่ใน Allowed Set ดังนั้นจึงมีเกณฑ์อีกเกณฑ์หนึ่งกำหนดขึ้นมาเรียกว่า Aspiration Criteria ซึ่งเป็นเกณฑ์ที่ใช้กำหนดว่า คำตอบที่เป็นไปตามเกณฑ์นี้ สามารถอยู่ใน Allowed Set ได้ ถึงแม้

จะถูกห้ามอยู่ใน Tabu List ตัวอย่าง Aspiration Criteria เช่น คำตอบที่ดีกว่าคำตอบที่ดีที่สุดสามารถอยู่ใน Allowed Set ได้ ถึงแม้คำตอบนั้นจะอยู่ใน Tabu List ก็ตาม (Blum and Roli, 2003)

วิธีการหาคำตอบของ Tabu Search สามารถอธิบายได้ด้วยขั้นตอนวิธีดังรูปที่ 2.6

```
Procedure Tabu Search
begin
    s ← GenerateInitialSolution()
    TabuList ← ∅
    k ← 0
    while termination conditions not met do
        AllowedSet(s,k) ← {s'∈ N(s) | s does not
                violate a Tabu List, or it satisfies
                at least one aspiration condition}
        s ← ChooseBestof(AllowedSet(s,k))
        UpdateTabuListandAspirationConditions()
        k ← k+1
    endwhile
end
```

รูปที่ 2.6 Algorithm : Tabu Search

2.4.2.2.3 Genetic Algorithms

Genetic Algorithms (GA) เป็นวิธีการค้นหาคำตอบโดยเลียนแบบมาจากการวิวัฒนาการที่เกิดขึ้นในธรรมชาติ เริ่มต้นจากประชากรรุ่นแรกที่มีอยู่ สมาชิกเหล่านี้จะผ่านกระบวนการคัดเลือกทางธรรมชาติ (Selection) เพื่อหาสมาชิกที่มีความเหมาะสมในการผลิตสมาชิกตัวใหม่ต่อไป ส่วนสมาชิกตัวที่ด้อยกว่า จะไม่สามารถอยู่รอดได้ก็ต้องตายจากไป สมาชิกที่ผ่านการคัดเลือกจะจับคู่กันอย่างสุ่มเพื่อทำการครอสโอเวอร์ (Crossover) กัน เรียกสมาชิก 2 ตัวที่นำมาทำการครอสโอเวอร์กันว่า Parents หลังจากผ่านการครอสโอเวอร์จะทำให้เกิดสมาชิกใหม่ขึ้นมาเรียกว่า Child นอกจากนี้ สมาชิกบางตัวอาจเกิดการ

กลายพันธุ์ (Mutation) ได้ สมาชิกทั้งหมดเหล่านี้จะกลายเป็นประชากรรุ่นใหม่ (New Population) และเกิดการวิวัฒนาการในแต่ละรุ่นเช่นนี้เรื่อยไป

วิธีการหาคำตอบโดยใช้ GA เป็นการค้นหาคำตอบจากสมาชิกทั้งหมดของประชากรรุ่นนั้น ไม่ใช่จากสมาชิกตัวใดตัวหนึ่ง และมีการใช้ความน่าจะเป็นในแต่ละกระบวนการ ซึ่งสามารถอธิบายได้ด้วยขั้นตอนวิธีดังรูปที่ 2.7

```
Procedure Genatic Algorithms
begin
    P ←— GenerateInitialPopulation()
    Evaluate(P)
    while termination conditions not met do
        P ←— Selection(P)
        P ←— Crossover(P)
        P ←— Mutation(P)
        Evaluate(P)
    endwhile
end
```

รูปที่ 2.7 Algorithm : Genetic Algorithms

การค้นหาคำตอบโดยวิธี GA นั้น ต้องมีการเข้ารหัส (Encoding) หรือแปลงคำตอบ (Solutions) ให้อยู่ในรูปของสายอักขระ (String) ก่อน หรือเรียกว่าโครโมโซม (Chromosome) โดยวิธีการเข้ารหัสนั้นขึ้นอยู่กับรูปแบบคำตอบของปัญหา ตัวอย่างหนึ่งของวิธีการเข้ารหัสคือ การเข้ารหัสแบบไบนารี (Binary Coding) ซึ่งสายอักขระที่ได้จะประกอบไปด้วยบิต 0 กับบิต 1

ขั้นตอนแรกของการค้นหาคำตอบ จะทำการสุ่มคำตอบขึ้นมาชุดหนึ่ง เรียกคำตอบชุดนี้ว่าประชากร (Population) จากนั้นทำการเข้ารหัสโดยใช้วิธีที่เหมาะสมกับปัญหา หลังจากเข้ารหัสแล้วจะเข้าสู่กระบวนการของการค้นหาคำตอบดังนี้ เริ่มต้นจะทำการคัดเลือกหาคำตอบที่มีคุณสมบัติเหมาะสมซึ่งสามารถทำได้หลายวิธีการ วิธีการหนึ่งคือการสร้างวงล้อรูลเล็ท (Roulette Wheel) ที่มีจำนวนช่องเท่ากับจำนวนประชากร และขนาดของช่องเป็นสัดส่วนกับค่าความเหมาะสม จากนั้นทำการหมุนวงล้อเป็นจำนวนเท่ากับ

ขนาดประชากร คำตอบทั้งหมดที่ได้รับการคัดเลือกจะเข้าสู่กระบวนการต่อไป ส่วนคำตอบที่ไม่ได้รับการคัดเลือกจะตายจากไป จากนั้นจะทำการสุ่มคำตอบเป็นคู่ เพื่อนำมาทำการครอสโอเวอร์ (Crossover) กัน ครอสโอเวอร์คือการจับคู่คำตอบ 2 คำตอบใดๆ ในประชากรอย่างสุ่ม แล้วทำการสลับไขว้ตำแหน่งกัน จะทำให้ได้คำตอบใหม่ขึ้นมา นอกจากนี้ คำตอบบางตัวอาจเกิดการกลายพันธุ์ (Mutation) ซึ่งคือการเปลี่ยนบิตภายในคำตอบ โดยเปลี่ยนบิตจาก 1 เป็น 0 หรือ 0 เป็น 1 โดยทำการเลือกตำแหน่งที่จะทำการเปลี่ยนบิตอย่างสุ่ม หลังจากเสร็จสิ้นกระบวนการทั้งหมด คำตอบทั้งหมดจะกลายเป็นประชากรรุ่นใหม่ต่อไป (New Population) และถูกทำซ้ำขั้นตอนเช่นนี้ไปเรื่อยๆ จนกว่าจะถึงจำนวนรุ่นที่ตั้งเกณฑ์ไว้

2.5 บทสรุป

ปัญหาการจัดเส้นทางเดินรถเป็นปัญหาที่เหมาะสมที่สุดเชิงการจัด ซึ่งได้รับการศึกษาอย่างแพร่หลายและแตกแขนงออกไปหลายรูปแบบ ความซับซ้อนของการคำนวณอยู่ในระดับเอ็นพีแบบยาก ขั้นตอนวิธีที่ใช้สำหรับการแก้ปัญหานั้นแบ่งได้เป็น 2 ประเภทใหญ่ๆ คือ ขั้นตอนวิธีเหมาะสมที่สุด (Optimization Algorithms) และขั้นตอนวิธีฮิวริสติก (Heuristic Algorithms) ขั้นตอนวิธีเหมาะสมที่สุดคือขั้นตอนวิธีที่สามารถแก้ปัญหาแล้วได้คำตอบที่เหมาะสมที่สุด แต่ใช้เวลานานในการค้นหาคำตอบ บางขั้นตอนวิธีต้องอาศัยการสร้างรูปแบบปัญหาให้อยู่ในรูปของการโปรแกรมเชิงจำนวนเต็มร่วมกับการแก้ปัญหาด้วย ส่วนขั้นตอนวิธีฮิวริสติกสามารถแก้ปัญหาโดยใช้เวลาไม่นาน แต่ไม่รับประกันว่าคำตอบที่ได้นั้นจะให้คำตอบที่เหมาะสมที่สุด ขั้นตอนวิธีฮิวริสติกแบ่งออกได้ 2 ประเภทตามยุคสมัยของการพัฒนา ในยุคแรกฮิวริสติกที่ใช้ในการแก้ปัญหาจะเป็นลักษณะของขั้นตอนวิธีเชิงสร้าง (Constructive Algorithms) และ Local Search Algorithms ซึ่งคำตอบที่ได้นั้นจะได้เพียงแค่ Local Optima จึงได้มีการพัฒนาขั้นตอนวิธีที่จะช่วยให้สามารถหลบหนีจาก Local Optima ได้ เรียกว่า Metaheuristic Algorithms ตัวอย่างเช่น Simulated Annealing Tabu Search และ Genetic Algorithms แต่ทุกวิธีการที่ใช้ในการหาคำตอบ ก็ยังเป็นที่นิยมสำหรับใช้แก้ปัญหาและถูกใช้อย่างแพร่หลายจนถึงทุกวันนี้