บทที่ 3

การทบทวนวรรณกรรมและผลงานที่ผ่านมาของการวิจัยดำเนินงาน

การวิจัยดำเนินงาน

การวิจัยดำเนินงาน (Operation Research) คือ วิธีการอย่างมีหลักเกณฑ์ ในการจัดรวบรวมข้อมูลและวิเคราะห์เป็นตัวเลขสำหรับช่วยตัดสินใจให้กับฝ่ายบริหาร โดยคำนึงว่าการทำงานนั้นต้องอยู่ภายใต้อำนาจการควบคุมด้วย

รูปแบบของปัญหาในการวิจัยดำเนินงาน สามารถจำแนกออกได้เป็น 10 รูปแบบ คือ

1. Allocation Models เป็นการจัดสรรองค์ประกอบหรือทรัพยากรที่มีอยู่จำกัดให้ได้ผลประโยชน์สูงสุด ตัวอย่างของปัญหา เช่น

- Assignment problem เป็นปัญหาทางการจัดคนเข้ากับงาน งานเข้ากับเครื่องจักร
- Transportation problem เป็นการจัดสรรทรัพยากรระหว่างหน่วยงาน

2. PERT/CPM เป็นการจัดโครงงานที่ประกอบด้วยงานย่อย ๆ ซึ่งใช้เวลาและแรงงานขนาดต่าง ๆ กัน ที่จะต้องการให้เสร็จสิ้นเมื่อไร

3. Competitive Models เป็นการแข่งขันของสองฝ่ายหรือมากกว่า ด้วยการตัดสินใจภายใต้สถานการณ์และการขัดแย้งผลประโยชน์ซึ่งกันและกัน โดยผลลัพธ์ที่ออกมาจะควบคุมการตัดสินใจของทุกฝ่าย

4. Dynamic Programming Models เป็นการแก้ปัญหาด้วยการช่วยตัดสินใจสำหรับการดำเนินงานของระบบที่มีความสัมพันธ์กับเวลาและสถานการณ์ในช่วงเวลาต่อเนื่อง

5. Inventory Models เป็นการกเก็บและรักษาวัสดุหรือสินค้าคงคลัง ซึ่งจะมีปัญหาในการตัดสินใจเกี่ยวกับปริมาณที่ต้องการ หรือการเก็บรักษาในช่วงระยะเวลาหนึ่ง ๆ

6. Queueing Models เป็นการจัดแถวคอยในงานบริการด้านต่าง ๆ

7. Replacement, Maintenance, Reliability Models เป็นการตัดสินใจในการกำหนดจำนวนครั้งและช่วงเวลาในการซ่อมบำรุงหรือเปลี่ยนแปลงเครื่องจักร

8. Routing Models เป็นปัญหาการขนส่งและการสื่อสาร ด้วยการหาเส้นทางจากจุดเริ่มต้นไปสู่จุดหมาย

9. Search and Heuristic Models เป็นกระบวนการที่ใช้แก้ปัญหาในเรื่องการสื่อสาร

10. Sequencing Models เป็นการลำดับงานหรือเหตุการณ์ให้เหมาะสม

จากรูปแบบของปัญหาที่กล่าวมาข้างต้นทั้งหมดนี้ จะเห็นว่ารูปแบบของปัญหาที่เกี่ยวข้องกับการหาเส้นทางเดินรถที่เหมาะสมนั้น จะอยู่ในข้อที่ 8 ซึ่ง Routing Models นี้สามารถแบ่งย่อยออกไปได้อีก 2 ประเภท คือ

- Singleroute Problems
- Multiroute Problems

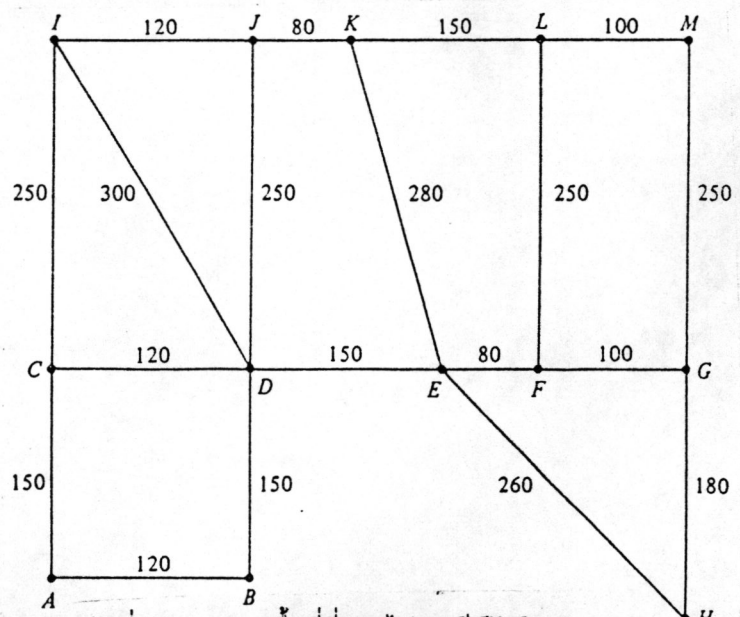
โดยรายละเอียดของปัญหาทั้ง 2 ประเภท จะกล่าวไว้ในหัวข้อต่อไป

Singleroute Problems

Singleroute Problems เป็นการแก้ปัญหาของรถคันเดียว โดยเส้นทางที่ได้ออกมาจะมีเพียงเส้นทางเดียวเท่านั้น และจะเดินทางซ้ำเหมือนกันทุกวัน สำหรับวิธีการแก้ปัญหาที่ใช้ก็จะแตกต่างกันไปตามรูปแบบของปัญหา ดังนี้

1. Edge Covering : The Chinese Postman's Problem (CPP)

เป็นวิธีการที่ใช้หาเส้นทางที่ทำให้การเดินทางมีค่าระยะทาง (ระยะเวลา, ค่าใช้จ่าย) น้อยที่สุด โดยที่การเดินทางจะไปให้ครบทุกถนน ตัวอย่างของปัญหานี้คือ การส่งไปรษณีย์ของบุรุษไปรษณีย์ในพื้นที่ที่กำหนด ดังเช่นในรูป 3.1 บุรุษไปรษณีย์จะเริ่มที่ Node A จากนั้นก็จะเดินทางไปให้ครบทุกถนนแล้วจึงกลับมาที่จุดเริ่มต้น โดยในบางถนนอาจจะต้องมีการเดินซ้ำ จึงจำเป็นที่จะต้องหาเส้นทางที่เมื่อเดินทางจนครบทุกถนนแล้วมีระยะรวมที่มีค่าน้อยที่สุด



รูปที่ 3.1 แสดงพื้นที่ที่บุรุษไปรษณีย์รับผิดชอบ

2. Node Covering : The Traveling Salesman Problem (TSP)

2.1 บทนำ

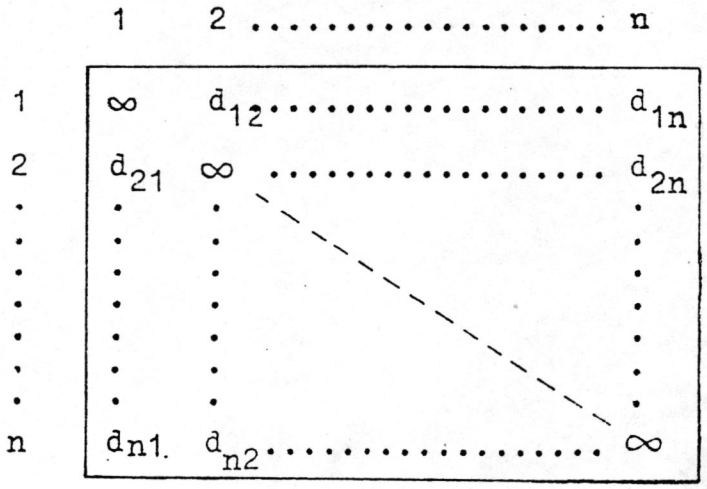
Traveling Salesman Problem (Anen Aung - aphinant, 1980 อ้างถึงใน Little, 1963) เป็นปัญหาที่ตรงกับการหาเส้นทางโดยการกำหนดให้มีจำนวน node ระยะเวลา (ระยะทาง, ค่าใช้จ่าย) จาก node i ถึง node j แทนด้วย d_{ij}, (i , j = 1,2,3, ...,n) และ $d_{ii} = \infty$ ในการหาเส้นทางของ traveling salesman จะเริ่มจาก node k ไปตาม node ที่เหลือจนครบแล้วจึงกลับมาที่ node k อีกครั้ง โดยที่

ก. แต่ละ node จะไปเพียง 1 ครั้ง

ข. ระยะเวลา (ระยะทาง, ค่าใช้จ่าย) ที่ไปเป็นระยะเวลา (ระยะทาง, ค่าใช้จ่าย) ที่น้อยที่สุด

* หมายเหตุ * node ในที่นี้หมายถึงจุดเก็บขน

เมตริกซ์ระยะเวลา (ระยะทาง, ค่าใช้จ่าย) ดังแสดงอยู่ในรูปที่ 3.2 โดยที่ d_{ij} คือ ระยะเวลา (ระยะทาง, ค่าใช้จ่าย) เดินทางจาก node i ไป node j และ $d_{ij} >= 0$



รูปที่ 3.2 แสดงเมตริกซ์ของ TSP

2.2 ความหมายของตัวแปร

ก. C แทนเมตริกซ์ของเวลาใน TSP

$C = [C(i , j)]$

ข. t แทนเซตของเส้นทางที่เป็นคู่ node กัน

$t = [(i_1 , i_2) (i_2 , i_3) ..... (i_{n-1} , i_n) (i_n , i_1)]$

ค. Z (t) แทนผลรวมของเวลา ในเส้นทาง t

$Z(t) = \sum_{(i,j)\in t} c(i,j)$

ง. X,Y ,$\overline{Y}$ แทนคู่ node ต่าง ๆ โดยที่ X แทนสับเซตคู่ node ของเส้นทาง, Y แทนคู่ node คู่ใหม่ที่ไปได้ , $\overline{Y}$ แทนคู่ node อื่นที่ไม่ใช่คู่ node Y ซึ่งเป็นทางที่ห้ามไป

จ. w (X) แทนเวลาของเส้นทาง x โดยที่ Z (t) > = w (X)

ฉ. Z_o แทนเวลาของเส้นทางที่เหมาะสมที่สุด

2.3 Lower bounds

จากตัวอย่างเมตริกซ์ระยะเวลาจำนวน 6 node (จุดที่ 1 เป็น depot ซึ่งหมายถึงจุดเริ่มต้นของการเดินทางและเป็นจุดสุดท้ายของการเดินทางเมื่อทำการเดินทางครบ 1 รอบหรือครบทุกจุดแล้ว, จุดที่ 2 - 6 เป็นจุดเก็บขน) ในรูปที่ 3.3 ซึ่งมีเส้นทาง คือ t = [(1,3) , (3,2) , (2,5) , (5,,6), (6,4) , (4,1)] จะมีระยะเวลาเท่ากับ 43+13+30+5+9+21 = 121 ซึ่งเมตริกซ์ระยะเวลานี้สามารถที่จะลดค่าตัวเลขที่อยู่ภายในลงได้ เพื่อที่จะให้ง่ายต่อการคำนวณโดยที่ในแต่ละแถวให้นำค่าที่น้อยที่สุดลบออกจากค่าทุกค่าในแถวนั้น ส่วนทางด้านคอลัมน์ก็ให้ทำเช่นเดียวกับแถว จากนี้ให้ตรวจสอบดูว่าทุกแถวและทุกคอลัมน์ จะต้องมีเลข 0 อยู่อย่างน้อย 1 ตัว และผลรวมของค่าที่ถูกลดในแต่ละแถวและคอลัมน์จะแทนด้วย h หรือเรียกว่า lower bounds ซึ่งจะทำให้ได้สมการที่ 3.1

$$Z(t) = h + Z_1(t) \qquad \text{..... (3.1)}$$

โดยที่ Z_1 (t) เป็นระยะเวลาหลังการลดเมตริกซ์ ดังแสดงในรูปที่ 3.4 ซึ่งจะมีค่า h = 48 นั่นคือ Z (t) > = 48 สำหรับทุกค่า t

จากรูปที่ 3.4 จะสังเกตเห็นว่ามีวงกลมที่มีตัวเลขอยู่ภายในเหนือเลข 0 ค่าที่อยู่ในนั่นเรียกว่า θ (i,j) เป็นค่าที่เกิดจากผลรวมของค่าที่น้อยที่สุดในแถวและคอลัมน์ที่คู่ node (i,j) โดยจะไม่

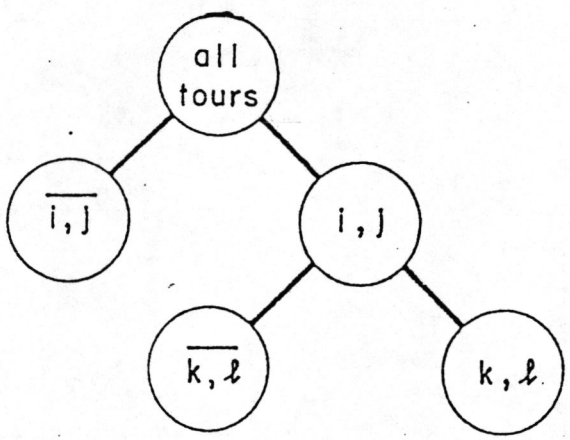
พิจารณาค่าของคู่ node (i,j) เช่น คู่ node (1,4) ค่าตัวเลขที่น้อยที่สุดในแถวคือ 10 ส่วนในคอลัมน์คือ 0 เพราะฉะนั้นผลรวมจึงเท่ากับ 10 ซึ่งเป็นค่าของ θ (1,4)

From \ To	1	2	3	4	5	6
1	∞	27	43	16	30	26
2	7	∞	16	1	30	25
3	20	13	∞	35	5	0
4	21	16	25	∞	18	18
5	12	46	27	48	∞	5
6	23	5	5	9	5	∞

รูปที่ 3.3 แสดงเมตริกซ์ของ 6 node

	1	2	3	4	5	6
1	∞	11	27	0 (10)	14	10
2	1	∞	15	0 (1)	29	24
3	15	13	∞	35	5	0 (5)
4	0 (1)	0 (0)	9	∞	2	2
5	2	41	22	43	∞	0 (2)
6	13	0 (0)	0 (9)	4	0 (2)	∞

รูปที่ 3.4 แสดงเมตริกซ์หลังการลดแล้ว



รูปที่ 3.5 แสดงการ branching

2.4 Branching

Branching เป็นการเลือกเส้นทางคู่ node ที่จะไป โดยจะมีการแตกจากคู่ node เดิมออกไปเป็นคู่ node ใหม่เหมือนกิ่งต้นไม้ ดังรูปที่ 3.5 ซึ่งจะเริ่มจาก "all tours" แตกออกไปเป็นคู่ node (i,j) และ $\overline{(i,j)}$ โดยคู่ node (i,j) แทน all tours ที่มี (i,j) ส่วน $\overline{(i,j)}$แทน all tours ที่ไม่มี (i,j)

เมื่อทำการ branching ต่อไปจาก (i,j) ก็จะได้ (k,l) และ $\overline{(k,l)}$ โดยที่ (k,l) แทน all tours ที่รวมทั้ง (i,j) และ (k,l) ส่วน $\overline{(k,l)}$ แทน all tours ที่รวม (i,j) แต่ไม่รวม (k,l)

2.5 ขั้นตอนการหาเส้นทาง

ขั้นตอนในการหาจะเป็นไปตามรูปที่ 3.6 และใช้ตัวอย่างในรูปที่ 3.3 ซึ่งมีรายละเอียดดังนี้

2.5.1 เริ่มใส่ค่าระยะเวลาลงในเมตริกซ์ ให้ X สำหรับแทน all tours

2.5.2 ลดค่าในเมตริกซ์และให้ค่า X เท่ากับ lower bound (w(X)) นั่นคือ X=48 ดังรูปที่ 3.7

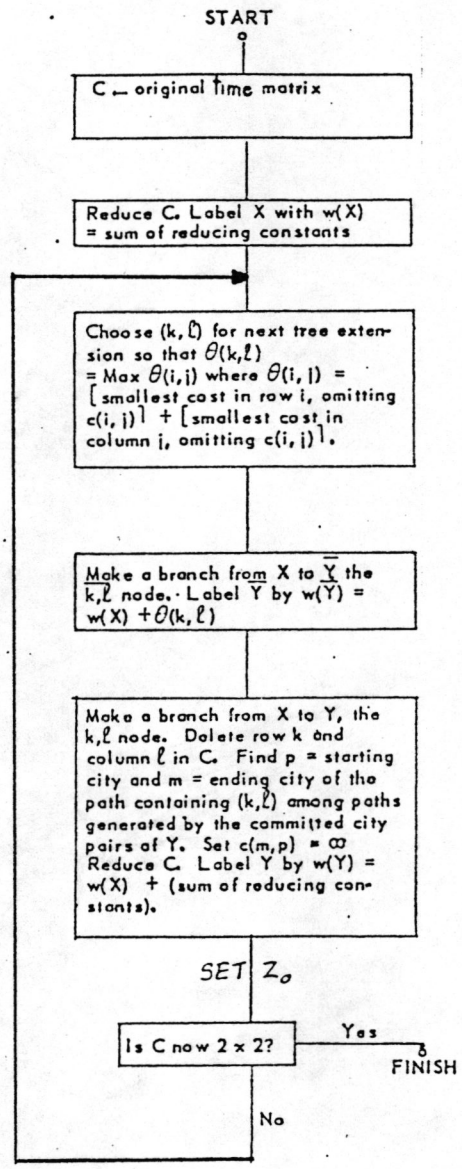
2.5.3 เลือก (k,l) โดยการดูจากคู่ node ที่มี θ (i,j) มากที่สุด ซึ่งจากตัวอย่างในรูปที่ 3.4 จะเลือก (1,4)

2.5.4 branching X ออกเป็น $\overline{Y} = \overline{(1,4)}$ โดยที่ w(Y) = w(X) + (k,l) ในตัวอย่างรูปที่ 3.7 Y จะมีค่าเท่ากับ 48+10 = 58

2.5.5 branching X ออกเป็น Y = (1,4) ในการที่จะหาค่า w(Y) จะต้องทำตามขั้นตอนดังนี้

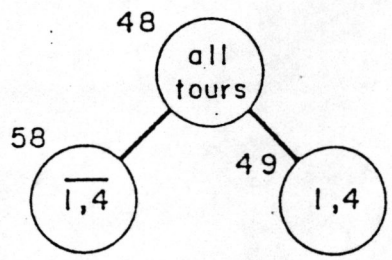
- ตัดแถว k และคอลัมน์ l ในเมตริกซ์เดิมออก จากตัวอย่างตัด k=1 และ l=4 ออกดังตัวอย่างรูป 3.8

- นำคู่ node ที่ได้เลือกมาแล้วทั้งหมดมาเรียงให้เป็นเส้นทางกัน โดยกำหนดให้ node ที่เริ่มต้นเท่ากับ p และ node สุดท้ายเท่ากับ m (อาจเป็นไปได้ที่ p=k หรือ m=l หรือทั้ง 2 อย่าง) นั่นหมายความว่าคู่ node (m,p) เป็นเส้นทางที่ห้ามไป ดังนั้น จึงต้องกำหนดให้ค่า c(m,p) = ∞ จากตัวอย่างมีเพียงคู่ node เดียวที่เลือกมา คือ (1,4) เพราะฉะนั้นค่า (m,p) = (4,1) จึงต้องกำหนดให้ค่า c(4,1) = ∞ ดังตัวอย่างรูป 3.8 สำหรับในกรณีที่ทำการ branching ต่อไปจนมีคู่ node คู่ใหม่ถูกเลือกขึ้นมา เช่นคู่ (2,1) เมื่อนำมาเรียงให้เป็นเส้นทางจะได้ว่า 2- - > 1- - > 4 นั่นคือค่า p=2 , m=4 และ (m,p) = (4,2) ดังนั้นจึงกำหนดให้ค่า c(4,2) = ∞



รูปที่ 3.6 แสดง Flow Chart ของ TSP

(Anen Aung - Aphinant , 1980)



รูปที่ 3.7 แสดงการ branching ครั้งแรก

	1	2	3	4	5	6
~~1~~	~~∞~~	~~11~~	~~27~~	~~0~~	~~14~~	~~10~~
2	0 +	∞	14 ~~15~~	0	28 ~~29~~	23 ~~24~~
3	15	13	∞	35	5	0
4	∞	0	9	∞	2	2
5	2	41	22	43	∞	0
6	13	0	0	4	0	∞

รูปที่ 3.8 แสดงการตัดแถวและคอลัมน์ในเมตริกซ์ออก

- เมื่อกำหนดค่า c(m,p) = ∞ แล้วให้ทำการลดค่าในเมตริกซ์ที่เหลือโดยวิธีการเดียวกับหัวข้อ 2.3 ดังตัวอย่างรูปที่ 3.8

- ผลรวมของค่าที่ลดไปจะนำไปบวกกับ w(x) เป็นค่าของ w(Y) นั่นคือ w(Y) = w(X) + h จากตัวอย่าง w(Y) = 48 + 1 = 49

2.5.6 กำหนดให้ค่า Z_0 = w(Y)

2.5.7 ตรวจสอบว่าเมตริกซ์ที่เหลือเป็นเมตริกซ์ 2 * 2 หรือไม่

- ถ้าเป็น ให้ข้ามไปทำขั้นตอนที่ 2.5.9

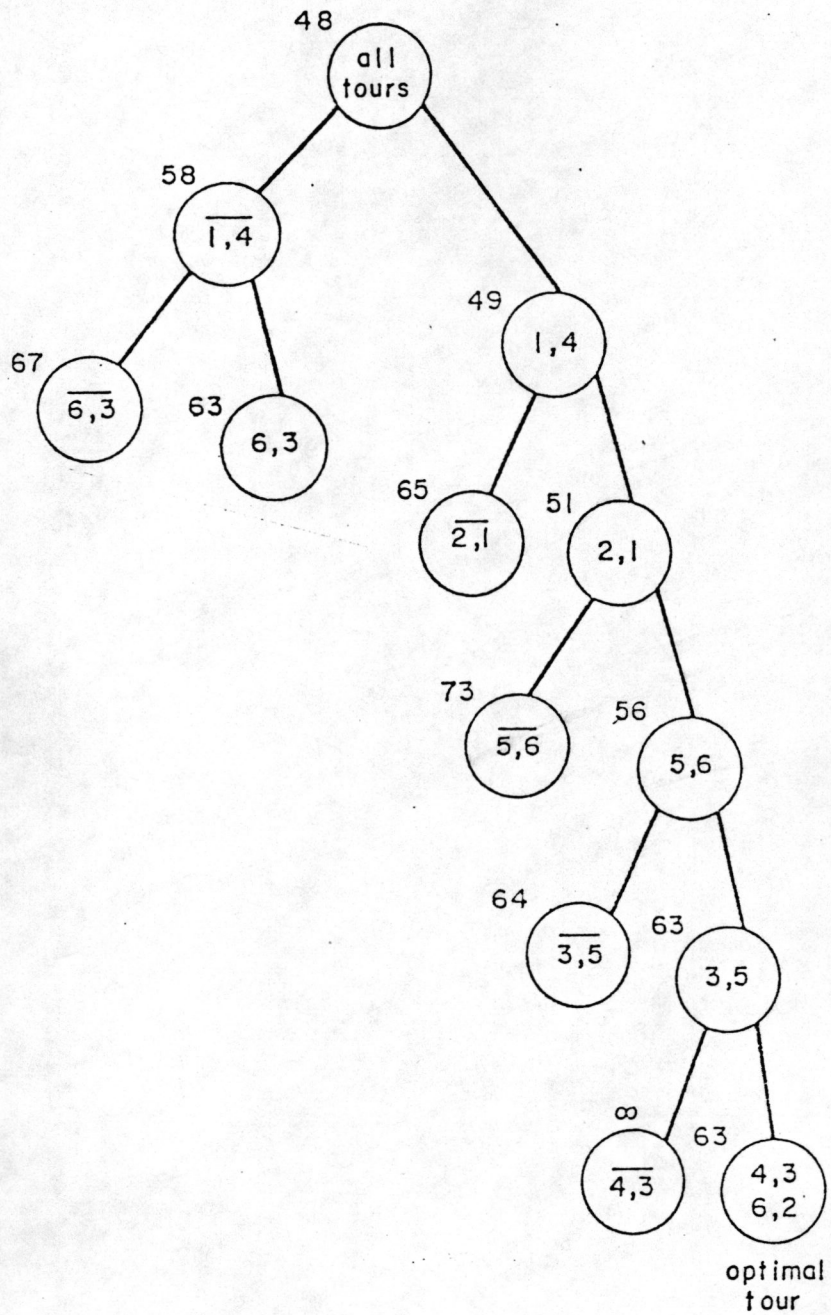
- ถ้าไม่เป็น ให้ทำขั้นตอนที่ 2.5.8 ต่อ

2.5.8 ทำการ branching ต่อจาก Y โดยย้อนกลับไปที่ขั้นตอนที่ 3

2.5.9 เมื่อเหลือเมตริกซ์ 2 * 2 แล้ว จะเหลือคู่ node 2 คู่ที่สามารถนำมาเชื่อมเป็นเส้นทางกับคู่ node ที่เลือกไปก่อนแล้วได้

จากตัวอย่างเมื่อทำครบทุกขั้นตอนจะได้เส้นทาง 1 - - > 4 - - > 3 - - > 5 - - > 6 - - > 2 - - > 1 ดังตัวอย่างรูปที่ 3.9

สำหรับในการที่จะหาระยะเวลา (ระยะทาง,ค่าใช้จ่าย) ระหว่าง node (จุดเก็บขน) เพื่อที่จะนำไปใส่ในแต่ละช่องของเมตริกซ์นั้น ค่าที่ได้ออกมาอาจจะมีได้หลายค่า เนื่องจากว่าเส้นทางที่จะไประหว่าง node นั้นสามารถที่จะไปได้หลายเส้นทาง ดังนั้นจึงสามารถที่จะกำหนดเส้นทางที่จะไประหว่าง node เองได้ หรืออาจจะใช้การวิจัยดำเนินงานเข้ามาช่วยหาก็ได้ โดยวิธีการนี้ เรียกว่า Shortest Time Paths ซึ่งจากวิธีการนี้จะได้เส้นทางระหว่าง node ที่มีระยะเวลาน้อยที่สุด



รูปที่ 3.9 แสดงผลลัพธ์ของ TSP

3. Shortest Time Paths

Shortest Time Paths (Papacostas และ Prevedouros, 1993 อ้างถึงใน Moore, 1957) เป็นการหาเส้นทางที่สั้นที่สุดจากจุดเริ่มต้นไปถึง node อื่นทุก node ต้องกำหนด node ทั้งที่เป็นจุดเก็บขนและจุดแยกของถนน ดังตัวอย่างในรูปที่ 3.10 จุดที่ 1 เป็น depot จุดที่ 2-5 เป็นจุดเก็บขนส่วนจุดที่ 6 - 14 เป็นแยกของถนน

ขั้นตอนในการหา Shortest Time Paths

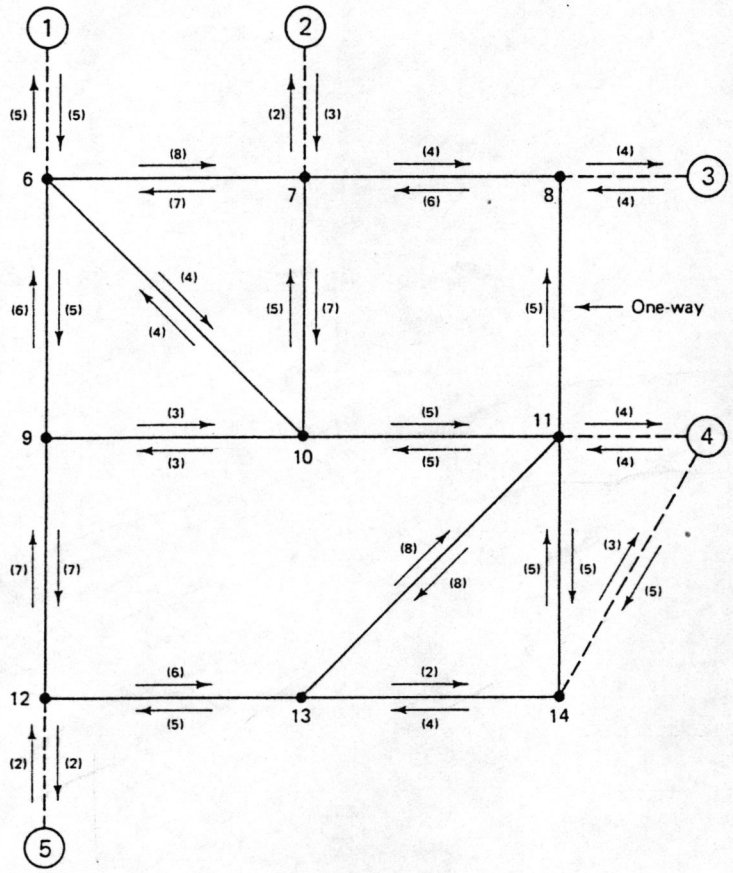
ขั้นตอนในการหาจะใช้ตัวอย่างในรูปที่ 3.10 และจะอธิบายขั้นตอนตามรูปที่ 3.11 ดังนี้

3.1 เริ่มจากจุดที่ 1 ลากเส้นทางที่พุ่งออกจาก node นี้ทุกเส้นทาง จากตัวอย่างจะพบว่ามีเส้นทางจากจุดที่ 1 ไปจุดที่ 6 เส้นทางเดียวเท่านั้น

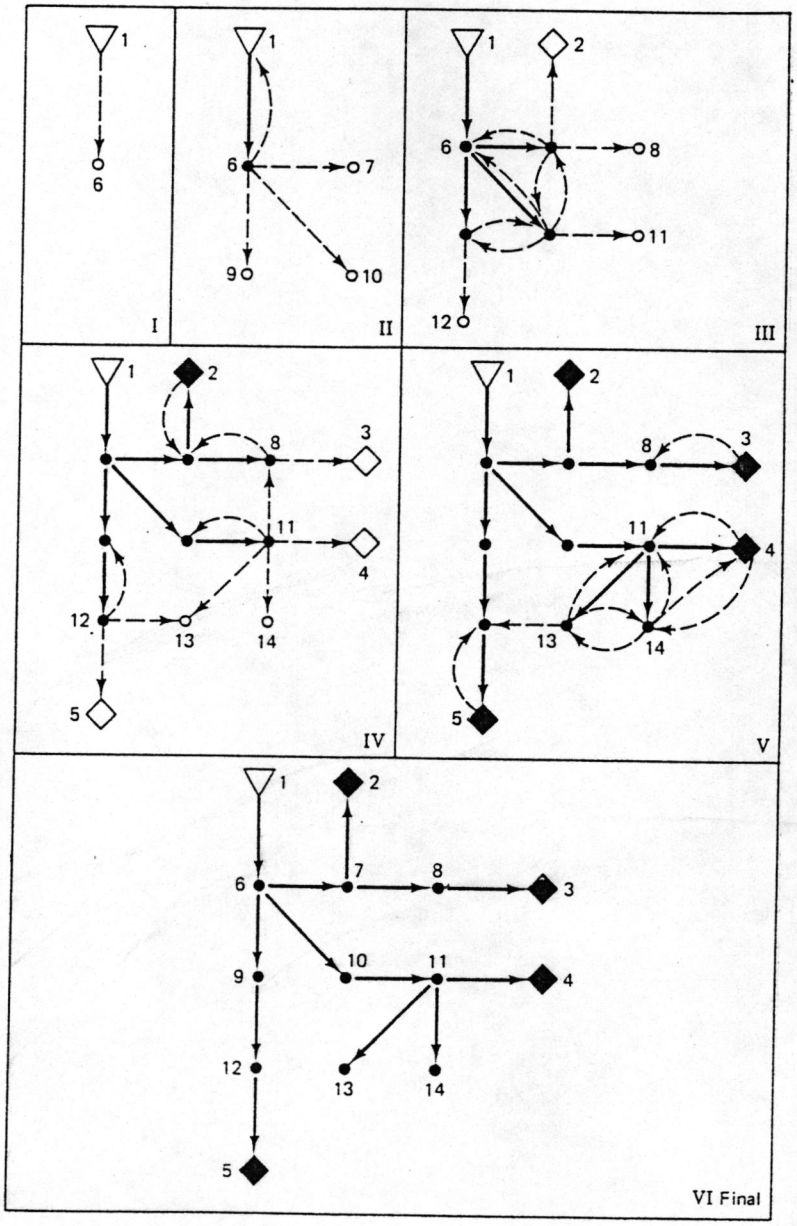
3.2 คำนวณระยะเวลา จากจุดที่ 6 (i=1) (j=6) ตามตารางที่ 3.1 โดยจุดที่ 1 มีค่าเท่ากับ 0 และค่าระหว่างจุดที่ 1 กับจุดที่ 6 มีค่าเท่ากับ 5 เมื่อนำมาบวกกันจะได้ค่าเท่ากับ 5 นำค่าไปเปรียบเทียบกับจุด j ที่ถูกกำหนดให้มีค่าเท่ากับ ∞ ทุก ๆ ค่า j เป็นค่าเริ่มต้นจะพบว่า $5 < \infty$ จึงสามารถ accept ได้ และจุดที่ 6 ก็จะเปลี่ยนไปมีค่าเท่ากับ 5

3.3 เริ่มจากจุดที่ 6 ลากเส้นพุ่งออกจาก node นี้ทุกเส้นทาง จะได้ออกไป 3 ทางคือไปที่จุด 7,9 และ 10 เปรียบเทียบกับค่า j ทั้งสาม (7,9,10) กับค่า j ที่กำหนดไว้ในตอนแรก ตามตารางที่ 3.1 จะเห็นว่า accept หมดทุกจุด และจุดที่ 7,9 และ 10 ก็จะเปลี่ยนไปมีค่าเท่ากับ 13 , 10 และ 9 ตามลำดับ

3.4 เริ่มจากจุดทั้ง 3 จุด คือ 7, 9, 10 หาเส้นทางที่พุ่งออกจากจุดทั้ง 3 จุด เช่นเดียวกับขั้นตอนที่ 1 และ 2 เปรียบเทียบค่าที่คำนวณได้ จากตารางที่ 3.1 จะเห็นว่าค่าของจุด j = 10 มี 2 ค่าเมื่อนำมาเปรียบเทียบกันค่าจากจุดที่ 7 ไปจุดที่ 10 มีค่ามากกว่าค่าจากจุดที่ 9 ไปจุดที่ 10 ดังนั้นจึงตัดค่าที่มากกว่าออก (reject) จากนั้นจึงเปรียบเทียบค่าที่เหลือกับค่าที่อยู่ในขั้นตอน N-1 นั่นคือขั้นตอนที่ 3 โดยดูจากจุด j ที่เหมือนกันเท่านั้น จากตารางที่ 3.1 จุด j จะเหมือนกัน 2 จุด คือ j = 7 และ j = 9 ซึ่งค่า j ทั้งสองค่าในขั้นตอนที่ 4 จะมีค่ามากกว่าค่า j ทั้งสองค่า ในขั้นตอนที่ 3 จึงต้อง reject j = 7 และ j = 9 ในขั้นตอนที่ 4 ออก ส่วนจุด j ที่เหลือคือ 2, 8, 11 และ 12 จะมีค่าน้อยกว่าค่า j ที่กำหนดไว้ในตอนแรก ดังนั้นค่า j ของทั้ง 4 จุดจึง accept



รูปที่ 3.10 แสดงผังตัวอย่างที่เป็นจุดเริ่มต้น, จุดเก็บขน และจุดแยกของถนน
(ค่าในวงเล็บ คือ ระยะเวลาที่ใช้ในการเดินทางระหว่าง node)



รูปที่ 3.11 แสดงขั้นตอนการพุ่งออกจาก node ในแต่ละ node

ตารางที่ 3.1 แสดงการคำนวณของ Shortest Time Paths

Stage N	Links i	j	Compute new path impedance	Compare to tree table stage $N-1$	Decision
II	1	6	0 + 5 = 5	$5 < \infty$	Accept
III	6	7	5 + 8 = 13	$13 < \infty$	Accept
		9	5 + 5 = 10	$10 < \infty$	Accept
		10	5 + 4 = 9	$9 < \infty$	Accept
IV	7	2	13 + 2 = 15	$15 < \infty$	Accept
		8	13 + 4 = 17	$17 < \infty$	Accept
		10	13 + 7 = 20 Reject	----------	Reject
	9	10	10 + 3 = 13	$13 > 9$	Reject
		12	10 + 7 = 17	$17 < \infty$	Accept
	10	7	9 + 5 = 14	$14 > 13$	Reject
		9	9 + 3 = 12	$12 > 10$	Reject
		11	9 + 5 = 14	$14 < \infty$	Accept
V	8	3	17 + 4 = 21	$21 < \infty$	Accept
	11	4	14 + 4 = 18	$18 < \infty$	Accept
		8	14 + 5 = 19	$19 > 17$	Reject
		13	14 + 8 = 22	$22 < \infty$	Accept
		14	14 + 5 = 19	$19 < \infty$	Accept
	12	5	17 + 2 = 19	$19 < \infty$	Accept
		13	17 + 6 = 23 Reject	----------	Reject
VI	All links emanating from nodes 3, 4, 5, 13, and 14 are rejected; the list is now empty and the procedure ends.				

ตารางที่ 3.2 แสดงผลสรุปในการหา Shortest Time Paths

Node	Total impedance to node j						Node preceding j					
(j)	I	II	III	IV	V	VI	I	II	III	IV	V	VI
1	0						—					
2	∞			15						7		N
3	∞				21						8	O
4	∞				18						11	
5	∞				19						12	
6	∞	5						1				C
7	∞		13						6			H
8	∞			17						7		A
9	∞		10						6			N
10	∞		9						6			G
11	∞			14						10		E
12	∞			17						9		
13	∞				22						11	
14	∞				19						11	

3.5 ทำเหมือนกับในขั้นตอนที่ 4 โดยจะหาเส้นทางพุ่งออกจากจุดที่ 8, 11, 12 เท่านั้นส่วนจุดที่ 2 ไม่สามารถหาเส้นทางที่พุ่งออกได้อีก เมื่อคำนวณค่าได้แล้วนำจุด j ภายในขั้นตอนที่ 5 ที่เหมือนกันมาเปรียบเทียบกัน รวมทั้งนำจุด j ในขั้นตอนที่ 5 ไปเปรียบเทียบกับจุด j ในขั้นตอนที่ 4 ที่เหมือนกันด้วยซึ่งจะพบว่าจุด j = 8, 13 ในขั้นตอนที่ 5 จะถูก reject ออก

3.6 เมื่อตรวจสอบดูจะเห็นว่าไม่สามารถหาจุดที่มีเส้นทางพุ่งออกไปได้อีก จึงนำค่าต่าง ๆ มาสรุปได้ตามตารางที่ 3.2 ซึ่งจะได้เส้นทางที่สั้นที่สุดจากจุดที่ 1 ไปถึงทุก ๆ จุดได้ โดยค่าที่จะเลือกมาใส่ในช่องเมตริกซ์แถวที่ 1 ของวิธี TSP จะประกอบด้วย 4 ค่า ดังตารางเมตริกซ์ข้างล่างนี้

TO

FROM					
	∞	15	21	18	19
		∞			
			∞		
				∞	
					∞

3.7 จากนั้นจึงเลือกจุดที่เหลือทั้ง 4 จุด (จุดที่ 2 - 5) ผลัดกันมาเป็นจุดเริ่มต้นและใช้วิธีที่กล่าวมาทำซ้ำอีกที จนกระทั่งจุดทั้ง 4 เป็นจุดเริ่มต้นครบทุกจุด โดยถ้าเลือก

-จุดที่ 2 เป็นจุดเริ่มต้นค่าที่ได้ทั้ง 4 ค่า จะใส่ในช่องเมตริกซ์แถวที่ 2

-จุดที่ 3 เป็นจุดเริ่มต้นค่าที่ได้ทั้ง 4 ค่า จะใส่ในช่องเมตริกซ์แถวที่ 3

-จุดที่ 4 เป็นจุดเริ่มต้นค่าที่ได้ทั้ง 4 ค่า จะใส่ในช่องเมตริกซ์แถวที่ 4

-จุดที่ 5 เป็นจุดเริ่มต้นค่าที่ได้ทั้ง 4 ค่า จะใส่ในช่องเมตริกซ์แถวที่ 5

ซึ่งเมื่อนำค่าที่หาได้มาใส่ในช่องเมตริกซ์ครบทุกช่องแล้วก็จะได้เป็นเมตริกซ์ 5*5 ออกมา เพื่อนำไปใช้ในวิธี TSP ต่อไป

Multiroute Problems

เป็นวิธีการที่ใช้จัดเส้นทางในกรณีที่มีรถหลายคันหรือรถ 1 คันแต่วิ่งหลายเที่ยว โดยจะต้องทำการจัดแบ่งงานสำหรับรถแต่ละคันหรือแต่ละเที่ยวในพื้นที่ที่รับผิดชอบ ในการที่จะจัดเส้นทางภายในพื้นที่ใหญ่ ๆ นั้นสามารถที่จะทำได้ 2 ทางคือ

ก. แบ่งพื้นที่ใหญ่ออกเป็นพื้นที่ย่อย ๆ จากนั้นจึงหาเส้นทางภายในแต่ละพื้นที่ย่อยนั้นซึ่งเรียกวิธีการนี้ว่า " Cluster first, Route second"

ข. หาเส้นทางทั้งพื้นที่ใหญ่นั้นก่อน จากนั้นจึงแบ่งเส้นทางย่อย ๆ ตามจำนวนรถ หรือจำนวนเที่ยว เรียกวิธีนี้ว่า "Route first, Cluster second"

1. Multiroute edge - covering problems

ในหัวข้อนี้จะเป็นการจัดเส้นทางแบบ "Route first, Cluster second" คือการหาเส้นทางในพื้นที่ทั้งหมดก่อนแล้วจึงแบ่งออกเป็นเส้นทางย่อย ๆ ตามจำนวนรถ หรือจำนวนเที่ยว แต่การที่จะได้เส้นทางที่ดีที่สุดหรือไม่นั้น จะต้องขึ้นอยู่กับความรู้ความสามารถและประสบการณ์ของผู้วิเคราะห์ในการจัดแบ่งเส้นทาง

2. Multiroute node - covering problems

ในการแก้ปัญหาแบบ Multiroute node - covering นี้สามารถที่จะแบ่งออกได้เป็น 3 แบบคือ

ก. The m - TSP problem

ข. The single - depot VRP problem

ค. The multidepot VRP problem

โดยการแก้ปัญหาแบบข้อ 1 จะเป็นแบบไม่มีเงื่อนไขอื่นใดนอกจากรถจำนวนหลายคัน หรือจำนวนเที่ยวที่มากกว่า 1 เที่ยว ซึ่งเมื่อหาเส้นทางออกมาแล้วจะต้องได้ m เส้นทาง (m = จำนวนรถหรือจำนวนเที่ยว) โดยมีวัตถุประสงค์คือการหาระยะทาง (ระยะเวลา, ค่าใช้จ่าย) รวมที่น้อยที่สุดของทั้ง m เส้นทาง

ส่วนในการแก้ปัญหาแบบข้อ 2 และข้อ 3 นั้น จะเป็นแบบที่มีการกำหนดเงื่อนไขขึ้นมา เช่น ความจุของรถหรือระยะทางที่ไปได้มากที่สุด ซึ่งเรียกว่าเป็นปัญหาแบบ VRP (vehicle routing problem) VRP นี้อาจจะมีได้ทั้ง single หรือ many origins - destinations ซึ่งเรียกว่า depot วัตถุประสงค์ของการแก้ปัญหาแบบนี้คือ ผลรวมของระยะทาง (ระยะเวลา, ค่าใช้จ่าย) หรือค่าใช้จ่ายทั้งหมดมีค่าน้อยที่สุด

2.1 m - TSP problem

m - TSP problem (Larson และ Odoni, 1981) นี้จัดเป็น single origin - destination problem

ขั้นตอนในการหาเส้นทาง

กำหนดให้ $d(v_1, x) = d(v_2, x) = \ldots\ldots = d(v_m, x) = d(v, x)$

โดยที่ v_m = depot ของรถคันที่ m หรือเที่ยวที่ m

x = จุดใดจุดหนึ่งในเส้นทาง

และ $d(v_i, v_j) = \infty$ สำหรับ $i, j = 1, 2, \ldots\ldots, m$

2.1.1 กำหนดจำนวน salesman หรือ จำนวนเที่ยว

2.1.2 เพิ่มแถวและคอลัมน์ในเมตริกซ์เช่น เมตริกซ์ระยะทางในรูปที่ 3.12 (ก) เมื่อกำหนดให้มีจำนวน salesman เท่ากับ 2 ดังนั้นจึงต้องเพิ่ม V เป็น V_1 และ V_2 เหมือนในรูปที่ 3.12(ข)

2.1.3 หาเส้นทางโดยการใช้ TSP จะได้ออกมาดังรูปที่ 3.12 (ค)

2.1.4 จับจุด V มารวมกันก็จะได้เส้นทางออกมาดังรูปที่ 3.12 (ง)

2.2 Single - Depot VRP

Single - Depot VRP (Larson และ Odoni อ้างถึงใน Clarke และ Weight, 1981) วิธีนี้ใช้ในการหาเส้นทางที่มีเพียง 1 depot โดยที่มีการสร้างเงื่อนไขขึ้นมาด้วย เช่น ความจุสูงสุดของรถ หรือ ระยะทางที่รถสามารถเดินทางได้มากที่สุดและสิ้นสุดที่จุด depot (D)

ขั้นตอนในการหาเส้นทาง

2.2.1 คำนวณ $s(i,j) = d(D,i) + d(D,j) - d(i,j)$ สำหรับทุก ๆ คู่ (i,j) โดยที่ s(i,j) คือ ระยะทางที่เดินทางทั้งหมดที่ลดลงเมื่อรถคันหนึ่งให้บริการจุด 2 จุด (เรียกว่า i และ j) ในเที่ยวเดียวกัน

2.2.2 จัดลำดับ s(i,j) โดยเรียงจากมากไปหาน้อย

2.2.3 นำ (i,j) ต่าง ๆ มาเชื่อมต่อกันเป็นเส้นทาง โดยดูว่าคู่ (i,j) ที่พิจารณาอยู่ตรงกับข้อใดดังต่อไปนี้

ก. ถ้า i หรือไม่ก็ j ไม่มีอยู่ในเส้นทางให้เริ่มต้นเป็นเส้นทางใหม่

ข. ถ้ามี i หรือ j จุดใดก็ได้มีอยู่ในเส้นทางทางด้านนอกให้รวมอีกจุดที่เหลือเข้าไปอยู่ในเส้นทางนั้นด้วย

ค. ถ้ามี i และ j อยู่กันคนละเส้นทางที่ต่างกัน และจุดทั้ง 2 อยู่ทางด้านนอกของเส้นทางให้นำทั้ง 2 เส้นทางมารวมกัน

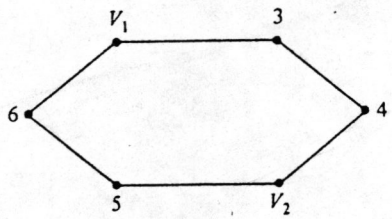
2.2.4 ตรวจสอบว่าเส้นทางแต่ละเส้นทางอยู่ภายใต้เงื่อนไขหรือไม่

	V	3	4	5	6
V	—	28	57	20	45
3	28	—	47	46	73
4	57	47	—	76	85
5	20	46	76	—	40
6	45	73	85	40	—

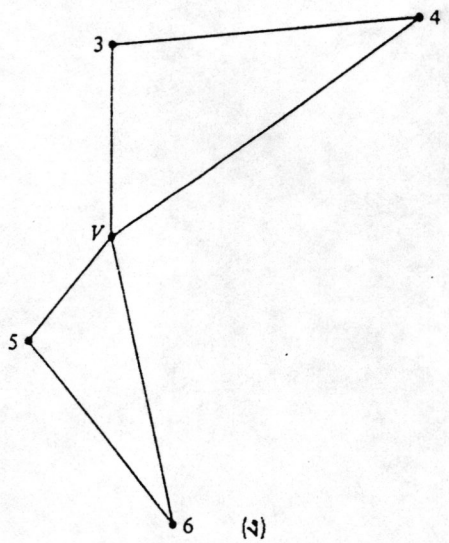
(ก)

	V_1	V_2	3	4	5	6
V_1	∞	∞	28	57	20	45
V_2	∞	∞	28	57	20	45
3	28	28	∞	47	46	73
4	57	57	47	∞	76	85
5	20	20	46	76	∞	40
6	45	45	73	85	40	∞

(ข)



(ค)



(ง)

รูปที่ 3.12 แสดงตัวอย่างในการหา m-TSP

2.2.5 ถ้า s(i,j) ยังมีไม่ครบทุกจุด ให้กลับไปทำที่ขั้นตอนที่ 3 แต่ถ้าครบแล้วให้หยุดการหาเส้นทาง

ข้อจำกัดของวิธีการนี้ คือ ค่าจาก (i,j) และ (j,i) จะต้องมีค่าเท่ากัน

<u>ตัวอย่างในการหาเส้นทาง</u>

ตารางที่ 3.3 แสดงตัวอย่างในการหา single - depot VRP

(ก) Distances (below diagonal) and savings (above the diagonal).

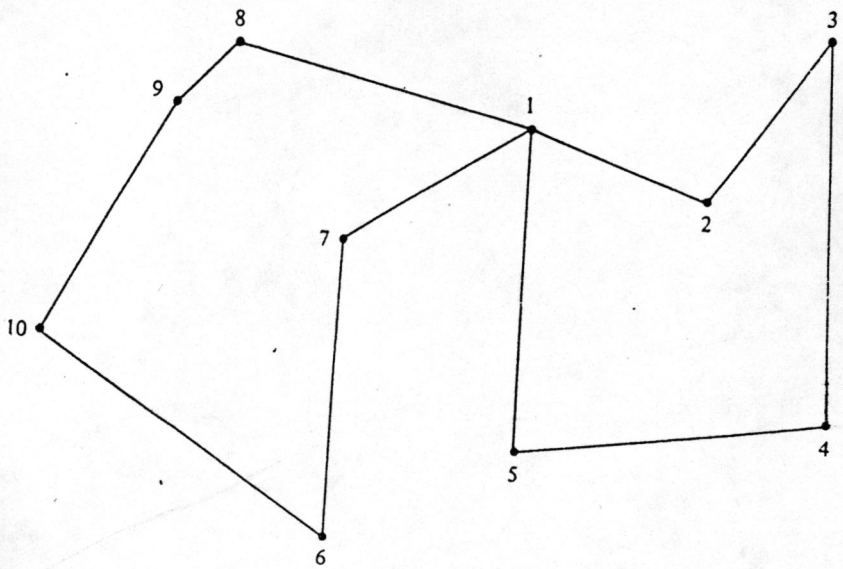
	1	2	3	4	5	6	7	8	9	10
1		—	—	—	—	—	—	—	—	—
2	25		39	48	25	18	5	0	1	5
3	43	29		48	14	8	0	3	2	23
4	57	34	52		55	47	15	3	6	20
5	43	43	72	45		77	36	19	26	49
6	61	68	96	71	27		50	36	47	86
7	29	49	72	71	36	40		39	46	57
8	41	66	81	95	65	66	31		78	66
9	48	72	89	99	65	62	31	11		83
10	71	91	114	108	65	46	43	46	36	

(ข) Quantities of refuse to be collected at each point for Example

Node	2	3	4	5	6	7	8	9	10
Quantity	4	6	5	4	7	3	5	4	4

(ค) Savings list for Example

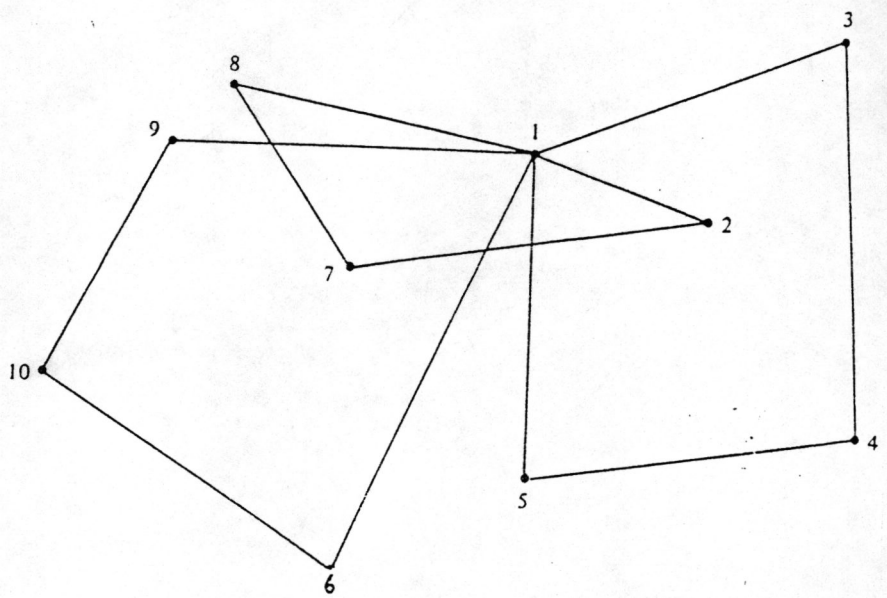
Link	*Savings*	*Link*	*Savings*	*Link*	*Savings*
(6, 10)	86	(4, 6)	47	(4, 7)	15
(9, 10)	83	(7, 9)	46	(3, 5)	14
(8, 9)	78	(7, 8)	39	(3, 6)	8
(5, 6)	77	(2, 3)	39	(4, 9)	6
(8, 10)	66	(5, 7)	36	(2, 7)	5
(7, 10)	57	(6, 8)	36	(2, 10)	5
(4, 5)	55	(5, 9)	26	(3, 8)	3
(6, 7)	50	(2, 5)	25	(4, 8)	3
(5, 10)	49	(3, 10)	23	(3, 9)	2
(3, 4)	48	(4, 10)	20	(2, 9)	1
(2, 4)	48	(5, 8)	19	(2, 8)	0
(6, 9)	47	(2, 6)	18	(3, 7)	0



รูปที่ 3.13 แสดงผลลัพธ์สุดท้ายของ single - depot VRP

จากตารางที่ 3.3 (ค) จะเริ่มจาก (6,10) โดยเส้นทางเริ่มต้นจะประกอบด้วย { 1,6,10,1 } จากนั้นค่าต่อไปคือ (9,10) และ (8,9) ก็จะได้เส้นทาง { 1,6,10,9,8,1 } เมื่อขยายเส้นทางต่อไปก็จะได้ { 1,5,6,10,9,8,1 } ซึ่งเมื่อตรวจสอบกับเงื่อนไขที่ให้ว่าความจุของรถจะต้องไม่เกิน 23 units แต่จากเส้นทางที่ขยายใหม่นี้จะมีเท่ากับ 24 units ดังนั้นจึงต้องตัด (5,6) ออกไป ต่อจากนั้น (8,10) อยู่ในเส้นทางอยู่แล้ว ส่วน (7,10) ใส่เข้าไปไม่ได้เนื่องจาก 10 อยู่ภายในเส้นทาง ต่อไปคู่ (4,5) ไม่สามารถนำไปรวมได้ให้เริ่มต้นเป็นเส้นทางใหม่ก็จะเป็น { 1,4,5,1 } คู่ต่อไปที่นำไปรวมได้คือ (6,7) ซึ่งจะได้ { 1,7,6,10,9,8,1 } กับภาระ 23 units ต่อจากนั้นก็ให้เริ่มต้นหาเส้นทางที่ 2 ซึ่งเมื่อหาเสร็จแล้วจะได้เป็น { 1,2,3,4,5,1 } กับภาระ 19 units เส้นทางที่จะได้ก็จะเป็นแบบในรูปที่ 3.13 และมีผลรวมระยะที่เดินทางทั้งหมดเท่ากับ 397 units ถ้าลองเปลี่ยนความจุของรถมาเป็น 16 units ก็จะได้เส้นทางใหม่ดังรูปที่ 3.14

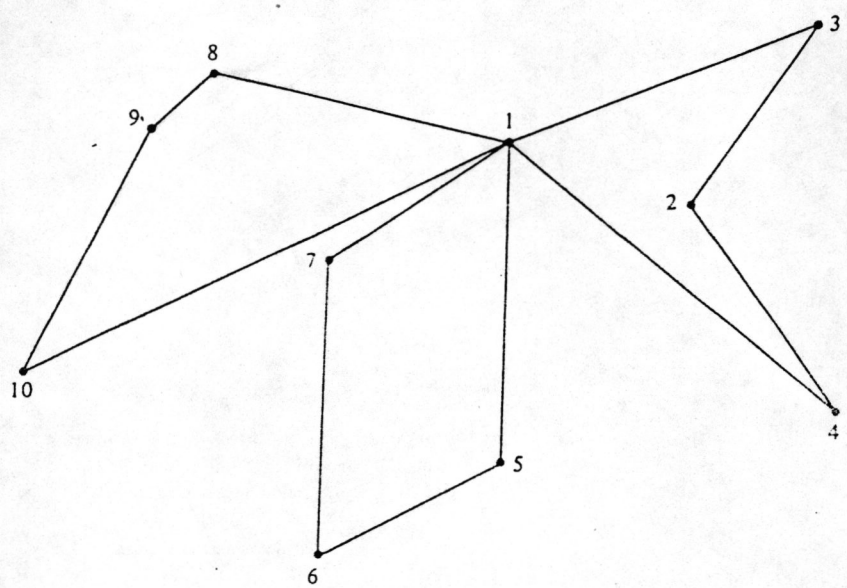
Route	*Load*	*Distance Covered*
{1, 6, 10, 9, 1}	15	191
{1, 3, 4, 5, 1}	15	183
{1, 2, 7, 8, 1}	12	146
		520 = total



รูปที่ 3.14 แสดงผลลัพธ์สุดท้ายของ single - depot VRP ที่ความจุ 16 units

อีกวิธีการหนึ่งที่สามารถแก้ปัญหา Single - depot VRP ได้คือ Sweep Algorithm for the VRP (Larson และ Odoni, 1981) ซึ่งเป็นการแก้ปัญหาแบบ " Cluster first - Route second " วิธีการหาเส้นทางก็โดยการใช้จุด depot เป็นจุดเริ่มต้นจากนั้นก็กวาดทิศทางออกไปเหมือนเข็มนาฬิกา โดยจะหมุนแบบตามเข็มนาฬิกาหรือทวนเข็มนาฬิกาก็ได้ซึ่งจากตัวอย่างข้างต้นที่รถจะมีความจุเท่ากับ 16 units ก็จะหาเส้นทางออกมาได้ดังรูปที่ 3.15

Route	*Load*	*Distance Covered*
{1, 3, 2, 4, 1}	15	163
{1, 5, 6, 7, 1}	14	139
{1, 10, 9, 8, 1}	13	159
		461 = total



รูปที่ 3.15 แสดงตัวอย่างในการหา single - depot VRP แบบ Sweep Algorithm

2.3 Multidepot VRP

Multidepot VRP (Larson และ Odoni, 1981 อ้างถึงใน Gill, 1974)วิธีนี้จะใช้การแก้ปัญหาแบบ "Cluster first, Route second" โดยในตอนแรกจะจัดสรร node ต่าง ๆ ให้อยู่ในแต่ละ depot ตามความเหมาะสม จากนั้นจึงใช้ single - depot VRP แก้ปัญหาในแต่ละ depot

ในการจัดสรร node ต่าง ๆ ให้อยู่ในแต่ละ depot ทำได้โดยคำนวณ r (i) ตามสูตรที่ 3.2

$$r(i) = d'(i) / d''(i) \qquad .....(3.2)$$

โดยที่ d' (i) และ d" (i) เป็นระยะทางจากจุด i ไปยัง depot ที่ใกล้ที่สุดเป็นแห่งแรกและแห่งที่สอง จากนั้นให้กำหนดค่า δ โดยที่ $0 < \delta < 1$ เพื่อให้เปรียบเทียบกับ r (i) ซึ่งถ้า $r(i) \leq \delta$ ก็ให้จุด i อยู่กับ depot ที่ใกล้ที่สุด แต่ถ้า $r(i) > \delta$ ก็ให้เก็บไว้พิจารณาภายหลัง หลังจากที่จุด i ถูกจัดสรรให้ depot หมดแล้ว ก็ให้นำจุด i ที่มีค่า $r(i) > \delta$ มาจัดสรรต่อโดยการเพิ่มจุด i นั้นเข้าไประหว่างจุด j และ k ซึ่งอยู่ใน depot เดียวกัน (D_s) ซึ่งจากการเพิ่มจุด i เข้าไปนี้จะทำให้เส้นทางที่มี D_s เดียวกันนั้นมีค่าเพิ่มขึ้นเท่ากับ $d_{jk}(i) = d(j, i) + d(i, k) - d(j, k)$ จุด i แต่ละจุดจะถูกจัดสรรให้อยู่ร่วมกับ depot ที่มีค่า $d_{jk}(i)$ น้อยที่สุด เมื่อจุด i ถูกจัดสรรครบทุกจุดแล้วให้ใช้ single - depot VRP เพื่อหาเส้นทางในแต่ละ depot

ผลงานงานวิจัยที่เกี่ยวข้องกับการหาเส้นทางเดินรถที่เหมาะสม

1. Fisher และ Jaikumar (1981) ได้คิดค้นวิธีการที่เรียกว่า " Generalized Assignment Approaches" โดยวิธีการนี้จะเป็นการหาเส้นทางแบบ Cluster first,route second และเป็น multiroute แบบ single - depot สำหรับเส้นทางที่หาได้ออกมานั้นจะใช้เป็นเส้นทางเดินรถสำหรับทุกวัน โดยออกวิ่งวันละ 1 เที่ยว เพราะว่าจะเป็นการหาเส้นทางสำหรับลูกค้าที่ใช้บริการทุกวัน โดยขั้นตอนในการหาจะแบ่งออกเป็น 2 ขั้นตอน คือ

1.1 การจัดสรรลูกค้า i ต่อรถ k

ก. แบบจำลองทางคณิตศาสตร์ของการจัดสรร

LP Model :

Objective Min $\sum_{i \in S} \sum_{k=1}^{W} b_{ik} Y_{ik}$ ..... (3.3)

Subject To $\sum_{k \in Q_i} Y_{ik} = 1$, ทุก $i \in S$ ..... (3.4)

$\sum_{i \in S} d_i Y_{ik} \leq q_k$, $k = 1,2,.....,W$ ..... (3.5)

$Y_{ik} \in \{0,1\}$, ทุก i และ k ..... (3.6)

โดยที่

b_{ik} = ระยะเวลาที่ใช้ในการจัดสรรลูกค้า i ต่อรถ k

Y_{ik} = $\begin{cases} 1 & \text{ถ้ารถ k ไปที่ลูกค้า i} \\ 0 & \text{อื่น ๆ} \end{cases}$

d_i = ปริมาณความต้องการของลูกค้า i

q_k = ความจุของรถ k

W = จำนวนรถทั้งหมด

S = เซ็ตของลูกค้าทั้งหมด

Q_i = เซ็ตของรถที่สามารถให้บริการที่จุด i ได้

แบบจำลองนี้จัดเป็นแบบจำลองแบบ Linear Programming Model (LP Model)

ข. ความหมายของสมการและอสมการ

สมการที่ 3.3 หมายถึง ผลรวมระยะทางทั้งหมดของรถทุกคันที่มีค่าน้อยที่สุดที่ใช้ใน 1 วัน โดยที่ได้มาจากการจัดสรรลูกค้าแต่ละรายให้กับรถแต่ละคัน

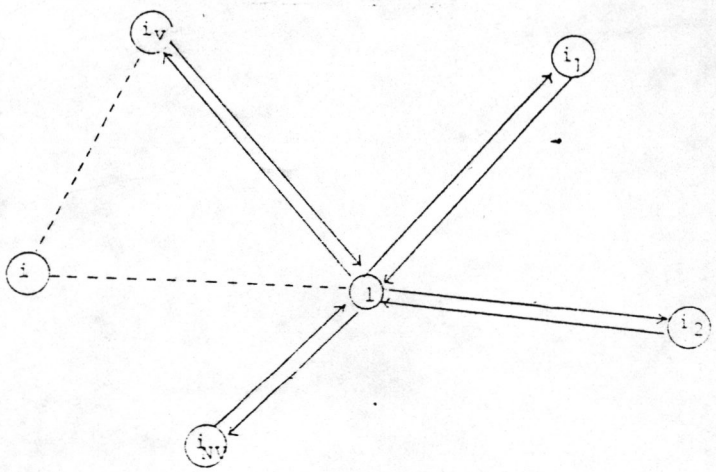
สมการที่ 3.4 หมายถึง ลูกค้า I แต่ละรายจะถูกจัดสรรให้กับรถเพียงคันเดียวเท่านั้น

สมการที่ 3.5 หมายถึง ปริมาณความต้องการของลูกค้า i ที่รถ k รับผิดชอบอยู่รวมกันทั้งหมดจะต้องมีปริมาณไม่เกินความจุของรถ k

สมการที่ 3.6 หมายถึง ตัวแปร Y_{ik} จะมีค่าเท่ากับ 0 หรือ 1 เท่านั้น

ค. <u>การหาค่า b_{ik}</u>

โดยทั่ว ๆ ไปเส้นทางที่ได้จะถูกหาโดยระยะทางรวมที่มีค่าน้อยที่สุด ดังนั้นค่า b_{ik} จึงเป็นค่าที่ใช้ประเมินระยะทางของรถ k ที่เดินทางไปถึงลูกค้า i แต่เนื่องจากว่าเส้นทางของรถ k ยังไม่ทราบ จึงไม่รู้ว่าจะต้องเดินทางจากจุดใดก่อนมาถึงลูกค้า i ดังนั้นจะต้องทำการสมมติจุดขึ้นมา 1 จุดที่เรียกว่า "Seed" โดยจะใช้หลักในการเดินทางว่ารถ k จะต้องเดินทางจาก depot ไปที่ Seed แล้วจึงกลับมาที่ depot อีกครั้ง ฉะนั้นค่า b_{ik} จะเป็นค่าของระยะทางที่เพิ่มขึ้นจากการใส่ลูกค้า i ลงไปในเส้นทาง D → ik → D (D= depot, ik = seed ของรถ k) ดังเช่นในรูปที่ 3.16



รูปที่ 3.16 แสดงการเดินทางในการเพิ่มลูกค้า i ลงบนเส้นทางจาก D → ik → D

ค่า b_{ik} หาได้จากสมการที่ 3.7

$$b_{ik} = \text{Min } \{d_{D,i} + d_{i,ik} + d_{ik,D} \, , \, d_{D,ik} + d_{ik,i} + d_{i,D}\} - \{d_{D,ik} + d_{ik,D}\} \quad \text{.....(3.7)}$$

โดยที่ $d_{D,i}$ คือ ระยะเวลาทางจาก depot ไปที่ลูกค้า i

$d_{i,ik}$ คือ ระยะเวลาจากลูกค้า i ไปที่ Seed ของรถ k

$d_{ik,D}$ คือ ระยะเวลาจาก Seed ของรถ k ไปที่ depot

$d_{D,ik}$ คือ ระยะเวลาจาก depot ไปที่ Seed ของรถ k

$d_{ik,i}$ คือ ระยะทางจาก Seed ของรถ k ไปที่ลูกค้า i

$d_{i,D}$ คือ ระยะเวลาจากลูกค้า i ไปที่ลูกค้า depot

ง. <u>การหา Seed (ik)</u>

ขั้นตอนในการหา Seed นี้จะเป็นการกำหนดเบื้องต้นเกี่ยวกับบริเวณพื้นที่ที่รถจะต้องรับผิดชอบ รวมถึงเส้นทางและปริมาณความต้องการที่จะต้องให้บริการ จากนั้นจึงนำไปปรับปรุงให้เหมาะสมมากขึ้นด้วยแบบจำลองทางคณิตศาสตร์ การหา Seed จะต้องหา Seed ประจำสำหรับรถแต่ละคัน โดยขั้นตอนในการหา Seed เริ่มจาก

1. การพล็อตจุด depot ลงบนกระดาษ

2. การหาว่า depot จะไปถึงจุด i ใดโดยที่มีระยะทางที่สั้นที่สุด จากนั้นพล็อตจุด i ที่เลือกมาลงบนเส้นรัศมีที่พุ่งออกจาก depot โดยมีระยะห่างเท่ากับระยะทางจาก depot ไปถึงจุด i

3. การหาว่าจุด i ในข้อ 2 จะไปถึงจุด i อื่นใดโดยที่มีระยะทางที่สั้นที่สุด จากนั้น พล็อตจุด i ที่เลือกใหม่ต่อจากจุด i ในข้อ 2 โดยที่การพล็อตจะต้องพล็อตลงบนเส้นรัศมีใหม่ที่หมุนไปทางด้านขวา หรือซ้าย ของรัศมีเดิมซึ่งมุมที่จะหมุนไปจะมีค่าเท่ากับระยะทางระหว่างจุด i ในข้อ 2 กับจุด i ที่เลือกมาใหม่ (ในการหมุนของเส้นรัศมี จะต้องหมุนไปทางด้านเดียวกันตลอดทุก ๆ เส้นรัศมี) สำหรับระยะห่างของ depot กับจุด i ใหม่ จะมีค่าเท่ากับระยะทางจาก depot ถึงจุด i ใหม่

4. ทำเหมือนกับในข้อ 3 จนกระทั่งพล็อตจุด i ได้ครบ โดยจุด i ทั้งหมดจะอยู่รอบ ๆ depot และระยะห่างรวมระหว่างจุด i ทั้งหมดจะเท่ากับมุม 360 องศา

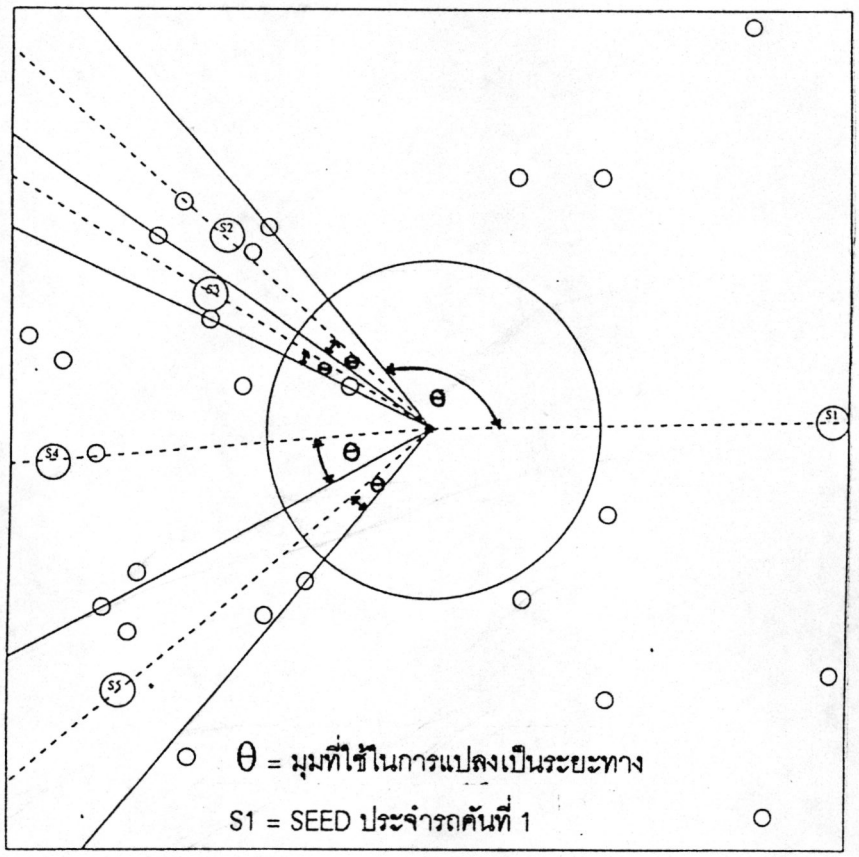
5. ทำการแบ่งพื้นที่ในรูปออกเป็นกรวยแต่ละกรวย โดยที่กรวยมีจำนวนเท่ากับรถและจุด i ที่อยู่ภายในกรวยเมื่อรวมปริมาณความต้องการของแต่ละจุดแล้วจะต้องไม่เกินความจุของรถที่รับผิดชอบที่ในกรวยนั้น

6. ทำการแบ่งครึ่งกรวยแต่ละอัน seed จะตั้งอยู่บนเส้นแบ่งครึ่งกรวยนั้น

7. ในการวางตำแหน่งของ seed ลงบนแผนที่ที่มีการระบุตำแหน่งของจุด i ไว้แล้วบนถนนต่าง ๆ นั้นจะเริ่มต้นจากการดูว่า seed มีตำแหน่ง อยู่ระหว่างจุด i ใด จากนั้นจึงนำ

มาพล็อตลงบนแผนที่ระหว่างจุด i ทั้ง 2 นั้น โดยระยะทางระหว่าง seed กับจุด i ที่เลือกมาด้านหนึ่งในแผนที่จะมีค่าเท่ากับระยะทางที่แปลงมาจากมุมระหว่างเส้นรัศมีของ seed กับจุด i ทางด้านนั้น ดังนั้นระยะทางระหว่าง seed กับจุด i ในด้านที่เหลือบนแผนที่เมื่อนำมาแปลงเป็นมุมก็จะได้เท่ากับมุมระหว่างเส้นรัศมีของ seed กับจุด i ในด้านที่เหลือ

ตัวอย่างการหา seed แสดงอยู่ในรูปที่ 3.17



รูปที่ 3.17 แสดงการหา seed

1.2 การหาเส้นทางโดยการใช้ Traveling Salesman Problem สำหรับรถแต่ละคัน

เมื่อทำการจัดสรรลูกค้า i ให้กับรถ k แต่ละคันแล้ว จากนั้นให้ทำการหาเส้นทางที่จะต้องเดินทางโดยการใช้หลักการและวิธีการของ TSP

2. Ball (1988) ได้ทำการดัดแปลงแบบจำลองทางคณิตศาสตร์ของ Christofides และ Beasley (1984) ซึ่งเป็นแบบจำลองที่ใช้หารูปแบบในการรับบริการของลูกค้า i แต่ละรายที่รถแต่ละคันรับผิดชอบอยู่ และเรียกวิธีนี้ว่า p - median approach ซึ่งเป็น multiroute แบบ single - depot

ในการที่จะหารูปแบบที่เหมาะสมให้กับลูกค้า i นั้น (สำหรับรถทีละ 1 คัน) ในขั้นแรกจะต้องทำการกำหนดข้อมูลต่าง ๆ ที่ใช้ในการสร้างรูปแบบ ดังในตัวอย่างรูปที่ 3.18 (ก) ซึ่งจะเริ่มจากการหาเซ็ตของรูปแบบเท่าที่จะเป็นไปได้ของวันที่จะได้รับบริการในช่วงแผนงาน (แผนงานในที่นี้หมายถึงระยะเวลาของวันที่ถูกกำหนดไว้สำหรับการทำงาน 1 รอบ เช่น ถ้าแผนงานมีระยะเวลาที่กำหนดไว้เท่ากับ 7 วัน รถก็จะทำงานตามแผนงานในวันที่ 1 จนกระทั่งถึงแผนงานในวันที่ 7 ซึ่งเป็นการครบรอบการทำงาน สำหรับในวันที่ 8 รถก็จะเริ่มรอบการทำงานใหม่โดยจะทำเหมือนแผนงานในวันที่ 1 กระทำเช่นนี้ไปเรื่อย ๆ จนครบรอบการทำงานอีกครั้ง) และในขั้นนี้จะทำการกำหนดอัตราส่วนความต้องการให้กับวันที่ได้รับบริการในแต่ละรูปแบบไปพร้อมกันด้วย ซึ่งเมื่อนำอัตราส่วนความต้องการภายในแต่ละรูปแบบนี้มารวมกันก็จะต้องมีค่าเท่ากับ 1 ในทุก ๆ รูปแบบ จากนั้นจึงหาปริมาณความต้องการทั้งหมดของลูกค้า i แต่ละราย เมื่อได้ข้อมูลจนครบทุกอย่างแล้วจึงทำการจัดเซ็ตของรูปแบบที่ลูกค้าต้องการและหาปริมาณความต้องการในวันที่รับบริการของรูปแบบที่เลือกไว้ ซึ่งอาจจะมีได้หลายรูปแบบดังเช่นตัวอย่างในรูปที่ 3.18 (ข) ดังนั้นจึงต้องทำการเลือกรูปแบบที่เหมาะสมกับลูกค้า i แต่ละรายโดยที่มีระยะทางในการเดินทางรวมน้อยที่สุด และรูปแบบนั้นก็เป็นที่พอใจของลูกค้า

PATTERNS

PAT #	M	T	W	TH	F	S
A	.4		.3		.3	
B		.4		.25		.35
C		.5			.5	
D	.5			.5		
E		1				
F			1			

DEMANDS

CUST #	WEEKLY DEMAND
1	100
2	120
3	200
4	150
5	40
6	30
7	80

(ก)

FEASIBLE DEMAND DISTRIBUTIONS

CUST #	PAT #	M	T	W	TH	F	S
1	A	40		30		30	
1	B		40		25		35
1	C		50			50	
2	A	48		36		36	
2	B		48		30		42
3	A	80		60		60	
3	B		80		50		70
4	A	60		45		45	
5	E		40				
5	F			40			
6	E		30				
6	F			30			
7	C		40			40	
7	D	40			40		

(ข)

รูปที่ 3.18 แสดงตัวอย่างของรูปแบบความต้องการ

(ก) การกำหนดข้อมูลต่าง ๆ เพื่อนำมาสร้างรูปแบบ

(ข) เซ็ตของรูปแบบที่ลูกค้าต้องการ

ขั้นตอนในการหา (สำหรับรถทีละ 1 คัน) จะแบ่งออกเป็น 2 ขั้นตอน คือ

2.1 <u>การเลือกรูปแบบให้กับลูกค้าแต่ละราย</u>

ก. <u>แบบจำลองทางคณิตศาสตร์</u> (สำหรับรถทีละ 1 คัน) ของการเลือกรูปแบบ คือ

LP Model :

Objective $\quad \text{Min} \sum_{i \in M} \sum_{h=1}^{n} b_{ih} Y_{ih}$ ..... (3.8)

Subject to $\quad \sum_{p \in P} Z_{ip} = 1 \quad$, ทุก $i \in M$ ..... (3.9)

$d_{ih} = \sum_{p \in P} (f_{ph} * d'_i) Z_{ip} \quad$, $h = 1,2,\ldots,n$ และ $i \in M$ ..... (3.10)

$y_{ih} = \sum_{p \in P} e_{ph} Z_{ip} \quad$, $h = 1,2,\ldots,n$ และ $i \in M$ ..... (3.11)

$\sum_{i \in M} d_{ih} \leq q \quad$, $h = 1,2,\ldots,n$ ..... (3.12)

$y_{ih}, Z_{ip} \in \{0,1\} \quad$, ทุก i ,h และ p ..... (3.13)

$d_{ih} \geq 0 \quad$, ทุก i และ h ..... (3.14)

โดยที่

b_{ih} = ระยะเวลาที่ใช้ในการเดินทางจาก Center ไปถึงลูกค้า i ในวัน h

Y_{ih} = $\begin{cases} 1 & \text{ถ้าลูกค้า i ถูกบริการในวัน h} \\ 0 & \text{อื่น ๆ} \end{cases}$

Z_{ip} = $\begin{cases} 1 & \text{ถ้ารูปแบบ p ถูกจัดให้กับลูกค้า i} \\ 0 & \text{อื่น ๆ} \end{cases}$

d_{ih} = ปริมาณความต้องการที่รถจะต้องส่งถึงลูกค้า i ในวัน h

f_{ph} = อัตราส่วนความต้องการในวัน h ของรูปแบบ p

d'_i = ปริมาณความต้องการทั้งหมดของลูกค้า i ตลอดทั้งแผนงาน

i_{ph} = $\begin{cases} 1 \text{ ถ้า } f_{ph} > 0 \\ 0 \text{ อื่น ๆ} \end{cases}$

q = ความจุของรถ

P = เซ็ตของรูปแบบ

M = เซ็ตของลูกค้า i ทั้งหมดที่รถรับผิดชอบอยู่

n = จำนวนวันทั้งหมดในแผนงาน

แบบจำลองนี้จัดเป็นแบบจำลอง Linear Programming Model (LP Model)

ข. <u>ความหมายของสมการและอสมการ</u>

สมการที่ 3.8 หมายถึง ผลรวมของระยะทางทั้งหมดตลอดแผนงานที่มีค่าน้อยที่สุดที่ใช้ในการเดินทางจาก Center ไปถึงสถานพยาบาลแต่ละแห่ง

สมการที่ 3.9 หมายถึง ลูกค้า i แต่ละรายจะมีรูปแบบได้เพียงรูปเดียวเท่านั้น

สมการที่ 3.10 หมายถึง ปริมาณความต้องการที่รถจะต้องส่งถึงลูกค้า i ในวัน h จะมาจากรูปแบบที่ถูกเลือก

สมการที่ 3.11 หมายถึง รถจะต้องไปที่ลูกค้า i ในวัน h ถ้าลูกค้า i ถูกจัดให้รับบริการในวัน h

อสมการที่ 3.12 หมายถึง ปริมาณความต้องการที่รถจะต้องส่งถึงลูกค้า i ที่รับบริการในวัน h รวมกันทั้งหมดจะต้องมีปริมาณไม่เกินความจุของรถ

สมการที่ 3.13 หมายถึง ตัวแปร Y_{ih} และ Z_{ip} จะต้องมีค่าเท่ากับ 0 หรือ 1 เท่านั้น

อสมการที่ 3.14 หมายถึง ปริมาณความต้องการที่รถจะต้องส่งถึงลูกค้า i ในวัน h จะต้องมีค่ามากกว่าหรือเท่ากับ 0

ค. <u>การหาค่า b_{ih}</u>

Christofides และ Eilon (1969) คาดว่าค่าใช้จ่ายรวมในการเดินทางจะมีค่าเพิ่มขึ้นในขณะที่ระยะทางจาก Center มีค่าเพิ่มขึ้น ดังนั้น Christofides และ Beasley จึงกำหนดให้มี Center ในแต่ละวัน จากนั้นจึงใช้ระยะทางจาก Center ถึงลูกค้า i เป็นตัวแทนสำหรับค่าใช้จ่ายในการจัดลูกค้า

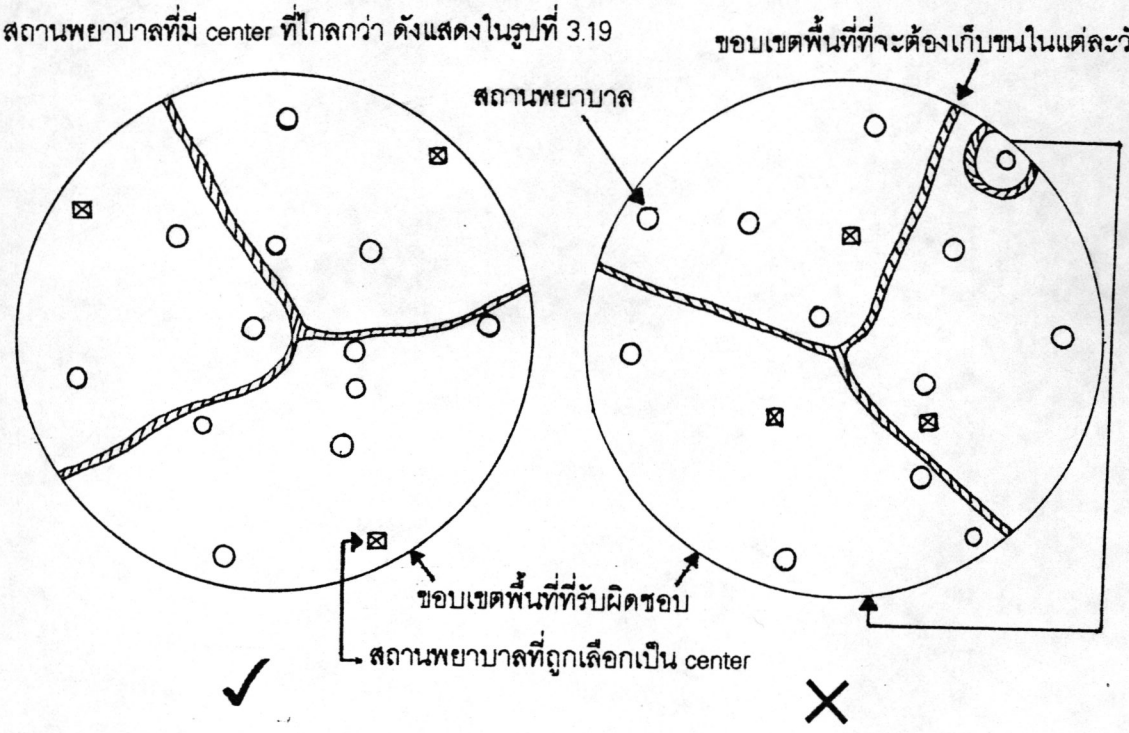
i ให้รับบริการในวัน h ซึ่งนั่นก็คือค่า b_{ih} โดยที่ในวัน h ลูกค้า i ที่รับบริการในวันนี้ก็จะมี Center ร่วมกัน ดังนั้นถ้าแผนงานมี h วันก็จะต้องมี h Center (ในขั้นตอนการหาค่า b_{ih} นี้จะกำหนดให้ระยะทางจาก Center ถึงลูกค้า i มีค่าเท่ากับระยะทางจากลูกค้า i ถึง Center)

ง. <u>การหา Centers</u>

ในการที่หา Centers นี้ขึ้นมาก็เพื่อที่จะใช้เป็นจุดกำหนดสำหรับการหาสถานพยาบาลที่จะต้องรับผิดชอบในแต่ละวัน โดยมีขั้นตอนในการหาดังนี้

1. การหาเซ็ตของ J (J = เซ็ตของ Center ทั้งหมด) โดยการเลือก i ที่ไกลจาก depot มากที่สุดเป็น center แรก จากนั้นเลือก i ที่ไกลจาก depot และ center แรกเป็น center จุดที่ 2 จนกระทั่งได้ center ครบ h centers

สำหรับเหตุผลในการเลือก center ให้อยู่ไกลจาก depot หรืออยู่ในบริเวณรอบนอกของพื้นที่ที่รับผิดชอบนั้น ก็เนื่องจากว่าต้องการให้การเลือกสถานพยาบาลที่จะต้องเก็บขนในแต่ละวันนั้นเริ่มเลือกมาจาก center เข้าสู่ศูนย์กลางของพื้นที่ เพราะถ้ากำหนด center เอาไว้ภายในของพื้นที่จะทำให้การเลือกสถานพยาบาลเริ่มเลือกจาก center ออกไปรอบนอกจนถึงขอบของพื้นที่ซึ่งอาจจะทำให้มีบางสถานพยาบาลที่ไม่ได้อยู่กับ center ที่อยู่ใกล้กับสถานพยาบาลนั้นเนื่องจากว่าจะทำให้ปริมาณเกินความจุของรถ ดังนั้นจึงต้องจัดสรรให้สถานพยาบาลที่เหลือนั้นไปอยู่กับกลุ่มของสถานพยาบาลที่มี center ที่ไกลกว่า ดังแสดงในรูปที่ 3.19



รูปที่ 3.19 แสดงวิธีการเลือก centers

2. สำหรับแต่ละลูกค้า i ให้หาเซ็ต J_i (J_i = เซ็ตของ centers ที่อยู่ใกล้กับจุด i ที่สุด ซึ่งอาจจะมีได้หลาย center)

3. การหาเมตริกซ์ (h x h) จากค่า m_{jh}

$$m_{jh} = \sum_{j \in J_i} [(\sum_{p \in P} a_{ph}) / | p_i |] \qquad \text{.....(3.15)}$$

โดยที่ $a_{ph} = \begin{cases} 1 & \text{ถ้าวัน h มีอยู่ในรูปแบบ p} \\ 0 & \text{อื่น ๆ} \end{cases}$

$| p_i |$ = จำนวนรูปแบบทั้งหมดของลูกค้า i

4. การแก้ปัญหา assignment problem แบบ Maximize กับเมตริกซ์ (h x h) เพื่อที่จะหา center j ใดควรจะอยู่กับวัน h ใด

2.2 <u>การหาเส้นทางโดยการใช้ Traveling Salesman Problem สำหรับแต่ละวันในแผนงาน</u>

เมื่อทำการเลือกรูปแบบให้กับลูกค้า i แล้ว ก็สามารถที่จะทราบว่าในแต่ละวันในแผนงานรถจะต้องมีลูกค้าที่จะต้องรับผิดชอบรายใดบ้าง จากนั้นให้ทำการหาเส้นทางที่จะต้องเดินทางโดยการใช้หลักการและวิธีการของ TSP