การเปรียบเทียบวิธีการสร้างแถวและวิธีการสร้างสดมภ์สำหรับปัญหากำหนดการเชิงเส้นขนาดใหญ่และการใช้หน่วยความจำรำรอง

ปวิธ ไกรสรนุเคราะห์[1*], พีรยุทธ์ ชาญเศรษฐิกุล[2], ธนพันธ์ คงทอง[3] และ พาพิศ วงศ์ชัยสุวัฒน์[4]

[1, 2, 4]สาขาวิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ (บางเขน) กรุงเทพ 10900

[3]สาขาวิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ (ศรีราชา) ชลบุรี 20230

Received: 31 January 2022; Revised: 12 April 2022; Accepted: 6 May 2022

บทคัดย่อ

งานวิจัยนี้มีวัตถุประสงค์เพื่อนำเสนอวิธีการสร้างแถวและสดมภ์ (Row-and-column generation) และทำการเปรียบเทียบวิธีการวิจัย 4 รูปแบบ คือ วิธีการตรงแบบเต็มรูป (Linear programming with full model, LPFM) วิธีการสลับการสร้างแถวและสดมภ์ (Switching row-and-column generation, SRCG) วิธีการสร้างสดมภ์ขึ้นกับแถว (Row-dependent-column generation, RDCG) และวิธีการสร้างแถวขึ้นกับสดมภ์ (Column-dependent-column generation, CDRG) โดยวัดประสิทธิภาพของแต่ละวิธีการด้วยคุณภาพของคำตอบ และระยะเวลาที่ใช้ในการประมวลผล จากการแก้ปัญหากำหนดการเชิงเส้นขนาดใหญ่ (Large-scale linear programming problem) พบว่าเมื่อขนาดปัญหาขนาดใหญ่ขึ้นถึงช่วงหนึ่ง จะทำให้วิธีการตรงแบบเต็มรูปไม่สามารถหาคำตอบได้เนื่องจากขีดจำกัดของทรัพยากรและทั้ง 4 วิธีการให้คุณภาพของคำตอบที่ใกล้เคียงกัน แต่จะมีความแตกต่างในด้านของระยะเวลาที่ใช้ในการประมวลผลซึ่งวิธีการสร้างแถวขึ้นกับสดมภ์ใช้ระยะเวลาในการประมวลผลโดยเฉลี่ย (Average processing time) ต่ำที่สุด และในบทขยายเพิ่มเติมได้แนะนำให้เก็บข้อมูลบางส่วนที่ไม่ได้ใช้ขณะนั้นจากหน่วยความจำหลัก (Primary storage) ไปยังหน่วยความจำสำรอง (Secondary storage) เพื่อเพิ่มพื้นที่ของหน่วยความจำหลักในการคำนวนปัญหา ซึ่งพบว่าสามารถแก้ปัญหาขนาดใหญ่ได้มากกว่าเดิม

คำสำคัญ: ปัญหากำหนดการเชิงเส้นขนาดใหญ่, การสร้างแถว, การสร้างสดมภ์, หน่วยความจำสำรอง

* Corresponding author: E-mail: pawit.krai@ku.th

[1]นิสิต หลักสูตรดุษฎีบัณฑิต ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ (บางเขน)

[2]รองศาสตราจารย์ ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ (บางเขน)

[3]อาจารย์ ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ (ศรีราชา)

[4]ผู้ช่วยศาสตราจารย์ ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ (บางเขน)

Comparison Simultaneous Row-and-Column Generation Approach for Large-Scale Linear Programming Problem and secondary memory utilization

Pawit Kraisornnukhor[1*], Peerayuth Charnsethikul [2], Thanapan Kongtong [3] and Papis Wongchaisuwat [4]

[1, 2, 4]Industrial Engineering Department of Industrial Engineering, Faculty of Engineering, Kasetsart University (Bangkaeng), Bangkok 10900

[3] Industrial Engineering Department of Industrial Engineering, Faculty of Engineering, Kasetsart University (sriracha), Bangkok 10900

Received: 31 January 2022; Revised: 12 April 2022; Accepted: 6 May 2022

Abstract

This research studied the methods to solve linear programming by simultaneous row-and-column generation approach and compared efficiency among 4 methods: Linear Programming with Full Model (LPFM), Switching Row-and-Column Generation (SRCG), Row-Dependent-Column Generation (RDCG) and Column-Dependent-Column Generation (CDRG). The study compares optimal objective value and processing time when solving large-scale linear programming problem. The result showed that some of large-scale linear programming problems cannot solve using LPFM directly due to out-of-core memory. The solutions quality solved by each method is the same. However, there are differences in processing time in which RDCG provided optimal solutions with the least average processing time. And in the last chapter recommended to store some unused data from the primary storage to the secondary storage to free up the space of the main memory. Which was found to be able to solve larger problems.

Keywords: Large-scale linear programming, Row generation, Column generation, Secondary storage

* Corresponding author: E-mail: pawit.krai@ku.th

[1]Doctoral Student in Department of Industrial Engineering, Faculty of Engineering, Kasetsart University (Bangkaeng)

[2]Associate Professor in Department of Industrial Engineering, Faculty of Engineering, Kasetsart University (Bangkaeng)

[3]Lecturer in Department of Industrial Engineering, Faculty of Engineering, Kasetsart University (sriracha)

[4]Assistant Professor in Department of Industrial Engineering, Faculty of Engineering, Kasetsart University (Bangkaeng)

1. คำนำ

เป็นที่รู้กันว่าปัจจุบันได้เข้ายุคที่มีข้อมูลมหาศาล (Big Data) ซึ่งหลายองค์กรต้องการที่จะนำข้อมูลดังกล่าวมาทำงานวิเคราะห์หาค่าที่เหมาะสมที่สุด (Optimization) เพื่อที่จะเพิ่มศักยภาพในการแข่งขันให้กับตนเอง ซึ่งการนำข้อมูลมหาศาลมาวิเคราะห์ จะส่งผลให้มีจำนวนตัวแปรและเงื่อนไขจำนวนมาก ดังนั้นจึงจำเป็นที่จะต้องมีเครื่องมือเพื่อจะช่วยหาคำตอบที่ดีที่สุด (Optimal solution) ในการเพิ่มประสิทธิ-ภาพในการหาคำตอบ

จากที่กล่าวมาเป็นเหตุให้เกิดการค้นคว้าวิจัยหลายรูปแบบที่จะช่วยเพิ่มประสิทธิภาพในการหาคำตอบ อย่างเช่น วิธีการฮวลิสติก (Heuristic Algorithm) [7 - 8] วิธีเมตาฮิวลิตติก (Meta-heuristic Algorithm) [9 - 10] และวิธีการแบ่งย่อยปัญหา (Decomposition algorithm) [1 - 6] สำหรับการแก้ไขปัญหาดังกล่าว แต่สำหรับงานวิจัยนี้จะมุ่งเน้นไปที่ปัญหากำหนดการเชิงเส้นขนาดใหญ่ (Large-scale linear programming, LSLP)

เป็นที่รู้กันว่าการแก้ปัญหา LSLP โดยวิธีการตรงแบบเต็มรูป (Full model of linear programming, FMLP) นั้นจะใช้ระยะเวลาและทรัพยากรเป็นอย่างมากในการประมวลผล อย่างงานวิจัยของ [4] และ [6] ได้แสดงให้เห็นว่าวิธีการแก้ปัญหากำหนดการเชิงเส้น (Linear Programming, LP) ด้วยวิธีการตรงแบบเต็มรูปแบบใช้ระยะเวลาที่นานกว่าวิธีเฉพาะทางอื่นที่ถูกนำมาปรับใช้

โดยงานของ [4] ได้ใช้วิธีการสร้างแถวสำหรับการแก้ปัญหาการจัดสรรเงินลงทุนเชิงจัดหมู่ ในกรณีที่มีจำนวนเงื่อนไข (Constraint) มากกว่าจำนวนตัวแปรตัดสินใจ (Decision Variable) มากๆ และงานของ [6] ได้ใช้วิธีการสร้างสดมภ์สำหรับการแก้ปัญหาการจัดสรรเงินลงทุนเชิงจัดหมู่ ในกรณีที่มีจำนวนตัวแปรตัดสินใจมากกว่าจำนวนเงื่อนไขมากๆ

สำหรับงานวิจัยนี้จะศึกษาปัญหากรณีที่มีจำนวนเงื่อนไขและตัวแปรตัดสินใจมากเกินขีดจำกัดของหน่วยความจำหลัก การศึกษาในงานวิจัยชิ้นนี้จึงเป็นการหาวิธีผสมผสานที่เหมาะสม จากการนำวิธีการสร้างสดมภ์และสร้างแถวมาหากลยุทธ์ในการประยุกต์ใช้แก้ปัญหาพร้อมกันเป็น 3 วิธีการ ภายใต้คุณลักษณะปลีกย่อยลงไปอีก เช่น กรณีที่จำนวนตัวแปรมากกว่าจำนวนข้อจำกัด หรือจำนวนข้อจำกัดมากกว่าตัวแปร หรือทั้งสองค่ามีปริมาณไม่แตกต่างกันมาก ซึ่งจะเปรียบเทียบกับการการแก้ปัญหาตรงแบบเต็มรูป (Linear programming with full model) ในเชิงของคุณภาพคำตอบ (Fitness value) และระยะเวลาที่ใช้ในการประมวลผลโดยเฉลี่ย (Average processing time)

2. ขั้นตอนวิธีการวิจัย

งานวิจัยนี้นำเสนอวิธีการแก้ปัญหากำหนดการเชิงเส้นขนาดใหญ่ ด้วยวิธีการ 4 รูปแบบ คือ วิธีการตรงแบบเต็มรูป (Linear Programming with Full Model, LPFM) วิธีการสลับการสร้างแถวและสดมภ์ (Switching row-and-column generation, SRCG) วิธีการสร้างสดมภ์ขึ้นกับแถว (Row-dependent-column generation, RDCG) และวิธีการสร้างแถวขึ้นกับสดมภ์ (Column-dependent-column generation, CDRG) และวัดประสิทธิภาพของแต่ละวิธีการด้วยคุณภาพของคำตอบ และระยะเวลาที่ใช้ในการประมวลผล

2.1 แบบจำลองทางคณิตศาสตร์แบบเต็มรูป (Mathematical Model: Full Model)

งานวิจัยนี้ได้กำหนดการสร้างแบบจำลองทางคณิตศาสตร์ขนาด MxN ซึ่งแสดงดังสมการดังต่อไปนี้

$$Minimize\ Z = \sum_{j=1}^{N} -c_j x_j \quad (1)$$

$$\sum_{j=1}^{N} a_{ij} x_j \le b_i \quad ; i = 1,2,\dots,M \quad (2)$$

$$x_j \ge 0 \quad ; j = 1,2,\dots,N \quad (3)$$

โดยที่ Z คือ ค่าเป้าหมายของแบบจำลองแบบเต็มรูป

c_j คือ สัมประสิทธิ์ที่ j ของสมการเป้าหมาย $; \forall_j$

x_j คือ ตัวแปรตัดสินใจที่ j $; \forall_j$

a_{ij} คือ สัมประสิทธิ์ของเงื่อนไขที่ i สำหรับตัวแปรที่ j $; \forall_{ij}$

b_i คือ ค่าจำกัดของเงื่อนไขที่ i $; \forall_i$

จากสมการที่ (1) – (3) เป็นแบบจำลองทางคณิตศาสร์แบบเต็มรูป (Full Model) ขนาด MXN โดยสมการที่ (1) แสดงสมการเป้าหมาย (Objective function) ของแบบจำลองทางคณิตศาสตร์ ซึ่งมีวัตถุประสงค์เพื่อหาคำตอบให้มีค่าน้อยที่สุด อสมการที่ (2) แสดงข้อจำกัด (Constraint) ของแบบจำลอง และอสมการที่ (3) เพื่อกำหนดค่าตัวแปรตัดสินใจต้องไม่น้อยกว่า 0

2.2 แบบจำลองของปัญหาย่อย (Mathematical Model: Sub Problem)

จากแบบจำลองทางคณิตศาสตร์แบบเต็มรูปที่มีขนาด MXN จะทำการย่อยปัญหาเป็นขนาด mxn เพื่อให้ง่ายต่อการแก้ไขปัญหา กล่าวคือปัญหาย่อยเป็นสับเซต (Subset) ของปัญหาแบบเต็มรูป

$$Minimize\ Z' = \sum_{k=1}^{n} -c_k x_k \quad (4)$$

$$\sum_{k=1}^{n} a_{rk} x_k \leq b_r \quad ;r = 1,2,\dots,m \quad (5)$$

$$x_k \geq 0 \quad ;\forall_k \quad (6)$$

โดยที่ Z' คือ ค่าเป้าหมายของแบบจำลองปัญหาย่อย

c_k คือ สัมประสิทธิ์ที่ k ของสมการเป้าหมาย $;k = 1,\dots,n$

x_k คือ ตัวแปรตัดสินใจที่ $k\ ;k = 1,\dots,n$

a_{rk} คือ สัมประสิทธิ์ของเงื่อนไขที่ r สำหรับตัวแปรที่ $k\ ;\forall_{rk}$

b_r คือ ค่าจำกัดของเงื่อนไขที่ $r\ ;\forall_r$

จากสมการที่ (4) – (6) เป็นลักษณะของปัญหาย่อยขนาด mXn โดยสมการที่ (4) แสดงสมการเป้าหมายของแบบจำลองทางคณิตศาสตร์ของปัญหาย่อย ซึ่งมีวัตถุประสงค์เพื่อหาคำตอบให้มีค่าน้อยที่สุด อสมการที่ (5) แสดงข้อจำกัดของแบบจำลองของปัญหาย่อย และอสมการที่ (6) เพื่อกำหนดค่าตัวแปรตัดสินใจต้องไม่น้อยกว่า 0

2.3 แบบจำลองของปัญหาคู่ควบ (Mathematical Model: Duality Problem)

เมื่อพิจารณาแบบจำลองทางคณิตศาสตร์แบบดั้งเดิม (Primal Problem) ของปัญหาย่อยนำมาแปลงเป็นปัญหาคู่ควบ (Dual problem) ได้ดังสมการต่อไปนี้

$$Minimize\ W = \sum_{i=1}^{m} b_i p_i \quad (7)$$

$$\sum_{i=1}^{m} a_{ij} p_i \geq c_j \quad ;j = 1,2,\dots,n \quad (8)$$

$$p_i \geq 0 \quad ;i = 1,2,\dots,m \quad (9)$$

โดยที่ W คือ ค่าเป้าหมายของแบบจำลองของปัญหาคู่ควบ

สมการที่ (7) แสดงสมการเป้าหมายของแบบจำลองทางคณิตศาสตร์ของปัญหาคู่ควบ ซึ่งเป็นสัมประสิทธิ์ของสมการเป้าหมายคือค่าด้านขวา (Right hand side, RHS) ของอสมการข้อจำกัดของปัญหาดั้งเดิม (5) และตัวแปรตัดสินใจของปัญหาคู่ควบคือ ค่าเงา (shadow price, p_i) ของแบบจำลองแบบเต็มรูป ต่อมาอสมการที่ (8) แสดงสมการข้อจำกัดของปัญหาคู่ควบ โดยค่าทางขวา (RHS) ของปัญหาคู่ควมคือสัมประสิทธิ์ของสมการที่ (1) รวมถึงเครื่องหมายของอสมการจะสอดคล้องกับอสมการที่ (6) และอสมการที่ (9) เพื่อกำหนดค่าตัวแปรตัดสินใจต้องไม่น้อยกว่า 0 ซึ่งมีความสอดคล้องกับเครื่องหมายอสมการที่ (5)

2.4 หลักการแก้ปัญหาด้วยการสร้างสดมภ์ (Principle of column generation approach)

วิธีการสร้างสดมภ์เริ่มจากการนำปัญหาขนาดย่อยอย่างสมการที่ (4) – (6) มาทำการแก้ปัญหา ซึ่งผลลัพธ์ที่ได้ (Output) จะได้ค่าเป้าหมาย (Fitness value) ค่าตัวแปรตัดสินใจ และค่าเงาหรือค่าตัวแปรตัดสินใจของปัญหาคู่ควบ เนื่องจากการสร้างสดมภ์เป็นการเพิ่มตัวแปรนำเข้าจากการใช้หลักของทฤษฎีคู่ควบจากสมการที่ (7) – (8) โดยจะตรวจสอบค่าการผิดเงื่อนของอสมการที่ (8) ซึ่งเขียนเป็นสมการดังนี้

$$\varepsilon_{column} = Min\{c_j - \sum_{i=1}^{m} a_{ij} p_i\}\ ;j = 1,\dots,N \quad (10)$$

โดยที่ ε_{column} คือ ค่าที่น้อยที่สุดของผลต่างระหว่างค่าทางขวามือและค่าทางซ้ายมือของอสมการเงื่อนไขของปัญหาคู่ควบ

จากสมการที่ (10) หากตัวแปรตัดสินใจใดที่ค่าผิดเงื่อนไขมากที่สุด จะถูกพิจารณาให้นำเข้าปัญหาย่อย โดยการสร้างสดมภ์ เพื่อเพิ่มตัวแปรเข้าไปในปัญหาย่อย โดยวัตถุประสงค์ของการสร้างสดมภ์คือการเลือกตัวแปรตัดสินใจที่จำเป็นจากแบบจำลองแบบเต็มรูปเข้าสู่ปัญหาย่อย

2.4.1 การเปรียบเทียบประสิทธิภาพวิธีการสร้างสดมภ์กับวิธีการตรงแบบเต็มรูป

เพื่อให้เห็นถึงประสิทธิภาพสำหรับการแก้ปัญหาที่มีจำนวนตัวแปรมากๆ ด้วยวิธีการสร้างสดมภ์ สำหรับโจทย์ปัญหาทั่วไปที่ใช้ในงานวิจัยนี้ ซึ่งได้ออกแบบการทดลองให้มีขนาดปัญหาแตกต่างกัน 8 ขนาด ซึ่งลองทดสอบ 1 รอบการทดลอง “N/A” หมายถึงไม่สามารถคำนวณได้ เนื่องจากหน่วยความจำหลักไม่เพียงพอ (Out-of-core memory) ดังตารางที่ 1 ซึ่งพบว่าวิธีการสร้างสดมภ์สามารถให้คุณภาพของคำตอบได้ใกล้เคียงกับวิธีการ LPFM อีกทั้งยังใช้เวลาน้อยกว่าทุกขนาดการทดลอง ดังนั้น เพื่อให้เกิดความชัดเจนในเรื่องประสิทธิภาพการหาคำตอบ จึงขยายผลการทดลองโดยทำการทดลองซ้ำเป็น 5 รอบการทดลอง ซึ่งได้ผลการทดลองดังตารางที่ 2 ซึ่งพบว่าระยะเวลาที่ใช้ในการประมวลผลโดยเฉลี่ยของวิธิการสร้างสดมภ์น้อยกว่าวิธีการ LPFM ในทุกกรณี อีกทั้งยังสามารถหาผลคำตอบในปัญหาที่มีจำนวนตัวแปรมากได้ ดังนั้นปัญหาที่ใช้ในงานวิจัยนี้สามารถนำวิธีการสร้างสดมภ์ไปประยุกต์ใช้เพิ่มประสิทธิภาพในการหาคำตอบได้

ตารางที่ 1 ผลลัพธ์ของประสิทธิภาพของคำตอบใน 1 รอบการทดลองของวิธีการสร้างสดมภ์เทียบกับวิธีการตรงแบบเต็มรูป

			วิธีการตรง		วิธีการสร้างสดมภ์	
		จำนวนตัว	ระยะเวลา		ระยะเวลา	
ลำดับที่	จำนวนเงื่อนไข	แปร	ค่าเป้าหมาย	ประมวลผล [วินาที]	ค่าเป้าหมาย	ประมวลผล [วินาที]
1	10	50000	-289.16587	10.54843	-289.16587	0.32134
2	10	100000	-294.61277	39.72316	-294.61277	0.22348
5	50	50000	-256.40719	18.89889	-256.40719	0.58067
3	10	500000	-324.23177	1153.27205	-324.23177	0.33896
6	50	100000	-261.19895	88.43738	-261.19895	0.74498
4	10	1000000	-317.0094	4449.32139	-317.0094	0.61156
7	50	500000	-262.65485	2636.20556	-262.65485	1.77896
8	50	1000000	-262.42872	9225.70931	-262.42872	2.55826
9	50	8000000	N/A	N/A	-269.5359	154.70088
10	80	5000000	N/A	N/A	-260.82205	34.38280
11	100	5000000	N/A	N/A	-259.07318	509.13599

ตารางที่ 2 ผลลัพธ์ของระยะเวลาที่ใช้ในการประมวลผลโดยเฉลี่ยของวิธีการสร้างสดมภ์เทียบกับวิธีการตรงแบบเต็มรูป

			วิธีการตรง	วิธีการสร้างสดมภ์
		จำนวนตัว	ระยะเวลาประมวลผล	ระยะเวลาประมวลผล
ลำดับที่	จำนวนเงื่อนไข	แปร	โดยเฉลี่ย [วินาที]	โดยเฉลี่ย [วินาที]
1	10	50000	7.89064	0.19644
2	10	100000	38.82950	0.19279
5	50	50000	19.61540	0.53788
3	10	500000	1098.75447	0.30851
6	50	100000	93.20430	0.68521
4	10	1000000	4539.17195	0.46050
7	50	500000	2361.35339	1.78414
8	50	1000000	9862.63822	2.60890
9	50	8000000	N/A	184.22584
10	80	5000000	N/A	31.43372
11	100	5000000	N/A	236.06913

2.5 หลักการแก้ปัญหาด้วยการสร้างแถว (Principle of row generation approach)

วิธีการสร้างแถวเริ่มจากการนำปัญหาขนาดย่อยอย่างสมการที่ (4) – (6) มาทำการแก้ปัญหา ซึ่งผลลัพธ์ที่ได้ จะได้ค่าเป้าหมาย ค่าตัวแปรตัดสินใจ และค่าเงาหรือค่าตัวแปรตัดสินใจของปัญหาคู่ควบ เนื่องจากการสร้างแถวเป็นการเพิ่มเงื่อนไข โดยจะตรวจสอบค่าการผิดเงื่อนของอสมการที่ (2) ซึ่งเขียนเป็นสมการดังนี้

$$\varepsilon_{row} = Min\{b_i - \sum_{j=1}^{n} a_{ij}x_j\}; i = 1, \dots, M \quad (11)$$

โดยที่ ε_{row} คือ ค่าที่น้อยที่สุดของผลต่างระหว่างค่าทางขวามือและค่าทางซ้ายมือของอสมการเงื่อนไขของปัญหาหลัก

จากสมการที่ (11) หากเงื่อนไขใดที่ค่าผิดเงื่อนไขมากที่สุด จะถูกพิจารณาให้นำเข้าปัญหาย่อย โดยการสร้างแถว เพื่อเพิ่มเงื่อนไขเข้าไปในปัญหาย่อย โดยวัตถุประสงค์ของการสร้างแถวคือการเลือกเงื่อนไขที่จำเป็นจากแบบจำลองแบบเต็มรูปเข้าสู่ปัญหาย่อย

2.5.1 เปรียบเทียบประสิทธิภาพวิธีการสร้างแถวกับวิธีการตรงแบบเต็มรูป

เพื่อให้เห็นถึงประสิทธิภาพสำหรับการแก้ปัญหาที่มีจำนวนตัวเงื่อนไขมากๆ ด้วยวิธีการสร้างแถว สำหรับโจทย์ปัญหาทั่วไปที่ใช้ในงานวิจัยนี้ ซึ่งได้ออกแบบการทดลองใช้มีขนาดปัญหาแตกต่างกัน 8 ขนาด ซึ่งลองทดสอบ 1 รอบการทดลอง ดังตารางที่ 3 ซึ่งพบว่าวิธีการสร้างแถวสามารถให้คุณภาพของคำตอบได้ใกล้เคียงกับวิธีการ LPFM อีกทั้งยังใช้เวลาน้อยกว่าทุกขนาดการทดลอง ดังนั้น เพื่อให้เกิดความชัดเจนในเรื่องประสิทธิภาพการหาคำตอบ จึงขยายผลการทดลองโดยทำการทดลองซ้ำเป็น 5 รอบการทดลอง ซึ่งได้ผลการทดลองดังตารางที่ 4 ซึ่งพบว่าระยะเวลาที่ใช้ในการประมวลผลโดยเฉลี่ยของวิธีการสร้างแถวน้อยกว่าวิธีการ LPFM ในทุกกรณี อีกทั้งยังสามารถหาผลคำตอบในปัญหาที่มีจำนวนเงื่อนไขมากได้ ดังนั้นปัญหาที่ใช้ในงานวิจัยนี้สามารถนำวิธีการสร้างแถวไปประยุกต์ใช้เพิ่มประสิทธิภาพในการหาคำตอบได้

ตารางที่ 3 ผลลัพธ์ของประสิทธิภาพของคำตอบใน 1 รอบการทดลองของวิธีการสร้างแถวเทียบกับวิธีการตรงแบบเต็มรูป

			วิธีการตรง		วิธีการสร้างแถว	
		จำนวนตัว	ระยะเวลาประมวลผล		ระยะเวลาประมวลผล	
ลำดับที่	จำนวนเงื่อนไข	แปร	ค่าเป้าหมาย	[วินาที]	ค่าเป้าหมาย	[วินาที]
1	50000	10	-161.98822	12.84708	-161.98822	0.09309
2	100000	10	-142.28526	52.28872	-142.28526	0.14483
5	50000	50	-176.4006	46.38095	-176.4006	0.25681
3	500000	10	-159.69959	1346.62239	-159.69959	0.12161
6	100000	50	-172.98185	245.68112	-172.98185	0.27790
4	1000000	10	-155.73181	5327.63624	-155.73181	0.17431
7	500000	50	-173.25679	6736.15155	-173.25679	0.76500
8	1000000	50	-173.47545	27006.47334	-173.47545	1.44540
9	8000000	50	N/A	N/A	-166.13673	42.21860
10	5000000	80	N/A	N/A	-174.29073	27.31059
11	5000000	100	N/A	N/A	-177.87075	64.98894

ตารางที่ 4 ผลลัพธ์ของระยะเวลาที่ใช้ในการประมวลผลโดยเฉลี่ยของวิธีการสร้างแถวเทียบกับวิธีการตรงแบบเต็มรูป

			วิธีการตรง	วิธีการสร้างแถว
		จำนวนตัว	ระยะเวลาประมวลผลโดย	ระยะเวลาประมวลผลโดย
ลำดับที่	จำนวนเงื่อนไข	แปร	เฉลี่ย [วินาที]	เฉลี่ย [วินาที]
1	50000	10	13.27460	0.05697
2	100000	10	53.66912	0.10775
5	50000	50	52.88390	0.33979
3	500000	10	1367.68030	0.11362
6	100000	50	245.70466	0.30596
4	1000000	10	5476.77261	0.21058
7	500000	50	6642.59914	0.83997
8	1000000	50	27065.45797	1.46140
9	8000000	50	N/A	39.73759
10	5000000	80	N/A	23.03271
11	5000000	100	N/A	73.91578

3 ขั้นตอนวิธีการสร้างแถวและวิธีการสร้างสดมภ์ (Simultaneous Row-and-Column Generation Approach)

ขั้นตอนวิธีการผสมขั้นตอนวิธีการการสร้างแถวและการสร้างสดมภ์ที่งานวิจัยนี้ จะมีขั้นตอนพื้นฐานที่เหมือนกันดังนี้

ขั้นตอนที่ 1: การสร้างแบบจำลองทางคณิตศาสตร์ขนาดย่อยขนาด mxn จากแบบจำลองแบบเต็มรูปขนาด MxN

ขั้นตอนที่ 2: ตามหลักการสร้างแถวและสร้างสดมภ์เพื่อให้สะดวกต่อการเพิ่มสัมประสิทธิ์เข้าแบบจำลองของปัญหาย่อยได้ถูกต้องตามตัวแปรและเงื่อนไขที่ถูกเลือกจากปัญหาใหญ่ จึงทำการสร้างสัมประสิทธิ์ย่อยแบบจำลองทางคณิตศาสตร์สำหรับวิธีการสร้างสดมภ์ (a_column) ที่จะเก็บสัมประสิทธิ์ของทุกตัวแปรตัดสินใจของแบบจำลองแบบเต็มรูปตามเงื่อนไขที่มีของแบบจำลองย่อย ซึ่งมีขนาด mxN และสำหรับวิธีการสร้างแถว (a_row) ที่จะเก็บสัมประสิทธิ์ของทุกเงื่อนไขของแบบจำลองแบบเต็มรูปตามตัวแปรตัดสินใจที่มีของแบบจำลองย่อย ซึ่งมีขนาด Mxn

ขั้นตอนที่ 3 การตัดสินใจขั้นตอนการเลือกเข้าใช้วิธีการสร้างแถวหรือวิธีการสร้างสดมภ์ ซึ่งทั้ง 3 วิธีการจะมีความแตกต่างกันในส่วนนี้ โดยมีขั้นตอนต่อไปนี้

3.1 วิธีการสลับการสร้างแถวและสดมภ์

ขั้นตอนที่ 1: หาคำตอบที่ดีที่สุดสำหรับปัญหาย่อย ซึ่งจะได้ผลลัพธ์เป็นค่าเป้าหมาย ค่าตัวแปรตัดสินใจ และค่าเงาหรือค่าตัวแปรตัดสินใจของปัญหาคู่ควบ

ขั้นตอนที่ 2: นำผลลัพธ์ที่ได้ไปคำนวนหา ε_{column} และ ε_{row}

ขั้นตอนที่ 3: เปรียบเทียบ ε_{column} และ ε_{row} หาก ε_{column} มีค่าน้อยกว่า ε_{row} ให้เข้าวิธีการสร้างสดมภ์ แต่หากตรงกันข้าม ให้เข้าวิธีการสร้างแถว

ขั้นตอนที่ 4: ทำขั้นตอนที่ 1 ถึงขั้นตอนที่ 3 วนซ้ำ จนกระทั้ง ε_{column} และ ε_{row} มีค่ามากกว่า 0

ขั้นตอนวิธีการสลับการสร้างแถวและสดมภ์ถูกแสดงดังรูปที่ 1

3.2 วิธีการสดมภ์ขึ้นกับแถว

ขั้นตอนที่ 1: หาคำตอบที่ดีที่สุดสำหรับปัญหาย่อย ซึ่งจะได้ผลลัพธ์เป็นค่าเป้าหมาย ค่าตัวแปรตัดสินใจ และค่าเงาหรือค่าตัวแปรตัดสินใจของปัญหาคู่ควบ

ขั้นตอนที่ 2: นำผลลัพธ์ที่ได้ไปคำนวนหา ε_{column} และทำการสร้างสดมภ์

ขั้นตอนที่ 3: ทำขั้นตอนที่ 1 ถึงขั้นตอนที่ 2 วนซ้ำ จนกระทั้ง ε_{column} มีค่ามากกว่า 0

ขั้นตอนที่ 4: นำผลลัพธ์ล่าสุดที่ได้ไปคำนวนหา ε_{row} และสร้างแถว

ขั้นตอนที่ 5: ทำขั้นตอนที่ 1 ถึงขั้นตอนที่ 4 วนซ้ำจนกระทั้ง ε_{row} มีค่ามากกว่า 0

วิธีการสร้างสดมภ์ขึ้นกับแถวถูกแสดงดังรูปที่ 2

3.2 วิธีการแถวขึ้นกับสดมภ์

ขั้นตอนที่ 1: หาคำตอบที่ดีที่สุดสำหรับปัญหาย่อย ซึ่งจะได้ผลลัพธ์เป็นค่าเป้าหมาย ค่าตัวแปรตัดสินใจ และค่าเงาหรือค่าตัวแปรตัดสินใจของปัญหาคู่ควบ

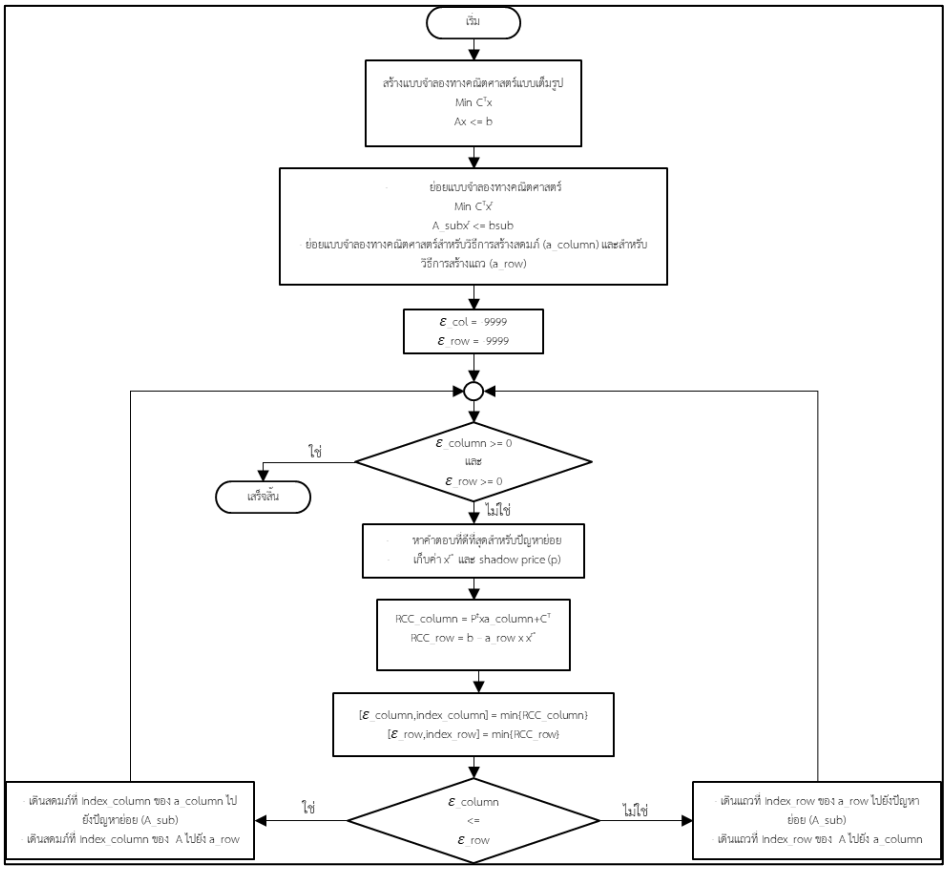
ขั้นตอนที่ 2: นำผลลัพธ์ที่ได้ไปคำนวนหา ε_{row} และทำการสร้างแถว

ขั้นตอนที่ 3: ทำขั้นตอนที่ 1 ถึงขั้นตอนที่ 2 วนซ้ำ จนกระทั้ง ε_{row} มีค่ามากกว่า 0

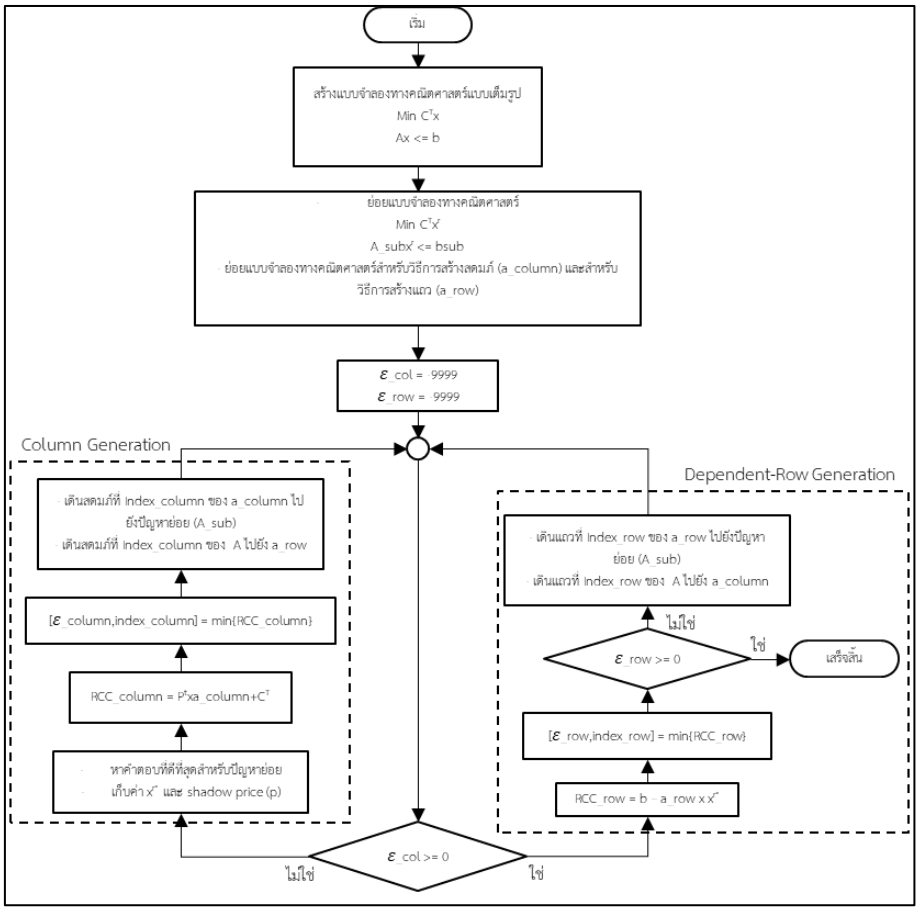
ขั้นตอนที่ 4: นำผลลัพธ์ล่าสุดที่ได้ไปคำนวนหา ε_{column} และทำการสร้างสดมภ์

ขั้นตอนที่ 5: ทำขั้นตอนที่ 1 ถึงขั้นตอนที่ 4 วนซ้ำจนกระทั้ง ε_{column} มีค่ามากกว่า 0

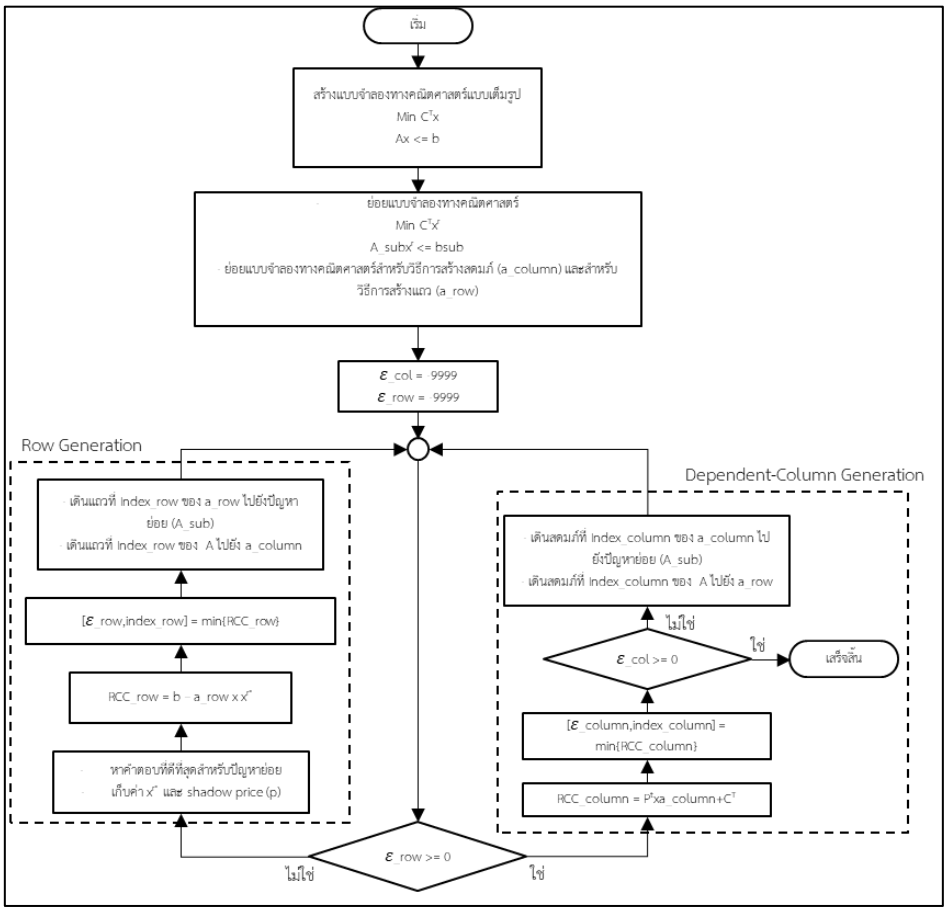
วิธีการสร้างแถวขึ้นกับสดมภ์ถูกแสดงดังรูปที่ 3



รูปที่ 1 แผนผังขั้นตอนวิธีการสลับการสร้างแถวและ



รูปที่ 2 แผนผังขั้นตอนวิธีการสร้างแถวขึ้นกับสดมภ์



รูปที่ 3 แผนผังขั้นตอนวิธีการสร้างสดมภ์ขึ้นกับแถว

4 การออกแบบการทดลองโจทย์ปัญหา

เพื่อการวัดผลการทดลอง งานวิจัยนี้ได้สร้างโจทย์ปัญหา ที่เป็นแบบสุ่มค่าและหนาแน่นไปด้วยตัวเลขที่ไม่ใช่ศูนย์ในเมตริกซ์ (Dense matrix) ข้อจำกัด สำหรับการทดลองหลายขนาดของจำนวนเงื่อนไข M และจำนวนตัวแปร N ให้แต่ละส่วนมีขนาดตั้งแต่ 5,000 ถึง 20,000 และกำหนดการสร้างโจทย์ปัญหาดังนี้

$c_j \sim \text{Uni}[0.1,1.1]$; $\forall_j$ คือมีการแจกแจงแบบยูนิฟอร์ม (Uniform Distribution) โดยกำหนดค่าไม่ต่ำว่า 0.1 และไม่เกิน 1.1

$a_{ij} \sim \text{Uni}[1,2]$; $\forall_{ij}$คือมีการแจกแจงแบบยูนิฟอร์ม (Uniform Distribution) โดยกำหนดค่าไม่ต่ำว่า 1 และไม่เกิน 2

$b_i \sim \text{Uni}[300,401]$; $\forall_i$คือมีการแจกแจงแบบยูนิฟอร์ม (Uniform Distribution) โดยกำหนดค่าไม่ต่ำว่า 300 และไม่เกิน 401

ซอฟต์แวร์ (Software)

วิธีการแก้ปัญหากำหนดการเชิงเส้น (linprog) บน MATLAB R2021b บนระบบปฏิบัติการไมโครซอฟท์วินโดว์ (Windows 10 PRO)

อุปกรณ์

- Intel i5-4690 3.5 GHZ RAM 8 GB

5 ผลการวิจัย

ทำการทดลองวัดประสิทธิภาพของทั้ง 4 วิธีการ โดยมีการกำหนดขนาดของปัญหา 14 แบบ เพื่อเปรียบเทียบประสิทธิภาพของคำตอบ ซึ่งพบว่าประสิทธิภาพของคำตอบของทั้ง 4 วิธีการให้ค่าที่ใกล้เคียงกัน ซึ่ง “N/A” หมายถึงไม่สามารถคำนวนได้ เนื่องจากหน่วยความจำหลักไม่เพียงพอ (Out-of-core memory) ดังตารางที่ 5

สืบเนื่องจากผลลัพธ์ของค่าเป้าหมายไม่มีความแตกต่างกันมากนักของทั้ง 4 วิธีการ จึงทำการวัด

ประสิทธิภาพด้วยระยะเวลาที่ใช้ในการประมวลผลโดยเฉลี่ย (Average processing time) โดยการทดลองซ้ำในแต่ละปัญหาอีก 5 รอบ ซึ่งในแต่ละรอบจะมีการสุ่มตัวเลขสำหรับการสร้างโจทย์ปัญหาที่แตกต่างกัน ดังตารางที่ 5

ตารางที่ 5 ผลลัพธ์ของประสิทธิภาพของคำตอบใน 1 รอบการทดลอง

			วิธีการตรง		วิธีการสลับการสร้างแถวและสดมภ์		วิธีการสร้างสดมภ์ขึ้นกับแถว		วิธีการสร้างแถวขึ้นกับสดมภ์	
ลำดับที่	จำนวนเงื่อนไข	จำนวนตัวแปร	ค่าเป้าหมาย	ระยะเวลาประมวลผล [วินาที]	ค่าเป้าหมาย	ระยะเวลาประมวลผล [วินาที]	ค่าเป้าหมาย	ระยะเวลาประมวลผล [วินาที]	ค่าเป้าหมาย	ระยะเวลาประมวลผล [วินาที]
1	5000	10000	-219.154	84.37381	-219.154	41.47466	-219.154	44.63120	-219.154	28.69066
2	10000	5000	-215.716	125.98470	-215.716	36.39134	-215.716	32.54340	-215.716	17.64044
3	5000	15000	-220.14	132.42278	-220.14	77.90746	-220.14	67.23225	-220.14	49.72490
4	15000	5000	-215.071	335.50706	-215.071	62.84183	-215.071	63.25313	-215.071	27.97648
5	5000	20000	-220.454	520.61781	-220.454	88.08352	-220.454	82.97043	-220.454	70.33993
6	10000	10000	-217.935	455.11545	-217.935	116.49054	-217.935	103.51096	-217.935	64.05637
7	10000	15000	-218.882	1133.09763	-218.882	170.26484	-218.882	167.55677	-218.882	89.16230
8	15000	10000	N/A	N/A	-217.143	153.94551	-217.143	143.91503	-217.143	77.13220
9	10000	20000	N/A	N/A	-219.789	237.52606	-219.789	225.54162	-219.789	165.88199
10	20000	10000	N/A	N/A	-216.469	165.29074	-216.469	182.03382	-216.469	87.53562
11	15000	15000	N/A	N/A	-217.726	245.86972	-217.726	248.68947	-217.726	142.29838
12	15000	20000	N/A	N/A	-218.546	379.93975	-218.546	371.23413	-218.546	243.95376
13	20000	15000	N/A	N/A	-217.551	356.82366	-217.551	369.62353	-217.551	205.18482
14	20000	20000	N/A	N/A	-218.284	609.52366	-218.284	643.15995	-218.284	342.08712

ตารางที่ 6 ผลลัพธ์ของระยะเวลาที่ใช้ในการประมวลผลโดยเฉลี่ย

			วิธีการตรง	วิธีการสลับการสร้างแถวและสดมภ์	วิธีการสร้างสดมภ์ขึ้นกับแถว	วิธีการสร้างแถวขึ้นกับสดมภ์
ลำดับที่	จำนวนเงื่อนไข	จำนวนตัวแปร	ประมวลผลโดยเฉลี่ย [วินาที]	ระยะเวลาประมวลผลโดยเฉลี่ย [วินาที]	ระยะเวลาประมวลผลโดยเฉลี่ย [วินาที]	ระยะเวลาประมวลผลโดยเฉลี่ย [วินาที]
1	5000	10000	91.48457	41.72101	41.99774	29.54423
2	10000	5000	133.07323	36.45976	37.87565	19.62240
3	5000	15000	181.36293	76.14838	70.07413	51.71818
4	15000	5000	295.38589	63.50171	62.76971	28.87450
5	5000	20000	269.07404	92.54860	89.90592	69.00494
6	10000	10000	344.18495	111.52845	102.36471	59.38972
7	10000	15000	1340.60110	193.98651	184.31089	106.66751
8	15000	10000	N/A	151.48364	143.95147	74.16020
9	10000	20000	N/A	252.64751	256.25022	161.23621
10	20000	10000	N/A	196.09125	201.42715	99.47513
11	15000	15000	N/A	274.86621	280.93340	150.25910
12	15000	20000	N/A	409.66976	401.06738	232.28657
13	20000	15000	N/A	402.88522	422.13514	213.38505
14	20000	20000	N/A	642.81913	681.93991	377.60577

จากตารางที่ 5 และตารางที่ 6 วิธีการกำหนดการเชิงเส้นแบบเต็มรูป (LPFM) จะคำนวณไม่ได้ตั้งแต่ปัญหาที่ 8 เป็นต้นไป เนื่องจากใช้ตัวแปรและเงื่อนไขทั้งหมดในการประมวลผลหาคำตอบ ซึ่งทำให้เกิดหน่วยความจำหลักไม่เพียงพอ แต่กลับกันวิธีการประยุกต์ใช้วิธีการสร้างแถวและสดมภ์สามารถคำนวณหาผลคำตอบในขนาดปัญหาดังกล่าวได้ อีกทั้งยังมีประสิทธิภาพของระยะเวลาที่ใช้ประมวลผลที่ดีกว่าในทุกกรณีเช่นกัน

เมื่อพิจารณาวิธีการประยุกต์ใช้วิธีการสร้างแถวและสดมภ์ทั้ง 3 วิธีการ พบว่า วิธีการสร้างสดมภ์ขึ้นกับแถว (CDR) จะมีประสิทธิภาพมากที่สุด อย่างที่รู้กันว่าระยะเวลาที่ใช้ในการแก้ปัญหาจะขึ้นอยู่กับจำนวนของเงื่อนไข ดังนั้นวิธีการสร้างสดมภ์ขึ้นกับแถว (CDR) ที่จะเน้นการเลือกตัวแปรที่สำคัญเข้ามาก่อนแล้วจึงเลือกเงื่อนไขที่สำคัญ ซึ่งเมื่อเทียบกับวิธีการสร้างแถวขึ้นกับสดมภ์ (RDC) ที่มีลักษณะตรงกันข้าม ส่งผลให้รูปแบบของปัญหาของวิธีการสร้างสดมภ์ขึ้นกับแถว (CDR) มีจำนวนเงื่อนไขในการพิจารณาน้อยกว่าวิธีการสร้างแถวขึ้นกับสดมภ์ (RDC) ดังนั้นจึงเป็นสาเหตุที่ใช้ระยะเวลาในการประมวลผลน้อยกว่า ส่วนวิธีการสลับวิธีการสร้างแถวและสดมภ์ (SRCG) นั้น เนื่องจากมิได้มีการเน้นสร้างแถวหรือสดมภ์เป็นหลัก จำนวนเงื่อนไขของปัญหาอาจจะมีค่ามากกว่าหรือน้อยกว่าเมื่อเทียบกับวิธีการสร้างแถวขึ้นกับสดมภ์ (RDC) ซึ่งก็จะส่งผลให้ระยะเวลาที่ใช้ในการประมวลผลในบางครั้งอาจจะใช้ระยะเวลาในการประมวลผลได้ดีว่าหรือแย่กว่าก็ได้เช่นกัน

5 สรุปและแนวทางในอนาคต

งานวิจัยนี้มีวัตถุประสงค์เพื่อเปรียบเทียบลำดับของวิธีการสร้างแถวและสดมภ์ ด้วยการเปรียบเทียกวิธีการวิจัย 4 รูปแบบ คือ วิธีการตรงแบบเต็ม วิธีการสลับการสร้างแถวและสดมภ์ วิธีการสร้างสดมภ์ขึ้นกับแถว และวิธีการสร้างแถวขึ้นกับสดมภ์ โดยวัดประสิทธิภาพของแต่ละวิธีการด้วยคุณภาพของคำตอบ และระยะเวลาที่ใช้ในการประมวลผล ซึ่งผลการทดลองได้แสดงให้เห็นว่าวิธีการนำการสร้างสดมภ์และการสร้างแถวมาประยุกต์ใช้สามารถหาคำตอบได้ดีพอกับวิธีการตรงแบบเต็มรูป และให้ประสิทธิภาพในเชิงของระยะเวลาที่ใช้ในการประมวลผลได้ดีกว่า อีกทั้งยังหาคำตอบในปัญหาขนาดใหญ่ได้ ในขณะที่การแก้ปัญหาวิธีการตรงแบบเต็มรูปทำให้เกิดปัญหาหน่วยความจำไม่เพียงพอ และวิธีการนำการสร้างสดมภ์และการสร้างแถวทั้ง 3 วิธีการ พบว่าประสิทธิภาพของวิธีการสร้างสดมภ์ขึ้นกับแถว (CDRG) ให้ประสิทธิภาพดีที่สุด รองลงมาเป็นวิธีการสร้างแถวขึ้นกับสดมภ์ (RDCG) และสุดท้าย (SRCG)

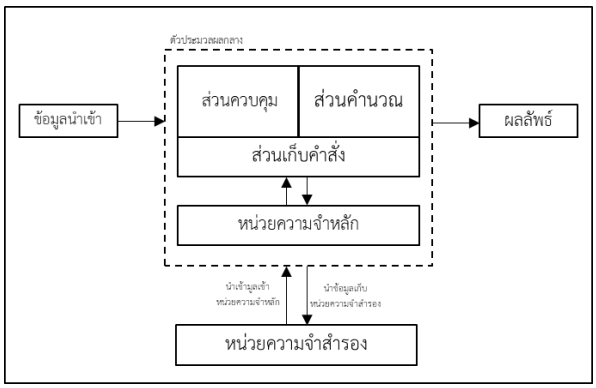
ซึ่งแนวทางในอนาคตอาจจะมีการวิจัยกลยุทธ์อื่นๆ มาปรับใช้ หรือมีการปรับพารามิเตอร์ในการสร้างโจทย์ปัญหา เพื่อเป็นการพิสูจน์และยืนยันว่าประสิทธิภาพของวิธีการสร้างสดมภ์ขึ้นกับแถว (CDRG) ยังคงดีที่สุด

6 ส่วนขยายเพิ่มเติม

สำหรับการขยายงานวิจัยในภายภาคหน้าอาจจะทำการหาโครงสร้างของโจทย์ปัญหาที่เหมาะสมกันของแต่

ละวิธีการ เพื่อหากลยุทธ์ในการเลือกใช้ในแต่ละวิธีการหรือปรับปรุงวิธีการเพื่อเพิ่มประสิทธิภาพรองรับโจทย์ปัญหาที่ใหญ่ขึ้น ดังเช่นตัวอย่างดังต่อไปนี้ จะเป็นการเก็บข้อมูลบางส่วนที่ไม่ได้ใช้ขณะนั้นจากหน่วยความจำหลัก (Primary storage) ไปยังหน่วยความจำสำรอง (Secondary storage) เพื่อเพิ่มพื้นที่ของหน่วยความจำหลักในการคำนวนปัญหา ดังรูปที่ 4

โดยการทดลองนี้ได้เก็บข้อมูลไปยังฮาร์ดดิสก์ (Hard Disk Drive, HDD) โดยใช้วิธีการสร้างสดมภ์ขึ้นกับแถว (CDR) สำหรับโจทย์ปัญหา MxN เป็น 1,000x1,000,000 และกำหนดแบบจำลองขนาดย่อย mxn เป็น 50x10,000 จากขนาดปัญหาหลักดังกล่าว หากมิได้ใช้การเก็บข้อมูลลงหน่วยความจำสำรองจะทำให้เกิดปัญหาหน่วยความจำหลักไม่เพียงพอ ซึ่งพบว่าสามารถประมวลหาคำตอบได้ แต่ผู้วิจัยได้สังเกตเห็นถึงระยะเวลาที่ใช้ในการถ่ายโอนข้อมูลระหว่างหน่วยความจำหลักและหน่วยความจำสำรอง ซึ่งทำให้ใช้ระยะเวลาในการถ่ายโอนข้อมูลนานพอสมควร ซึ่งผลการทดลองครั้งเดียวใช้เวลา 4.65 ชั่วโมง ดังนั้นจึงเสนอให้หาแนวทางในการเพิ่มประสิทธิภาพการถ่ายโอนข้อมูล หรือการคำนวนแบบคู่ขนาน (Parallel processing) เพื่อเพิ่มประสิทธิภาพให้ดีมากยิ่งขึ้น



รูปที่ 4 แผนผังขั้นตอนการประมวลผลของหน่วยประมวลผล
อ้างอิง: https://computersciencewiki.org/

7. เอกสารอ้างอิง

[1] G. B. Dantzig and P. Wolfe, “Decomposition Principle for Linear Programs,” *Operations Research.*, vol. 8, no. 1, pp. 101–111.

[2] R. E. Bixby, J. W. Gregory, I. J. Lustig, R. E. Marsten and D. F. Shanno, “Very Large-Scale Linear Programming: A Case Study in Combining Interior Point and Simplex Methods,” *Operations Research.*, vol. 40, pp. 885-897, 1992.

[3] J. F. Benders, “Partitioning procedures for solving mixed-variables programming problems,” *Numerische Mathematik.*, vol. 4, no. 3, pp. 238–252, 1962.

[4] อภิศักดิ์ วิทยาประภากร, เอราวิล ถาวร, เริงทิวา ทิพยศักดิ์ และ พีรยุทธ์ ชาญเศรษฐิกุล, “การแก้ปัญหาการจัดสรรเงินลงทุนเชิงจัดหมู่ด้วยเทคนิคการสร้างแถว,” *วารสารไทยการวิจัยดำเนินงาน.*, ปีที่ 6, ฉบับที่ 2, น. 10-21, 2561.

[5] G. Dantzig, R. Fulkerson and S. Jonhson, “Solution of a large-scale travelling-salesman problem,” *Journal of the Operation Research Society of America.*, vol. 2, no. 4, pp. 393-410, 1954.

[6] รัฐพล สังคะสุข, ฐิติรัตน์ วิวิธเกยูรวงศ์, เบญจพร พวงจาปี, ศรินทิพย์ อนุรักษ์, สหชาติ เลิศศิริเภสัช และ พีรยุทธ์ ชาญเศรษฐิกุล, “การแก้ปัญหาการจัดสรรเงินลงทุนเชิงจัดหมู่ ด้วยวิธีการสร้างสดมภ์,” *วารสารไทยการวิจัยดำเนินงาน.*, ปีที่ 8, ฉบับที่ 1, น. 37-48, 2563.

[7] A. Sangiovanni-Vincentelli, Li-Kuan Chen and L. Chua, “An efficient heuristic cluster algorithm for tearing large-scale networks,” in *IEEE Transactions on Circuits and Systems*, vol. 24, no. 12, pp. 709-717, 1977.

[8] J. Kytöjoki, T. Nuortio, O. Bräysy and M. Gendreau, “An efficient variable neighborhood search heuristic for very large-scale vehicle routing problems,” *Computers & Operations Research.*, vol. 34, no. 9, pp.2743-2757, 2007.

[9] ปวิธ ไกรสรนุเคราะห์, รมิดายุ อยู่สุข และ นราภรณ์ เภาประเสริฐ, “การปรับปรุงวิธีอาณาจักรมดสาหรับการแก้ปัญหาการจัดเส้นทางขนส่งสินค้าแบบมีขอบเขตเวลาและแบ่งส่งสินค้า,” *วิศวกรรมสารธรรมศาสตร์ มหาวิทยาลัยธรรมศาสตร์.*, ปีที่ 6, ฉบับที่ 1, น.1-8, 2020.

[10] ปวิธ ไกรสรนุเคราะห์, รมิดายุ อยู่สุข, แพรวพรรณ ประหยัดทรัพย์ และ นราภรณ์ เภาประเสริฐ, “การพัฒนาโปรแกรมในการจัดลำดับเครื่องเล่นในสวนสนุกโดยขั้นตอนวิธีเชิงพันธุกรรมและขั้นตอนวิธีเชิงวิวัฒนาการ,” in *The 5th Thai-Nichi Institute of Technology Academic Conference 2019*, 2019.