วิศวกรรมสารฉบับวิจัยและพัฒนา
Engineering Journal of Research and Development
วิศวกรรมสถานแห่งประเทศไทย ในพระบรมราชูปถัมภ์
The Engineering Institute of Thailand under H.M. The King's Patronage
ปีที่ 30 ฉบับที่ 1 มกราคม-มีนาคม 2562
Volume 30 Issue 1 January-March 2019
Received 18 April 2018
Revised 31 May 2018
Accepted 10 June 2018
Physical Sciences

อัลกอริทึมการแตกกิ่งและกำหนดขอบเขตสำหรับปัญหาการเดินทางของพนักงานขายภายใต้ความน่าจะเป็นที่กลับไปยังจุดเริ่มเดินทาง

A BRANCH AND BOUND ALGORITHM FOR THE PROBABILISTIC TRAVELING SALESMAN PROBLEM WITH RETURN

ศรีรักษ์ ศรีทองชัย
ผู้ช่วยศาสตราจารย์, ภาควิชาวิศวกรรมอุตสาหการและการจัดการ คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม
มหาวิทยาลัยศิลปากร

บทคัดย่อ

งานวิจัยนี้ได้นำหลักการของการแตกกิ่งและกำหนดขอบเขตมาสร้างอัลกอริทึมที่ใช้สำหรับแก้ปัญหาการเดินทางของพนักงานขายภายใต้ความน่าจะเป็นที่กลับไปยังจุดเริ่มเดินทาง กระบวนการประกอบด้วย 3 ขั้นตอนหลัก ขั้นตอนการแตกกิ่งเป็นขั้นแรกของการดำเนินการเกี่ยวข้องกับการแบ่งแยกปัญหาออกเป็นปัญหาย่อยขนาดเล็กลงเพื่อช่วยให้แก้ไขปัญหาได้ง่ายและรวดเร็วขึ้น คำตอบของแต่ละปัญหาย่อยจะถูกตรวจสอบความสมเหตุสมผลต่อไปโดยเปรียบเทียบกับขีดจำกัดบนและขีดจำกัดล่าง ขั้นที่สองคือการกำหนดค่าขอบเขตทั้งขอบเขตบนและขอบเขตล่างจะได้รับการปรับค่าให้ดีขึ้นกว่าเดิม โดยคำตอบที่เป็นไปได้จะอยู่ในช่วงขอบเขตบนและขอบเขตล่างเท่านั้น ขั้นที่สามคือการเลือกโหนดด้วยวิธีการโดดข้ามกิ่งไปแก้ปัญหาย่อยที่ให้ผลลัพธ์ดีที่สุดก่อน อัลกอริทึมจะทำงานไปเรื่อยๆ ผ่านการคำนวณหลายรอบจนกระทั่งค้นพบคำตอบที่เหมาะสมที่สุด คำตอบดังกล่าวคือลำดับการเดินทางที่ให้เส้นทางที่สั้นที่สุด ขั้นตอนข้างต้นยังได้ถูกนำไปพัฒนาต่อเป็นโปรแกรมคอมพิวเตอร์ที่สามารถหาผลเฉลยได้ในเวลาที่ยอมรับได้ เช่น ขนาดเมือง 15 เมืองใช้เวลาประมวลผลไม่ถึงวินาทีและจำนวนเมือง 800 เมืองใช้เวลาคำนวณประมาณ 30 นาที

คำสำคัญ: อัลกอริทึม, การแตกกิ่งและกำหนดขอบเขต, ปัญหาการเดินทางของพนักงานขาย

ABSTRACT

This paper presents a branch and bound algorithm for the solution of the Probabilistic Traveling Salesman Problem with Return. The algorithm is composed of three main steps. In the first step, branching divides the problem into smaller subproblems that are easier and quicker to handle. All possible solutions are then checked for feasibility by comparing with the boundaries (lower and upper bounds). The second step, known as bounding, is to find lower and upper bounds for the solution at each node that are continually refined over time. A feasible solution must lie between these bounds. In the third step, called node selection, the jump-tracking strategy chooses a node with the best result to process. For obtaining the optimal solution, many iterations are executed until the shortest route is identified. A computer program was developed based on the algorithm, and number of test cases have been solved in an acceptable time; a 15 city problem took less than a second, and a 800 city problem required about 30 minutes.

KEYWORDS: algorithm, branch and bound, traveling salesman problem

Sriruk Srithongchai
Assistant Professor, Department of Industrial Engineering and Management,
Faculty of Engineering and Industrial Technology Silpakorn University

1. บทนำ

ปัญหาการเดินทางของพนักงานขายภายใต้ความน่าจะเป็นที่กลับไปยังจุดเริ่มเดินทาง (Probabilistic Traveling Salesman Problem with Return: PTSP-R) เป็นปัญหาที่ขยายมาจากปัญหาการเดินทางของพนักงานขาย (Traveling Salesman Problem: TSP) ซึ่งเป็นที่รู้จักกันอย่างกว้างขวางและได้รับความสนใจจากนักวิจัยอย่างต่อเนื่องเป็นระยะเวลานาน ตั้งแต่ศตวรรษที่ 18 ถึงปัจจุบันมีนักคณิตศาสตร์ที่มีชื่อเสียงหลายท่านมีส่วนร่วมในการพัฒนาหลักการและทฤษฎีที่เกี่ยวข้องกับปัญหา TSP ในยุคแรก Euler และ Hamilton [1] เป็นผู้ริเริ่มนำวิธีการทางคณิตศาสตร์ไปใช้ในการค้นหาคำตอบของปัญหา ปี ค.ศ.1736 Euler สนใจหาเส้นทางเดินเท้าที่ผ่านทุกเกาะของเมืองเคอนิจส์แบร์ก (Konigsberg Bridge Problem) โดยข้ามสะพานทั้งเจ็ดแห่งแต่ละสะพานเพียงครั้งเดียว ต่อมาในปี 1857 Hamilton ได้ศึกษาเส้นทางเดินบนขอบ (Edge) ของกราฟไอโคเซียน (Icosian) ที่ผ่านทุกจุดยอดมุม (Vertex) แต่ไม่ผ่านจุดยอดใดจุดยอดหนึ่งมากกว่าหนึ่งครั้งโดยมีจุดเริ่มต้นกับสิ้นสุดเป็นจุดเดียวกัน จากนั้นช่วงปี ค.ศ.1930 Menger [2] นำเสนอปัญหาเส้นทางการเดินทางที่สั้นที่สุดของผู้ส่งสาร (Messenger Problem) หากแต่ไม่กลับไปเมืองเริ่มต้นเมื่อจบสิ้นการเดินทาง ผลงานของ Flood ปี 1937 [3] ในเรื่องการจัดเส้นทางการเดินรถของรถรับ-ส่งนักเรียน (School Bus-Routing Problem) ให้เหมาะสม ก่อให้เกิดความตื่นตัวต่อปัญหา TSP เป็นอย่างสูง ในปี ค.ศ.1954 Dantzig, Fulkerson และ Johnson [4] ถือว่าเป็นกลุ่มบุคคลที่มีส่วนสำคัญยิ่งต่อความรุดหน้าในการแก้ปัญหา TSP เขาได้เสนอวิธีระนาบตัด (Cutting Plane Method) สำหรับปัญหากำหนดการจำนวนเต็ม (Integer Programming: IP) และยังใช้แก้ปัญหาการเดินทางที่มีจำนวนเมือง 49 เมือง (49-City Problem) เมื่อเข้าสู่ยุคคอมพิวเตอร์จึงมีการพัฒนาโปรแกรมสำเร็จรูปมาใช้งานเพื่อให้การดำเนินงานสำเร็จด้วยความรวดเร็ว ถูกต้องมากยิ่งขึ้น และยังช่วยแก้ปัญหาที่ซับซ้อนต่างๆ ปัจจุบันมีซอฟแวร์ชื่อว่า Concorde [5] เป็นโปรแกรมที่พัฒนาจากกลุ่มนักวิจัยนำทีมโดย Applegate มีประสิทธิภาพสูงใช้แก้ปัญหา TSP ขนาดใหญ่ที่มีจำนวนเมืองมากถึง 85,900 เมือง ให้ใช้งานฟรีเพื่อการศึกษา (Academic Use)

ปัญหา TSP จัดเป็นปัญหาการหาค่าเหมาะสมที่สุดเชิงการจัด (Combinatorial Optimization Problem) แต่ปัญหาจริงที่พบทั่วไปจะมีเรื่องความไม่แน่นอนมาเกี่ยวข้องนั่นคือเหตุการณ์ใดๆ มีโอกาสเกิดขึ้นได้ด้วยความน่าจะเป็น แนวคิดของปัญหาการเดินทางของพนักงานขายภายใต้ความน่าจะเป็น (Probabilistic Traveling Salesman Problem: PTSP) กล่าวครั้งแรกในปี 1985 โดย Jaillet [6, 7] ถึงความน่าจะเป็นของการเยือนเมืองต่างๆ เส้นทางเบื้องต้น (Prior Tour) ที่สั้นที่สุดในการเดินทางผ่านเมืองทั้งหมดจะถูกนำมาใช้หาเส้นทางใหม่ (Posterior Tour) ด้วยการตัดเมืองที่ไม่จำเป็นต้องแวะออกจากเส้นทางเบื้องต้น โดยลำดับของการแวะเมืองยังคงเหมือนเดิมเพียงแต่ข้ามเมืองที่ไม่ได้แวะเท่านั้น เช่น ลูกค้าบางรายไม่ได้สั่งสินค้าในทุกรอบการขนส่ง สำหรับปัญหา PTSP-R [8] เป็นรูปแบบใหม่ของปัญหา TSP ส่วนที่แตกต่างจากปัญหา PTSP คือการเดินทางจะสิ้นสุดลงหากตรวจพบเป้าหมายที่สนใจ (Object of Interest) สามารถนำไปประยุกต์ใช้ให้เกิดประโยชน์ในงานค้นหาบุคคลหรือสิ่งของที่สูญหายในพื้นที่ที่กำหนดซึ่งมักอาศัยทฤษฎีค้นหา (Search Theory) [9] เป็นเครื่องมือในการแก้ปัญหา PTSP-R ถือเป็นทางเลือกหนึ่งที่ช่วยในการวางแผนการค้นหาอย่างมีระบบโดยเฉพาะการค้นหากู้ภัย (Search and Rescue: SAR) เพราะทุกๆ วินาทีที่ผ่านไปหมายถึงชีวิตของผู้ประสบเหตุ

เนื่องจากนักวิจัยส่วนใหญ่นิยมแก้ปัญหา TSP ขนาดใหญ่โดยใช้การประมาณค่า [10] ด้วยวิธีฮิวริสติก (Heuristic Method) หรือเมตาฮิวริสติก (Metaheuristic Method) เพราะให้คำตอบได้อย่างรวดเร็ว แม้ว่าจะเป็นเพียงคำตอบใกล้เคียง (Approximate Solution) และอาจไม่ใช่คำตอบที่ดีที่สุด (Optimal Solution) เสมอไป งานวิจัยนี้จึงมีวัตถุประสงค์ที่จะพัฒนาอัลกอริทึม (Algorithm) วิธีแตกกิ่งและกำหนดขอบเขต (Branch and Bound Method: B&B) เพื่อหาคำตอบแม่นตรง (Exact Solution) ของปัญหาการเดินทางของพนักงานขายภายใต้ความน่าจะเป็นที่เดินทางกลับสู่จุดตั้งต้น ซึ่งยังอาจใช้เป็นแนวทางการตรวจสอบความถูกต้องของคำตอบใกล้เคียงรวมทั้งประเมินความเหมาะสมของวิธีการที่ใช้หาผลลัพธ์ดังกล่าว

2. ปัญหาการเดินทางของพนักงานขายด้วยค่าความน่าจะเป็นที่กลับไปยังจุดเริ่มเดินทาง

ปัญหา PTSP-R และปัญหา TSP มีลักษณะคล้ายคลึงกัน แต่มีความแตกต่างกันโดยการเดินทางแบบ PTSP-R ไม่จำต้องผ่านเมืองทุกเมืองและมีค่าความน่าจะเป็นระหว่าง 0 ถึง 1 ในการเยี่ยมเมืองแต่ละเมือง (ขณะที่ TSP มีค่าเพียง 2 ค่า คือ 0 หรือ 1) หลังจากค้นพบสิ่งที่สนใจก็จะเดินทางกลับสู่จุดตั้งต้นและยุติการเดินทาง กรณีตัวอย่างของปัญหา PTSP-R ในการใช้งานจริง อาทิเช่น ภารกิจการปฏิบัติการใต้น้ำของยานใต้น้ำไร้คนขับ (Unmanned Underwater Vehicle: UUV) ซึ่งกองทัพเรือนำไปใช้เพื่อสำรวจทุ่นระเบิดในท้องทะเล, ค้นหาทำลายเรือดำน้ำของข้าศึกและกู้ภัยเครื่องบินตกกลางทะเล ปัญหา PTSP-R ถูกเสนอในปี 2009 โดย Gudbrandsen [8] ได้ศึกษาหาเส้นทางที่ใช้ต้นทุนต่ำสุดในการค้นหาสิ่งที่สูญหายโดยคำนึงถึงระยะทางการเดินทางและโอกาสที่จะประสบความสำเร็จของการพบเป้าหมาย ด้วยการอาศัยวิธีการฮิวริสติกแบบหาตำแหน่งที่ใกล้ที่สุด (Nearest Neighbor Heuristic) มาใช้แก้ปัญหาที่มีจำนวนเมือง 50 เมือง

หลักการและแนวคิดของปัญหา TSP จะช่วยทำให้เข้าใจปัญหา PTSP-R ได้ง่ายขึ้น ปัญหา TSP นั้นเป็นการหาลำดับการเดินทางที่ใช้ระยะทางรวม (อาจแทนด้วยเวลาหรือค่าใช้จ่าย) น้อยที่สุดของพนักงานขายหนึ่งคนที่ต้องเดินทางไปยังเมืองต่างๆ ทุกเมืองและกลับมาที่จุดเริ่มต้นเดินทางโดยให้ผ่านแต่ละเมืองได้เพียงครั้งเดียวเท่านั้น ในที่นี้จะเป็นการพิจารณาปัญหาการเดินทางของพนักงานขายที่มีระยะทางไปและกลับเท่ากัน (Symmetric Traveling Salesman Problem) ถ้ากำหนดให้มีเมือง N เมืองจะได้เส้นทางการเดินทางที่เป็นไปได้ทั้งหมด $(N\text{-}1)!/2$ เช่น การไปไหว้พระ 9 วัดเพื่อเสริมสิริมงคลในจังหวัดอยุธยาจะมีจำนวนเส้นทางให้เลือกเดินทางเท่ากับ 8!/2 คือ 20,160 เส้นทาง จึงเป็นเรื่องยากมากสำหรับการหาเส้นทางที่สั้นที่สุด ในปี ค.ศ. 1954 Dantzig และคณะ [3] ได้กล่าวถึงตัวแบบทางคณิตศาสตร์ซึ่งมีรูปแบบเป็นกำหนดการจำนวนเต็มที่มีเป้าหมายในการหาเส้นทางที่มีระยะการเดินทางรวมต่ำที่สุดและอยู่ภายใต้ข้อกำหนดต่างๆ (Constraint)

ตัวแบบกำหนดการจำนวนเต็มที่ใช้แทนปัญหา TSP มีตัวแปรตัดสินใจ (Decision Variable) เป็นตัวแปรแบบไบนารี (Binary Variable) ที่อาจเป็นค่า 0 หรือ 1 ดังนี้

$$x_{ij} = \begin{cases} 1 & \text{กรณีที่พนักงานขายเดินทางจากเมือง } i \text{ ไปเมือง } j \\ 0 & \text{อื่นๆ} \end{cases} \quad (1)$$

เราต้องการค่าระยะทางรวมสั้นที่สุดซึ่งคำนวณได้จากผลรวมระยะทางระหว่างเมืองที่พนักงานขายเดินผ่าน เขียนแทนด้วยตัวแปร z โดยที่ระยะห่างระหว่างเมือง i และเมือง j คือ d_{ij} จะได้สมการเป้าหมายหรือฟังก์ชั่นวัตถุประสงค์ (Objective Function) คือ

$$\min z = \sum_i \sum_j d_{ij} x_{ij} \quad (2)$$

พนักงานขายเดินทางไปแวะเมืองแต่ละเมืองได้เพียงครั้งเดียวเท่านั้น นั่นคือเดินทางเข้าเมืองใดๆ ได้ครั้งเดียว และเดินทางออกได้เพียงครั้งเดียวดังสมการที่ (3) และ (4) ตามลำดับ นอกจากนี้ยังมีเงื่อนไขว่าจะต้องไม่เกิดการเดินทางย่อย (Subtour) ที่เดินทางไม่ครบทุกเมืองแสดงในสมการที่ (5) กำหนดตัวแปรให้ u_i แทนหมายเลขแสดงลำดับของเมืองที่แวะ i

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (\text{for } j = 1, 2, \ldots, N) \quad (3)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{(for } i = 1, 2, ..., N\text{)} \tag{4}$$

$$u_i - u_j + N x_{ij} \leq N - 1 \qquad \text{(for } i \neq j;\ i = 1, 2, ..., N;\ j = 1, 2, ..., N;\ u_j \geq 0\text{)} \tag{5}$$

ตัวแบบกำหนดการจำนวนเต็มของปัญหา TSP [11] ดังกล่าว จะเขียนได้ดังนี้

$$\min z = \sum_i \sum_j d_{ij} x_{ij} \tag{2}$$

s.t.

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \text{(for } j = 1, 2, ..., N\text{)} \tag{3}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{(for } i = 1, 2, ..., N\text{)} \tag{4}$$

$$u_i - u_j + N x_{ij} \leq N - 1 \qquad \text{(for } i \neq j;\ i = 1, 2, ..., N;\ j = 1, 2, ..., N\text{)} \tag{5}$$

$$\text{All } x_{ij} = 0 \text{ or } 1;\ \text{All } u_j \geq 0 \tag{1}$$

ปัญหา PTSP-R มีวัตถุประสงค์เพื่อให้การเดินทางจากต้นทางไปยังปลายทางได้ระยะทาง, เวลา หรือค่าใช้จ่ายรวมน้อยที่สุด ตารางที่ 1 [12] เป็นข้อมูล (ได้จากวิธีการสุ่ม) ระยะห่างจากเมืองหนึ่งไปยังอีกเมืองหนึ่ง (d_{ij}) จำนวน 6 เมือง แบบมีระยะทางเท่ากันทั้งไปและกลับ และค่าความน่าจะเป็นที่จะพบความสำเร็จในการค้นหาที่เมืองนั้นๆ (p_i) หากกำหนดให้เริ่มต้นออกเดินทางจากเมือง 1 คำตอบที่ดีที่สุดที่ได้จากอัลกอริทึมซึ่งพัฒนามาจากวิธีแตกกิ่งและกำหนดขอบเขตคือเดินทางออกจากเมือง 1 ไปเมือง 5, เมือง 3, เมือง 6, เมือง 2 และเมือง 4 ตามลำดับแล้วกลับมายังจุดเริ่มต้น ตามเส้นทาง 1-5-3-6-2-4-1 ได้ระยะทางตลอดเส้นทางรวมกัน 37.98 หน่วย (ดูรายละเอียดในหัวข้อ 3) เขียนเป็นแผนภาพต้นไม้ (Tree Diagram) ได้ดังรูปที่ 1 มีลักษณะลดหลั่นกันเป็นชั้นๆ จากระดับบนสุดหรือเรียกว่าระดับ (Level) 0, ระดับ 1 ไปถึงระดับล่างสุด ความสูงของต้นไม้เท่ากับ (จำนวนเมือง–1) ใช้สัญลักษณ์รูปวงกลมแทนโหนด (Node) ของต้นไม้ ตัวเลขด้านบนของวงกลมแทนหมายเลขเมืองและตัวเลขในวงกลมคือค่าระยะทางรวม

ตารางที่ 1 ระยะทางและความน่าจะเป็นของความสำเร็จในการค้นหาที่มีจำนวนเมือง 6 เมือง

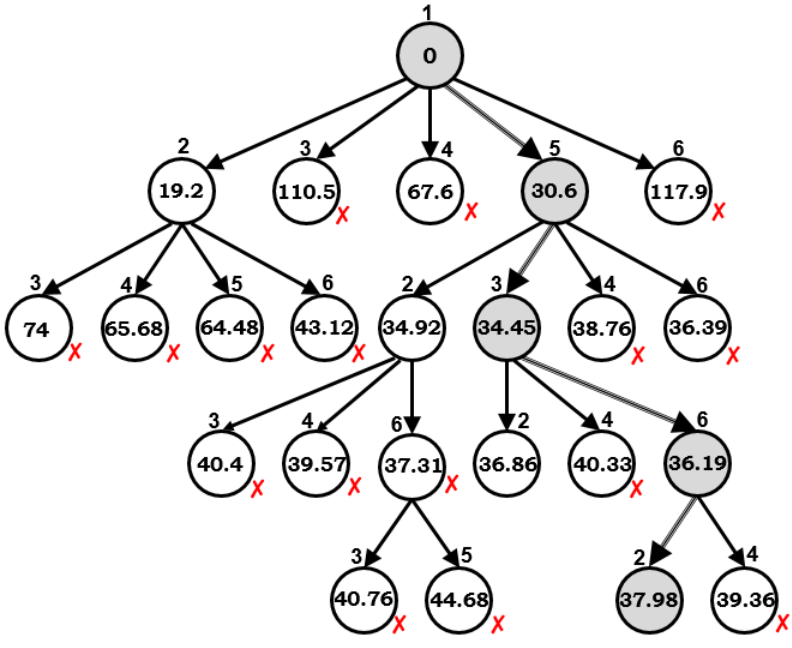
	ระยะทางระหว่างเมือง (d_{ij})						ความน่าจะเป็น
เมือง	1	2	3	4	5	6	(p_i)
1	0	8	72	35	15	88	0.0
2	8	0	42	27	53	12	0.2
3	72	42	0	90	21	5	0.5
4	35	27	90	0	70	62	0.7
5	15	53	21	70	0	49	0.9
6	88	12	5	62	49	0	0.3

3. อัลกอริทึมแตกกิ่งและกำหนดขอบเขต (Branch and Bound Algorithm)

งานวิจัยนี้ใช้อัลกอริทึมที่อาศัยหลักการของการแตกกิ่งและกำหนดขอบเขตเป็นเครื่องมือช่วยในการแก้ปัญหา PTSP-R และยังนำมาใช้เป็นโครงร่างการเขียนโปรแกรมคอมพิวเตอร์เพื่อความสะดวกในการใช้งาน ช่วยประหยัดเวลาและลดข้อผิดพลาดในการคำนวณที่ซับซ้อน ผังงาน (Flowchart) ของอัลกอริทึมแสดงดังรูปที่ 2 ตั้งแต่ขั้นตอนแรกจนกระทั่งถึงขั้นตอนสุดท้าย บางขั้นตอน

อาจทำงานแบบวนซ้ำๆ ตามเงื่อนไขที่กำหนด ประกอบด้วย 3 ขั้นตอนหลักได้แก่

(1) การแตกกิ่ง (Branching) เป็นการแบ่งปัญหาใหญ่ให้เป็นปัญหาย่อย (Subproblem) เปรียบเสมือนโครงสร้างต้นไม้ที่แตกกิ่งก้านหรือแขนงออกไป มีลักษณะคล้ายต้นไม้จริงแบบกลับหัวโดยมีโหนดราก (Root Node) อยู่ด้านบนสุดแทนปัญหาหลัก โหนดใดๆที่ได้จากการแตกกิ่งออกไปใช้แทนปัญหาย่อยตามลำดับชั้น แต่ละโหนดของกิ่งสาขาจะให้ค่าผลลัพธ์ของปัญหาย่อยออกมาเป็นผลเฉลยที่เป็นไปได้ (Feasible Solution) หรืออาจเป็นผลเฉลยที่เป็นไปไม่ได้ (Infeasible Solution) ปัญหาย่อยที่ทำให้เกิดผลเฉลยที่เป็นไปได้จะถูกแยกย่อยอีกเพื่อค้นหาคำตอบที่ดีขึ้นอันจะนำไปสู่คำตอบที่ดีที่สุด (Optimal Solution) ขณะที่ปัญหาย่อยที่ไม่มีคำตอบที่เป็นไปได้หรือได้ให้ผลเฉลยที่สามารถเลือกได้ (Candidate Solution) จะไม่ถูกแตกย่อยต่อไปอีก (Fathoming)



รูปที่ 1 แผนภาพต้นไม้แทนเส้นทางการเดินทางผ่านเมือง 6 เมือง

(2) การกำหนดค่าขอบเขต (Bounding) ของคำตอบสำหรับปัญหาย่อย จะได้ขอบเขตบนและล่าง (Upper and Lower bounds) เพื่อนำไปใช้ในการตัดปัญหาย่อยที่ไม่สามารถให้คำตอบที่ดีที่สุดออกไป กล่าวคือคำตอบที่เป็นไปได้จะต้องมีค่าไม่น้อยกว่าขอบเขตล่าง (Lower Bound) และไม่มากกว่าขอบเขตบน (Upper Bound) กรณีปัญหาการหาค่าต่ำสุด (Minimization Problem) ค่าขอบเขตล่างพัฒนาจากคำตอบที่เป็นไปได้ ค่านี้จะสูงขึ้นเรื่อยๆ ขณะที่ค่าขอบเขตบนมาจากคำตอบที่สามารถเลือกได้ มีค่าดีขึ้นเรื่อยๆ เช่นกัน

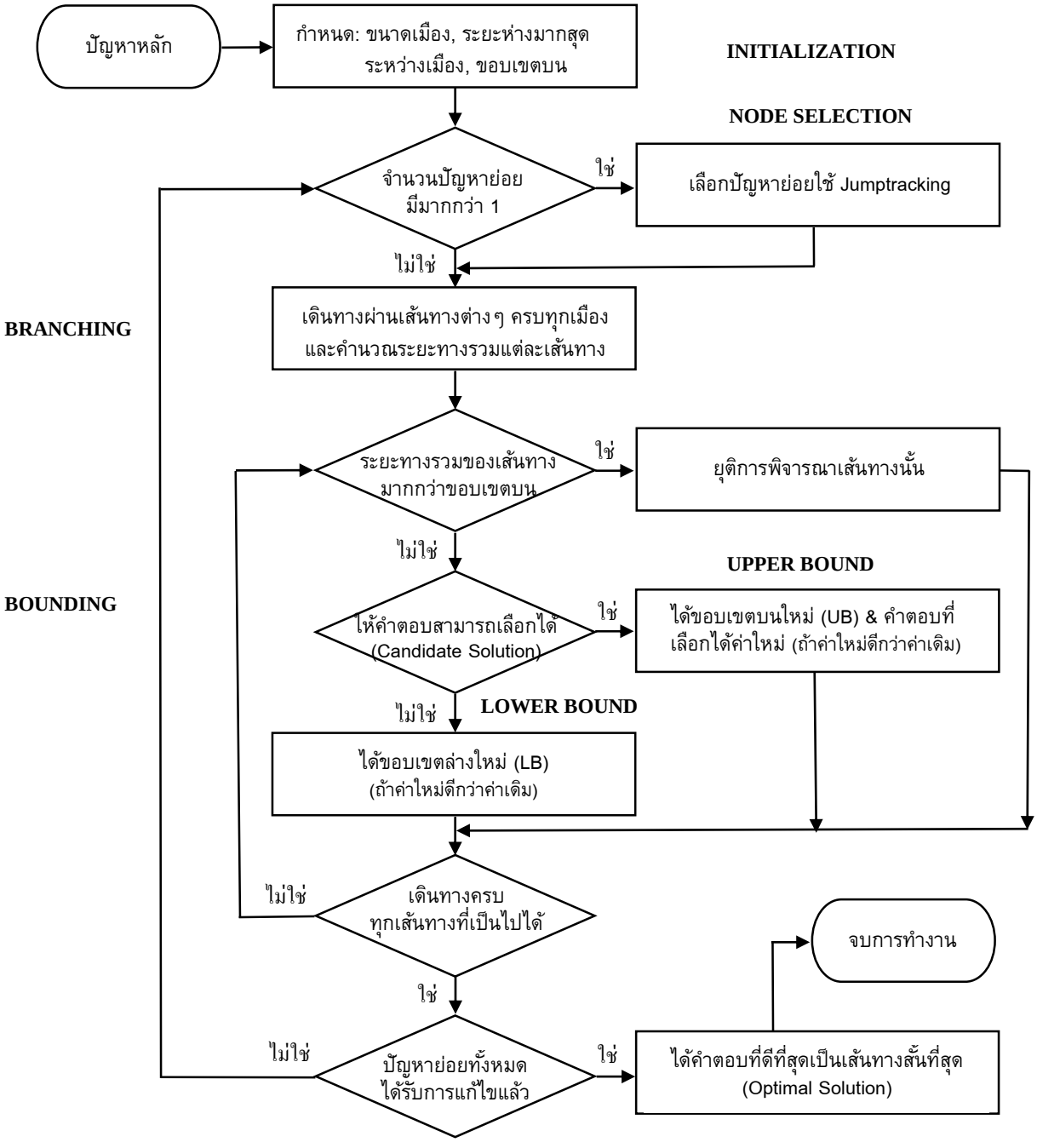
(3) การเลือกโหนด (Node Selection) กรณีที่มีมากกว่าหนึ่งปัญหาย่อยหรือโหนด เราใช้วิธีการโดดข้ามกิ่ง [11] จัดลำดับความสำคัญเพื่อให้ทราบว่าปัญหาย่อยใดควรดำเนินการแก้ไขในลำดับถัดไป ปัญหาย่อยที่ได้คำตอบที่ดีที่สุดจะได้รับเลือกก่อน

ตารางที่ 1 เป็นตัวอย่างปัญหาที่มีขนาดเมือง 6 เมือง เมือง 1 คือเมืองเริ่มต้นในการเดินทาง การหาเส้นทางที่สั้นที่สุดซึ่งคือเส้นทางที่ให้ผลรวมของระยะทางน้อยที่สุดโดยประยุกต์ใช้อัลกอริทึมข้างต้นต้องทำการคำนวณหลายๆ รอบ เพื่อให้คำตอบถูกต้องมากยิ่งขึ้น ผลลัพธ์ในแต่ละรอบการคำนวณ (Iteration) ตั้งแต่รอบที่ 1 ถึงรอบที่ 3 สรุปเป็นขั้นตอนได้ดังนี้

<u>รอบที่ 1</u> (ดูรูปที่ 3 ก)

ขั้นตอนที่ 1 (การแตกกิ่ง): เริ่มต้นออกเดินทางจากเมือง 1 ไปยังเมืองอื่น จึงแตกกิ่งที่โหนด 1 ซึ่งแทนเมือง 1

หากพิจารณาเมือง 1 ไปเมือง 2 จากเมือง 2 อาจจะกลับมาที่จุดเริ่มต้นหรือเดินทางต่อไปยังเมืองใกล้ที่สุดจากเมืองปัจจุบัน ในทำนองเดียวกันเมื่อเดินทางต่อไปจนครบทุกเมืองจะได้เส้นทางการเดินทาง 1-2-X-X-X-X-1 มีระยะทางรวมเท่ากับ

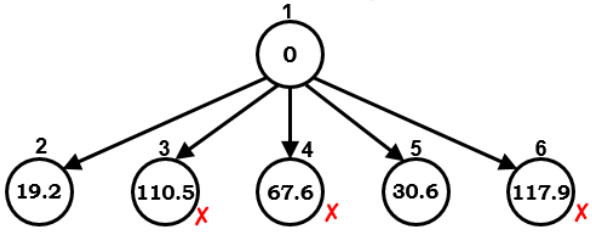


รูปที่ 2 ผังงานของอัลกอริทึมวิธีแตกกิ่งและกำหนดขอบเขตสำหรับปัญหา PTSP-R

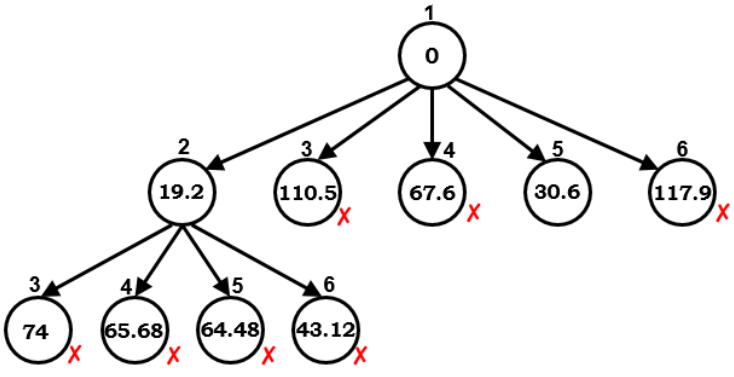
$$d_{12}+p_2 d_{21}+(1-p_2)\{d_{26}+p_6 d_{61}+(1-p_6)\{d_{63}+p_3 d_{31}+(1-p_3)\{d_{35}+p_5 d_{51}+(1-p_5)\{d_{54}+d_{41}\}\}\}$$

$$\cong d_{12}+p_2 d_{21}+(1-p_2)d_{26}=8+0.2*8+0.8*12=19.2$$

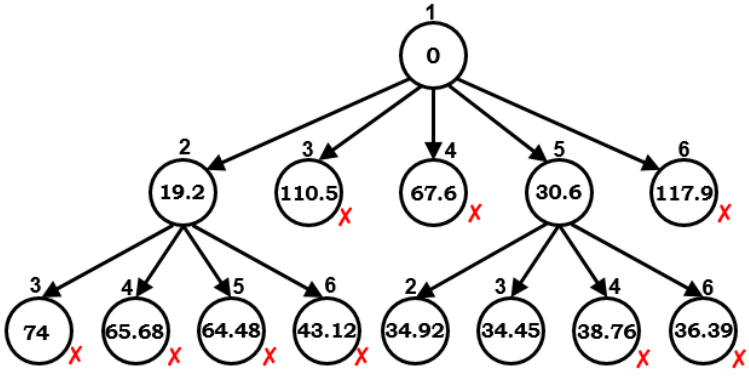
เนื่องจากเป็นปัญหาการหาค่าต่ำสุดและระยะระหว่างเมือง 2 ไปเมือง 6 มีค่าน้อยสุด จึงแทนระยะจากเมือง 2 ไปอีก 4 เมืองด้วย d_{26}



(ก)



(ข)



(ค)

รูปที่ 3 เส้นทางการเดินทางไปยังเมือง 6 เมือง ในแต่ละรอบการคำนวณ

(ก) ผลการคำนวณในรอบแรก เริ่มต้นจากเมือง 1

(ข) ผลการคำนวณในรอบสอง ออกจากเมือง 1 ไปเมือง 2 และเดินทางต่อไปยังเมืองอื่น

(ค) ผลการคำนวณในรอบสาม ออกจากเมือง 1 ไปเมือง 5 และเดินทางต่อไปยังเมืองอื่น

ดังนั้น จากเมือง 1 ไปเมือง 2 ได้เส้นทาง 1-2-X-X-X-X-1 ระยะทาง 19.2 หน่วย
จากเมือง 1 ไปเมือง 3 ได้เส้นทาง 1-3-X-X-X-X-1 ระยะทาง 110.5 หน่วย
จากเมือง 1 ไปเมือง 4 ได้เส้นทาง 1-4-X-X-X-X-1 ระยะทาง 67.6 หน่วย
จากเมือง 1 ไปเมือง 5 ได้เส้นทาง 1-5-X-X-X-X-1 ระยะทาง 30.6 หน่วย
จากเมือง 1 ไปเมือง 6 ได้เส้นทาง 1-6-X-X-X-X-1 ระยะทาง 117.9 หน่วย

ขั้นตอนที่ 2 (กำหนดค่าขอบเขต)

รูปที่ 3 ก แสดงเส้นทางการเดินทางจากเมืองตั้งต้น เส้นตรงเชื่อมระหว่างโหนดแสดงการเดินทางระหว่างเมือง ค่าเริ่มแรกสำหรับขอบเขตบน (Initial Upper Bound) กำหนดให้เท่ากับ 38 (ใช้วิธี Trial & Error ถ้าค่ามากเกินไประยะทางของทุกเส้นทางจะเกินขอบเขตบนทำให้ไม่สามารถหาคำตอบได้ ถ้าหากค่าน้อยไปจะใช้เวลาในการคำนวณนาน) ด้วยเหตุนี้การเดินทางที่เป็นไปได้มีเพียงสองเส้นทางคือ 1-2-X-X-X-X-1 และ 1-5-X-X-X-X-1 ระยะทางรวมเท่ากับ 19.2 หน่วย และ 30.6 หน่วย ตามลำดับ เส้นทางอื่นมีระยะทางเกินค่าขอบเขตบนจึงถูกตัดทิ้ง (แสดงเครื่องหมายกากบาทในรูปที่ 3)

ขั้นตอนที่ 3 (การเลือกโหนด)

เส้นทางที่ผ่านเมือง 2 เป็นลำดับสองของการเดินทางมีระยะทางน้อยกว่าเมือง 5 จึงเลือกโหนด 2

<u>รอบที่ 2</u> (ดูรูปที่ 3 ข)

ขั้นตอนที่ 1 (การแตกกิ่ง): เมือง 2 เป็นจุดหมายอันดับสองของการเดินทางจากนั้นเดินต่อไปยังเมืองอื่น จึงแตกกิ่งที่โหนด 2

พิจารณาการเดินทางจากเมือง 1 ผ่านเมือง 2 ไปเมือง 3 จนครบทุกเมือง มีการเดินทางเป็น 1-2-3-X-X-X-1 ระยะทางรวมเท่ากับ

$$d_{12}+p_2d_{21}+(1-p_2)\{d_{23}+p_3d_{31}+(1-p_3)\{d_{36}+p_6d_{61}+(1-p_6)\{d_{65}+p_5d_{51}+(1-p_5)\{d_{54}+d_{41}\}\}\}$$

$$\cong d_{12}+p_2d_{21}+(1-p_2)\{d_{23}+p_3d_{31}+(1-p_3)d_{36}\}=74$$

จากการเปรียบเทียบระยะทางระหว่างสองเมืองที่เริ่มต้นจากเมือง 3 พบว่าการเดินทางไปเมือง 6 ใกล้ที่สุด จึงแทนระยะทางจากเมือง 3 ไปอีก 3 เมืองด้วยค่า d_{36}

ดังนั้น เส้นทาง 1-2-3-X-X-X-1 ได้ระยะทางรวม 74 หน่วย เส้นทาง 1-2-5-X-X-X-1 ได้ระยะทางรวม 64.48 หน่วย
เส้นทาง 1-2-4-X-X-X-1 ได้ระยะทางรวม 65.68 หน่วย เส้นทาง 1-2-6-X-X-X-1 ได้ระยะทางรวม 43.12 หน่วย

ขั้นตอนที่ 2 (กำหนดค่าขอบเขต)

ผลลัพธ์ในขั้นตอน 1 รอบที่ 2 ทุกเส้นทางมีความยาวเกินค่าขอบเขตบน ดังนั้นไม่จำเป็นต้องพิจารณาเส้นทางเหล่านั้นอีกต่อไป

ขั้นตอนที่ 3 (การเลือกโหนด)

โหนดที่เหลือมีเพียงโหนดเดียวคือ โหนด 5

<u>รอบที่ 3</u> (ดูรูปที่ 3 ค)

ขั้นตอนที่ 1 (การแตกกิ่ง): เชื่อมต่อเมือง 5 กับเมืองอื่นๆ จึงแตกกิ่งที่โหนด 5

ออกเดินทางจากเมือง 5 มุ่งหน้าสู่เมือง 2 จนแวะครบทุกเมือง มีเส้นทางการเดินทาง 1-5-2-X-X-X-1 รวมระยะทางเท่ากับ

$$d_{15}+p_5d_{51}+(1-p_5)\{d_{52}+p_2d_{21}+(1-p_2)\{d_{26}+p_6d_{61}+(1-p_6)\{d_{63}+p_3d_{31}+(1-p_3)\{d_{34}+d_{41}\}\}\}$$

$$\cong d_{15}+p_5d_{51}+(1-p_5)\{d_{52}+p_2d_{21}+(1-p_2)d_{26}\}=34.92$$

แทนค่าระยะทางจากเมือง 2 ไปเมืองที่เหลือ 3 เมืองด้วย d_{26}

ดังนั้น เส้นทาง 1-5-2-X-X-X-1 ได้ระยะทางรวม 34.92 หน่วย เส้นทาง 1-5-4-X-X-X-1 ได้ระยะทางรวม 38.76 หน่วย
เส้นทาง 1-5-3-X-X-X-1 ได้ระยะทางรวม 34.45 หน่วย เส้นทาง 1-5-6-X-X-X-1 ได้ระยะทางรวม 36.39 หน่วย

ขั้นตอนที่ 2 (กำหนดค่าขอบเขต)

เส้นทางที่เป็นไปได้ 3 เส้นทาง ออกจากเมือง 1 ไปยังเมือง 5 และไปสู่เมืองจุดหมายลำดับสามคือเมือง 2, เมือง 3 และเมือง 6

ขั้นตอนที่ 3 (การเลือกโหนด)

เลือกโหนด 3 จากเซตของโหนดซึ่งประกอบด้วยโหนด 2, โหนด 3 และโหนด 6

หลังจากทำการคำนวณซ้ำ 8 รอบจะได้เส้นทางสั้นที่สุดเป็น 1-5-3-6-2-4-1 มีระยะรวม 37.98 หน่วย

4. ผลการวิจัย

ภาษา C++ ถูกนำมาใช้สร้างโปรแกรมคอมพิวเตอร์เนื่องจากเป็นภาษาที่มีโครงสร้างข้อมูลแบบลิงค์ลิสต์ (Linked List) ซึ่งประกอบด้วยสมาชิกหลายตัวเชื่อมต่อกัน สมาชิกแต่ละตัวมีองค์ประกอบ 2 ส่วนคือส่วนเก็บข้อมูลและส่วนพอยน์เตอร์ (Pointer) ที่ชี้ไปยังสมาชิกตัวอื่น พอยน์เตอร์หรือตัวชี้ช่วยให้การเข้าถึงสมาชิกในตำแหน่งที่ต้องการได้ง่ายทำให้สามารถเพิ่ม, ลบหรือจัดเรียงลำดับสมาชิกเป็นไปอย่างสะดวกรวดเร็ว งานวิจัยนี้เก็บเฉพาะข้อมูลของโหนดที่ให้ผลลัพธ์เป็นไปได้เท่านั้นด้วยการเพิ่มสมาชิกใหม่ลงในลิสต์และเรียงสมาชิกตามลำดับค่าระยะทางรวมจากน้อยไปมาก (Ascending Order) และทำการลบทิ้งสมาชิกที่เก็บข้อมูลของโหนดแม่ (Parent Node) ที่แบ่งย่อยเป็นโหนดลูกแล้วรวมถึงโหนดที่มีระยะรวมของเส้นทางมากกว่าขอบเขตบนใหม่ นอกจากนี้การจัดลำดับความสำคัญของปัญหาย่อยเพื่อให้รู้ว่าปัญหาอันใดควรทำก่อนทำหลังเป็นอีกวิธีการหนึ่งในการเพิ่มประสิทธิภาพการทำงานของโปรแกรมส่งผลให้พบคำตอบของปัญหาได้เร็วขึ้น ในที่นี้ใช้วิธีการโดดข้ามกิ่งไปแก้ปัญหาย่อยที่ให้ระยะทางต่ำสุดก่อน

ตารางที่ 2 เวลาที่ใช้ในการแก้ปัญหา PTSP-R ที่มีขนาดของเมืองแตกต่างกัน

ข้อมูลป้อนเข้า			**ผลลัพธ์**	
จำนวนเมือง	**ค่ามากสุดของระยะห่างระหว่างเมือง**	**ค่าเริ่มต้นของขอบเขตบน**	**จำนวนโหนดที่สร้างขึ้น**	**เวลาประมวลผล (วินาที)**
5	100	100	10	0*
10	100	100	109	0*
15	100	100	347	0*
100	100	30	80,415	15
400	100	20	392,774	556
800	100	20	834,470	1,679
1,300	100	20	1,972,784	8,247

จากข้อมูลนำเข้า (Input Data) ที่ประกอบด้วยจำนวนเมือง, ระยะห่างมากสุดระหว่างเมืองและค่าเริ่มต้นขอบเขตบนที่ระบุขึ้นเอง และการใช้เทคนิคการสุ่มทำให้ได้เมตริกซ์ระยะทาง (Distance Matrix) และเมตริกซ์ความน่าจะเป็นของการค้นพบ (Probability Matrix) สำหรับใช้ดำเนินการคำนวณในโปรแกรม ตารางที่ 2 แสดงผลที่ได้จากการแก้ปัญหาที่มีขนาดแตกต่างกัน พบว่าจำนวนเมือง

* โปรแกรม C++ ที่เขียนขึ้นใช้หน่วยวัดเวลาเป็นวินาที ค่าที่น้อยกว่า 1 วินาทีจึงแสดงผลเป็น 0

5, 10 และ15 เมืองใช้เวลาในการหาคำตอบน้อยกว่า 1 วินาที เมื่อปัญหามีขนาดใหญ่ขึ้นจะใช้เวลาในการคำนวณเพิ่มขึ้นอย่างเอ็กโพ-เนนเชียลและสามารถรองรับปัญหาขนาด 1,300 เมืองได้ในเวลาประมาณ 2.5 ชั่วโมง

5. สรุป

อัลกอริทึมและโปรแกรมคอมพิวเตอร์สำหรับปัญหาการเดินทางของพนักงานขายที่เดินทางไปพบลูกค้าด้วยค่าความน่าจะเป็นแล้วกลับมายังจุดเริ่มต้น ได้ถูกพัฒนาขึ้นด้วยเทคนิควิธีการแตกกิ่งและกำหนดขอบเขตเพื่อใช้หาผลเฉลยแม่นตรงโดยค้นหาผลเฉลยไปจนกว่าจะพบเส้นทางที่มีระยะสั้นที่สุด อย่างไรก็ตามเรื่องของประสิทธิภาพในการประมวลผลยังคงมีขีดจำกัด การแก้ปัญหาใหญ่ซับซ้อนใช้เวลายาวนานและบางครั้งอาจหาคำตอบไม่ได้ ด้วยเหตุนี้ข้อพิจารณาเพิ่มเติมในงานวิจัยนี้คือการเพิ่มความเร็วการทำงานของโปรแกรมจะทำให้สามารถเกิดการประยุกต์ใช้งานได้จริง และยังใช้ตรวจสอบความถูกต้องของวิธีการหาคำตอบใกล้เคียง การประมวลผลแบบขนาน (Parallel Computing) เป็นวิธีการหนึ่งช่วยให้งานขนาดใหญ่ทำงานเสร็จเร็วขึ้นโดยแบ่งงานออกเป็นส่วนย่อยที่เป็นอิสระต่อกันหรือไม่เกี่ยวข้องกันเพื่อนำไปประมวลผลพร้อมกันบนคอมพิวเตอร์ที่มีโปรเซสเซอร์ (Processor) หลายตัว

กิตติกรรมประกาศ

ผู้วิจัยขอขอบพระคุณ Professor Manbir Sodhi (Department of Mechanical, Industrial and Systems Engineering, University of Rhode island, USA) เป็นอย่างสูงที่กรุณาสละเวลาให้คำปรึกษาและความรู้ในศาสตร์แขนงการวิจัยดำเนินงาน

เอกสารอ้างอิง

[1] Cook, W. J. *In pursuit of the traveling salesman: mathematics at the limits of computation*. Princeton: Princeton University Press, 2012.

[2] Menger, K. Bericht iiber ein mathematisches Kolloquium. *Monatshefte fur Mathematik und Physik*, 1931, 38, pp. 17-38.

[3] Applegate, D. L., Bixby, R. E., Chvátal, V. and Cook, W. L. *The traveling salesman problem: a computational study*, Princeton: Princeton University Press, 2007.

[4] Dantzig, G., Fulkerson, R. and Johnson, S. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 1954, 2, pp. 393-410.

[5] Cook, W. L. *The Traveling Salesman Problem*, 2015. Available from http://www.math.uwaterloo.ca/tsp/concorde.html [Accessed 11 April 2018].

[6] Jaillet, P. *The probabilistic traveling salesman problems*. Technical Report 185, Operations Research Center, MIT, Cambridge MA, 1985.

[7] Jaillet, P. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, 1988, 36, pp. 929-936.

[8] Gudbrandsen, D. *Traveling salesman problem with returns (TSP-R) a new type of probabilistic TSP.* Thesis, University of Rhode Island, 2009.

[9] Haley, K.B. Applications of search theory. *European Journal of Operational Research*, 1981, 7, pp. 227-331.

[10] Laporte, G. The traveling salesman problem: an overview of exact and approximate algorithms. *European Journal of Operational Research*, 1992, 59, pp. 231-247.

[11] Winston, W. L. *Operations research: applications and algorithms*, 4th ed. Belmont: Duxbury Press, 2004.

[12] Srithongchai, S. Exact algorithms for the probabilistic traveling salesman problem with return. In: Proceedings IE Network 2016 Conference, Khon Kaen Thailand, 7-8 July 2016, pp. 51-57.