บรรณานุกรม

ภาษาไทย

วันชัย ริจิรวนิช. การวิเคราะห์โครงข่าย. พิมพ์ครั้งที่ 1. กรุงเทพมหานคร : สำนักพิมพ์ซีเอ็ดยูเคชั่น, 2528.

ศิริจันทร์ ทองประเสริฐ. ระบบพัสดุคงคลัง. พิมพ์ครั้งที่ 2. กรุงเทพมหานคร : โรงพิมพ์จุฬาลงกรณ์มหาวิทยาลัย, 2530.

อรวรรณ ตันศิริเจริญกุล. การใช้วิธีฮิวริสติกแก้ปัญหาเส้นทางเดินรถในการเก็บขนขยะมูลฝอยในพื้นที่บางเขน. วิทยานิพนธ์ปริญญามหาบัณฑิต จุฬาลงกรณ์มหาวิทยาลัย, 2534.

ภาษาอังกฤษ

Bodin, D.L., et al. Routing and scheduling of vehicles and crews. Comput & Ops Res 10 (1983) No.2: 63-211.

Clark, G., and Wright, W.J. Scheduling of vehicles from a central depot to a number of delivery point. Operation Research 12 (1964) No.3: 568-581.

ภาคผนวก ก

การหาเส้นทางเป็นเลิศของทุกๆคู่ของจุดเชื่อมโดยวิธี RCM

การหาเส้นทางเป็นเลิศของทุกๆคู่ของจุดเชื่อม วิธี Revision Cascade Method (RCM) (วันชัย ริจิรวนิช,2528 [1]) มีวิธีการดังนี้

เราจะต้องกำหนดหมายเลขให้จุดเชื่อม 1 ถึง n และให้ d^*_{ik} คือระยะทางของเส้นทางที่เป็นเลิศจาก i ถึง k ผ่านจุดเชื่อม j นั่นคือ

$$d^*_{ik} <= d_{ij}+d_{jk} \quad i = j = k$$

ถ้าพิจารณาแล้วปรากฏว่าเกิดกรณี $d^*_{ik} > d_{ij}+d_{jk}$ ก็แสดงว่าเราต้องเปลี่ยน d^*_{ik} ให้เป็น $d_{ij}+d_{jk}$ แทน ดังนั้นถ้าเรากำหนด d^*_{ik}ไว้เป็นค่าใดๆ ที่มีค่าสูงไว้ก่อนแล้ว พิจารณาให้ d^*_{ik} ผ่านจุดเชื่อม j = 1,2,.....,n จุดใดที่ $d^*_{ik} > d_{ij}+d_{jk}$ ก็เปลี่ยน d^*_{ik} เป็นการปรับปรุงให้มีเส้นทางที่ดีขึ้น กระบวนการเช่นนี้เราอาจจะเขียนได้เป็น

$$d^m_{ik} = \min \{ d^{m-1}_{ik} ; d^{m-1}_{ij} + d^{m-1}_{jk} \} \quad i = j = k \qquad (1)$$

โดยมี d^{m-1}_{ik} คือ d^*_{ik} เมื่ออยู่ในขั้นตอนที่ m-1 และมีค่า d^*_{ik} ใหม่ คือ d^m_{ik} สำหรับขั้นตอนที่ m กระบวนการนี้เราเรียกว่า triple operation เพราะมีการกำหนด d^*_{ik} หาผลลัพธ์ของ $d_{ij}+d_{jk}$ แล้วเปรียบเทียบกันเป็นการดำเนินงาน 3 ประการ

ให้ $D^0 = [d^0_{ik}]$ เป็นเมตริกซ์ของระยะทางสำหรับโครงข่าย โครงข่ายนี้อาจจะมี d_{ik} เป็นลบก็ได้ แต่ระยะทางของวงจรต่างๆ ต้องมากกว่า 0 สำหรับ d_{ik} ที่ไม่มีเส้นเชื่อม (i,k) จะมีค่าเป็น α ใน D^0 แล้วมี $d_{ii} = 0$ เมื่อได้ D^0 แล้วต่อไปก็จะต้องหา D^m สำหรับ m = 1,2...,n โดยอาศัยสมการ (1) และได้ D^m เป็นผลลัพธ์ที่ต้องการ

นอกจากการหาเมตริกซ์ของเส้นทางที่เป็นเลิศของทุก ๆ คู่ของจุดเชื่อมได้เป็น D^m แล้วเรายังต้องหาเมตริกซ์ที่แสดงเส้นทางด้วย ซึ่งเรียกว่า Route Matrix

ให้ $R^m = [r^m_{ik}]$ เป็นเมตริกซ์แสดงเส้นทาง r^m_{ik} จะเป็นจุดเชื่อมทางผ่านของระยะทางเป็นเลิศจาก i ไป k สำหรับ m = 1,2...,n ดังนั้น $R^0 = [r^0_{ik}] = k$ และ r^m_{ik} จะหาได้โดย

$$r^m_{ik} = \begin{cases} j & \text{ถ้า } \min\{d^{m-1}_{ik}, d^{m-1}_{ij} + d^{m-1}_{ik}\} = d^{m-1}_{ij} + d^{m-1}_{ik} \\ r^{m-1}_{ik} & \text{ถ้า } \min\{d^{m-1}_{ik}, d^{m-1}_{ij} + d^{m-1}_{ik}\} = d^{m-1}_{ik} \end{cases} \quad (2)$$

สมการ (2.2) มีความหมายว่า ถ้า $\min.\{d^{m-1}_{ik}, d^{m-1}_{ij} + d^{m-1}_{ik}\} = d^{m-1}_{ik}$ คือเส้นทางเดิมจาก i ไป k โดยไม่ผ่าน j ดีอยู่แล้วจึงให้ $r^m_{ik} = r^{m-1}_{ik}$ คือจุดเชื่อมทางผ่านไม่เปลี่ยนแปลง แต่ถ้า $\min\{d^{m-1}_{ik}, d^{m-1}_{ij} + d^{m-1}_{jk}\} = d^{m-1}_{ij}$ แสดงว่าการผ่านจุดเชื่อมๆ เป็นเส้นทางที่ดีกว่า เราจึงต้องเปลี่ยน r^m_{ik} ให้เป็น j

ขั้นตอนการหาเส้นทางเป็นเลิศของทุกๆ คู่ของจุดเชื่อมโดยวิธี RCM พอสรุปได้ดังนี้

ขั้นตอนที่ 1 : กำหนดหมายเลขประจำจุดเชื่อมจาก 1 ถึง n

ขั้นตอนที่ 2 : ให้ $D^0 = [d^0_{ik}] = [d_{ik}]$, $d_{ii} = 0$, $d = \alpha$ ถ้าไม่มีเส้นเชื่อม (i,k)
ให้ $R^0 = [r_{ik}] = k$ และ m = 1

ขั้นตอนที่ 3 : จาก D^{m-1} ขีดคร่อมเส้นแนวนอนและแนวยืนผ่านจุดเชื่อม m ซึ่งเรียกว่าจุดหมุน (pivot point) บนเส้นแนวนอนถ้ามีค่า $d_{ik} = \alpha$ ให้ขีดเส้นแนวยืนผ่าน บนเส้นแนวยืนถ้ามีค่า $d_{ik} = \alpha$ ให้ขีดเส้นแนวนอนผ่าน สำหรับ $d^{m-1}_{ik}, i = k$ ที่เหลืออยู่ให้ใช้วิธี Triple Operations

หา $d^m_{ik} = \min\{d^{m-1}_{ik}, d^{m-1}_{ij} + d^{m-1}_{jk}\}$

(1) ถ้า $d^{m}_{ik} = d^{m-1}_{ij} + d^{m-1}_{jk}$
ให้เปลี่ยน d^{m-1}_{ik} เป็น d^{m}_{ik} และ $r^{m}_{ik} = j$

(2) ถ้า $d^{m}_{ik} = d^{m-1}_{ik} + d^{m}_{ik}$ และ r^{m}_{ik} จะเหมือน d^{m-1}_{ik} และ r^{m-1}_{ik} จะได้ D^{m} และ R^{m} ตามต้องการ

ขั้นตอนที่ 4 : ให้ m = m+1 ถ้า m<n ให้กลับไปขั้นตอนที่ 3 จนกว่าจะได้ D^{n} และ R^{n} เป็นผลลัพธ์ที่ต้องการ

<u>ตัวอย่าง</u> ให้หาเส้นทางเป็นเลิศของทุก ๆ คู่ของจุดเชื่อมในโครงข่าย

$$D^{0} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 3 & \alpha & \alpha \\ 2 & 0 & 1 & \alpha & \alpha \\ \alpha & \alpha & 0 & \alpha & 2 \\ \alpha & 4 & \alpha & 0 & \alpha \\ \alpha & \alpha & \alpha & 1 & 0 \end{bmatrix} \end{array}$$

$$R^{0} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \end{array}$$

m = 1

$$D^{0} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 3 & \alpha & \alpha \\ 2 & 0 & 1 & \alpha & \alpha \\ \alpha & \alpha & 0 & \alpha & 2 \\ \alpha & 4 & \alpha & 0 & \alpha \\ \alpha & \alpha & \alpha & 1 & 0 \end{bmatrix} \end{array}$$

$d^{1}_{23} = \min \{d^{0}_{23}, d^{0}_{21} + d^{0}_{13}\}$
$= \min \{1, 2+3\} = 1 = d^{0}_{23}$

$r^{1}_{23} = r^{0}_{23} = 3$

$D^{1} = D^{0}$, $R^{1} = R^{0}$

m = 2

$$D^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 3 & \alpha & \alpha \\ 2 & 0 & 1 & \alpha & \alpha \\ \alpha & \alpha & 0 & \alpha & 2 \\ \alpha & 4 & \alpha & 0 & \alpha \\ \alpha & \alpha & \alpha & 1 & 0 \end{bmatrix} \end{array}$$

$d^1_{13} = \min \{d^0_{13}, d^0_{12} + d^0_{23}\}$

$= \min \{3, 1+1\} = 2$

$d^2_{13} = d^1_{12} + d^1_{23} = 2$

$r^2_{13} = 2$

$d^2_{41} = \min \{d^1_{41}, d^1_{42} + d^1_{21}\}$

$= \min \{\alpha, 4+2\} = 6$

$d^2_{41} = 6, \; r^2_{41} = 2$

$d^2_{43} = \min \{d^1_{43}, d^1_{42} + d^0_{23}\}$

$= \min \{\alpha, 4+1\} = 5$

$d^2_{43} = 5, \; r^2_{43} = 2$

$$D^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & \alpha & \alpha \\ 2 & 0 & 1 & \alpha & \alpha \\ \alpha & \alpha & 0 & \alpha & 2 \\ 6 & 4 & 5 & 0 & \alpha \\ \alpha & \alpha & \alpha & 1 & 0 \end{bmatrix} \end{array}$$

$$R^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & 2 & 2 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 2 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \end{array}$$

m = 3

$$D^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & \alpha & \alpha \\ 2 & 0 & 1 & \alpha & \alpha \\ \alpha & \alpha & 0 & \alpha & 2 \\ 6 & 4 & 5 & 0 & \alpha \\ \alpha & \alpha & \alpha & 1 & 0 \end{bmatrix} \end{array}$$

$d^3_{15} = \min \{d^2_{15}, d^2_{13} + d^2_{35}\}$

$= \min \{\alpha, 2+2\} = 4$

$d^3_{15} = 4, \; r^3_{15} = 3$

$d^3_{25} = \min \{d^2_{25}, d^2_{23} + d^2_{35}\}$

$= \min \{\alpha, 1+2\} = 3$

$d^3_{25} = 3, \; r^3_{25} = 3$

$$d^3_{45} = \min \{d^2_{45}, d^2_{43} + d^2_{35}\}$$

$$= \min \{\alpha, 5+2\} = 7$$

$$d^3_{45} = 7, \; r^3_{45} = 3$$

$$D^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & \alpha & 4 \\ 2 & 0 & \alpha & \alpha & 3 \\ \alpha & \alpha & 0 & \alpha & \alpha \\ 6 & 4 & 5 & 0 & 7 \\ \alpha & \alpha & \alpha & 1 & 0 \end{bmatrix} \end{array}$$

$$R^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & 2 & 2 & 4 & 3 \\ 1 & 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 & 5 \\ 2 & 2 & 2 & 4 & 3 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \end{array}$$

m = 4

$$D^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & \alpha & 4 \\ 2 & 0 & 1 & \alpha & 3 \\ \alpha & \alpha & 0 & \alpha & 2 \\ 6 & 4 & 5 & 0 & 7 \\ \alpha & \alpha & \alpha & 1 & 0 \end{bmatrix} \end{array}$$

$$d^4_{51} = \min \{d^3_{51}, d^3_{54} + d^3_{41}\}$$

$$= \min \{\alpha, 1+6\} = 7$$

$$d^4_{51} = 7, \; r^4_{51} = 4$$

$$d^4_{52} = \min \{d^3_{52}, d^3_{54} + d^3_{42}\}$$

$$= \min \{\alpha, 1+4\} = 5$$

$$d^4_{52} = 5, \; r^4_{52} = 4$$

$$d^4_{53} = \min \{d^3_{53}, d^3_{54} + d^3_{43}\}$$

$$= \min \{\alpha, 1+5\} = 6$$

$$d^4_{53} = 6, \; r^4_{53} = 4$$

$$D^4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & \alpha & 4 \\ 2 & 0 & 1 & \alpha & 3 \\ \alpha & \alpha & 0 & \alpha & 2 \\ 6 & 4 & 5 & 0 & 7 \\ 7 & 5 & 6 & 1 & 0 \end{bmatrix} \end{array}$$

$$R^4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & 2 & 2 & 4 & 3 \\ 1 & 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 & 5 \\ 2 & 2 & 2 & 4 & 3 \\ 4 & 4 & 3 & 4 & 5 \end{bmatrix} \end{array}$$

m = 5

$$D^4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & \alpha & 4 \\ 2 & 0 & 1 & \alpha & 3 \\ \alpha & \alpha & 0 & \alpha & 2 \\ 6 & 4 & 5 & 0 & 7 \\ 7 & 5 & 6 & 1 & 0 \end{bmatrix} \end{array}$$

$d^5_{12} = \min \{d^4_{12}, d^4_{15} + d^4_{52}\}$

$= \min \{1, 4+5\} = d^4_{12}$

$d^5_{13} = d^4_{13}$

$d^5_{14} = \min \{d^4_{14}, d^4_{15} + d^4_{54}\}$

$= \min \{\alpha, 4+1\} = 5$

$d^5_{14} = 5, \; r^5_{14} = 5$

$d^5_{21} = d^4_{21} ; d^5_{23} + d^4_{54}\}$

$d^5_{24} = \min \{d^4_{24}, d^4_{25} + d^4_{54}\}$

$d^5_{24} = 4, \; r^5_{24} = 5$

$d^5_{31} = \min \{d^4_{31}, d^4_{35} + d^4_{51}\}$

$= \min \{\alpha, 2+7\} = 9$

$d^5_{31} = 9, \; r^5_{31} = 5$

$d^5_{32} = \min \{d^4_{32}. \; d^4_{35} + d^4_{52}\}$

$= \min \{\alpha, 2+5\} = 7$

$d^5_{32} = 7, \; r^5_{32} = 5$

$d^5_{34} = \min \{d^4_{34}. \; d^4_{35} + d^4_{54}\}$

$= \min \{\alpha, 2+1\} = 3$

$d^5_{34} = 3, \; r^5_{34} = 5$

$d^5_{41} = d^4_{41} ; \; d^5_{42} = d^4_{42}$

$$d^5_{43} = d^4_{43}$$

$$D^5 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 5 & 4 \\ 2 & 0 & 1 & 4 & 3 \\ 9 & 7 & 0 & 3 & 2 \\ 6 & 4 & 5 & 0 & 7 \\ 7 & 5 & 6 & 1 & 0 \end{bmatrix} \end{array}$$

$$R^5 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & 2 & 2 & 5 & 3 \\ 1 & 2 & 3 & 5 & 3 \\ 5 & 5 & 3 & 5 & 5 \\ 2 & 2 & 2 & 4 & 3 \\ 4 & 4 & 4 & 4 & 5 \end{bmatrix} \end{array}$$

จากผลลัพธ์ของ D^5 เราสามารถกำหนดระยะทางเป็นเลิศจากจุดหนึ่งไปอีกจุดหนึ่งได้ ส่วน R^5 จะบอกเส้นทางหรือจุดเชื่อมต่างๆ ของเส้นทางที่เป็นเลิศนั้น ตัวอย่างเช่น

$i = 3$, $k = 1$ ระยะทางที่เป็นเลิศคือ 9

เส้นทางเดินเริ่มจาก $r_{31} = 5$, $r_{51} = 4$, $r_{41} = 2$, $r_{21} = 1$

เส้นทางคือ 3 - 5 - 4 - 2 - 1

ถ้า $i = 5$, $k = 3$ ระยะทางที่เป็นเลิศคือ 6

เส้นทางเดินเริ่มจาก $r_{53} = 4$, $r_{43} = 2$, $r_{23} = 3$

เส้นทางคือ 5 - 4 -2 - 3

ภาคผนวก ข

ปัญหาเส้นทางเดินรถโดยวิธี The Saving Algorithm

<u>ปัญหาเส้นทางเดินรถ โดยวิธี The Saving Algorithm (G.Clarke and I.W.Wright, 1964)</u>

เป็นวิธีการจัดตารางเดินรถจากคลังสินค้าไปยังจุดขนถ่ายต่างๆ ที่จะหาเส้นทางที่ใกล้เคียงที่เหมาะสมวิธีหนึ่งจะใช้พิจารณาสำหรับกรณีที่มีจุดขนถ่ายจำนวนมาก ซึ่งถ้าพิจารณาโดยใช้ปัญหาเส้นทางเดินรถที่ให้คำตอบที่ดีที่สุด เช่น วิธี Branch and Bound Techniques เวลาที่ใช้ในการคำนวณอาจจะนานและยุ่งยาก วิธี The Saving Algorithm สามารถคำนวณได้โดยใช้โปรแกรมคอมพิวเตอร์ วิธีดังกล่าวเป็นวิธีการคำนวณที่ไม่ซับซ้อนและไม่มีข้อจำกัดมาก มีวิธีการดังนี้

ให้จำนวนรถบรรทุก X_i มีความจุในการบรรทุก C_i (i = 1, ...n) และมีความต้องการของจุดขนถ่าย q_i จุดขนถ่ายต่าง P_i และคลังสินค้า P_o ระยะทางที่สั้นที่สุดระหว่างจุดเชื่อม $d_{y,z}$ หาได้จากวิธี RCM นั่นคือความจุในการบรรทุกของรถแต่ละคันจะมากกว่าความต้องการของจุดขนถ่ายต่างๆ ดังสมการ

$$C_n \ll \Sigma^n_{j=1} q_j$$

และถ้า $C_n \geq \Sigma^n_{j=1} q_j$ รูปแบบปัญหาจะเป็นรูปแบบปัญหา Traveling Salesman Problem

จากเส้นทางการเดินรถดังรูป ข.1 คลังสินค้า P_o และพิจารณาจุดขนถ่าย P_y,P_z สมมุติ P_{y-1}, P_{y+1}, P_{z-1}, P_{z+1} เป็นจุดเชื่อมของ P_y และ P_z จากรูป a เป็นรูปแบบการขนส่งแบบปกติ รูป b, c, d, c เป็นการพิจารณาจัดเดินทางการขนส่งรูปแบบต่างๆ ซึ่งระยะทางที่สามารถประหยัดได้ของแต่ละรูปเมื่อเปรียบเทียบกับรูป 3 เป็นดังนี้

(b) $d_{y,y+1} - d_{o,y+1} + d_{z,z+1} - d_{o,z+1} - d_{y,z}$

(c) $d_{y-1,y} - d_{o,y-1} + d_{z,z+1} - d_{o,z+1} - d_{y,z}$

(d) $d_{y,y+1} - d_{o,y+1} + d_{z,z-1} - d_{o,z-1} - d_{y,z}$

(e) $d_{y-1,y} - d_{o,y-1} + d_{z,z-1} - d_{o,z-1} - d_{y,z}$

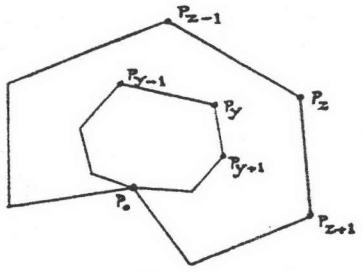

Figure a

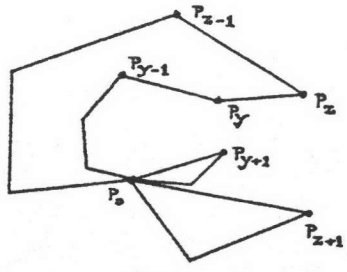

Figure b

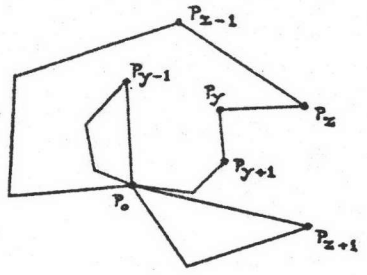

Figure c

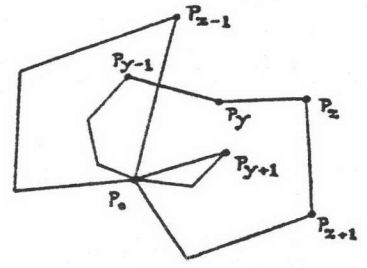

Figure d

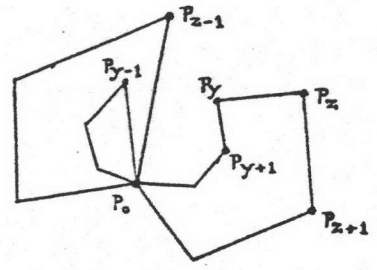

Figure e

รูปที่ ข.1 แสดงวิธีการเดินทางแบบต่างๆของรถขนส่ง

ตัวอย่างการคำนวณ

Q	P_0											
1200	9	P_1										
1700	14	18 5	P_2									
1500	21	18 12	28 7	P_3								
1400	23	10 21	20 17	34 10	P_4							
1700	22	10 21	20 16	22 21	26 19	P_5						
1400	25	10 24	16 23	18 30	20 28	38 9	P_6					
1200	32	10 31	20 26	26 27	30 25	44 10	50 7	P_7				
1900	36	10 35	20 30	20 37	24 35	42 16	50 11	58 10	P_8			
1800	38	10 37	16 36	16 43	20 41	38 22	50 13	54 16	68 6	P_9		
1600	42	10 41	20 36	32 31	36 29	44 20	50 17	64 10	72 6	68 12	P_{10}	
1700	50	10 49	20 44	34 37	42 31	44 28	50 25	64 18	72 14	76 12	84 8	P_{11}
1100	52	10 51	20 46	34 39	46 29	44 30	50 27	64 20	72 16	70 20	84 10	92 10	P_{12}

กำหนดให้

1. Q เป็นความต้องการของจุดขนถ่าย P ใดๆ Q_3 = 1,500 คือ ความต้องการของจุดขนถ่าย P_3 = 1,500 หน่วย

2. ตัวเลขมุมล่างขวามือของตารางแต่ละช่องเป็นระยะทางที่สั้นที่สุดของจุดต่างๆ 2 จุด เช่น 7 เป็นระยะทางที่สั้นที่สุดของ P_2 P_3 (จาก RCM Method)

3. ตัวเลขมุมล่างซ้ายมือเป็นระยะทางที่ประหยัดได้ (Saving)โดยสมมติให้จุด P_0 เป็นจุดเริ่มต้น และ P_y, P_z เป็นจุดขนถ่าย

ระยะที่ประหยัดได้จาก $P_yP_z = 2d_{0,y} + 2d_{0,z} - (d_{0,y} + d_{y,z} + 2d_{0,z})$

$= d_{0,y} + d_{0,z} - d_{y,z}$

จากตัวอย่าง Saving P_2P_3 = 14+21-7=28

4. ความจุของรถบรรทุก สมมติให้แต่ละคันมีความจุ 6,000 หน่วยจำนวนรถบรรทุกที่ต้องการอย่างน้อยที่สุด 4 คัน (ความต้องการทั้งหมดของแต่ละจุด หารด้วยความจุของรถ)

การคำนวณ

1. เลือกจุดขนถ่ายที่ P_{12} เปิดจุดขนถ่ายแรก เนื่องจากมีระยะทางจาก P_0 มากกว่าจุดอื่น ๆ มีความต้องการ 1,100 หน่วย เหลือความจุของรถบรรทุก = 6,000-1,100 = 4,900 หน่วย

2. พิจารณาหาระยะทางที่ประหยัดที่สุด (Saving)จุดขนถ่ายที่ 12 ไปยังจุดขนถ่ายที่ 11 จะประหยัดระยะทางมากที่สุด ฉะนั้นจุดขนถ่ายจุดที่สองจะเป็นจุด P_{11} มีความต้องการ 1,700 หน่วย เหลือความจุของรถบรรทุก = 4,900 - 1,700 = 3,200 หน่วย

3. พิจารณาจุดถัดไป ตามข้อ 2 จนกระทั่งความจุของรถไม่สามารถจะขนส่งไปอีกแล้ว

4. ถ้ายังมีความต้องการของแต่ละจุดเหลืออยู่ ให้นำรถบรรทุกคันต่อไปมาใช้ขนส่งและดำเนินตามข้อ 1 จนทุกจุดไม่มีความต้องการเหลืออีก

5. ตามตัวอย่างข้างต้น จะมีเส้นทางการขนส่งของรถแต่ละคันดังนี้

คันที่ 1	เส้นทาง	$P_0\ P_1\ P_2\ P_3\ P_4\ P_0$	บรรทุก	5,800 หน่วย
คันที่ 2	เส้นทาง	$P_0\ P_5\ P_0$	บรรทุก	1,700 หน่วย
คันที่ 3	เส้นทาง	$P_0\ P_6\ P_8\ P_9\ P_0$	บรรทุก	5,100 หน่วย
คันที่ 4	เส้นทาง	$P_0\ P_{10}\ P_{12}\ P_{11}\ P_7\ P_0$	บรรทุก	5,600 หน่วย

ภาคผนวก ค

โปรแกรมคอมพิวเตอร์

```
PROGRAM DataOfTruck;
USES Crt,Printer;
VAR
   Filvar:Text;
   Capacity:ARRAY[1..100,1..10]OF Integer;
   NoOfTruck,i,j:Integer;
   Answer:Char;
   FileName:STRING[14];
BEGIN
     Clrscr;
     WriteLn('===ENTER NAME OF FILE TO WRITE:===');
     ReadLn(FileName);
     Assign(FilVar,FileName);
     ReWrite(FilVar);
     WriteLn('1.ENTER NUMBER OF TRUCKS');
          ReadIn(NoOfTruck);
          WriteLn(FilVar,NoOfTruck);
          WriteLn('2.ENTER CAPACITY FOR EACH TRUCK (kl/each)');
          WriteLn('  ONE TRUCK HAS TEN CHANNELS');
          i:=1;
          REPEAT
               WriteLn('NO OF TRUCK  ',i);
               FOR j:=1 TO 10 DO{ one truck has ten chennels}
               BEGIN
               WriteLn('CAPACITY NO  '.j,'  ?  ');
               ReadLn(Capacity[i,j]):
               WriteLn(FilVar,Capacity[i,j]);
               END;
               i:=i+1;
          UNTIL i>NoOfTruck;
          Close(FilVar);
          WriteLn('===PRINT DATA OF TRUCKS===  Y/N ');
          ReadLn(Answer);
          IF (Answer='y')or(Answer='Y') THEN
          BEGIN
          WriteLn(Lst,'NO OF TRUCK  ',NoOfTruck);
          FOR i:=1 TO NoOfTruck DO
          FOR j:=1 TO 10 DO
          WriteLn(Lst,'TRUCK NO  ',i,'  CAPACITY NO ',j,'  '.Capacity[i,j]);
          END;
          WriteLn('END OF FILE.');
END.
```

```
PROGRAM DataOfDistance;
USES Crt,Printer;
VAR
   Filvar:Text;
   Distance:ARRAY[1..100,1..100]OF Integer;
   Nodes,i,j :Integer;
   Answer:Char;
   FileName:STRING[14];

BEGIN
     Clrscr;
     WriteLn('===ENTER NAME OF FILE TO WRITE===');
     ReadLn(FileName);
     Assign(FilVar,FileName);
     ReWrite(FilVar);
     WriteLn('1.ENTER NUMBER OF NODES');
     ReadLn(Nodes);
     WriteLn(FilVar,Nodes);
     WriteLn('2.ENTER DISTANCE BETWEEN NODES');
     i:=1;
     REPEAT
           J:=1;
           REPEAT
                 WriteLn('FROM NODE NO  ',i,'  TO NODE NO  ',j);
                 ReadLn(Distance[i,j]);
                 WriteLn(FilVar,Distance[i,j]);
           j:=j+1;
           UNTIL j>Nodes;
     i:=i+1;
     UNTIL i>Nodes;
     Close(FilVar);
     WriteLn('PRINT DATA DISTANCE   Y/N ?');
     ReadLn(Answer);
     IF (Answer='Y')or(Answer='y') THEN
     BEGIN
     WriteLn(Lst,'NUMBER OF NODES  ',Nodes);
     FOR i:=1 TO Nodes DO
     FOR j:=1 TO Nodes DO
     WriteLn(Lst,'FROM NODE NO  ',i,'  TO NODE NO  ',J,' DISTANCE  ',Distance[i,j]);
     END;
     WriteLn('END OF FILE.');
END.
```

```
PROGRAM DataOfDemand;
USES Crt,Printer;
VAR
   Filvar:Text;
   Demand,NoNodes:ARRAY[1..100]OF Integer;
   Nodes,i,j :Integer;
   Answer:Char;
   FileName:STRING[14];
BEGIN
    Clrscr;
    WriteLn('===ENTER NAME OF FILE TO WRITE===');
    ReadLn(FileName);
    Assign(FilVar,FileName);
    ReWrite(FilVar);
    WriteLn('1.ENTER NUMBER OF NODES');
    ReadLn(Nodes);{nodes of demand must be equal nodes of distance}
    {WriteLn(FilVar,Nodes);}
    WriteLn('2.ENTER NODES DEMAND');
    i:=1;
    REPEAT
          WriteLn('FROM NODE NO  ',i); {i=1 : deposit}
          ReadLn(Demand[i]);
          WriteLn(FilVar,Demand[i]);
          i:=i+1;
    UNTIL i>Nodes;
    Close(FilVar);
    WriteLn('PRINT DATA OF DEMAND   Y/N ?');
    ReadLn(Answer);
    IF (Answer='y')or(Answer='Y') THEN
    BEGIN
         WriteLn(Lst,'NUMBER OF NODES  ',Nodes);
         FOR i:=1 TO Nodes DO
         WriteLn(Lst,'NODE NO  ',i,'  DEMAND  ',Demand[i]);
    END;
    WriteLn('END OF FILE');
END.
```

```
PROGRAM RCM;
USES Crt,Printer;
CONST Infinity=999;
VAR Filvar:Text;
    Distance: ARRAY[1..100,1..100] OF Integer;
    L,N:array[1..100] of integer;
    Nodes,i,j,k,p,r,s,m:Integer;
    z:Boolean;
    Answer:Char;
    FileName:STRING[14];
BEGIN
     Clrscr;
     WriteLn('===START RCM METHOD===');
     WriteLn('ENTER NAME OF FILE TO READ:DISTANCE BETWEEN NODES');
     ReadLn(FileName);{receive data distance between nodes}
     Assign(Filvar,FileName);
     Reset(Filvar);
     WHILE NOT Eof(Filvar)DO
     BEGIN
         ReadLn(Filvar,Nodes);
         i:=1;
         REPEAT
               j:=1;
               REPEAT
                     ReadLn(Filvar,Distance[i,j]);
                     j:=j+1;
               UNTIL j>Nodes;
               i:=i+1;
         UNTIL i>Nodes;
     END;
     Close(FilVar);
     m:=1;{start rcm method}
     REPEAT
           k:=1;
           REPEAT
                 IF Distance[k,m]>Infinity THEN L[k]:=0
                                           ELSE L[k]:=k;
                 k:=k+1;
           UNTIL k > Nodes;
           p:=1;
           REPEAT
                 IF Distance[m,p]>Infinity THEN N[p]:=0
                                           ELSE N[p]:=p;
                 p:=p+1;
           UNTIL p > Nodes;
           r:=1;
           REPEAT
                 s:=1;
                 REPEAT
                       z:=(L[r]>0)AND(N[s]>0)AND(r<>m)AND(s<>m)AND(r<>s);
                       IF z THEN
                       BEGIN
                             IF Distance[L[r],N[s]]>=(Distance[L[r],m]+Distance[m,N[s]])
                             THEN Distance[L[r],N[s]]:=(Distance[L[r],m]+Distance[m,N[s]])
                             ELSE Distance[L[r],N[s]]:=Distance[L[r],N[s]];
                       END;
                       s:=s+1;
                 UNTIL s > Nodes;
                 r:=r+1;
           UNTIL r > Nodes;
           m:=m+1;
     UNTIL m > Nodes;
     WriteLn('===SOLUTION FOR RCM METHOD===');
     WriteLn('ENTER NAME OF FILE TO WRITE');
     ReadLn(FileName);
     Assign(Filvar,FileName);
```

```
    ReWrite(Filvar);
    WriteLn(Filvar,Nodes);
    WriteLn('NO OF NODE   ',Nodes);
    FOR i:=1 TO Nodes DO
    FOR j:=1 TO Nodes DO
    BEGIN
        WriteLn(Filvar,Distance[i,j]);
        WriteLn('FROM  NODE  ',i,'  TO  NODE  ',j,'  SHORTPATH  ',Distance[i,j]);
    END;
    CLOSE(Filvar);
    WriteLn('PRINT SOLUTION OF RCM METHOD  Y/N  ? ');
    ReadLn(Answer);
    IF (Answer='y')or(Answer='Y') THEN
    BEGIN
         WriteLn(Lst,'NO OF NODE  ',Nodes);
         FOR i:=1 TO Nodes DO
         FOR J:=1 TO Nodes DO
         WriteLn(Lst,'FROM  NODE  ',i,'  TO  NODE  ',j,'  SHORTPATH  ',Distance[i,j]);
   END;
   WriteLn('END OF SOLUTION');
END.
```

```
PROGRAM SavingMethod;
USES Crt,Printer;
Var FilVar:Text;
    Saving,Shortpath,RestoreSaving:ARRAY[1..100,1..100] OF Integer;
    MaxCapacity,Capacity,SeqDemand,SeqNode:ARRAY[1..20,1..10] OF Integer;
    Demand,spt:ARRAY[1..100] OF Integer;
    SumDistance,SumCapacity,SumSeqDemand,CheckCapacity:ARRAY[1..20] OF Integer;
    i,j,Nodes,NoOfTruck,L,hz,vt,index,SumDemand,Sequence,loop:Integer;
    k,sp,spp,no,noo,recordd,MaxShortpath,MaxSaving,check,m,n:Integer;
    Answer:Char;
    FileName:STRING[14];
BEGIN
    Clrscr;
    WriteLn('====START SAVING METHOD====');
    WriteLn('1.ENTER NAME OF FILE TO READ SHORTPATH BETWEEN NODES:RCM METHOD');
    ReadLn(FileName);
    Assign(FilVar,FileName);
    Reset(FilVar);
    ReadLn(FilVar,Nodes);
    FOR i:=1 TO Nodes DO
    FOR j:=1 TO Nodes DO
    BEGIN
        ReadLn(FilVar,Shortpath[i,j]);
    END;
    Close(FilVar);
    FOR i:=3 TO Nodes DO {make saving}
    FOR j:=2 TO i DO
    BEGIN
        Saving[i,j]:=Shortpath[i,1]+Shortpath[j,1]-Shortpath[i,j];
    END;
    FOR i:=1 TO Nodes DO {make half matrix}
    FOR j:=i TO Nodes DO
    BEGIN
        Saving[i,j]:=0;
    END;
    FOR i:=1 TO Nodes DO
    BEGIN
        Saving[i,1]:=0;
    END;
    WriteLn('2.ENTER NAME OF FILE TO READ DATA OF TRUCK');
    ReadLn(FileName);
    Assign(FilVar,FileName);
    Reset(FilVar);
    ReadLn(FilVar,NoOfTruck);
    FOR i:=1 TO NoOfTruck DO
    FOR j:=1 TO 10 DO
    BEGIN
        ReadLn(FilVar,Capacity[i,j]);
    END;
    Close(FilVar);
    WriteLn('3.ENTER NAME OF FILE TO READ DEMAND OF NODES ');
    ReadLn(FileName);
    Assign(FilVar,FileName);
    Reset(FilVar);
    FOR i:=1 TO Nodes DO
    BEGIN
        ReadLn(FilVar,Demand[i]);
    END;
    Close(FilVar);
    i:=1;{check demand=0,make shortpath=0,saving=0}
    REPEAT
        IF Demand[i]=0 THEN
        BEGIN
            Shortpath[i,1]:=0;
            FOR j:=1 TO Nodes DO
            BEGIN
```

```
                Saving[i,j]:=0;
                Saving[j,i]:=0;
            END;
        END;
i:=i+1;
UNTIL i>Nodes;
FOR i:=1 TO NoOfTruck DO {make variable = 0}
FOR j:=1 TO 10 DO
BEGIN
    SeqDemand[i,j]:=0;
    SeqNode[i,j]:=0;
    MaxCapacity[i,j]:=0;
END;
FOR i:=1 TO NoOfTruck DO
BEGIN
     SumDistance[i]:=0;
     SumCapacity[i]:=0;
     SumSeqDemand[i]:=0;
     CheckCapacity[i]:=0;
END;
FOR i:=1 TO Nodes DO
FOR j:=1 TO Nodes DO
BEGIN
     RestoreSaving[i,j]:=0;
END;
FOR i:=1 TO NoOfTruck DO{find sum capacity of truck}
FOR j:=1 TO 10 DO
BEGIN
     SumCapacity[i]:=SumCapacity[i]+Capacity[i,j];
END;
i:=1;{no of truck: start saving method}
REPEAT
      MaxShortpath:=0;
      Sequence:=0;
      j:=1;{make max distance}
      REPEAT
            IF Shortpath[j,1]>MaxShortpath THEN
            BEGIN
                 MaxShortpath:=Shortpath[j,1];
                 no:=j;
                 recordd:=no;
            END;
      j:=j+1;
      UNTIL j>Nodes;
      L:=1;
      REPEAT
            IF (Demand[no]>0)and(Capacity[i,L]>0)and(MaxCapacity[i,L]=0) THEN
            BEGIN
                 IF Demand[no]>=Capacity[i,L] THEN
                 BEGIN
                      index:=Demand[no]-Capacity[i,L];
                      IF (index<>1)and(index<>2)and(index<>5) THEN
                      BEGIN
                           Sequence:=Sequence+1;
                           MaxCapacity[i,L]:=Capacity[i,L];
                           Demand[no]:=Demand[no]-Capacity[i,L];
                           SeqNode[i,Sequence]:=no;
                           SeqDemand[i,Sequence]:=Capacity[i,L];
                           SumSeqDemand[i]:=SumSeqDemand[i]+SeqDemand[i,Sequence];
                           recordd:=no;
                           Capacity[i,L]:=0;
                           IF Sequence=1 THEN noo:=1
                                         ELSE noo:=SeqNode[i,Sequence-1];
                           SumDistance[i]:=SumDistance[i]+Shortpath[noo,no];
                           FOR k:=1 TO Nodes DO
                           FOR n:=1 TO Nodes DO
```

```
                BEGIN
                    RestoreSaving[k,n]:=Saving[k,n];
                END;
            END;
        END;
    END;
L:=L+1;
UNTIL L>10;
IF Demand[no]=0 THEN
BEGIN
    Shortpath[no,1]:=0;
END;
FOR m:=1 TO Nodes DO
BEGIN
spt[m]:=0;
END;
   loop:=1;
   REPEAT
      MaxSaving:=0;
      hz:=1;{make saving for horizontal}
      REPEAT
          IF spt[hz]<>hz THEN
          BEGIN
              IF Saving[no,hz]>MaxSaving THEN
              BEGIN
                  MaxSaving:=Saving[no,hz];
                  sp:=hz;
              END;
          END;
      hz:=hz+1;
      UNTIL hz>Nodes;
      vt:=1;{make saving for vertical}
      REPEAT
          IF spt[vt]<>vt THEN
          BEGIN
              IF saving[vt,no]>MaxSaving THEN
              BEGIN
                  MaxSaving:=Saving[vt,no];
                  sp:=vt;
              END;
          END;
      vt:=vt+1;
      UNTIL vt>Nodes;
      IF Demand[no]=0 THEN
      BEGIN
          FOR k:=1 TO Nodes DO
          BEGIN
              Saving[no,k]:=0;
              Saving[k,no]:=0;
          END;
      END;
      L:=1;
      REPEAT
          IF (Demand[sp]>0)and(Capacity[i,L]>0)and(MaxCapacity[i,L]=0) THEN
          BEGIN
              IF Demand[sp]>=Capacity[i,L] THEN
              BEGIN
                  index:=Demand[sp]-Capacity[i,L];
                  IF (index<>1)and(index<>2)and(index<>5) THEN
                  BEGIN
                      Sequence:=Sequence+1;
                      MaxCapacity[i,L]:=Capacity[i,L];
                      Demand[sp]:=Demand[sp]-Capacity[i,L];
                      SeqDemand[i,Sequence]:=Capacity[i,L];
                      SumSeqDemand[i]:=SumSeqDemand[i]+SeqDemand[i,Sequence];
                      SeqNode[i,Sequence]:=sp;
```

```
                recordd:=sp;
                Capacity[i,L]:=0;
                IF Sequence=1 THEN spp:=1
                              ELSE spp:=SeqNode[i,Sequence-1];
                SumDistance[i]:=SumDistance[i]+Shortpath[spp,sp];
                FOR k:=1 TO Nodes DO
                FOR n:=1 TO Nodes DO
                BEGIN
                     RestoreSaving[k,n]:=Saving[k,n];
                END;
           END;
      END;
   END;
  L:=L+1;
  UNTIL L>10;
  IF Demand[sp]=0 THEN
  BEGIN
       Shortpath[sp,1]:=0;
  END;
  spt[no]:=no;
  no:=sp;
loop:=loop+1;
UNTIL loop>Nodes;
FOR L:=1 TO 10 DO
BEGIN
     CheckCapacity[i]:=CheckCapacity[i]+Capacity[i,L];
END;
IF (CheckCapacity[i]>0)and(Demand[recordd]>0) THEN
BEGIN
L:=1;
REPEAT
IF (Capacity[i,L]>0) and (Demand[recordd]>0) THEN
BEGIN
      no:=recordd;
      IF (Demand[no]>0)and(Capacity[i,L]>0)and(MaxCapacity[i,L]=0) THEN
      BEGIN
           IF Demand[no]>=Capacity[i,L] THEN
           BEGIN
                    Sequence:=Sequence+1;
                    MaxCapacity[i,L]:=Capacity[i,L];
                    Demand[no]:=Demand[no]-Capacity[i,L];
                    SeqDemand[i,Sequence]:=Capacity[i,L];
                    SumSeqDemand[i]:=SumSeqDemand[i]+SeqDemand[i,Sequence];
                    SeqNode[i,Sequence]:=no;
                    recordd:=no;
                    Capacity[i,L]:=0;
                    IF Sequence=1 THEN noo:=1
                                  ELSE noo:=SeqNode[i,Sequence-1];
                    FOR k:=1 TO Nodes DO
                    FOR n:=1 TO Nodes DO
                    BEGIN
                         RestoreSaving[k,n]:=Saving[k,n];
                    END;
                    SumDistance[i]:=SumDistance[i]+Shortpath[noo,no];
           END;
           IF (Demand[no]<Capacity[i,L]) and (Capacity[i,L]>0) and (Demand[no]>0) THEN
           BEGIN
                    Sequence:=Sequence+1;
                    MaxCapacity[i,L]:=Demand[no];
                    SeqDemand[i,Sequence]:=Demand[no];
                    SumSeqDemand[i]:=SumSeqDemand[i]+SeqDemand[i,Sequence];
                    SeqNode[i,Sequence]:=no;
                    recordd:=no;
                    Capacity[i,L]:=0;
                    Demand[no]:=0;
                    IF Sequence=1 THEN noo:=1
```

```
                                ELSE noo:=SeqNode[i,Sequence-1];
                        FOR k:=1 TO Nodes DO
                        FOR n:=1 TO Nodes DO
                        BEGIN
                             RestoreSaving[k,n]:=Saving[k,n];
                        END;
                        SumDistance[i]:=SumDistance[i]+Shortpath[noo,no];
                END;
           END;
      END;
      L:=L+1;
      UNTIL L>10;
      END;
      CheckCapacity[i]:=0;
      FOR L:=1 TO 10 DO
      BEGIN
           CheckCapacity[i]:=CheckCapacity[i]+Capacity[i,L];
      END;
IF (CheckCapacity[i]>0)and(Demand[recordd]=0) THEN
BEGIN
IF Demand[no]=0 THEN
BEGIN
     Shortpath[no,1]:=0;
END;
FOR m:=1 TO Nodes DO
BEGIN
spt[m]:=0;
END;
      FOR k:=1 TO Nodes DO
      FOR n:=1 TO Nodes DO
      BEGIN
           Saving[k,n]:=RestoreSaving[k,n];
      END;
      no:=recordd;
      loop:=1;{make sure  one truck has 10 channels}
      REPEAT
         MaxSaving:=0;
         hz:=1;{make saving for horizontal}
         REPEAT
               IF spt[hz]<>hz THEN
               BEGIN
                    IF Saving[no,hz]>MaxSaving THEN
                    BEGIN
                         MaxSaving:=Saving[no,hz];
                         sp:=hz;
                    END;
               END;
         hz:=hz+1;
         UNTIL hz>Nodes;
         vt:=1;{make saving for vertical}
         REPEAT
               IF spt[vt]<>vt THEN
               BEGIN
                    IF saving[vt,no]>MaxSaving THEN
                    BEGIN
                         MaxSaving:=Saving[vt,no];
                         sp:=vt;
                    END;
               END;
         vt:=vt+1;
         UNTIL vt>Nodes;
         IF Demand[no]=0 THEN
         BEGIN
              FOR k:=1 TO Nodes DO
              BEGIN
                   Saving[no,k]:=0;
```

```
                Saving[k,no]:=0;
            END;
        END;
      L:=1;
      REPEAT
      IF (Capacity[i,L]>0) and (Demand[sp]>0) THEN
      BEGIN
          IF (Demand[sp]>0)and(Capacity[i,L]>0)and(MaxCapacity[i,L]=0) THEN
          BEGIN
              IF Demand[sp]>=Capacity[i,L] THEN
              BEGIN
                      Sequence:=Sequence+1;
                      MaxCapacity[i,L]:=Capacity[i,L];
                      Demand[sp]:=Demand[sp]-Capacity[i,L];
                      SeqDemand[i,Sequence]:=Capacity[i,L];
                      SumSeqDemand[i]:=SumSeqDemand[i]+SeqDemand[i,Sequence];
                      SeqNode[i,Sequence]:=sp;
                      recordd:=sp;
                      Capacity[i,L]:=0;
                      IF Sequence=1 THEN spp:=1
                                    ELSE spp:=SeqNode[i,Sequence-1];
                      FOR k:=1 TO Nodes DO
                      FOR n:=1 TO Nodes DO
                      BEGIN
                          RestoreSaving[k,n]:=Saving[k,n];
                      END;
                      SumDistance[i]:=SumDistance[i]+Shortpath[spp,sp];
              END;
              IF (Demand[sp]<Capacity[i,L]) and (Capacity[i,L]>0) and (Demand[sp]>0) THEN
              BEGIN
                      Sequence:=Sequence+1;
                      MaxCapacity[i,L]:=Demand[sp];
                      SeqDemand[i,Sequence]:=Demand[sp];
                      SumSeqDemand[i]:=SumSeqDemand[i]+SeqDemand[i,Sequence];
                      SeqNode[i,Sequence]:=sp;
                      recordd:=sp;
                      Demand[sp]:=0;
                      Capacity[i,L]:=0;
                      IF Sequence=1 THEN spp:=1
                                    ELSE spp:=SeqNode[i,Sequence-1];
                      FOR k:=1 TO Nodes DO
                      FOR n:=1 TO Nodes DO
                      BEGIN
                          RestoreSaving[k,n]:=Saving[k,n];
                      END;
                      SumDistance[i]:=SumDistance[i]+Shortpath[spp,sp];
              END;
          END;
      END;
      L:=L+1;
      UNTIL L>10;
      IF Demand[sp]=0 THEN
      BEGIN
          Shortpath[sp,1]:=0;
      END;
      spt[no]:=no;
      no:=sp;
    loop:=loop+1;
    UNTIL loop>10;
    END;
    IF Sequence=0 THEN SumDistance[i]:=0
                  ELSE SumDistance[i]:=SumDistance[i]+Shortpath[1,SeqNode[i,Sequence]];
i:=i+1;
UNTIL i>NoOfTruck;
FOR i:=1 TO NoOfTruck DO
FOR j:=1 TO 10 DO
```

```
BEGIN
END;
SumDemand:=0;
FOR i:=1 TO Nodes DO
BEGIN
SumDemand:=SumDemand+Demand[i];
END;
Clrscr;
WriteLn('===SOLUTION OF SAVING METHOD  ===');
i:=1;
REPEAT
     IF i>10 THEN  REPEAT UNTIL KeyPressed ;
     Write(i,' TRUCK NO ',i,' CAPACITY ',SumCapacity[i]);
     WriteLn(' Total DO.  ',SumSeqDemand[i],'  Total  Dist. ' ,SumDistance[i]);
     Write('D-NODE  ');
     FOR j:=1 TO 10 DO
     BEGIN
          Write(SeqNode[i,j],'-');
     END;
     {WriteLn;}
     Write('  D-ORDER  ');
     FOR j:=1 TO 10 DO
     BEGIN
          Write(SeqDemand[i,j],'-');
     END;
     WriteLn;
     {WriteLn('SUM OF DELIVERY ORDER   ',SumSeqDemand[i],'   SUM OF DISTANCE  ',SumDistance[i]); }
i:=i+1;
UNTIL i>NoOfTruck;
WriteLn('-------------------------------------------------');
WriteLn('  LACK OF DEMAND     ',SumDemand);
WriteLn('PRINT DATA SOLUTION OF SAVING METHOD  Y/N  ? ');
ReadLn(Answer);
IF(Answer='y')or(Answer='Y') THEN
BEGIN
WriteLn(Lst,'=== SOLUTION  OF  SAVING  METHOD ===');
FOR i:=1 TO NoOfTruck DO
     BEGIN
          WriteLn(Lst);
          WriteLn(Lst,'  TRUCK  NO  ',i,'  CAPACITY  ',SumCapacity[i]);
          Write(Lst,'DELIVERY NODE  ');
          FOR j:=1 TO 10 DO
          BEGIN
               Write(Lst,SeqNode[i,j],'-');
          END;
          WriteLn(Lst);
          Write(Lst,'DELIVERY ORDER  ');
          FOR j:=1 TO 10 DO
          BEGIN
               Write(Lst,SeqDemand[i,j],'-');
          END;
          WriteLn(Lst);
          WriteLn(Lst,'SUM OF DELIVERY ORDER   ',SumSeqDemand[i],'   SUM OF DISTANCE  ',SumDistance[I]);
     END;
     WriteLn(Lst,'-----------------------------------------');
     WriteLn(Lst,'  LACK OF DEMAND     ',SumDemand);
     WriteLn(Lst,'-----------------------------------------');
     WriteLn(Lst,'END OF SOLUTION');
END;
WriteLn('END OF SOLUTION');
END.
```

ประวัติผู้เขียน

นาย กฤษไกร มนิมนากร เกิดวันที่ 20 มกราคม 2507 จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมโยธา คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2527 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ที่ จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ.2533