

ขั้นตอนวิธีการคัดกรองแบบไม่เอนเอียงเพื่อลดขนาดของปัญหาและ
เวลาประมวลผลแบบขนานสำหรับปัญหามัลติเพิลแนปแซค

UNBIASED FILTERING ALGORITHM FOR PARALLEL TIME-SPACE
REDUCTION IN THE MULTIPLE KNAPSACK PROBLEM



พัชรินทร์ บัวเย็น
PATCHARIN BUAYEN

วิทยานิพนธ์นี้สำหรับการศึกษาตามหลักสูตร
ปริญญาปรัชญาดุษฎีบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2565

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้ภายในเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

UNBIASED FILTERING ALGORITHM FOR PARALLEL TIME-SPACE
REDUCTION IN THE MULTIPLE KNAPSACK PROBLEM

PATCHARIN BUAYEN

A THESIS SUBMITTED IN FULFILLMENT OF THE REQUIREMENT FOR THE
DEGREE OF DOCTOR OF PHILOSOPHY PROGRAM IN COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE SCHOOL OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2022

KMITL-2022-SC-D-002-115

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



COPYRIGHT 2022

SCHOOL OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการศึกษาเท่านั้น เมื่อได้รู้เห็นนำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อวิทยานิพนธ์	ขั้นตอนวิธีการคัดกรองแบบไม่เอนเอียงเพื่อลดขนาดของปัญหาและเวลาประมวลผลแบบขนานสำหรับปัญหามัลติเพิลแนปแซค
ชื่อนักศึกษา	นางสาวพัชรินทร์ บัวเย็น
รหัสประจำตัว	62605116
ปริญญา	ปรัชญาดุษฎีบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชา	วิทยาการคอมพิวเตอร์
พ.ศ.	2565
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รองศาสตราจารย์ ดร.จิรพร วีระพันธุ์

บทคัดย่อ

ปัญหาซิงเกิลแนปแซค (0/1-KP) และมัลติเพิลแนปแซค (0/1-mKP) เป็นปัญหากลุ่มเอ็นพีแบบยาก (NP-Hard) ที่ต้องการคำตอบที่ดีที่สุดและเวลาที่รวดเร็วในการประมวลผล เพื่อให้สามารถนำไปใช้ต่อยอดในการปฏิบัติได้จริง จึงทำให้ปัญหานี้เป็นปัญหาที่ได้รับความสนใจอย่างต่อเนื่อง สำหรับวิทยานิพนธ์ฉบับนี้ ผู้วิจัยได้นำเสนอขั้นตอนวิธีใหม่สำหรับการแก้ปัญหแนปแซคและมัลติเพิลแนปแซคที่หาคำตอบที่ดีที่สุดได้อย่างน้อยร้อยละ 99 ของชุดข้อมูลทั้งหมดที่ใช้ในการทดลอง ด้วยความซับซ้อนด้านเวลาแบบโพลิโนเมียล โดยได้แบ่งการพัฒนาขั้นตอนวิธีออกเป็น 2 ช่วง ได้แก่ ช่วงที่ 1 ขั้นตอนวิธีทีเอสรีดักชัน ($TS_{Reduction}$) สำหรับปัญหาซิงเกิลแนปแซค โดยได้เสนอกระบวนการคัดกรองแบบไม่เอนเอียง (Unbiased Filtering) ที่แบ่งสิ่งของในชุดข้อมูลออกเป็น 3 กลุ่ม คือ กลุ่มสิ่งของที่ควรเลือก กลุ่มสิ่งของที่ไม่ควรเลือก และกลุ่มสิ่งของที่ตัดสินใจไม่ได้ ซึ่งสามารถหาคำตอบที่ดีที่สุดได้สำหรับข้อมูลปกติและคำตอบที่ใกล้เคียงคำตอบที่ดีที่สุดสำหรับข้อมูลที่ไม่ปกติ ด้วยความซับซ้อนด้านเวลาในการประมวลผลแบบอนุกรมที่ $O(n+C')$ และแบบขนานที่ $O(n/p \log 2p + C'/p)$ (กรณี $p < n$) และช่วงที่ 2 ขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ ($DPT-List_{TSR}$: $O(n^2, nC)$) สามารถหาคำตอบที่ดีที่สุดได้สำหรับปัญหาซิงเกิลแนปแซคที่นำเสนอการแปลงรูปของขั้นตอนวิธีโปรแกรมเชิงพลวัตจากเดิมที่ใช้การเก็บแบบอาร์เรย์ให้อยู่ในรูปแบบลิสต์ จากนั้นพัฒนาต่อเป็นขั้นตอนวิธีมัลติดีพีทีลิสต์ทีเอสอาร์ ($Multi-DPT-List_{TSR}$: $O(m^2[n^2, nC])$) สำหรับปัญหามัลติเพิลแนปแซค ซึ่งได้ใช้ขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์สำหรับการหาคำตอบในแต่ละถุงเป้ และเพิ่มรูปแบบการเรียงถุงเป้ที่เหมาะสมที่สุดเพื่อให้ได้คำตอบโดยรวมที่ดีที่สุด อีกทั้งได้พัฒนาการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (Robust Unbiased Filtering: $O(m^2n)$) เพื่อลดขนาดปัญหาและเวลาในการประมวลผลให้เป็นโพลิโนเมียลฟังก์ชัน ในการทดลองโดยใช้ชุดข้อมูลสุ่มและชุดข้อมูลมาตรฐาน ($n \leq 10000$ และ $m \leq 100$) ขั้นตอนวิธีที่นำเสนอสามารถหาคำตอบที่ดีที่สุดได้อย่างน้อยร้อยละ 99 จากชุดข้อมูลทั้งหมดที่ใช้ในการทดลอง

คำสำคัญ : การคัดกรองแบบไม่เอนเอียง ขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ ขั้นตอนวิธีทีเอสรีดักชัน

ขั้นตอนวิธีมัลติดีพีทีลิสต์ทีเอสอาร์ ปัญหาซิงเกิลแนปแซค ปัญหามัลติเพิลแนปแซค

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปเผยแพร่โดยไม่ได้รับอนุญาต

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Thesis Title	Unbiased Filtering Algorithm for Parallel Time-Space Reduction in The Multiple Knapsack Problem
Student Name	Miss Patcharin Buayen
Student ID	62605116
Degree	Doctor of Philosophy (Computer Science)
Department	Computer Science
Year	2022
Thesis Advisor	Associate Professor Dr.Jeeraporn Weerapun

Abstract

The 0/1-Knapsack problem (0/1-KP) and Multiple 0/1 Knapsack problem (0/1-mKP) are the NP-hard problems since its optimal solutions are meaningful in real-world applications. Therefore, these interested KP-problems have been researched to improve time performance while retaining the optimal-solution performance as much as possible. This thesis is proposed with the novel algorithms for solving 0/1-KP and 0/1-mKP with at least 99% optimal solutions in polynomial time. In this study, the efficient designing of the algorithms is divided into two parts (for solving the 0/1-KP and the 0/1-mKP). In Part 1, Time-Space Reduction algorithm ($TS_{Reduction}$ algorithm) is proposed to solve the 0/1-KP with unbiased filtering that can classify the objects into 3 groups: selected objects, unselected objects, and uncertain objects (solved with the modified-DP). The proposed $TS_{Reduction}$ algorithm can find the optimal solutions (for regular datasets) and near optimal solutions (for irregular datasets) in $O(n+C')$ and $O(\log 2p+C'/p)$ according to sequential and parallel processing. In Part 2, first the DP-transformation to list-based time-space reduction algorithm (DPT-List_{TSR} algorithm: $O([n^2, nC])$) is proposed to find the optimal solutions for each of m knapsacks by transforming from the 2D-array (nC) of the exact Dynamic Programming (DP). Then, the multi-DPT-List_{TSR} algorithm is proposed for the 0/1-mKP with the exact-fit (best) knapsack order in $O(m^2[n^2, nC])$. Finally, robust unbiased filtering is incorporated to solve the 0/1-mKP in polynomial time ($O(m^2n)$). In experiments, our efficient 0/1-mKP reduction confirmed 99% optimal solutions on random and benchmark datasets ($n \leq 10000$, $m \leq 100$).

Keywords : Unbiased Filtering, DPT-List_{TSR} Algorithm, $TS_{Reduction}$, Multi-DPT-List_{TSR} Algorithm, 0/1-Knapsack Problem (0/1-KP), Multiple 0/1-knapsack Problem (0/1-mKP)

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ ผู้วิจัยได้รับแรงบันดาลใจและความมุ่งมั่นจากพระบรมราโชวาทและพระราชกรณียกิจของพระบาทสมเด็จพระบรมชนกาธิเบศร มหาภูมิพลอดุลยเดชมหาราช บรมนาถบพิตร ที่แสดงถึงความอดทน ความมุ่งมั่นอย่างไม่ย่อท้อต่ออุปสรรคและการคิดค้นสร้างสรรค์นวัตกรรมใหม่ ๆ บนพื้นฐานความเรียบง่าย เพื่ออำนวยความสะดวกสูงสุดให้แก่ประชาชนให้สามารถดำรงตนอยู่ได้ด้วยหลักเศรษฐกิจพอเพียงร่วมกับการสร้างสรรค์นวัตกรรมชุมชนภายใต้พื้นฐานความสามัคคี ซึ่งเป็นรากฐานที่สำคัญของประเทศและเป็นพื้นฐานที่ทำให้ทุกคนในสังคมอยู่ร่วมกันได้อย่างมีความสุขและยั่งยืน ผู้วิจัยขอเทิดพระเกียรติและขอน้อมนำพระบรมราโชวาทเพื่อการพัฒนาและคิดค้นผลงานที่ดีตลอดไป

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความสะดวกตา กรุณาและเอาใจใส่ดูแลอย่างดีมาตลอดจาก รองศาสตราจารย์ ดร.จิรพร วีระพันธุ์ ผู้เป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งได้ให้คำแนะนำทั้งในด้านวิชาการ ความรู้ ทักษะ และการดำเนินชีวิตเพื่อส่งเสริมให้ผู้วิจัยเป็นผู้ที่มีความรู้ ความสามารถ และทักษะ เพื่อนำสิ่งที่ได้รับจากการศึกษาเล่าเรียนไปคิดค้นสร้างสรรค์ผลงานเพื่อสังคมบนพื้นฐานของความดีและการให้อย่างไม่มีขอบเขต ผู้วิจัยใคร่ขอขอบพระคุณเป็นอย่างสูง และจะนำคำสอนที่ได้รับไปเป็นแนวทางในการพัฒนาผลงานและดำเนินชีวิตให้เป็นประโยชน์ต่อตนเองและสังคมต่อไป

ผู้วิจัยขอขอบพระคุณ รองศาสตราจารย์ ดร.ศุภกานต์ พิมลธเรศ ประธานกรรมการสอบวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร.วรางคณา กัมปาน ผู้ช่วยศาสตราจารย์ ดร.อนันตพร หรรษคุณาภย์ และผู้ช่วยศาสตราจารย์ ดร.นวลสวาท หิรัญสกุลวงศ์ คณะกรรมการสอบวิทยานิพนธ์ที่กรุณาให้คำแนะนำ เพื่อให้วิทยานิพนธ์ฉบับนี้มีความสมบูรณ์มากยิ่งขึ้น อีกทั้งเป็นการส่งเสริมให้ผู้วิจัยมีวิสัยทัศน์และเป็นแนวทางในการพัฒนางานวิจัยที่ดีขึ้นต่อไป

ขอขอบพระคุณคณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ให้โอกาสทางด้านการศึกษาและเปิดโอกาสให้ผู้วิจัยได้รู้จักสังคมแห่งการเรียนรู้ที่ดี

ขอขอบพระคุณทีมงานวารสาร Journal of Parallel and Distributed Computing และวารสาร Algorithms ของสำนักพิมพ์ MDPI สำหรับคำแนะนำที่เป็นประโยชน์ต่อการพัฒนางานวิจัยให้มีคุณภาพมากขึ้น และมอบโอกาสในการเผยแพร่งานวิจัยสู่สาธารณชนเพื่อให้เกิดเป็นประโยชน์ต่อไป

ขอขอบพระคุณบิดามารดาและครอบครัวที่เป็นแรงบันดาลใจให้ผู้วิจัยมุ่งมั่นพัฒนางานตนเองแสวงหาความรู้ รวมถึงปลูกฝังความเป็นครู การเป็นผู้ให้ และเป็นกำลังใจในการเรียนตลอดมา และขอขอบคุณกัลยาณมิตรทุกคนที่ได้ให้กำลังใจ ช่วยเหลือและส่งเสริมสนับสนุนมาโดยตลอด

สำหรับคุณงามความดีและประโยชน์อันใดที่พึงเกิดจากวิทยานิพนธ์ฉบับนี้ ผู้วิจัยขอถวายเป็นพระราชกุศลแด่พระบาทสมเด็จพระบรมชนกาธิเบศร มหาภูมิพลอดุลยเดชมหาราช บรมนาถบพิตร และขอบใจแด่บิดา มารดา คณาจารย์ที่เคารพรัก ตลอดจนญาติพี่น้อง และกัลยาณมิตรทุกท่าน

พัชรินทร์ บัวเย็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

	หน้า
บทคัดย่อภาษาไทย.....	ก
บทคัดย่อภาษาอังกฤษ.....	ข
กิตติกรรมประกาศ.....	ค
สารบัญ.....	ง
สารบัญตาราง.....	ช
สารบัญรูป.....	ญ
สารบัญขั้นตอนวิธี.....	ฎ
รายการคำย่อและสัญลักษณ์.....	ท
บทที่ 1 บทนำ.....	1
1.1 ความเป็นมาและความสำคัญของงานวิจัย.....	1
1.2 วัตถุประสงค์ของงานวิจัย	4
1.3 ขอบเขตของงานวิจัย	5
1.4 ประโยชน์ที่ได้รับ	5
บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง.....	6
2.1 ขั้นตอนวิธีกริดีฮิวริสติก (Greedy Heuristic Algorithm)	6
2.2 ขั้นตอนวิธีโปรแกรมเชิงพลวัต (Dynamic Programming Algorithm)	7
2.2.1 ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื้องต้น (Basic-DP Algorithm)	8
2.2.2 ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็ว (Fast-DP Algorithm).....	9
2.2.3 ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ (Complete-DP Algorithm)	10
2.3 ขั้นตอนวิธีการหาผลรวมพรีฟิกซ์แบบขนาน (Parallel Prefix Sum Algorithm).....	11
2.4 การเรียงลำดับข้อมูลแบบขนานด้วยขั้นตอนวิธีโอบีเอส (Parallel Sorting by OBS Algorithm).....	13
2.5 ขั้นตอนวิธีพอยเตอร์จัมป์ (Pointer-Jumping Algorithm)	15
2.6 แผนการทดลองแบบลาตินสแควร์ (Latin Squares Design).....	16
2.7 งานวิจัยที่เกี่ยวข้อง (Related Work).....	18
2.7.1 ขั้นตอนวิธีกริดีฮิวริสติกด้วยการคาดการณ์อย่างมีประสิทธิภาพ (Greedy with Dynamic Expectation Efficiency)	18
2.7.2 การหาคำตอบที่เหมาะสมแบบกลุ่มด้วยวิธีชฟเฟิลฟรอก (Shuffled Frog Swarm Optimization)	19

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

2.7.3 การหาคำตอบที่เหมาะสมแบบฝูงแมลงวันผลไม้สำหรับปัญหามัลติไดเมนชันแนปแซค (Improved Fruit Fly Optimization Algorithm for the Multidimensional Knapsack Problem)	21
2.7.4 การหาคำตอบที่เหมาะสมตามรูปแบบการเรียนรู้ของมนุษย์สำหรับปัญหามัลติไดเมนชันแนปแซค (Human Learning Optimization Algorithm to Multidimensional Knapsack Problems)	22
2.7.5 ขั้นตอนวิธีการประเมินที่แตกต่างตามแนวทางควอนตัมเพื่อการแก้ปัญหาซิงเกิลแนปแซค (0/1-KP Solving by Quantum -Inspired Differential Evolution Algorithm)	23
2.7.6 ขั้นตอนวิธีแบบจำลองทางคณิตศาสตร์สำหรับปัญหามัลติเพิลแนปแซค (Mathematical Hy-MKP).....	23
บทที่ 3 ขั้นตอนวิธีที่ไฮสรีดักชันสำหรับปัญหาซิงเกิลแนปแซค (0/1-KP)	28
3.1 ขั้นตอนวิธีฮิวริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพ (Greedy Heuristic with Efficient Feature Sorting)	29
3.2 ขั้นตอนการแบ่งกลุ่มและการคัดกรองแบบไม่เอนเอียง (Object Classification and Unbiased Filtering).....	30
3.2.1 การปรับปรุงผลลัพธ์ให้ดีขึ้นด้วยขั้นตอนวิธีจีเอชพลัส (Improve Solutions by the GH+ Strategy).....	31
3.2.2 การกำหนดค่าน้ำหนักและคำนวณค่าความเหมาะสมแบบพลวัตเพื่อการแบ่งกลุ่มสมาชิก (Compute Dynamic Weighting to Classify Objects).....	32
3.2.3 การคัดกรองแบบไม่เอนเอียง (Unbiased Filtering)	33
3.3 ขั้นตอนวิธีเอ็มดีพี (MDP)	36
3.4 ขั้นตอนวิธีที่ไฮสรีดักชันแบบขนาน (Parallel TS _{Reduction} Algorithm)	40
3.4.1 ขั้นตอนวิธีฮิวริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพในแบบขนาน (Parallel-GH).....	41
3.4.2 ขั้นตอนการแบ่งกลุ่มและการคัดกรองแบบไม่เอนเอียงในแบบขนาน (Parallel-Classification and Filtering)	43
3.4.3 ขั้นตอนวิธีเอ็มดีพีแบบขนาน (Parallel-MDP).....	45

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

	หน้า
บทที่ 4 ขั้นตอนวิธีมัลติดีพีทีลิสต์ทีเอสอาร์สำหรับปัญหามัลติเพิลแนปแซค (0/1-mKP)	48
4.1 ขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์สำหรับปัญหาซิงเกิลแนปแซค (DPT-List _{TSR} Algorithm for the 0/1-KP).....	48
4.1.1 ขั้นตอนพรีโพรเซสซิง (Pre-processing).....	49
4.1.2 ขั้นตอนการเอ็กซ์แทร็ก (X-tracking)	55
4.2 ขั้นตอนวิธีมัลติดีพีทีลิสต์ทีเอสอาร์ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่งสำหรับปัญหามัลติเพิลแนปแซค (multi-DPT-List _{TSR} plus Robust Filtering for 0/1-mKP)	57
4.2.1 ขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์สำหรับปัญหามัลติเพิลแนปแซค (Efficient multi-DPT-List _{TSR} for solving 0/1-mKP)	58
4.2.2 การคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (Efficient Robust Unbiased Filtering).....	63
บทที่ 5 การวิเคราะห์ความถูกต้องแม่นยำและความซับซ้อนด้านเวลา	69
5.1 ขั้นตอนวิธีทีเอสรีดักชันแบบขนาน (TS _{Reduction} Algorithm)	69
5.1.1 ความแม่นยำของขั้นตอนวิธีทีเอสรีดักชัน (Precision of TS _{Reduction} Algorithm)	
5.1.2 ความซับซ้อนด้านเวลาของขั้นตอนวิธีทีเอสรีดักชัน (Time Complexity of TS _{Reduction} Algorithm).....	73
5.2 ขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ (DPT-List _{TSR} Algorithm)	77
5.2.1 ความถูกต้องของขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ (Correctness of the DPT-List _{TSR} Algorithm).....	77
5.2.2 ความซับซ้อนของขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ (Complexity Analysis of the DPT-List _{TSR} Algorithm).....	79
5.2.3 ความแม่นยำของการเรียงถุงเป้ที่เหมาะสมพอดี (Exact Precision (as m! Orders) of the Exact-fit (Best) Knapsack Order).....	80
5.2.4 ความแม่นยำของการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (High Exact Precision of the Robust Unbiased Filtering).....	82
บทที่ 6 ผลการทดลอง	84
6.1 ผลการทดลองของขั้นตอนวิธีทีเอสรีดักชัน (Experimental Result of TS _{Reduction} Algorithm).....	84

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

	หน้า
6.2 ผลการทดลองของขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์และมัลติดีพีทีลิสต์ทีเอสอาร์	88
6.2.1 ผลการทดลองของขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ร่วมกับการคัดกรองแบบ ไม่เอนเอียงที่แข็งแกร่ง (Results of the DPT-List _{TSR} and Robust Unbiased Filtering).....	88
6.2.2 ผลการทดลองของขั้นตอนวิธีมัลติดีพีทีลิสต์ทีเอสอาร์ร่วมกับการคัดกรอง แบบไม่เอนเอียงที่แข็งแกร่ง (Results of the Multi-DPT-List _{TSR} and Robust Unbiased Filtering).....	89
6.2.3 ผลการทดลองเพิ่มเติมและการปรับปรุงเพิ่มเติมเกี่ยวกับชุดข้อมูลพิเศษ ...	96
บทที่ 7 สรุปผลการทดลองและแนวทางการพัฒนางานวิจัย.....	100
7.1 สรุปผลการวิจัย	100
7.2 ข้อเสนอแนะ	101
บรรณานุกรม.....	102
ภาคผนวก.....	108
ภาคผนวก ก.....	109
ภาคผนวก ข.....	124
ประวัติผู้เขียน	159

สารบัญตาราง

ตารางที่	หน้า
2.1 เปรียบเทียบประสิทธิภาพขั้นตอนวิธีที่ศึกษา.....	26
3.1 การเรียงลำดับแบบหลายระดับของขั้นตอนวิธีจีเอช	30
3.2 การแบ่งกลุ่มสมาชิกตามค่าน้ำหนักพลวัต (dw : dynamic weight).....	33
3.3 เปรียบเทียบข้อดีและข้อจำกัดของขั้นตอนวิธีที่ได้ศึกษากับขั้นตอนวิธีทีเอสรีดักชัน	38
3.4 ความซับซ้อนด้านเวลาของขั้นตอนวิธีทีเอสรีดักชันแบบขนาน (Parallel-TS _{Reduction})	46
5.1 เปรียบเทียบความสมบูรณ์ของผลลัพธ์ ประสิทธิภาพด้านความถูกต้องและความซับซ้อนด้านเวลาของขั้นตอนวิธีทีเอสรีดักชัน (TS _{Reduction}) ขั้นตอนวิธีกรีดี้ฮิวริสติก (Greedy Heuristic : GH) และขั้นตอนวิธีโปรแกรมเชิงพลวัต (Dynamic Programming : DP).....	74
5.2 เปรียบเทียบวิธีการ จุดแข็ง และจุดอ่อนของขั้นตอนวิธี.....	76
5.3 เปรียบเทียบความซับซ้อนด้านเวลาของขั้นตอนวิธีแบบขนาน	76
5.4 เปรียบเทียบประสิทธิภาพการจัดเรียงถุงเป้สำหรับปัญหา 0/1-mKP	81
6.1 ผลการทดลองส่วนมูลค่ารวมสำหรับชุดข้อมูลมาตรฐาน	85
6.2 ผลการทดลองส่วนมูลค่ารวมสำหรับชุดข้อมูลแบบสุ่ม	85
6.3 ผลการทดลองส่วนน้ำหนักรวมสำหรับชุดข้อมูลแบบสุ่ม	86
6.4 เวลาประมวลผลของขั้นตอนวิธีทีเอสรีดักชัน ขั้นตอนวิธีกรีดี้ฮิวริสติก และขั้นตอนวิธีโปรแกรมเชิงพลวัต	86
6.5 ประสิทธิภาพด้านความเร็วสำหรับชุดข้อมูลขนาด $n = 5000000$ ด้วยจำนวนโปรเซสเซอร์ที่ $p \leq 4096$	87
6.6 ผลการทดลองสำหรับชุดข้อมูลไม่ปกติ (จากชุดข้อมูลบางส่วนใน 223 จาก 10000)	88
6.7 ความแม่นยำของขั้นตอนวิธีดีฟิทีลิสต์ทีเอสอาร์ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (DPT-List _{TSR} + Robust Unbiased Filtering) เมื่อ $(n \leq 10000)$	88
6.8 ประสิทธิภาพการลดเนื้อที่เก็บข้อมูลของขั้นตอนวิธีดีฟิทีลิสต์ทีเอสอาร์ (DPT-List _{TSR}).....	89
6.9 ผลลัพธ์ส่วน Total Profit ของชุดข้อมูลที่มีความจุของถุงเป้ขนาด $C \pm 10$ ($m = 2$).....	91
6.10 ผลลัพธ์ส่วน Total Profit ของชุดข้อมูลที่มีความจุของถุงเป้ขนาด $C \pm 15$ ($m = 3$).....	91
6.11 ผลลัพธ์ส่วน Total Profit ของชุดข้อมูลที่มีความจุของถุงเป้ขนาด $C \pm 20$ ($m = 4$).....	92
6.12 ผลลัพธ์ของชุดข้อมูลที $C \pm 20$ สำหรับ $n = 1000$ ถึง 5000 ที $m = 6$ ถึง 50	92
6.13 ผลลัพธ์ของชุดข้อมูลที $C \pm 10, 20$ สำหรับ $n = 9000$ ที $m = 40$ ถึง 90	92
6.14 ผลลัพธ์ส่วน Total Profit ของชุดข้อมูลผิดปกติ ที $m = 3$ ถึง 25 และ $n \leq 10000$	93
6.15 จำนวนชุดข้อมูลที่ไม่ได้คำตอบที่ดีที่สุดสำหรับ $n \leq 10000$ ต่อ m (ที $m < 100$).....	93

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง (ต่อ)

ตารางที่	หน้า
6.16 ผลลัพธ์ของชุดข้อมูลมาตรฐานขนาด $n = 100$ และ $m = 10$	94
6.17 ผลลัพธ์ของชุดข้อมูลมาตรฐานขนาด $200 \leq n \leq 500$ และ $20 \leq m \leq 50$	95
6.18 การเปรียบเทียบขั้นตอนวิธี mDPT-List _{TSR} ที่นำเสนอและขั้นตอนวิธี HyMKP [16].....	98



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

รูปที่	หน้า
2.1 ตัวอย่างการแก้ปัญหาถุงเป้ด้วยขั้นตอนวิธีกรีดีฮิวริสติก	7
2.2 ตัวอย่างการประมวลผลด้วยขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื้องต้น	9
2.3 ตัวอย่างการประมวลผลด้วยขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็ว	9
2.4 ตัวอย่างการประมวลผลด้วยขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์.....	11
2.5 เทคนิคดีไวด์แอนด์คอนเคอร์ (Divide-and-Conquer)	12
2.6 ขั้นตอนวิธีการหาผลรวมพรีฟิกซ์แบบขนาน (Parallel Prefix Sum).....	13
2.7 ตัวอย่างของขั้นตอนวิธีการหาผลรวมพรีฟิกซ์แบบขนาน.....	13
2.8 การเรียงลำดับข้อมูลแบบขนานด้วยขั้นตอนวิธีโอบีเอส (Parallel Sorting by OBS Algorithm)..	14
2.9 ตัวอย่างการเรียงลำดับข้อมูลแบบโอบีเอสด้วยค่ามิติพอยท์	15
2.10 แผนภาพแสดงการทำงานของขั้นตอนวิธีพอยเตอร์จัมป์.....	16
2.11 ตัวอย่างการทำงานของขั้นตอนวิธีพอยเตอร์จัมป์สำหรับ $n = 8$ และ $p = 8$	16
2.12 ตัวอย่างการทำงานของแผนการทดลองแบบลาตินสแควร์ขนาด 4×4	17
2.13 การแบ่งกลุ่มสมาชิกของปัญหาด้วยขั้นตอนวิธีจีดีอี (Classified Objects by GDEE Algorithm)	18
2.14 ขั้นตอนวิธีดีเอสเอฟแอลเอแบบดัดแปลง (Modified-DSFLA).....	20
2.15 ฟังก์ชันกระบวนการของการหาคำตอบที่เหมาะสมแบบฝูงแมลงวันผลไม้ (IFFOA)	21
2.16 ฟังก์ชันกระบวนการของการหาคำตอบที่เหมาะสมตามการเรียนรู้ของมนุษย์ (HLO)	22
2.17 กรอบแนวคิดของขั้นตอนวิธี QDGWO (Quantum-inspired Differential (QD) evolution with adaptive Grey Wolf Optimizer (GWO)).....	23
2.18 แบบจำลองการไหลโมเดลอาร์คโฟลว์ (Arc-flow Graph and Optimal Solution) สำหรับถุงเป้ 4 สิ่งของขั้นตอนวิธีดีเอสเอฟแอลเอแบบดัดแปลง (Modified-DSFLA).....	25
2.19 แบบจำลองรีเฟลกและการหาคำตอบที่ดีที่สุดเนื่องจากรูปที่ 2.18 (Reflect Graph and Optimal Solution for the instant of Fig.2.18)	26
3.1 ตัวอย่างผลลัพธ์ของขั้นตอนวิธีจีเอช	30
3.2 ตัวอย่างผลลัพธ์ของขั้นตอนวิธีจีเอชพลัส	31
3.3 ตัวอย่างการกำหนดค่าน้ำหนักและผลรวมค่าน้ำหนักพลวัต (dynamic weight : dw).....	32
3.4 การคัดกรองระดับ 1 (Filtering 1) และตัวอย่าง	34
3.5 การคัดกรองระดับ 2 (Filtering 2) และตัวอย่าง	35
3.6 ตัวอย่างขั้นตอนวิธีเอ็มดีพี	36

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

รูปที่	หน้า
3.7 ตัวอย่างการประมวลผลของขั้นตอนวิธีทีเอสรีดักชัน.....	37
3.8 ตัวอย่างการติดต่อสื่อสารของ MINs ⁺ สำหรับโปรเซสเซอร์ขนาด $n = 8$	40
3.9 การติดต่อสื่อสารแบบขนานเพื่อการคำนวณค่า tw แบบพรีฟิกซ์ที่ $p < n$ เพื่อเลือก/ไม่เลือก ($x_j = 0/1$) และตัวอย่างการหาค่าผลรวม (tp และ tw) และตำแหน่งสุดท้ายในลำดับที่เลือก (point).....	42
3.10 ตัวอย่างการประมวลผลของขั้นตอนวิธีทีเอสรีดักชันแบบขนาน (Parallel-TS _{Reduction}).....	44
3.11 ตัวอย่างขั้นตอนวิธีเอ็มดีพีแบบขนาน (Parallel-MDP).....	46
4.1 ตัวอย่างขั้นตอนพรีโปรเซสซิ่งของ DPT-List _{TSR}	51
4.2 ตัวอย่างแสดงการลดจำนวนโหนดที่ใช้ใน F-list และ B-list สำหรับ $n=15$, $C=40$	54
4.3 ตัวอย่างการแทรกข้อมูลในลิสต์ด้วยขั้นตอนวิธีการเอ็กซ์แทรก สำหรับ $n=5$, $C=18$	55
4.4 ตัวอย่างการลดจำนวนข้อมูลด้วย B-reduction ของปัญหาขนาด $n=15$, $C=40$	56
4.5 แนวทางการพัฒนางานวิจัยสำหรับปัญหามัลติเฟลแนปแซค และภาพรวมของขั้นตอนวิธีมัลติดีพีทีลิสต์ทีเอสอาร์และรูปแบบสำหรับการประมวลผลแบบขนาน	57
4.6 ตัวอย่างการเรียงถุงเป้แบบเก้าอันดับสูงสุดร่วมกับลาดินสแควร์สำหรับปัญหา 5 ถุงเป้	59
4.7 ตัวอย่างการเอ็กซ์แทรก 2 แบบสำหรับผลลัพธ์ที่ไม่ได้มีเพียงหนึ่งค่าที่เกิดขึ้นในถุงเป้ K_3 ..	60
4.8 หลักการเรียงถุงเป้แบบเหมาะสมพอดีและตัวอย่างของวิธีการเลือกถุงเป้ตามรูปแบบการเรียงถุงเป้แบบเหมาะสมพอดี	63
4.9 แผนภาพแสดงการคัดกรองด้วยวิธีการคัดกรองแบบไม่เอนเอียง (บทที่ 3) และการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง.....	64
4.10 ตัวอย่างการแบ่งกลุ่มข้อมูลของปัญหาขนาด $n = 25$, $m = 2$, $C = (30, 40)$ ในถุงเป้หมายเลข K_1	66
4.11 ตัวอย่างการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่งและขั้นตอนดีพีทีลิสต์ทีเอสอาร์สำหรับ n' ใน K_1 และ K_3	66
4.12 รูปแบบการเพิ่มประสิทธิภาพการแก้ปัญหามัลติเฟลแนปแซคโดยใช้การเรียงถุงเป้ที่เหมาะสมพอดีร่วมกับกระบวนการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่งในแต่ละเป้ในขั้นตอนดีพีทีลิสต์ทีเอสอาร์และตัวอย่าง.....	67
5.1 ตัวอย่างชุดข้อมูลพิเศษขนาด $n = 5$ ที่ไม่สามารถหาคำตอบที่ดีที่สุดได้ด้วยวิธี GH (P/W).....	70
5.2 ตัวอย่างชุดข้อมูลพิเศษขนาด $n = 35$ ที่ไม่สามารถหาคำตอบที่ดีที่สุดได้ด้วยวิธี GH (P/W).....	70

สารบัญรูป (ต่อ)

รูปที่	หน้า
5.3 แผนภาพแสดงความสามารถคัดแยกสิ่งของบริเวณจุดวิกฤติ (α และ β) มาไว้ในกลุ่มที่ไม่แน่นอนสำหรับปัญหาขนาดเล็กและปัญหาขนาดใหญ่	71
5.4 แผนภาพแสดงการคัดกรองแบบไม่เอนเอียง (Unbiased Filtering) และการจัดการสมาชิกที่เป็นค่าผิดปกติ (Outlier Handling) ด้วยขั้นตอนวิธีทีเอสรีดักชัน ($TS_{Reduction}$).....	72
5.5 ตัวอย่างการเพิ่ม e-node ใน F-list j	78
5.6 ภาพรวมของการลดขนาดข้อมูลด้วย F-reduction และ B-reduction ของขั้นตอนวิธีดีพีที่ลิสต์ทีเอสอาร์	80
5.7 แผนภาพพีรามิดแสดงการเรียงลำดับความจุของถุงเป้จากน้อยไปมาก กรณีปัญหาถุงเป้แบบหลายมิติ.....	80
5.8 ความแม่นยำของการคัดกรองแบบแข็งแกร่งสำหรับปัญหาซิงเกิลแนปแซค (0/1-KP).....	82
6.1 แผนภาพแสดงเวลาประมวลผลของขั้นตอนวิธีทีเอสรีดักชัน ($TS_{Reduction}$) กับขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็ว (Fast-DP) และแบบสมบูรณ์ (Complete-DP).....	87
6.2 แผนภาพแสดงประสิทธิภาพด้านเวลาของขั้นตอนวิธีทีเอสรีดักชันแบบขนาน (Parallel- $TS_{Reduction}$) ขนาด $n = 5000000$ ด้วยจำนวนหน่วยประมวลผลที่ $p \leq 4096$	87
6.3 การบรรจุเกิน (Over Packing) ของปัญหาซิงเกิลแนปแซค (0/1-KP) ในชุดข้อมูลวิกฤติ.....	95
6.4 การปรับปรุงแนวทาง multi-DPT-ListTSR สำหรับปัญหามัลติเพิลแนปแซค (0/1-mKP) ..	97

สารบัญขั้นตอนวิธี

ขั้นตอนวิธีที่	หน้า
2.1 ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื้องต้น (Basic-DP Algorithm).....	8
2.2 ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็ว (Fast-DP Algorithm).....	9
2.3 ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ (Complete-DP Algorithm).....	10
2.4 ขั้นตอนวิธีการหาผลรวมพรีฟิกซ์แบบขนาน (Parallel Prefix Sum Algorithm).....	12
2.5 ขั้นตอนวิธีลิสต์เรียงกิ่ง (List-Ranking Algorithm)	16
2.6 ขั้นตอนวิธีการหาคำตอบที่เหมาะสมแบบกลุ่มด้วยวิธีชัฟเฟิลฟรอก (Shuffled Frog Swarm Optimization Algorithm).....	19
2.7 ขั้นตอนวิธีแบบจำลองทางคณิตศาสตร์สำหรับปัญหามัลติเพิลแนปแซค (Mathematical HyMKP).....	24
2.8 ขั้นตอนวิธีการสร้างรีเฟล็กมัลติกราฟสำหรับปัญหามัลติเพิลแนปแซค (Create-reflect-multigraph-MKP).....	25
3.1 ขั้นตอนวิธีทีเอสรีดักชัน (TS _{Reduction} Algorithm).....	28
3.2 ขั้นตอนวิธีจีเอชพลัส (GH ⁺ : Greedy Heuristic Plus Algorithm).....	31
3.3 ขั้นตอนวิธีทีเอสรีดักชันแบบขนาน (Parallel-TS _{Reduction} Algorithm).....	40
3.4 ขั้นตอนวิธีจีเอชแบบขนาน (Parallel-GH Algorithm).....	41
3.5 ขั้นตอนวิธีจีเอชพลัสและการคัดกรองแบบขนาน (Parallel-GH ⁺ and Filtering Algorithm)	43
3.6 ขั้นตอนวิธีเอ็มดีพีแบบขนาน (Parallel-MDP Algorithm).....	45
4.1 ขั้นตอนวิธีโพรเซสซิง (Pre-processing).....	50
4.2 ขั้นตอนวิธีการเฝ้าติดตาม (X-tracking).....	55
4.3 ขั้นตอนวิธีมัลติดีพีทีลิสต์ที่เอสอาร์ด้วยการจัดเรียงถุงเป้หนึ่งในเค้ารูปแบบ (Multi-DPT-List _{TSR} for one order Algorithm).....	59
4.4 ขั้นตอนวิธีมัลติดีพีทีลิสต์ที่เอสอาร์โดยใช้กลยุทธ์การเรียงถุงเป้ที่ดีที่สุด (multi-DPT-List _{TSR} with exact-fit (best) order)).....	62
4.5 ขั้นตอนวิธีมัลติดีพีทีลิสต์ที่เอสอาร์ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (Multi-DPT-List _{TSR} plus Robust Unbiased Filtering).....	65

รายการคำย่อและสัญลักษณ์

0/1-KP	ปัญหาแนปแซค (0/1-Knapsack Problem) ที่มีถุงเป้เดียวในชุดข้อมูล
0/1-mKP	ปัญหามัลติเพิลแนปแซค (0/1-Multiple Knapsack Problem) ที่มีหลายถุงเป้ในชุดข้อมูล
GH	ขั้นตอนวิธีกรีดีฮิวริสติก (Greedy Heuristic Algorithm)
GH ⁺	ขั้นตอนวิธีจีเอชพลัส (Greedy Heuristic Plus Algorithm)
DP	ขั้นตอนวิธีโปรแกรมเชิงพลวัต (Dynamic Programming)
DPT	ขั้นตอนวิธีที่แปลงรูปจากขั้นตอนวิธีโปรแกรมเชิงพลวัต (DP Transformation)
MDP	ขั้นตอนวิธีเอ็มดีพี (Modified Dynamic Programming)
m	จำนวนถุงเป้ในปัญหาถุงเป้แบบหลายถุง
n	จำนวนสมาชิกทั้งหมดในปัญหา
n'	จำนวนสมาชิกที่เหลือในแต่ละถุงเป้
n*	จำนวนสมาชิกที่เหลือทั้งหมด
C	ขนาดความจุสูงสุดของถุงเป้ (Capacity)
C'	ขนาดความจุที่เหลือ (Remaining Capacity)
W	น้ำหนักของสิ่งของ (Weight)
P	มูลค่าของสิ่งของ (Profit)
P/W	สัดส่วนของมูลค่าต่อน้ำหนักของสิ่งของ (Profit/Weight)
p	โปรเซสเซอร์ (Processor)
w	จำนวนสมาชิกของปัญหาในแต่ละโปรเซสเซอร์ (Workload : $w = n/p$)
tw	น้ำหนักรวม (Total Weight)
tp	มูลค่ารวม (Total Profit)
soltw	น้ำหนักรวมของสิ่งของในถุงที่เป็นคำตอบ
soltp	มูลค่ารวมที่มากที่สุดของสิ่งของในถุงที่เป็นคำตอบ
solx	รายการดัชนีสิ่งของที่ถูกเลือกเป็นคำตอบ (Solution X)
dw	ค่าน้ำหนักพลวัต (Dynamic Weight) จาก 3 คำตอบ (P/W, P, W)
γ	ขอบเขตวิกฤติในกลุ่ม 1 ($dw = 9$)
α	ขอบเขตวิกฤติในกลุ่ม 2 ($dw = 8, 6, 5$)
u	ขอบเขตวิกฤติในกลุ่ม 3 ($dw = 4, 3, 1$)
β	ขอบเขตวิกฤติในกลุ่ม 4 ($dw = 0$)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของงานวิจัย

การพัฒนาเทคโนโลยีในยุคปัจจุบันได้มีการประยุกต์ขั้นตอนวิธีทางคอมพิวเตอร์เพื่อการใช้งานจริงในหลากหลายสาขา ซึ่งส่วนใหญ่จัดอยู่ในกลุ่มปัญหาเอ็นพีแบบยาก (NP-hard Problem) ที่ไม่สามารถประมวลผลได้ในเวลาโพลิโนเมียล (Polynomial Time) ดังนั้น จึงได้มีการปรับปรุงและบูรณาการขั้นตอนวิธีให้มีประสิทธิภาพมากขึ้นเพื่อให้หาคำตอบที่ดีที่สุดได้ในเวลาที่รวดเร็ว ซึ่งปัญหาหนึ่งที่ได้รับคามนิยมในการนำไปประยุกต์ใช้ คือ ปัญหาในกลุ่มแนปแซค อาทิ ปัญหาซิงเกิลแนปแซค (0/1-Knapsack Problem: 0/1-KP) ที่มีหนึ่งถุงเป้ในชุดปัญหาและปัญหามัลติเพิลแนปแซค (Multiple Knapsack Problem: 0/1-mKP) ที่มีหลายถุงเป้ในชุดปัญหา ซึ่งเป็นปัญหาเอ็นพีแบบยาก (NP-hard Problem) ที่มีการศึกษาวิจัยอย่างกว้างขวางทั้งในอดีตและปัจจุบัน รวมถึงเป็นปัญหาเอ็นพีพื้นฐานที่นิยมใช้ในการทดลองและเปรียบเทียบผลลัพธ์สำหรับศึกษาและพัฒนาขั้นตอนวิธีทางคอมพิวเตอร์ใหม่ ๆ เพื่อแก้ปัญหาเอ็นพีในกรณีอื่น ๆ โดยวัตถุประสงค์ของปัญหากลุ่มแนปแซค คือ การคัดเลือกสิ่งของ (Object) ที่มีน้ำหนัก (Weight: W) และมูลค่า (Profit: P) ที่แตกต่างกันบรรจุลงถุงเป้ที่รองรับน้ำหนักได้จำกัด (Capacity: C) โดยที่จะต้องได้มูลค่ารวมของทุกถุงเป้มากที่สุด (Maximum Profit) และเป็นคำตอบที่ดีที่สุด (Optimal Solution)

นิยามและลักษณะเฉพาะของปัญหาซิงเกิลแนปแซค (0/1-KP) เป็นการคัดเลือกสิ่งของที่เหมาะสมบรรจุลงถุงเป้หนึ่งถุงเป้ที่มีขนาดจำกัด ซึ่งมีคุณสมบัติของปัญหา ได้ดังนี้ [30]

$$\sum_{j=0}^{n-1} p_j x_j \quad (1.1)$$

$$\sum_{j=0}^{n-1} w_j x_j \leq C \quad (1.2)$$

$$x_j \in \{0, 1\}; j = 0, 1, 2, \dots, n-1 \quad (1.3)$$

นิยามและลักษณะเฉพาะของปัญหามัลติเพิลแนปแซค (0/1-mKP) เป็นการคัดเลือกสิ่งของจาก n จำนวน ลงถุงเป้ที่มีจำนวนมากกว่าหนึ่งถุงเป้ซึ่งมีขนาดความจุแตกต่างกัน โดยจะต้องได้มูลค่ารวมของสิ่งของที่ถูกเลือกจากถุงเป้ทั้งหมดมากที่สุด ซึ่งมีคุณสมบัติของปัญหา ได้ดังนี้ [30]

$$\sum_{i=1}^m \sum_{j=0}^{n-1} p_{ij} x_{ij} \quad (1.4)$$

$$\sum_{j=0}^{n-1} w_j x_{ij} \leq C_i; i = 1, 2, \dots, m; j = 0, 1, 2, \dots, n-1 \quad (1.5)$$

$$\sum_{i=1}^m x_{ij} \leq 1, x_{ij} \in \{0, 1\}; \forall i, j \quad (1.6)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ด้วยคุณลักษณะของปัญหาแนปแซคที่มุ่งเน้นการคัดเลือกสิ่งของที่ได้มูลค่ารวมในถุงเป๋ามากที่สุดในถุงเป๋ที่มีขนาดจำกัด จึงได้มีการนำเอาคุณสมบัติของปัญหานี้ไปประยุกต์ใช้ในหลากหลายด้าน อาทิ การพัฒนาประสิทธิภาพเกี่ยวกับระบบเครือข่าย เช่น การเพิ่มประสิทธิภาพการหน่วงเวลาสำหรับการรับส่งข้อมูลมัลติมีเดียผ่าน MANET [3] การจัดสรรทรัพยากรบนระบบกริด (Grid) [56] การนำไปใช้สำหรับการจัดการบัพเฟอร์ในระบบเครือข่ายที่ล่าช้า [57] สำหรับในด้านฐานข้อมูลได้มีการพัฒนาขั้นตอนวิธีร่วมกันระหว่างเครือข่ายของยีนส์กับปัญหาแนปแซคเพื่อรองรับการบันทึกและกระจายฐานข้อมูลในแต่ละกลุ่มข้อมูล [60] รวมถึงการรักรักษาความปลอดภัยความปลอดภัยบนระบบเครือข่ายโดยการซ่อนข้อมูลและลายเซ็นดิจิทัลประตูดของระบบรักษาความปลอดภัยหรือการเข้ารหัสข้อมูล (Encryption) [40] การจัดการบริการเกี่ยวกับทรัพยากรด้านซอฟต์แวร์ (SaaS) [4] ทั้งยังนำไปใช้ประยุกต์ในด้านธุรกิจและอุตสาหกรรมขนส่งโลจิสติก เช่น การจัดสรรสินค้าและปัญหาการขนส่งที่มีอุปสงค์แบบสุ่ม [38] การเลือกรุ่นของสินค้าที่จัดวางบนชั้นจำหน่ายสินค้า [6] นอกจากนี้งานที่เกี่ยวข้องกับระบบคอมพิวเตอร์และการขนส่งแล้ว ยังมี การควบคุมงบประมาณเพื่อป้องกันการลุกลามไม่ให้เกิดการก่อการจลาจล [8] การแก้ไขปัญหาเกี่ยวกับการวางแผนการผลิตในวางแผนในโรงหล่อขนาดเล็ก [13] การจัดการกลยุทธ์ของมหาวิทยาลัยเอกชนชั้นนำ [32] การจัดการความยั่งยืนเพื่อความสมดุลของทรัพยากรชายฝั่งทะเล [52]

จากการประยุกต์ใช้ขั้นตอนวิธีในข้างต้น เป็นขั้นตอนวิธีสำหรับแก้ปัญหาแนปแซคได้รับความนิยมและสามารถนำไปต่อยอดเพื่อสร้างสรรค์นวัตกรรมในหลายศาสตร์ได้ ดังนั้น จึงได้มีนักวิจัยได้นำเสนอผลงานการคิดค้นและพัฒนาขั้นตอนวิธีสำหรับแก้ปัญหากลุ่มแนปแซค [3][5][7][13][26][33][36][41][43][56][60][61] ปัญหากลุ่มซิงเกิลแนปแซค [8][10][11][12][14][15][23][29][31][34][35][44][46][48][49][47][50][51][55][62][64][65][66] รวมถึงปัญหากลุ่มมัลติเฟลแนปแซคและมัลติไดเมนชันนอลแนปแซค [6][16][17][19][20][21][25][32][37][38][39][52][58][63] ทั้งขั้นตอนวิธีในกลุ่มที่สามารถหาคำตอบที่ดีที่สุดได้ (Exact Algorithms) อาทิ ขั้นตอนวิธีที่มีการพัฒนาจากขั้นตอนวิธีบริวนซ์และบาวนด์ (BnB: Branch and Bound) เช่น ขั้นตอนการจัดการปริมาณของสิ่งของในส่วน Lower Bound ด้วยขั้นตอนวิธีแบ็คแทร็ก (Backtracking) [33] ขั้นตอนวิธีบริวนซ์และบาวนด์ซึ่งใช้พัฒนามัลติเฟลแนปแซค [19] การพัฒนาขั้นตอนวิธีบริวนซ์แอนด์บาวนด์ด้วยกระบวนการรีแลกซ์แอนด์คัท (Relax-and-Cut Procedure) เพื่อลดเวลาประมวลผล [15] การพัฒนาขั้นตอนวิธีโปรแกรมเชิงพลวัต (Dynamic Programming) [47] การพัฒนาขั้นตอนวิธีโปรแกรมเชิงพลวัต (Backward Reduced-state DP : BRDS) ทั้งกรณีหลายเงื่อนไข [49] กรณีเงื่อนไขเดียว [47] และกรณีที่ขนาดความจุของถุงเป๋ไม่แน่นอน (Robust Knapsack Problem) [41]

ขั้นตอนวิธีในกลุ่มที่สามารถหาคำตอบที่ดีที่สุดซึ่งสามารถนำไปใช้ประโยชน์ได้ (Approximate Algorithms) อาทิ ขั้นตอนวิธีแบบกลุ่ม (Swarm Algorithm) อาทิ การเพิ่มประสิทธิภาพฝูงอนุภาคไบนารี (Binary Partial Swarm) ที่ได้รับการดัดแปลง [7] รูปแบบการกระโดดของฝูงกบ (Shuffled Frog Leaping Algorithms) เพื่อเลือกแหล่งอาหาร [10] และขั้นตอนวิธีแบบกบกระโดดเพื่อสับเปลี่ยนฝูง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(Meta-Heuristic) แบบ Memetic [17] เพิ่มประสิทธิภาพอาณานิคมมด (Ant Colony) ที่ดีที่สุด ซึ่งใช้พัฒนากับซิงเกิลแนปแซคที่มีความคลุมเครือ [14] การศึกษาและพัฒนาขั้นตอนวิธีด้วยประสิทธิภาพที่เลียนแบบมาจากหาอาหารของฝูงแมลงวันผลไม้ (Fruit Fly Algorithm) [39] การศึกษาและพัฒนาขั้นตอนวิธีด้วยประสิทธิภาพที่เลียนแบบมาจากหาอาหารของฝูงลิงในหาจำฝูงและการย้ายถิ่นฐานตามแหล่งอาหาร (Monkey Algorithm) [65] การจัดทีมสำหรับการแข่งขันฟุตบอล [43] การเพิ่มประสิทธิภาพฝูงอนุภาคควอนตัมไฮบริดซึ่งเป็นการศึกษาพฤติกรรมทางสังคมของมนุษย์ร่วมกับปฏิกิริยาควอนตัม (Quantum Inspired Social Evolution Algorithm) [44] การตอบสนองของสารเคมีร่วมกับขั้นตอนวิธีวิวัฒนาการเพื่อแก้ปัญหาซิงเกิลแนปแซค [55] การใช้หลักการคำนวณแบบควอนตัมในการหาเส้นทางเพื่อการเลือกบรรจุสิ่งของลงถุงเป้ในปัญหาซิงเกิลแนปแซค [59] ขั้นตอนวิธีการที่เป็นการผสมผสาน (Hybrid Algorithms) ฟังก์ชันลาคองราจ (Lagrangian) ร่วมกับขั้นตอนวิธีแอนด์บาวนด์ซึ่งใช้พัฒนากับซิงเกิลแนปแซค [51]

นอกจากนี้ ได้มีการรวบรวมขั้นตอนวิธีอย่างหลากหลายต่าง ๆ ที่มีการเปรียบเทียบขั้นตอนวิธีเพื่อช่วยต่อการนำไปใช้ [61] และการพัฒนาขั้นตอนวิธีสำหรับปัญหาแนปแซคด้วยวิธีการใหม่ อาทิ ขั้นตอนวิธีที่ศึกษาและพัฒนาจากการเลียนแบบโครงสร้างของอะมิบา [62] ขั้นตอนวิธีที่ศึกษาและพัฒนาจากความชาญฉลาดของสายหอยในการแพร่ขยาย [63] การพัฒนาขั้นตอนวิธีใหม่ที่ใช้การประสานเสียงในภาพรวมของวงดนตรีในการค้นหาคำตอบที่ดีของปัญหา [66] แบบจำลองทางคณิตศาสตร์และวิธีการสลายตัว (Mathematical Models and Decomposition Methods) ซึ่งใช้พัฒนากับมัลติเพิลแนปแซค [16] ขั้นตอนวิธีบินแพ็คกิ่งสำหรับหลายคอนเทนเนอร์รวมถึงการควบคุมปัญหาการบรรจุเกิน [20] วิธีการแฉ่งนั้ด้วย Pseudo-gap แบบวนซ้ำสำหรับปัญหามัลติเพิลแนปแซคและมัลติไดเมนชันแนปแซค [21] ขั้นตอนวิธีการค้นหาที่ประสานกลมกลืน (Harmony Search Algorithm) [31][66] การคัดกรองแบบไม่เอนเอียงเพื่อลดขนาดและเวลาในการเลือกสิ่งของลงถุงเป้ [11]] การคำนวณค่าของ DNA เพื่อใช้ในปัญหาแนปแซค [26] การเพิ่มประสิทธิภาพการเลือกสิ่งของในกลุ่มย่อยของปัญหาซิงเกิลแนปแซค (Discounted 0/1-KP) [48]

ซึ่งจากการพัฒนาขั้นตอนวิธีการแก้ปัญหาในกลุ่มแนปแซค แบ่งได้เป็น 2 กลุ่มใหญ่ โดยมีวัตถุประสงค์แตกต่างกัน ดังนี้

1) กลุ่มขั้นตอนวิธีที่ได้คำตอบที่ดีที่สุด (Exact Algorithm) เป็นกลุ่มขั้นตอนวิธีที่สามารถหาคำตอบที่ดีที่สุด (Optimal Solution) ของปัญหาได้ ได้แก่ ขั้นตอนวิธีแอนด์บาวนด์ (Branch-and-Bound) ที่มีความซับซ้อนด้านเวลาที่ $O(2^n)$ ขั้นตอนวิธีแบบย้อนรอย (Backtracking Algorithm) ที่มีความซับซ้อนด้านเวลาที่ $O(2^n)$ และขั้นตอนวิธีโปรแกรมเชิงพลวัต (Dynamic Programming : DP) ที่มีความซับซ้อนด้านเวลาที่ $O(nC)$ ซึ่งเป็นขั้นตอนวิธีที่ดีที่สุดของกลุ่มนี้ โดยสามารถหาคำตอบที่ดีที่สุดได้เร็วเมื่อ n มีค่าน้อย ๆ ($n < 500$) เนื่องจากความจุของถุงเป้ (C) อาจจะมีค่ามาก ($C > n^2$)

2) กลุ่มขั้นตอนวิธีที่ได้คำตอบโดยประมาณ (Approximate Algorithm หรือ Heuristic Algorithm) เป็นกลุ่มขั้นตอนวิธีที่สามารถหาคำตอบได้รวดเร็วเพื่อรองรับปัญหาที่มีจำนวนสิ่งของ (n) เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มาก แต่คำตอบที่ได้อาจไม่เป็นคำตอบที่ดีที่สุดเป็นเพียงคำตอบที่ด้อยอมรับได้ โดยขั้นตอนวิธีในกลุ่มนี้ เน้นความเร็วในการประมวลผลที่เป็นโพลิโนเมียล (Polynomial Time) ซึ่งมีหลากหลายขั้นตอนวิธีที่ได้ผลลัพธ์ที่ดีขึ้น อาทิ ขั้นตอนวิธีเชิงพันธุกรรม (Genetic Algorithm : GA) ขั้นตอนวิธีการหาคำตอบที่เหมาะสมแบบกลุ่ม (Swarm Optimization Algorithm) รวมถึงขั้นตอนวิธีแบบผสม (Hybrid Algorithm) ซึ่งได้รับความนิยมและมีการศึกษาพัฒนาอย่างต่อเนื่องในปัจจุบัน เพื่อเพิ่มประสิทธิภาพด้านความถูกต้องของคำตอบให้ดีขึ้นด้วยเวลาประมวลผลที่รวดเร็ว

สำหรับขั้นตอนวิธีในวิทยานิพนธ์ฉบับนี้ ผู้วิจัยได้ให้ความสนใจเกี่ยวกับการคิดค้น ออกแบบ และพัฒนาขั้นตอนวิธีสำหรับแก้ปัญหาซิงเกิลแนปแซค (0/1-KP) และมัลติเพิลแนปแซค (0/1-mKP) ที่มุ่งเน้นการหาคำตอบที่ดีที่สุด (Optimal Solution) ไม่น้อยกว่าร้อยละ 99 จากชุดปัญหาทั้งหมดในเวลาโพลิโนเมียล (Polynomial Time) ด้วยองค์ประกอบของผลลัพธ์ที่สมบูรณ์ ที่ประกอบด้วยมูลค่ารวม (Total Profit: soltp) น้ำหนักรวม (Total Weight: soltw) และเซตดัชนีของสิ่งของที่ถูกเลือกบรรจุลงถุงเป้ (Solution X: solx) เพื่อประโยชน์ต่อการนำไปประยุกต์ใช้จริงในงานด้านต่าง ๆ ซึ่งจากความสนใจข้างต้นผู้วิจัยสามารถสรุปประเด็นความสนใจได้ดังนี้

- 1) ศึกษาธรรมชาติของข้อมูลในปัญหาซิงเกิลแนปแซคและปัญหามัลติเพิลแนปแซค
- 2) การลดขนาดของปัญหาอย่างมีประสิทธิภาพและเหมาะสมตามรูปแบบของปัญหา
- 3) การเพิ่มความเร็วในการประมวลผล

โดยมีสมมุติฐานงานวิจัย ดังนี้

1) การลดขนาดของปัญหาด้วยกระบวนการคัดกรองแบบไม่เอนเอียงที่มีประสิทธิภาพจะสามารถลดเวลาประมวลผลลงได้

2) การประยุกต์ใช้ขั้นตอนวิธีโปรแกรมเชิงพลวัตเพื่อหาคำตอบชุดสุดท้ายสำหรับกลุ่มปัญหาที่เหลือจากกระบวนการคัดกรองซึ่งมีจำนวนจำกัด จะทำให้เวลาประมวลผลโดยรวมเป็นโพลิโนเมียล และได้คำตอบโดยรวมเป็นคำตอบที่ดีที่สุด หรือใกล้เคียงคำตอบที่ดีที่สุดมากที่สุดได้

จากประเด็นความสนใจและสมมุติฐานงานวิจัยข้างต้น ผู้วิจัยได้เปรียบเทียบประสิทธิภาพของขั้นตอนวิธีที่นำเสนอในด้านความถูกต้องของคำตอบ (Confidentiality) กับขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ ซึ่งมีองค์ประกอบของผลลัพธ์ทั้งมูลค่ารวม (Total Profit : tp) และเซตของดัชนีสิ่งของที่ถูกเลือก (Solution X : Solx) และด้านเวลาในการประมวลผล (Response Time) กับขั้นตอนวิธีอิวิริสติกบนมาตรฐานภาษาซี (C Language) และมาตรฐานภาษาเอ็มพีไอ (MPI Standard) สำหรับการพัฒนาขั้นตอนวิธีแบบขนานบนระบบมัลติคอร์ (Multi-Core Computer)

1.2 วัตถุประสงค์ของงานวิจัย

งานวิจัยฉบับนี้เป็นการศึกษาและออกแบบขั้นตอนวิธีสำหรับปัญหาซิงเกิลแนปแซค (0/1-KP) และปัญหามัลติเพิลแนปแซค (0/1-mKP) โดยได้นำเสนอขั้นตอนวิธีใหม่ที่มีประสิทธิภาพทั้งด้านความถูกต้องของคำตอบและด้านความเร็วในการประมวลผล ซึ่งมีรายละเอียดดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- 1) เพื่อออกแบบกระบวนการคัดกรองสิ่งของที่มีคุณสมบัติแตกต่างกันมาก ทำให้สามารถลดขนาดของปัญหาขนาดใหญ่ (n) ลงเหลือปัญหาขนาดเล็ก (n') และยังสามารถลดเวลาประมวลผลรวมได้อย่างมีประสิทธิภาพ
- 2) เพื่อประยุกต์และแปลงรูปขั้นตอนวิธีโปรแกรมเชิงพลวัต เพื่อหาคำตอบชุดสุดท้ายสำหรับกลุ่มปัญหาที่เหลือจากกระบวนการคัดกรอง (ขนาด n')
- 3) เพื่อออกแบบขั้นตอนวิธีสำหรับปัญหาซิงเกิลแนปแซคและปัญหามัลติเพิลแนปแซค ที่สามารถได้คำตอบที่ดีที่สุด (Optimal Solution) อย่างน้อยร้อยละ 99 ของชุดปัญหาทั้งหมด พร้อมองค์ประกอบของผลลัพธ์ที่สมบูรณ์ ในเวลาโพลิโนเมียล (Polynomial Time)

1.3 ขอบเขตของงานวิจัย

งานวิจัยฉบับนี้เป็นการศึกษาและออกแบบคิดค้นขั้นตอนวิธีสำหรับปัญหาซิงเกิลแนปแซคและปัญหามัลติเพิลแนปแซค ซึ่งมีขอบเขตของงานวิจัย ดังนี้

- 1) งานวิจัยนี้เป็นการนำเสนอนวัตกรรมด้านขั้นตอนวิธีทางคอมพิวเตอร์ที่เป็นการบูรณาการขั้นตอนวิธีที่มีประสิทธิภาพร่วมกับขั้นตอนวิธีการคัดกรองข้อมูลแบบไม่เอนเอียงที่คิดค้นขึ้นเพื่อการแก้ปัญหาซิงเกิลแนปแซคและปัญหามัลติเพิลแนปแซค
- 2) การทดสอบประสิทธิภาพของขั้นตอนวิธีที่นำเสนอในด้านความเร็ว ได้เปรียบเทียบกับขั้นตอนวิธีฮิวริสติก สำหรับการทดสอบประสิทธิภาพด้านความถูกต้องของคำตอบได้เปรียบเทียบกับขั้นตอนวิธีโปรแกรมเชิงพลวัต

1.4 ประโยชน์ที่ได้รับ

ประโยชน์ที่ได้รับจากงานวิจัยนี้ เมื่อสามารถคัดกรองได้อย่างเป็นธรรมและถูกต้อง ได้แก่

- 1) ได้ขั้นตอนวิธีสำหรับการแก้ปัญหาซิงเกิลแนปแซค (เพื่อรองรับ n ขนาดใหญ่) และปัญหามัลติเพิลแนปแซค ที่สามารถได้คำตอบที่ดีที่สุด (Optimal Solution) อย่างน้อยร้อยละ 99 จากชุดปัญหาทั้งหมด ในเวลาโพลิโนเมียล (Polynomial Time)
- 2) ได้กระบวนการคัดกรองแบบไม่เอนเอียงและแข็งแกร่งเพื่อลดขนาดของปัญหาและเวลาประมวลผลได้อย่างมีประสิทธิภาพ และสามารถใช้เป็นต้นแบบเพื่อแก้ปัญหาเอ็นพีแบบยากชนิดอื่น ๆ ได้เร็วขึ้นและถูกต้องมากขึ้น
- 3) ได้ขั้นตอนวิธีใหม่ที่แปลงรูปมาจากขั้นตอนวิธีโปรแกรมเชิงพลวัตที่สามารถหาคำตอบที่ดีที่สุด เช่นเดียวกับขั้นตอนวิธีโปรแกรมเชิงพลวัต กรณีข้อมูลขนาดใหญ่ ($n \leq 10000$) ได้เร็วขึ้น และประยุกต์ใช้หลังจากการคัดกรองกรณี $n > 10000$ ได้อย่างมีประสิทธิภาพ

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

เนื้อหาในบทนี้เป็นการนำเสนอผลสรุปการศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง ที่ผู้วิจัยได้ศึกษาค้นคว้าเพื่อประกอบการคิดค้นออกแบบและพัฒนาขั้นตอนวิธีทางคอมพิวเตอร์สำหรับการแก้ปัญหาแนปแซค (Knapsack) โดยทฤษฎีที่ผู้วิจัยได้ศึกษา ได้แก่ ขั้นตอนวิธีกริดีฮิวริสติก (Greedy Heuristic Algorithm) ขั้นตอนวิธีโปรแกรมเชิงพลวัต (Dynamic Programming) การหาผลรวมพรีฟิกซ์แบบขนาน (Parallel Prefix Sum) และแผนการทดลองแบบลาตินสแควร์ (Latin Square Design: LS) สำหรับในวิทยานิพนธ์ฉบับนี้มีงานวิจัยที่เกี่ยวข้อง ได้แก่ การพัฒนาขั้นตอนวิธีกริดีฮิวริสติกด้วยการคาดการณ์อย่างมีประสิทธิภาพ (Greedy with Dynamic Expectation Efficiency : GDEE) การหาคำตอบที่เหมาะสมแบบกลุ่มด้วยวิธีชัฟเฟิลฟรอก (Shuffled Frog Swarm Optimization) และการเรียงลำดับข้อมูลแบบขนานด้วยขั้นตอนวิธีโอบีเอส (Parallel Sorting by OBS Algorithm) ขั้นตอนวิธีการประเมินที่แตกต่างตามแนวทางควอนตัมเพื่อการแก้ปัญหาซิงเกิลแนปแซค (0/1-KP Solving by Quantum-Inspired Differential Evolution Algorithm) และขั้นตอนวิธีแบบจำลองทางคณิตศาสตร์สำหรับปัญหามัลติเพิลแนปแซค (Mathematical HyMKP)

2.1 ขั้นตอนวิธีกริดีฮิวริสติก (Greedy Heuristic Algorithm)

ขั้นตอนวิธีฮิวริสติก (Heuristic Algorithm) เป็นขั้นตอนวิธีหนึ่งที่สามารถหาคำตอบที่ดีที่สุดหรือคำตอบที่ยอมรับได้ในเวลาที่รวดเร็ว ทำให้ขั้นตอนวิธีฮิวริสติกได้รับความนิยมในการศึกษาและพัฒนา ซึ่งหนึ่งในขั้นตอนวิธีฮิวริสติกที่ได้รับความนิยมนำไปพัฒนาร่วมกับขั้นตอนวิธีอื่นเพื่อเพิ่มประสิทธิภาพของผลลัพธ์ให้ดีขึ้น คือ ขั้นตอนวิธีกริดีฮิวริสติก (Greedy Heuristic Algorithm : GH)

ขั้นตอนวิธีฮิวริสติกแบบนี้ เป็นวิธีการหาคำตอบในเวลารวดเร็วด้วยการเลือกสมาชิกของปัญหาตามการเรียงลำดับคุณสมบัติของปัญหา สำหรับการแก้ปัญหาแนปแซคได้ใช้การเรียงลำดับสิ่งของตามมูลค่าและน้ำหนักซึ่งเป็นคุณสมบัติของปัญหาถุงเป้ จากนั้นจึงบรรจุสิ่งของเหล่านั้นลงถุงเป้ทีละชิ้นตามลำดับที่เรียงไว้ โดยที่น้ำหนักรวมของสิ่งของที่บรรจุลงถุงเป้จะต้องไม่เกินน้ำหนักที่ถุงเป้สามารถรองรับได้ (C) ซึ่งขั้นตอนวิธีนี้มีความซับซ้อนด้านเวลาที่ $O(n)$

ขั้นตอนวิธีกริดีฮิวริสติกสำหรับการแก้ปัญหาแนปแซค มีขั้นตอนโดยสรุป ดังนี้

ขั้นที่ 1 คำนวณค่าสัดส่วน P/W ของสิ่งของแต่ละชิ้น

ขั้นที่ 2 เรียงลำดับสิ่งของทั้งหมดตามค่า P/W จากมากไปน้อย

ขั้นที่ 3 บรรจุสิ่งของลงถุงเป้ทีละชิ้นตามลำดับที่ได้จัดเรียงไว้แล้ว จนกว่าจะเต็มถุงเป้

ในขั้นตอนการเลือกบรรจุสิ่งของลงถุงเป้ สิ่งของจะถูกกำหนดค่าสถานะการเลือก 2 สถานะ

คือ สถานะที่ถูกเลือก (1) กับสถานะที่ไม่ถูกเลือก (0)

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อใช้ในการเรียนการสอนเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนวิธีกริดฮิวริสติกข้างต้น สามารถนำไปประยุกต์ใช้ได้กับปัญหาอื่น ด้วยการกำหนดเงื่อนไขให้เหมาะสมกับคุณสมบัติของปัญหา สำหรับปัญหาถุงเป้สามารถประยุกต์ได้ตามคุณสมบัติของสิ่งของที่ประกอบด้วยมูลค่า (P) และน้ำหนัก (W) ซึ่งแตกต่างกันและไม่สัมพันธ์กัน จึงได้มีการพัฒนาขั้นตอนวิธีนี้ด้วยการเรียงลำดับสิ่งของตามคุณสมบัติใน 3 รูปแบบ ได้แก่ การเรียงลำดับตามค่า P/W จากมากไปน้อย (Descending Order by P/W) การเรียงลำดับตามค่า P จากมากไปน้อย (Descending order by P) และการเรียงลำดับตามค่า W จากน้อยไปมาก (Ascending order by W) [1] ซึ่งผลลัพธ์ที่ได้แต่ละรูปแบบอาจจะเหมือนกันหรือต่างกันก็ได้ และเมื่อนำมาเปรียบเทียบกับกันจะทำให้ได้คำตอบที่ดีที่สุดจากทั้ง 3 รูปแบบ ดังแสดงตัวอย่างในรูปที่ 2.1

X	1	2	3	4	5	6						
W	10	11	11	8	10	9	n = 6 C = 18					
P	42	39	38	37	35	38						

Greedy Heuristic 1 (GH1)
(Decreasing order by P/W)

Y	1	2	3	4	5	6
X	4	6	1	2	5	3
W	8	9	10	11	10	11
P	37	38	42	39	35	38
P/W	4.6	4.2	4.2	3.5	3.5	3.5
	1	1	0	0	0	0
tw1	= 17					
tp1	= 75					
Not Optimal						

Greedy Heuristic 2 (GH2)
(Decreasing order by P)

Y	1	2	3	4	5	6
X	1	2	3	6	4	5
W	10	11	11	9	8	10
P	42	39	38	38	37	35
	1	0	0	0	0	0
tw2	= 10					
tp2	= 42					
Not Optimal						

Greedy Heuristic 3 (GH3)
(Increasing order by W)

Y	1	2	3	4	5	6
X	4	6	1	5	2	3
W	8	9	10	10	11	11
P	37	38	42	35	39	38
	1	1	0	0	0	0
tw3	= 17					
tp3	= 75					
Not Optimal						

Total Weight = 17

Total Profit = 75

Solution Set = {4, 6}

รูปที่ 2.1 ตัวอย่างการแก้ปัญหาถุงเป้ด้วยขั้นตอนวิธีกริดฮิวริสติก

ตัวอย่างในรูปที่ 2.1 คำตอบที่ดีที่สุด (Optimal Solution) ของชุดข้อมูลนี้ คือ soltw = 18 soltp = 79 และ solx = {1, 4} (ดังแสดงสิ่งของที่เป็นคำตอบด้วยแถบสี) ซึ่งจากตัวอย่างจะเห็นว่าคำตอบของกฎที่ 1 (P/W ranking) และกฎที่ 3 (W ranking) ได้ค่า tp = 75 เท่ากัน จึงทำให้ได้คำตอบโดยรวมเป็น soltw = 17 และ soltp = 75 จากการประยุกต์ขั้นตอนวิธีกริดฮิวริสติกนี้ อาจจะทำให้ได้คำตอบที่ดีขึ้นกับหลายชุดปัญหา แต่จากชุดปัญหาในตัวอย่างจะเห็นว่าวิธีนี้ยังไม่สามารถหาคำตอบที่ดีที่สุดได้

2.2 ขั้นตอนวิธีโปรแกรมเชิงพลวัต (Dynamic Programming Algorithm)

ขั้นตอนวิธีโปรแกรมเชิงพลวัต (Dynamic Programming : DP) เป็นขั้นตอนวิธีในกลุ่มที่สามารถหาคำตอบที่ดีที่สุดได้ ซึ่งนิยมนำไปใช้ในการแก้ปัญหาทางคอมพิวเตอร์หลายปัญหาในกลุ่มปัญหาเชิงเกิลแนปแซค วิธีการนี้เป็นการนำสิ่งของทุกชิ้นของปัญหาเข้าสู่กระบวนการประมวลผลทีละชิ้นเพื่อคำนวณและเปรียบเทียบเงื่อนไขหามูลค่ารวมที่มากที่สุดในแต่ละรอบของการประมวลผลตามขนาด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ของความจุสูงสุดของถุงเป้ จากการศึกษาสามารถสรุปขั้นตอนวิธีโปรแกรมเชิงพลวัตสำหรับปัญหาชิงเกลแนปแซค (0/1-KP) ได้ 3 วิธี [30][36] ได้แก่

- 1) ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื้องต้น (Basic-DP Algorithm)
- 2) ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็ว (Fast-DP Algorithm)
- 3) ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ (Complete-DP Algorithm)

2.2.1 ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื้องต้น (Basic-DP Algorithm)

ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื้องต้น หรือขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบดั้งเดิม เป็นขั้นตอนวิธีที่ใช้เนื้อที่ในการประมวลผลด้วยอาร์เรย์ 2 มิติ ที่มีขนาด $n \times C$ (ดังตัวอย่างในรูปที่ 2.2) โดยนำค่าน้ำหนัก (P) และมูลค่า (W) เข้าสู่กระบวนการคำนวณและเปรียบเทียบเงื่อนไขทีละชิ้น ซึ่งหากสิ่งของใดบรรจุลงถุงเป้แล้วได้มูลค่ารวม (tp) มากกว่าเดิมและน้ำหนักรวม (tw) ไม่เกินค่า C สิ่งของนั้นจะถูกเลือกเป็นคำตอบ แต่หากได้มูลค่ารวมน้อยกว่าค่าเดิม ให้ข้ามไปพิจารณาสิ่งของชิ้นถัดไป ประมวลผลเช่นนี้ไปเรื่อย ๆ จนถึงสมาชิกชิ้นสุดท้าย (ดังกระบวนการในขั้นตอนวิธีที่ 2.1) องค์ประกอบของผลลัพธ์ที่ได้จากการวิธีการนี้ประกอบด้วย $solt_p$, $solt_w$ และ $solt_x$ ด้วยความซับซ้อนด้านเวลาที่ $O(nC)$

ขั้นตอนวิธีที่ 2.1 ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื้องต้น (Basic-DP Algorithm)

Basic-DP Algorithm

```

for (d=0 to C) do
    tp[0][d] ← 0
for (j=1 to n) do
    for (d=0 to W-1) do
        tp[j][d] ← tp[j-1][d]
    for (d=Wj to C) do
        if (tp[j-1][d-Wj]+Pj > tp[j-1][d]) then
            tp[j][d] ← tp[j-1][d-Wj]+Pj
        else
            tp[j][d] ← tp[j-1][d]
        end for d
    end for j
soltp ← tp[n][C]
d ← C; j ← n; soltx ← ∅
do
    while (tp[j][d]=tp[j-1][d])
        j ← j-1
    soltx ← soltx U {j}
    pp ← tp[j][d]-Pj
    j ← j-1
    while (tp[j][d-1] ≥ pp AND pp > 0 )
        d ← d-1
    while(pp > 0 AND j ≥ 1)

```

d	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
j	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42
2	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42
3	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42
4	0	0	0	0	0	0	0	0	37	37	42	42	42	42	42	42	42	42	79
5	0	0	0	0	0	0	0	0	37	37	42	42	42	42	42	42	42	42	79
6	0	0	0	0	0	0	0	0	37	38	42	42	42	42	42	42	42	75	79

รูปที่ 2.2 ตัวอย่างการประมวลผลด้วยขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื่องต้น

2.2.2 ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็ว (Fast-DP Algorithm)

ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็ว เป็นขั้นตอนวิธีที่พัฒนาจากขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื่องต้นที่มีกระบวนการทำงานเช่นเดียวกัน แต่ได้มีการลดขนาดเนื้อที่ในการประมวลผลลงด้วยการใช้อาร์เรย์ 1 มิติขนาด C คอลัมน์ จึงสามารถเรียกวิธีนี้ในอีกชื่อหนึ่งว่า ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบอาร์เรย์เดี่ยว (Single Array-DP Algorithm) [30][36] การประมวลผลของขั้นตอนวิธีนี้ใช้วิธีการแทนที่ของมูลค่ารวม (tp) ใหม่ที่ดีกว่าในแต่ละคอลัมน์ (ขั้นตอนวิธีที่ 2.2 และรูปที่ 2.3) จุดประสงค์ของการพัฒนาขั้นตอนวิธีนี้ คือ มุ่งเน้นการลดขนาดของเนื้อที่ที่ใช้ในการประมวลผล แต่ยังคงประกอบของผลลัพธ์ที่ได้จากวิธีการนี้มีเพียงค่าผลลัพธ์มูลค่ารวม (soltp) เท่านั้น เนื่องจากไม่มีกระบวนการแบ็คแทร็ค (Backtracking) เพื่อค้นหาชิ้นของสิ่งของ (x_j) ที่ถูกเลือกเป็นคำตอบ ดังนั้นจึงไม่มีองค์ประกอบของผลลัพธ์ในส่วน ค่าผลลัพธ์น้ำหนักรวม (soltw) และผลลัพธ์เซตดัชนีสิ่งของที่ถูกเลือก (solx) สำหรับเวลาในการประมวลผล ด้วยขั้นตอนวิธีนี้มีกระบวนการเช่นเดียวกับขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื่องต้นจึงทำให้ขั้นตอนวิธีนี้มีความซับซ้อนด้านเวลาที่ $O(nC)$

ขั้นตอนวิธีที่ 2.2 ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็ว (Fast-DP Algorithm)

Fast-DP Algorithm

```

for (d=0 to C)
    tp[d] ← 0
for (j=1 to n) do
    for (d=C down to Wj) do
        if (tp[d-Wj]+Pj > tp[d]) then
            tp[d] ← tp[d-Wj]+Pj
    end for d
end for j
soltp ← tp[C]
```

d	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
j=1	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42
j=2	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42
j=3	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42
j=4	0	0	0	0	0	0	0	0	37	37	42	42	42	42	42	42	42	42	79
j=5	0	0	0	0	0	0	0	0	37	37	42	42	42	42	42	42	42	42	79
j=6	0	0	0	0	0	0	0	0	37	38	42	42	42	42	42	42	42	75	79

รูปที่ 2.3 ตัวอย่างการประมวลผลด้วยขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.3 ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ (Complete-DP Algorithm)

ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์พัฒนามาจากขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็วที่ใช้อาร์เรย์ 1 มิติในการประมวลผล รวมถึงการคำนวณ เปรียบเทียบ และแทนค่าเช่นเดียวกับขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื้องต้น โดยจะมีการประมวลผลซ้ำ คือ ได้ปรับให้มีการลดขนาดของปัญหาในรอบที่สองเป็นต้นไป ซึ่งลดจำนวนสมาชิกของปัญหาและความจุของถุงแปลงตามสิ่งของที่ถูกละเลือกในรอบก่อนหน้า อีกทั้ง ได้เพิ่มอาร์เรย์ 1 มิติสำหรับเก็บดัชนีของสิ่งของล่าสุดที่ถูกละเลือกในแต่ละคอลัมน์ (ดังแสดงในขั้นตอนวิธีที่ 2.3 และตัวอย่างในรูปที่ 2.4) เพื่อลดขนาดเนื้อที่ที่ใช้ในการประมวลผล และคงความสมบูรณ์ขององค์ประกอบผลลัพธ์ (soltp, soltw, solx)

วิธีการนี้ใช้หลักการพื้นฐานจากขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็วที่ใช้อาร์เรย์ 1 มิติสำหรับเก็บค่า tp และเพิ่มอาร์เรย์ 1 มิติอีกหนึ่งชุดเพื่อเก็บค่า x_j ที่ถูกละเลือก ซึ่งจะเปลี่ยนแปลงสอดคล้องกันตลอดการประมวลผล จึงเรียกวิธีการนี้อีกชื่อหนึ่งว่า ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบโต้ตอบ (Interactive-DP Algorithm) [30][36] ซึ่งมีความซับซ้อนด้านเวลาที่ $O(knC)$ หรือ $O(n^2C)$ เมื่อ k คือจำนวนสิ่งของที่ถูกเลือก

หลักการการทำงานของขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ สรุปได้ดังนี้

1. เพิ่มอาร์เรย์ 1 มิติสำหรับเก็บดัชนีของสิ่งของที่ถูกเลือกใหม่ควบคู่กับมูลค่ารวมใหม่ในแต่ละ คอลัมน์น้ำหนักร
2. ประมวลผลเพื่อหามูลค่ารวมด้วยการแทนค่า tp ใหม่ที่ดีกว่า และค่า x_j ของสิ่งของที่ถูกเลือกในแต่ละคอลัมน์ตั้งแต่ C จนถึง w_j ของแต่ละ x_j ($1 \leq j \leq n$) ดังนั้น ในรอบแรกของการประมวลผล จะได้ soltp และ x_j ส่วนแรก โดยขนาดความจุถุงเป้และสมาชิกที่เหลือที่จะถูกนำไปประมวลผลซ้ำเพื่อหา x_j ส่วนที่เหลือในรอบถัดไป จะได้เป็น $C' = C - w_j$ และ $n' = n - x_j - 1$
3. ในรอบที่ 2 เป็นต้นไป ประมวลผลเช่นเดียวกับขั้นตอนที่ 2 จนกระทั่ง $C' \leq 0$

ขั้นตอนวิธีที่ 2.3 ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ (Complete-DP Algorithm)

Complete-DP Algorithm

```

solx  $\leftarrow \emptyset$ ;  $C' \leftarrow 0$ ;  $n' \leftarrow n$ 
do
  for ( $d=0$  to  $C'$ ) do
     $tp[d] \leftarrow 0$ ;  $x[d] \leftarrow 1$ 
  for ( $j=1$  to  $n'$ ) do
    for ( $d=C'$  down to  $w_j$ ) do
      if ( $tp[d-w_j]+P_j > tp[d]$ ) then
         $x[d] \leftarrow j$ 
         $tp[d] \leftarrow tp[d-w_j]+P_j$ 
    end for d
  end for j
   $r \leftarrow x[C']$ 
   $solx \leftarrow solx \cup \{r\}$ 
   $k \leftarrow k+1$ ;  $n' \leftarrow n-k$ 
   $C' \leftarrow C - w_r$ 
  while( $C' > 0$ )
    soltp  $\leftarrow tp[C]$ 

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้เฉพาะเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้แก้ไขโดยไม่ได้รับอนุญาต

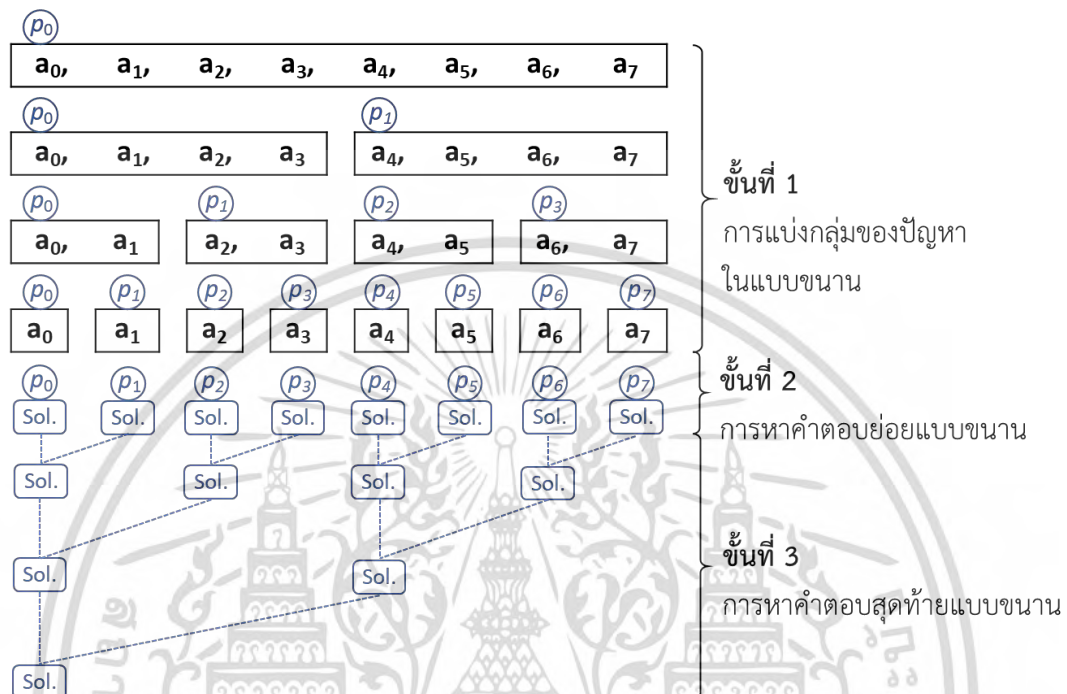
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในรูปที่ 2.4 แสดงตัวอย่างการประมวลผลของขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื่องต้นแบบเร็ว และแบบสมบูรณ์ จะเห็นว่าทั้งสามรูปแบบสามารถหาคำตอบที่ดีที่สุดได้ คือ $solt_p = 79$ $solt_w = 18$ และ $solt_x = \{1, 4\}$ ซึ่งจากการศึกษาพบว่า ขั้นตอนวิธีโปรแกรมเชิงพลวัตสามารถแบ่งตามรูปแบบการใช้เนื้อที่ที่เก็บข้อมูลระหว่างประมวลผล ได้ 2 แบบ คือ

1. ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบพื้นฐาน (Basic-DP) ใช้เนื้อที่แบบ 2 มิติ ขนาด $n \times C$ ที่มีความซับซ้อนด้านเวลา $O(nC)$
2. ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบลดเนื้อที่ (Reduced-DP) ใช้เนื้อที่แบบ 1 มิติ ขนาด C แต่มีความซับซ้อนด้านเวลาเพิ่มขึ้นเป็น $O(n^2C)$ ซึ่งในทางปฏิบัติ ขั้นตอนวิธีโปรแกรมเชิงพลวัตทั้ง 2 แบบ ไม่เหมาะสำหรับการประยุกต์ใช้ เมื่อ C มีค่ามาก ๆ โดยเฉพาะเมื่อ $C > n^2$ แต่มีประโยชน์ต่อการพัฒนาขั้นตอนวิธีทางคอมพิวเตอร์เพื่อแก้ปัญหาในหลายปัญหา

C'=18		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18				
initial	tp	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
	r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
j=1	tp	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42				
	r	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1				
j=2	tp	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42				
	r	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1				
j=3	tp	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42				
	r	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1				
j=4	tp	0	0	0	0	0	0	0	0	0	37	37	42	42	42	42	42	42	42	79				
	r	0	0	0	0	0	0	0	0	0	4	4	1	1	1	1	1	1	1	4				
j=5	tp	0	0	0	0	0	0	0	0	0	37	37	42	42	42	42	42	42	42	79				
	r	0	0	0	0	0	0	0	0	0	4	4	1	1	1	1	1	1	1	4				
j=6	tp	0	0	0	0	0	0	0	0	0	37	38	42	42	42	42	42	42	75	79				
	r	0	0	0	0	0	0	0	0	0	4	6	6	5	5	5	5	5	6	4				
tp[C]=79 solx={4}																								
C'=10		0	1	2	3	4	5	6	7	8	9	10												
initial	tp	0	0	0	0	0	0	0	0	0	0	0												
	r	0	0	0	0	0	0	0	0	0	0	0												
j=1	tp	0	0	0	0	0	0	0	0	0	0	42												
	r	0	0	0	0	0	0	0	0	0	0	1												
j=2	tp	0	0	0	0	0	0	0	0	0	0	42												
	r	0	0	0	0	0	0	0	0	0	0	1												
j=3	tp	0	0	0	0	0	0	0	0	0	0	42												
	r	0	0	0	0	0	0	0	0	0	0	1	solx={4, 1}											

ขั้นที่ 3 แต่ละหน่วยประมวลผลติดต่อสื่อสาร (Communication) เพื่อส่งคำตอบของกลุ่มย่อยที่ประมวลผลได้ให้กับหน่วยประมวลผลอื่น (ดังแสดงในรูปที่ 2.5 ขั้นที่ 3) จากนั้นแต่ละหน่วยประมวลผลที่ได้รับคำตอบจะทำการประมวลผลต่อจนกระทั่งได้คำตอบสุดท้าย



รูปที่ 2.5 เทคนิคดีไวด์แอนด์คอนเคอร์ (Divide-and-Conquer)

รูปที่ 2.5 แสดงรูปแบบการแบ่งสมาชิกของปัญหาและการติดต่อสื่อสารเพื่อการประมวลผลของเทคนิคดีไวด์แอนด์คอนเคอร์สำหรับปัญหขนาด $n = 8$ และ $p = 8$ ประกอบด้วย 3 ขั้นตอน ได้แก่ 1) แบ่งสมาชิกของปัญหาให้มีขนาดเล็กที่สุด 2) ประมวลผลเพื่อหาคำตอบพร้อมกันทุกหน่วยประมวลผล และ 3) ติดต่อสื่อสารเพื่อนำคำตอบที่ได้ในแต่ละหน่วยประมวลผลมาประมวลผลร่วมกันเพื่อให้ได้คำตอบโดยรวมของปัญหา

ขั้นตอนวิธีที่ 2.4 ขั้นตอนวิธีการหาผลรวมพรีฟิกซ์แบบขนาน (Parallel Prefix Sum Algorithm)

Parallel_Prefix_Sum

Begin

for all p_i ($0 \leq i \leq n-1$)

for $j = 0$ to $\lceil \log n \rceil - 1$ do

if $(i - 2^j \geq 0)$

$A[i] = A[i] + A[i - 2^j]$

end if

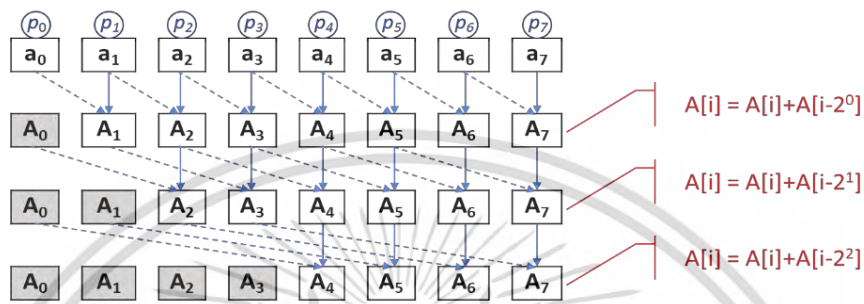
end for j

end for all

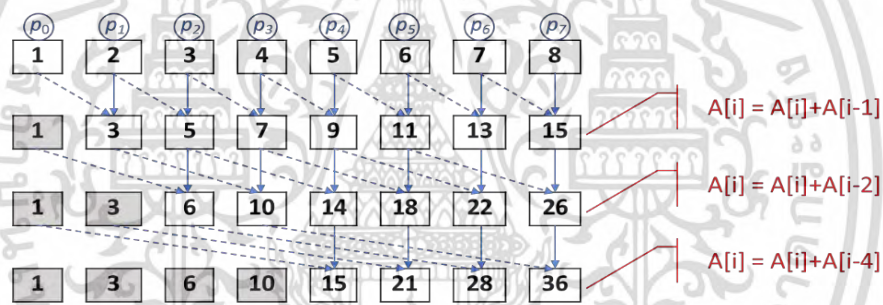
End

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เทคนิคดีไวด์แอนด์คอนเคอร์เป็นแนวคิดในการพัฒนาขั้นตอนวิธีการหาผลรวมพรีฟิกซ์แบบขนาน ซึ่งสามารถเข้าถึงข้อมูลขนาด n สมาชิก โดยใช้หน่วยความจำร่วมแบบ CREW-PRAM (Concurrent-Read Exclusive-Write Parallel RAM) ที่มีจำนวนของหน่วยประมวลผลที่ $p = n$ ซึ่งมีความซับซ้อนด้านเวลาที่ $O(\log_2 n)$ ดังในขั้นตอนวิธีที่ 2.4 และแผนภาพแสดงการทำงานในรูปที่ 2.6 พร้อมตัวอย่างในรูปที่ 2.7



รูปที่ 2.6 ขั้นตอนวิธีการหาผลรวมพรีฟิกซ์แบบขนาน (Parallel Prefix Sum)



รูปที่ 2.7 ตัวอย่างของขั้นตอนวิธีการหาผลรวมพรีฟิกซ์แบบขนาน

2.4 การเรียงลำดับข้อมูลแบบขนานด้วยขั้นตอนวิธีโอบีเอส (Parallel Sorting by OBS Algorithm)

การเรียงลำดับข้อมูลแบบขนานด้วยขั้นตอนวิธีโอบีเอส (Parallel Sorting by OBS Algorithm) [54] เป็นการเรียงลำดับข้อมูลแบบขนานโดยอาศัยค่ากลางหรือเรียกว่า ค่ามิดพอยท์ (Mid-Point) เพื่อเป็นตัวแทนของชุดข้อมูลกลุ่มย่อย ร่วมกับการประยุกต์กระบวนการเรียงลำดับข้อมูลแบบไบโทนิค (Bitonic Sort) ในส่วนของการเรียงลำดับตัวแทนชุดข้อมูล ดังแสดงในรูปที่ 2.8

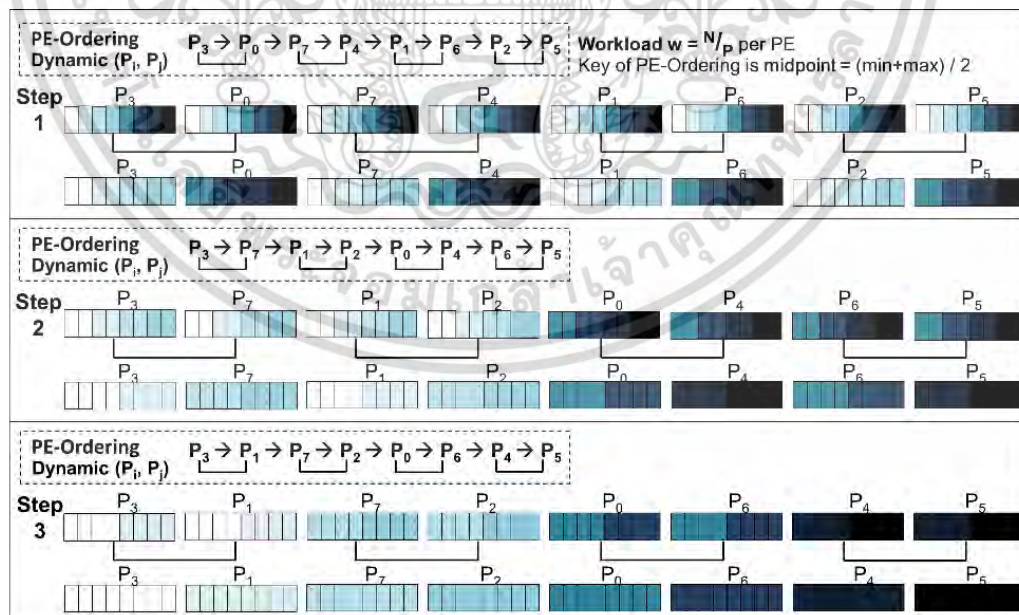
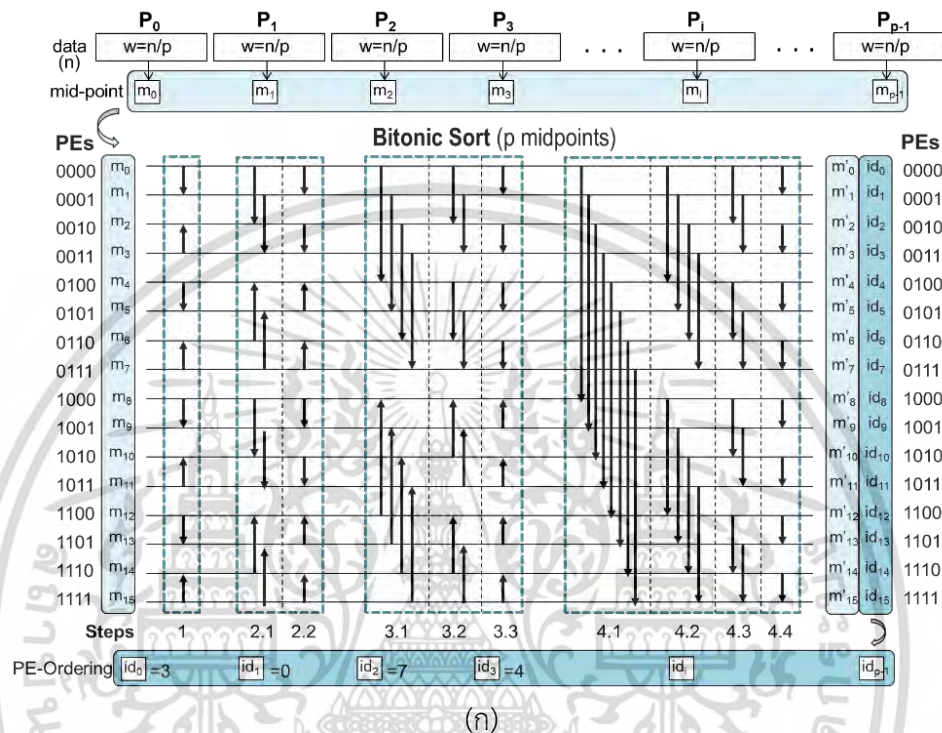
ขั้นตอนวิธีการเรียงลำดับข้อมูลแบบโอบีเอส มีขั้นตอนโดยสรุป ดังนี้

ขั้นที่ 1 จัดสรรข้อมูลให้แต่ละหน่วยประมวลผลอย่างเท่าเทียมกัน (Workload : $w = n/p$)

ขั้นที่ 2 แต่ละหน่วยประมวลผลเรียงลำดับข้อมูลด้วยขั้นตอนวิธีเรียงลำดับแบบเร็ว (Quick Sort) พร้อมกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ขั้นที่ 3 แต่ละหน่วยประมวลผลเก็บค่าน้อย (min) ค่ามาก (max) ดัชนีของหน่วยประมวลผล (id) และค่ามิดพอยท์ (mid)
- ขั้นที่ 4 แต่ละหน่วยประมวลผลจะทำการแลกเปลี่ยนข้อมูลซึ่งกันและกันเพื่อเรียงลำดับข้อมูลในแบบไบนารี

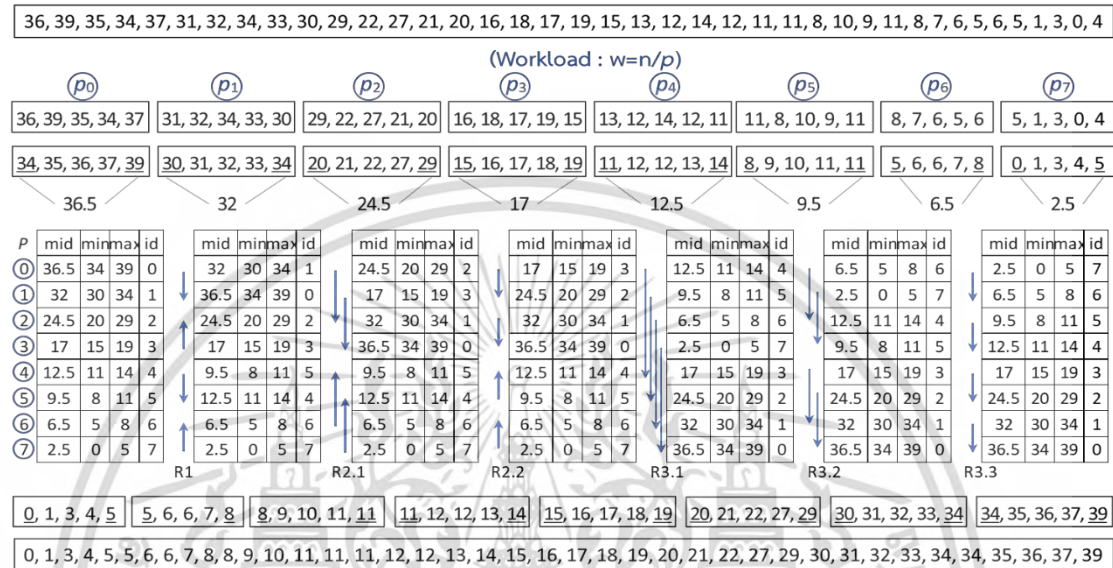


(ก)

รูปที่ 2.8 การเรียงลำดับข้อมูลแบบขนานด้วยขั้นตอนวิธีโอบีเอส (Parallel Sorting by OBS Algorithm)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 2.9 แสดงตัวอย่างการเรียงลำดับข้อมูลด้วยขั้นตอนวิธีการเรียงลำดับข้อมูลแบบโอบีเอส สำหรับชุดปัญหาขนาด $n = 40$ ประกอบด้วยค่า 36, 39, 35, 34, 37, 31, 32, 34, 33, 30, 29, 22, 27, 21, 20, 16, 18, 17, 19, 15, 13, 12, 14, 12, 11, 11, 8, 10, 9, 11, 8, 7, 6, 5, 6, 5, 1, 3, 0, 4 ด้วยจำนวนหน่วยประมวลผลที่ $p = 8$



รูปที่ 2.9 ตัวอย่างการเรียงลำดับข้อมูลแบบโอบีเอสด้วยค่ามิดพอยท์

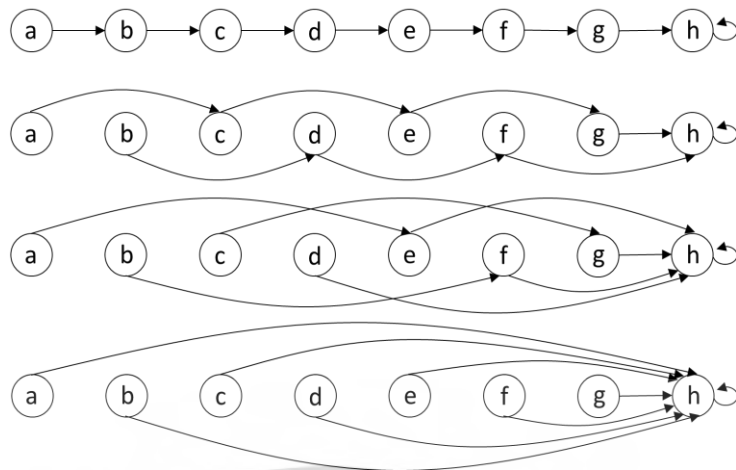
2.5 ขั้นตอนวิธีพอยเตอร์จัมป์ (Pointer-Jumping Algorithm)

ขั้นตอนวิธีพอยเตอร์จัมป์ (Pointer-Jumping Algorithm) หรือขั้นตอนวิธีช็อตคัตติ้ง (Short Cutting Algorithm) [2] เป็นขั้นตอนวิธีที่ใช้สำหรับโครงสร้างข้อมูลแบบลิงคีสต์ (Linked-List) และโครงสร้างข้อมูลแบบทรี (Tree) เป็นเทคนิคที่ออกแบบเพื่อขั้นตอนวิธีแบบขนานที่ประมวลผลภายใต้โครงสร้างการกระโดดของตัวชี้ที่มีประสิทธิภาพเป็น $O(\log_2 p)$ ที่สามารถหาหุ้ (root) ของทรีและสามารถประยุกต์ใช้กับขั้นตอนวิธีเกี่ยวกับกราฟเพื่อการประมวลผลแบบขนานที่มีประสิทธิภาพ หนึ่งในตัวอย่างงานที่ประยุกต์ใช้ได้ด้วยขั้นตอนวิธีพอยเตอร์จัมป์ หลักการกระโดดของตัวชี้ในขั้นตอนวิธีพอยเตอร์จัมป์ เป็นการกำหนดตัวชี้ของแต่ละโหนดแบบ $\text{all } v, \text{parent}(v) = \text{parent}(\text{parent}(v))$ ดังแสดงตัวอย่างการกระโดดของตัวชี้ของขั้นตอนวิธีพอยเตอร์จัมป์ในโครงสร้างข้อมูลแบบลิงคีสต์เป็นจำนวน $\log_2 p$ รอบ ดังในรูปที่ 2.10 ที่มีการกระโดดของตัวชี้จำนวน 3 รอบ เมื่อ $p = 8$

ในการใช้ขั้นตอนวิธีพอยเตอร์จัมป์ในการหาระยะทางจากแต่ละโหนดไปยังโหนดสุดท้ายในลิสต์ของโครงสร้างข้อมูลแบบลิงคีสต์ เรียกว่า ขั้นตอนวิธีลิสต์เรงกิง (List Ranking Algorithm) (ขั้นตอนวิธีที่ 2.5) ซึ่งมี 2 กระบวนการหลัก คือ ทุกหน่วยประมวลผลทำการคำนวณระยะทางของตัวชี้แต่ละโหนด และปรับตัวชี้ของ parent ขึ้นไปยัง parent ของ parent ซึ่งการหาระยะทางของแต่ละโหนด

ไปยังโหนดสุดท้ายของชุดปัญหาขนาด $n = 8$ และ $p = 8$ แสดงตัวอย่างในรูปที่ 2.11

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่ออนุญาตให้ไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



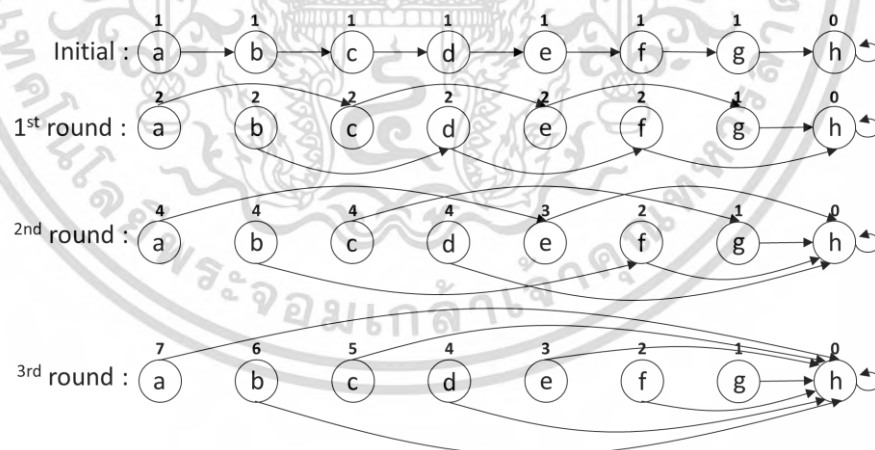
รูปที่ 2.10 แผนภาพแสดงการทำงานของขั้นตอนวิธีพอยเตอร์จัมป์

ขั้นตอนวิธีที่ 2.5 ขั้นตอนวิธีลิสต์เรียงกัน (List-Ranking Algorithm)

```

ListRanking (L, rank(..))
  all v, Pv: rank(v) = 1;
  Plast: rank(last) = 0;
  for k=1 to  $\lceil \log n \rceil$  do
    all v, Pv: rank(v) = rank(v) + rank(parent(v));
    parent(v) = parent(parent(v));
  end for
end

```



รูปที่ 2.11 ตัวอย่างการทำงานของขั้นตอนวิธีพอยเตอร์จัมป์สำหรับ $n = 8$ และ $p = 8$

2.6 แผนการทดลองแบบลาตินสแควร์ (Latin Square Design)

แผนการทดลองแบบลาตินสแควร์ (Latin Square Design) [4][9][22] มีลักษณะเป็นตารางอาเรย์ ขนาด $n \times n$ ซึ่งแทนปัจจัยที่จำนวนเท่ากัน 2 ปัจจัย เช่น แถวแทนช่วงเวลาที่เก็บข้อมูล 4 ช่วง คอลัมน์แทนสถานที่เก็บข้อมูล 4 แห่ง โดยเอกลักษณ์ของแผนการทดลองแบบลาตินสแควร์คือค่าในเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แต่ละแถวแต่ละคอลัมน์จะมีค่าไม่ซ้ำกัน ดังนั้น จึงเป็นวิธีที่สามารถนำมาวางแผนเพื่อการดำเนินงานวิจัยได้ ตัวอย่างเช่น หากต้องการวางแผนการสำรวจจำนวนช้างป่าในเนื้อที่ที่สนใจจำนวน 4 แห่ง ในช่วงระยะเวลา 1 ปี ซึ่งสามารถแบ่งออกได้เป็น 4 ไตรมาส โดยใช้แผนการทดลองแบบลาตินสแควร์สามารถทำได้ดังนี้ (แสดงในรูปที่ 2.12)

1. สร้างผังลาตินสแควร์ในขนาด 4x4 โดยกำหนดอักษรเบื้องต้นในแต่ละโดยอักษรไม่ซ้ำกันในแต่ละแถวแต่ละคอลัมน์

2. จัดลำดับแถวในแบบสุ่ม

3. จัดลำดับคอลัมน์ในแบบสุ่ม

4. กำหนดค่าพารามิเตอร์ให้กับแต่ละอักษรในผังลาตินสแควร์ในแบบสุ่ม

	Column 1	Column 2	Column 3	Column 4
Row 1	A	B	C	D
Row 2	B	A	D	C
Row 3	C	D	B	A
Row 4	D	C	A	B

(ก)

	Column 1	Column 2	Column 3	Column 4
Row 2	B	A	D	C
Row 4	D	C	A	B
Row 1	A	B	C	D
Row 3	C	D	B	A

(ข)

	Column 1	Column 4	Column 3	Column 2
Row 2	B	C	D	A
Row 4	D	B	A	C
Row 1	A	D	C	B
Row 3	C	A	B	D

(ค)

	Column 1	Column 4	Column 3	Column 2
Row 2	T1	T3	T2	T4
Row 4	T2	T1	T4	T3
Row 1	T4	T2	T3	T1
Row 3	T3	T4	T1	T2

(ง)

	พื้นที่ 1	พื้นที่ 4	พื้นที่ 3	พื้นที่ 2
ไตรมาส 2	T1	T3	T2	T4
ไตรมาส 4	T2	T1	T4	T3
ไตรมาส 1	T4	T2	T3	T1
ไตรมาส 3	T3	T4	T1	T2

(จ)

รูปที่ 2.12 ตัวอย่างการทำงานของแผนการทดลองแบบลาตินสแควร์ขนาด 4x4

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปเผยแพร่โดยไม่ได้รับอนุญาต
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 2.12 (ก) เป็นการสร้างผังลาดินสแควร์เพื่อกำหนดอักษรเบื้องต้นให้กับผังลาดินสแควร์ จากนั้นทำการสลับแถวแบบสุ่ม (รูปที่ 2.12 (ข)) แล้วจึงสลับคอลัมน์แบบสุ่ม (รูปที่ 2.12 (ค)) เมื่อสลับแถวและคอลัมน์ในแบบสุ่มแล้ว จากนั้นกำหนดค่าทริกเม้นต์ให้กับแต่ละอักษรในแบบสุ่ม ซึ่งในที่นี้ แทนอักษร A ด้วยทริกเม้นต์ T4 แทนอักษร B ด้วยทริกเม้นต์ T1 แทนอักษร C ด้วยทริกเม้นต์ T3 และแทนอักษร D ด้วยทริกเม้นต์ T2 (รูปที่ 2.12 (ง)) จากนั้นกำหนดค่าในคอลัมน์และแถวด้วยปัจจัยที่สนใจ แล้วเลือกรูปแบบใดรูปแบบหนึ่งในผังลาดินสแควร์ ดังในตัวอย่างจะเป็นการกำหนดให้แถวเป็นระยะไตรมาสที่สำรวจและกำหนดให้คอลัมน์เป็นเนื้อที่เก็บข้อมูล (รูปที่ 2.12 (จ))

2.7 งานวิจัยที่เกี่ยวข้อง (Related Work)

2.7.1 ขั้นตอนวิธีกรีดีฮิวริสติกด้วยการคาดการณ์อย่างมีประสิทธิภาพ (Greedy with Dynamic Expectation Efficiency)

จากขั้นตอนวิธีกรีดีฮิวริสติกที่มีความเร็วในการประมวลผลสูงอาจไม่ได้คำตอบที่ดีที่สุด หรือได้เพียงคำตอบที่ยอมรับได้แต่อาจห่างจากคำตอบที่ดีที่สุดมาก จึงมีนักวิจัยหลายท่านสนใจพัฒนาและประยุกต์ขั้นตอนวิธีกรีดีฮิวริสติกให้มีประสิทธิภาพมากขึ้น อาทิ ในปี ค.ศ. 2016 Jianhui Lv และคณะ [35] ได้มีการพัฒนาขั้นตอนวิธีกรีดีฮิวริสติกด้วยการคาดการณ์อย่างมีประสิทธิภาพ หรือขั้นตอนวิธีจีดีอี (Greedy with Dynamic Expectation Efficiency : GDEE) ซึ่งใช้รูปแบบการเรียงลำดับค่า P/W แล้วแบ่งกลุ่มสมาชิกจาก $n/2, n/4, \dots$ จนถึงเงื่อนไขที่กำหนดในสมการที่ 2.1 (รูปที่ 2.13 (Step I)) ทำให้ได้กลุ่มของสมาชิกขนาด m ที่คาดว่าจะเป็คำตอบที่ดีที่สุดเพื่อนำไปบรรจุลงถุงเป้ และกลุ่มของสมาชิกขนาด $n - m$ ที่ไม่แน่ใจว่าสิ่งของใดควรเลือกหรือไม่ควรเลือก ดังแสดงในรูปที่ 2.13 (Step II)

$$a_1 \wedge a_2 \leq \gamma, a_1^+ \wedge a_2^+ > \gamma, b_1 \wedge b_2 < \delta, b_1^+ \wedge b_2^+ \geq \delta \quad (2.1)$$

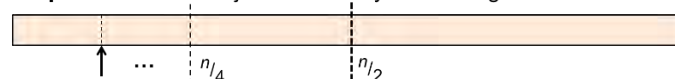
โดยที่

$$a_1 = \frac{\sum_{i=1}^j w_i}{Gw}, a_2 = \frac{\sum_{i=1}^j p_i}{Gtp}, a_1^+ = \frac{\sum_{i=1}^{j+1} w_i}{Gw}, a_2^+ = \frac{\sum_{i=1}^{j+1} p_i}{Gtp}$$

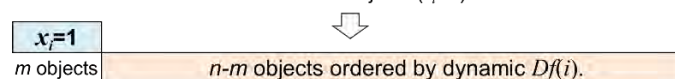
$$b_1 = \frac{\sum_{i=1}^{n/2} w_i + \sum_{i=n/2+1}^j w_i}{Gw}, b_2 = \frac{\sum_{i=1}^{n/2} p_i + \sum_{i=n/2+1}^j p_i}{Gtp}$$

$$b_1^+ = \frac{\sum_{i=1}^{n/2} w_i + \sum_{i=n/2+1}^{j+1} w_i}{Gw}, b_2^+ = \frac{\sum_{i=1}^{n/2} p_i + \sum_{i=n/2+1}^{j+1} p_i}{Gtp}$$

Step I: start with n objects ranked by decreasing P/W .



do ($n=n/2$) until satisfy all conditions in equation (2.1) in order to select the first m objects ($x_i=1$).



Step II: determine $x_i=1/0$, according to $Df(i)$, while $\sum x_i w_i \leq C$.

รูปที่ 2.13 การแบ่งกลุ่มสมาชิกของปัญหาด้วยขั้นตอนวิธีจีดีอี (Classified Objects by GDEE Algorithm)

เอกสารนี้เป็นเอกสารที่สวทช. อนุญาตให้รับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.2 การหาคำตอบที่เหมาะสมแบบกลุ่มด้วยวิธีชัฟเฟิลฟรอก (Shuffled Frog Swarm Optimization)

วิธีการหาคำตอบที่เหมาะสมแบบกลุ่มด้วยวิธีชัฟเฟิลฟรอกเป็นขั้นตอนวิธีทางคอมพิวเตอร์ที่ได้แนวคิดมาจากพฤติกรรมการหาแหล่งอาหารของฝูงกบ (Shuffled Frog Leaping) ซึ่งถูกคิดค้นและนำเสนอโดย Eusuff และ Lansay ในปี ค.ศ. 2003 ในชื่อเอสเอฟแอลเอ (Shuffled Frog Leaping Algorithm : SFLA) [10] (ดังแสดงในขั้นตอนวิธีที่ 2.6)

ขั้นตอนวิธีที่ 2.6 ขั้นตอนวิธีการหาคำตอบที่เหมาะสมแบบกลุ่มด้วยวิธีชัฟเฟิลฟรอก (Shuffled Frog Swarm Optimization Algorithm)

0. Initialize parameters (Population size = P , #memplexes= m , #iterations per memplex= n/IT)
1. Generate population ($P=m \times n$) and compute fitness of each value
2. Sort & partition P into m memplexes
3. Local search to determine X_b and X_w and improve X_w
4. Shuffle frog between m memplex
5. Repeat step 2-5 until satisfy the terminate condition

องค์ประกอบของการหาคำตอบที่เหมาะสมแบบกลุ่มด้วยวิธีชัฟเฟิลฟรอก ได้แก่ กบ แทนสมาชิกของปัญหา (P : Population) และฝูงกบ เรียกว่า มีมิเพล็กซ์ (m : Memplex) แทนกลุ่มย่อยของสมาชิก การทำงานของขั้นตอนวิธีนี้ เริ่มจากการเรียงลำดับกบทั้งหมดด้วยค่าฟิตเนส (Fitness) จากมากไปน้อย แล้วแบ่งกลุ่มสมาชิกกบทั้งหมด โดยส่งกบที่ละตัวไปแต่ละมีมิเพล็กซ์แบบหนึ่งต่อหนึ่งวนไปจนถึงมีมิเพล็กซ์สุดท้าย ($P_1 \rightarrow m_1, P_2 \rightarrow m_2, P_3 \rightarrow m_3$ จนถึง $P_r \rightarrow m_{สุดท้าย}$) แบ่งกลุ่มวนเช่นนี้จนถึงกบตัวสุดท้าย ในรอบแรก เมื่อแบ่งกลุ่มสมาชิกของปัญหาแล้ว แต่ละกลุ่มจะได้กบที่มีค่าฟิตเนสมากที่สุด (X_b) และกบที่มีค่าฟิตเนสน้อยที่สุด (X_w) กบที่มีค่าฟิตเนสมากที่สุดในแต่ละมีมิเพล็กซ์จะถูกกำหนดให้เป็นคำตอบสำหรับกลุ่มย่อย ซึ่งจะถูกนำไปเปรียบเทียบกับกบที่มีค่าฟิตเนสมากที่สุดของมีมิเพล็กซ์อื่น ซึ่งจะได้อีกกบที่มีค่าฟิตเนสมากที่สุดจากทั้งหมดเพื่อกำหนดเป็นคำตอบเริ่มต้นของปัญหา (X_g) จากนั้นแต่ละมีมิเพล็กซ์จะพัฒนากบที่มีค่าฟิตเนสน้อยที่สุด (X_w) ให้มีค่าดีขึ้น โดยใช้สมการที่ 2.2

$$D_i = \text{Rand}() \times (X_b - X_w) \quad \text{โดย } D_i \text{ คือ ตำแหน่งของกบที่ถูกสุ่ม (Rand()~U(0,1))} \quad (2.2)$$

$$X_w(\text{new}) = X_w + D_i \quad \text{โดย } X_w(\text{new}) \text{ คือ ตำแหน่งใหม่}$$

$$-D_{\max} \leq D_i \leq D_{\max} \quad \text{โดย } D_{\max} \text{ คือ ตำแหน่งใหม่ที่มากที่สุด}$$

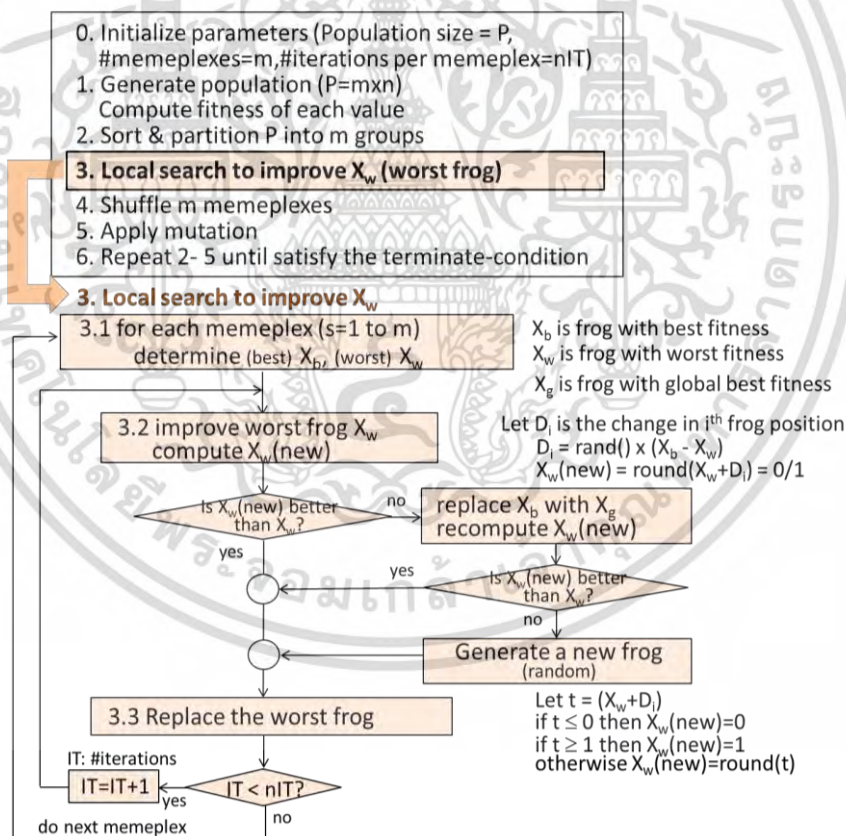
หลังจากปรับค่าฟิตเนสแล้วจะทำให้มีการปรับเปลี่ยนตำแหน่งของกบในแต่ละมีมิเพล็กซ์ ซึ่งทำให้มีการเปลี่ยนแปลงค่าคำตอบโดยรวมที่ดีขึ้น แต่ขั้นตอนวิธีนี้อาจจะเกิดเหตุการณ์ที่สมาชิกที่เป็นคำตอบติดอยู่ในบางมีมิเพล็กซ์ได้ ซึ่งส่งผลให้คำตอบโดยรวมไม่ใช่คำตอบที่ดีที่สุด จึงได้มีการปรับปรุงเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนวิธีเอสเอฟแอลเอ (SFLA) เป็นดีเอสเอฟแอลเอ (Discrete Shuffled Frog Leaping Algorithm : DSFLA) ด้วยการปรับค่าฟิตเนสของกบที่มีค่าฟิตเนสต่ำที่สุดในกลุ่มเพื่อพัฒนาคำตอบของกลุ่มย่อยให้ดีขึ้น ด้วยสมการที่ 2.3

$$D_i = \begin{cases} \min\{\text{int}[\text{Rand} \times (X_b - X_w)], D_{\max}\} & \text{สำหรับตำแหน่งที่เป็นค่าบวก} \\ \min\{\text{int}[\text{Rand} \times (X_b - X_w)], -D_{\max}\} & \text{สำหรับตำแหน่งที่เป็นค่าลบ} \end{cases} \quad (2.3)$$

$$X_w(\text{new}) = X_w + D_i$$

ต่อมา Kaushik Kumar และ S.P. Sarmah ได้นำเสนอขั้นตอนวิธีดีเอสเอฟแอลเอแบบดัดแปลง (Modified-DSFLA) ในปี ค.ศ.2014 [10] โดยเพิ่มขั้นตอนการกลายพันธุ์ (Mutation) ของขั้นตอนวิธีแบบพันธุกรรม (Genetic Algorithm) และปรับปรุงกระบวนการในการค้นหาคำตอบในกลุ่มย่อย (Local Solution) เพื่อเพิ่มประสิทธิภาพในการค้นหาคำตอบโดยรวม (Global Solution) ให้ดีขึ้น ดังแสดงในรูปที่ 2.14



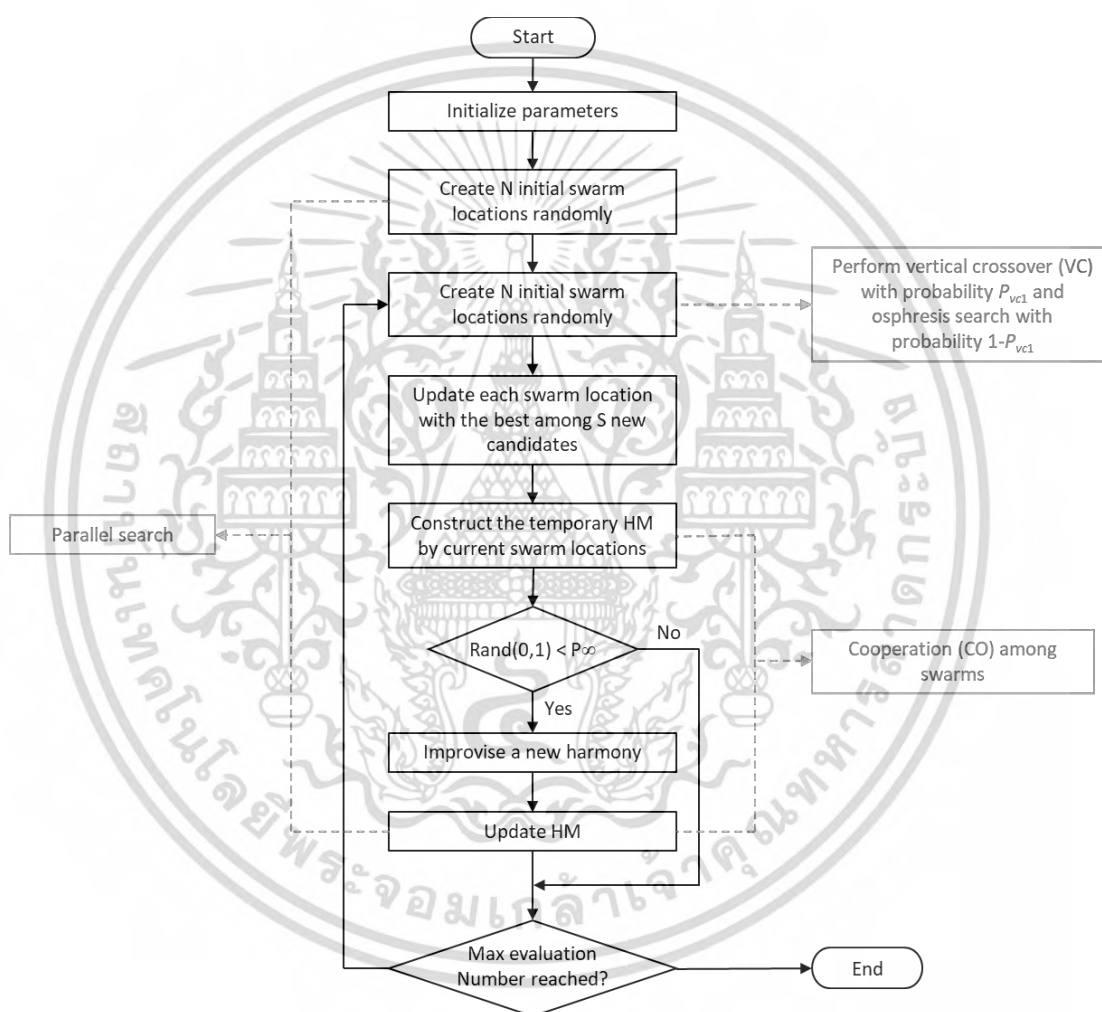
รูปที่ 2.14 ขั้นตอนวิธีดีเอสเอฟแอลเอแบบดัดแปลง (Modified-DSFLA)

ขั้นตอนวิธี Modified-DSFLA เป็นขั้นตอนที่ได้รับการศึกษาและพัฒนาอย่างมากซึ่งหนึ่งในการนำขั้นตอนวิธีนี้ไปประยุกต์คือการค้นหาลักษณะของคนที่มอิทธิพล (Influencer) ในสื่อสังคมออนไลน์ [53] โดย Jianxin Tang และทีมงาน เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อใช้ในการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.3 การหาคำตอบที่เหมาะสมแบบฝูงแมลงวันผลไม้สำหรับปัญหามัลติไดเมนชันแนปแซค (Improved Fruit Fly Optimization Algorithm for the Multidimensional Knapsack Problem)

การแก้ปัญหาถุงเป้แบบหลายมิติด้วยขั้นตอนวิธีการหาคำตอบที่เหมาะสมแบบฝูงแมลงวันผลไม้ (Osphrsis Foraging and Vision Foraging) [39] เป็นการพัฒนาขั้นตอนวิธีทางคอมพิวเตอร์ที่ได้จากการศึกษาพฤติกรรมการหาอาหารของฝูงแมลงวันผลไม้ ซึ่งมุ่งเน้นการหาคำตอบโดยรวมที่เหมาะสม (Global Optimization)



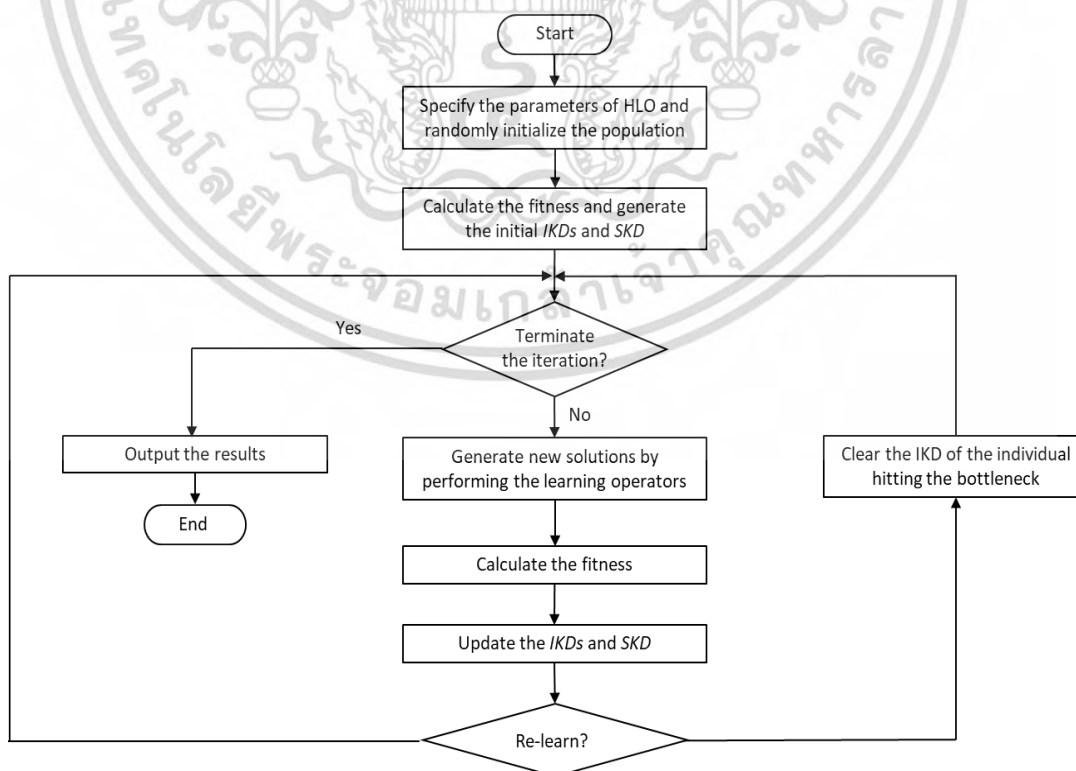
รูปที่ 2.15 ผังงานกระบวนการของการหาคำตอบที่เหมาะสมแบบฝูงแมลงวันผลไม้ (IFFOA)

รูปที่ 2.15 แสดงการประยุกต์ขั้นตอนวิธีการหาคำตอบที่เหมาะสมแบบฝูงแมลงวันผลไม้สำหรับปัญหาถุงเป้แบบหลายมิติ ซึ่งประกอบด้วยการกำหนดค่าในด้วยรหัสบิต (Binary Representation Scheme) ซึ่งเป็นขั้นตอนการแทนค่าการเลือกสิ่งของของแต่ละชิ้น (แมลงวันผลไม้) ด้วยรหัสบิตร่วมกับถุงเป้ (แหล่งอาหาร) จากนั้นเป็นการจำลองการหาคำตอบแบบขนาน (Parallel Search) ซึ่งเปรียบเสมือนกับการค้นหาแหล่งอาหารของแมลงวันผลไม้แต่ละตัวที่กระจายบินไปหาแหล่งอาหารแบบสุ่มในบริเวณเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตำแหน่งเริ่มต้น แล้วประเมินค่าคำตอบของสิ่งของและถุงเป้ซึ่งเป็นรูปแบบของการรับรู้ด้วยกลิ่นและการมองเห็นแหล่งอาหารของแมลงวันผลไม้แต่ละตัวกับแหล่งอาหาร (Osmophresis Foraging and Vision Foraging) โดยเป็นขั้นตอนการประเมินค่าเพื่อเลือกสิ่งของลงถุงเป้แต่ละถุง ตามความเข้มข้นของกลิ่นและระยะทางของแมลงวันผลไม้แต่ละตัวที่ใกล้แหล่งอาหารแต่ละตำแหน่งที่ไปค้นหา นอกจากนี้ ได้มีการประสานความร่วมมือระหว่างฝูงแมลงวันผลไม้ (Cooperation Among Swarm) ซึ่งเป็นขั้นตอนการหาคำตอบที่เหมาะสมโดยประยุกต์ใช้ขั้นตอนวิธีการค้นหาที่ประสานกลมกลืน (Harmony Search Algorithm) เพื่อปรับปรุงค่าคำตอบให้ดีขึ้น

2.7.4 การหาคำตอบที่เหมาะสมตามรูปแบบการเรียนรู้ของมนุษย์สำหรับปัญหามัลติไดเมนชันแนปแซค (Human Learning Optimization Algorithm to Multidimensional Knapsack Problems)

การหาคำตอบที่เหมาะสมตามรูปแบบการเรียนรู้ของมนุษย์ (Human Learning Optimization Algorithm : HLO) [58] เป็นขั้นตอนวิธีทางคอมพิวเตอร์ที่ออกแบบตามพฤติกรรมการเรียนรู้ของมนุษย์ ซึ่งประกอบด้วย 1) กระบวนการเรียนรู้รายบุคคล (Individual Learning Operator) 2) กระบวนการเรียนรู้ทางสังคม (Social Learning Operator) 3) กระบวนการสำรวจการเรียนรู้แบบสุ่ม (Random Exploration Learning operator) และ 4) กระบวนการเรียนรู้ซ้ำ (Re-learning Operator) ซึ่งเป็นการพัฒนาคำตอบใหม่และสามารถนำไปค้นหาคำตอบที่เหมาะสมได้ ดังแสดงในรูปที่ 2.16

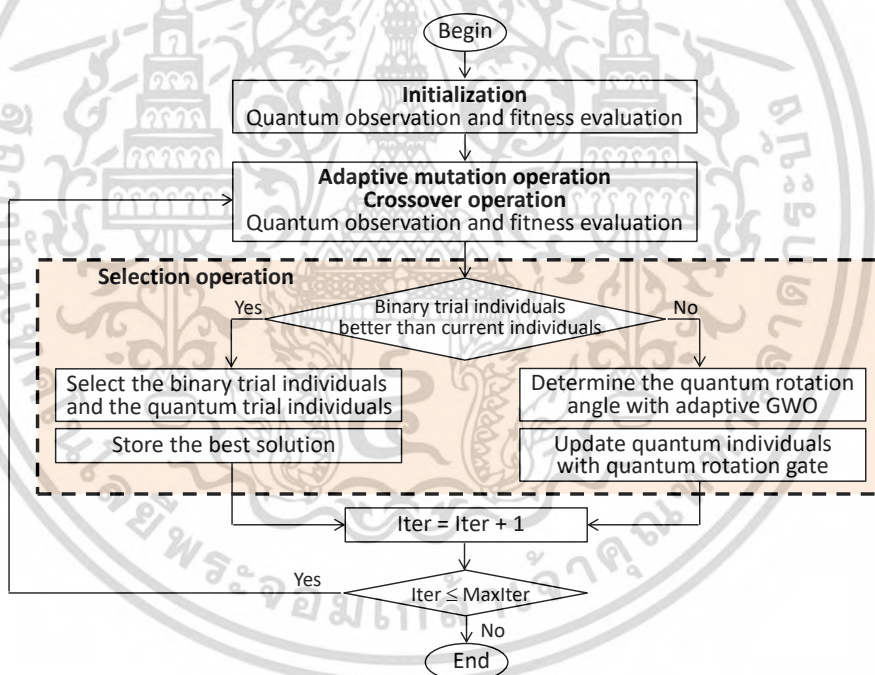


รูปที่ 2.16 ฟังก์ชันกระบวนการของการหาคำตอบที่เหมาะสมตามการเรียนรู้ของมนุษย์ (HLO)

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการศึกษาเท่านั้น เมื่อผู้ใดเห็นประโยชน์ในการนำเอกสารนี้ไปเผยแพร่โดยไม่ได้รับอนุญาตให้ถือว่าผิดกฎหมาย และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.5 ขั้นตอนวิธีการประเมินที่แตกต่างตามแนวทางควอนตัมเพื่อการแก้ปัญหาซิงเกิลแนปแซค (0/1-KP Solving by Quantum -Inspired Differential Evolution Algorithm)

ในการศึกษางานวิจัย พบว่าม้งานวิจัยหนึ่งที่นำเสนอในปี 2021 โดย Yule Wang และ Wanliang Wang [59] ได้นำเสนอขั้นตอนวิธี QDGWO (Quantum-inspired differential evolution with adaptive grey wolf optimizer) สำหรับการแก้ปัญหาซิงเกิลแนปแซค ด้วยการประยุกต์หลักการคำนวณแบบควอนตัมรวมกับการมิวเทชัน (Mutation) และการครอสโอเวอร์ (Crossover Operations) โดยได้ใช้การคำนวณแบบควอนตัมในการสร้างผลลัพธ์ใหม่ที่ใช้ในการทดลอง ซึ่งผลการทดลองของขั้นตอนวิธีนี้ ภายใต้ข้อมูลขนาด $n = 50$ ถึง $n = 3000$ ได้เปรียบเทียบกับขั้นตอนวิธีที่ใช้การคำนวณแบบควอนตัมเป็นพื้นฐาน ได้แก่ QEA, AQDE และ QSE ที่ใช้จำนวนรอบในการประมวลผลสูงสุดที่ 1000 รอบ ขั้นตอนวิธี QDGWO ได้ผลลัพธ์ที่ดีกว่าทั้งหมด แต่อย่างไรก็ตาม ขั้นตอนวิธี QDGWO นี้ ยังไม่สามารถรับรองประสิทธิภาพของผลลัพธ์โดยเฉพาะหากเป็นชุดข้อมูลผลิตปกติ



รูปที่ 2.17 กรอบแนวคิดของขั้นตอนวิธี QDGWO (Quantum-inspired Differential (QD) evolution with adaptive Grey Wolf Optimizer (GWO))

2.7.6 ขั้นตอนวิธีแบบจำลองทางคณิตศาสตร์สำหรับปัญหามัลติเพิลแนปแซค (Mathematical HyMKP)

ขั้นตอนวิธีไฮเอ็มเคพี HyMKP [16] ที่ Mauro Dell'Amico และคณะได้นำเสนอเพื่อการแก้ปัญหามัลติเพิลแนปแซค (0/1-mKP) ซึ่งได้นำเสนอโมเดลไฮเอ็มเคพีทางคณิตศาสตร์ (Mathematical HyMKP) ซึ่งเป็นการผสมผสานระหว่าง MULKNAP Program ซึ่งพัฒนามาจากเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษเท่านั้น เมื่ออนุญาตให้เผยแพร่ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนวิธีbranch and bound (Partial Branch and Bound) กับขั้นตอนวิธีการสร้างรีเฟล็กมัลติกราฟสำหรับปัญหามัลติเพิลแนปแซค (Create-reflect-multigraph-MKP) (ขั้นตอนวิธีที่ 2.8) และวิธีการแบ่งส่วน ในขั้นตอนวิธีที่ 2.8 ได้ปรับปรุงมาจากแบบจำลองการไหลโมเดลอาร์คโฟลว์และการสะท้อนแสง (Arc-Flow Model and The Reflect) ที่มีเวลาประมวลผลที่ $O(mnC)$ ที่เริ่มจากการเรียงลำดับสิ่งของตามน้ำหนักจากมากไปน้อยซึ่งทำให้ได้ผลลัพธ์เริ่มต้น (Initial Good Solution) ในแต่ละถุงเป้ ถ้าเป็นไปตามเงื่อนไข $C \leq n^2$ ($C = \max(C_i)_{i=1,...,m}$) ซึ่งในอดีต เนื่องจากแบบจำลองทางคณิตศาสตร์ถูกในการแก้ปัญหาเกี่ยวกับการจัดการคลังสินค้าแบบ (CSP: 1999 - 2017) ผลการทดลองจากชุดข้อมูลมาตรฐาน ขั้นตอนวิธีไฮเอ็มเคพี (HyMKP) สามารถหาคำตอบที่ดีที่สุดได้ถึง 99.9% เมื่อ $n \leq 500$

ขั้นตอนวิธีที่ 2.7 ขั้นตอนวิธีแบบจำลองทางคณิตศาสตร์สำหรับปัญหามัลติเพิลแนปแซค (Mathematical HyMKP)

Algorithm: Mathematical HyMKP.

- Step 1:** perform the existing pre-processing: *Instance reduction, Capacity lifting, and Item dominance.*
- Step 2:** call MULKnap branch-and-bound (BnB) for τ seconds.
if (the solution is optimal) then return.
- Step 3:** call Create-reflect-multigraph-MKP (Algorithm 2.8).
- Step 4:** for ($i=1$ to v) do
 execute the knapsack-based decomposition.
 if (an optimal solution has been obtained) then return.
 else add the resulting no-good-cut.
end for i
- Step 5:** if (the instance is not solved) then
 execute the reflect-based decomposition and return.

หลักการแนวคิดของแบบจำลองการไหลตามโมเดลอาร์คโฟลว์เพื่อการเลือกสิ่งของลงในถุงเป้ในกราฟซึ่งแต่ละเส้นหรืออาร์ค (Arc) ของแต่ละสิ่งของ ซึ่งเงื่อนไขหลักของขั้นตอนวิธีที่ 2.8 แสดงอยู่ในบรรทัดที่ 5-16 ซึ่งเหมือนกับขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื้องต้นสำหรับหาคำตอบของแต่ละถุงเป้ด้วยเวลา $O(mnC)$ ซึ่งในขั้นตอนวิธีนี้ได้กำหนดเส้น (Arc) ในเซต A จากโหนด d ถึง e ดังนี้

$A_s = \{(d, d+w_j, j, 0), 1 \leq j \leq n\}$ เป็นเซตของไอเท็มมาตรฐานของ Arc

$A_r = \{(d, C_r - (d+w_j), j, i), 1 \leq i \leq m, 1 \leq j \leq n\}$ เป็นเซตของ Arc item สะท้อน
(satisfying $d+w_j > C_r/2$ และ $d \leq C_r - w_j$).

$A_c = \{(C_r/2, C_r/2, 0, i)\}$ เป็นเซตของส่วนโค้งเชื่อมต่อที่สะท้อนออกมา

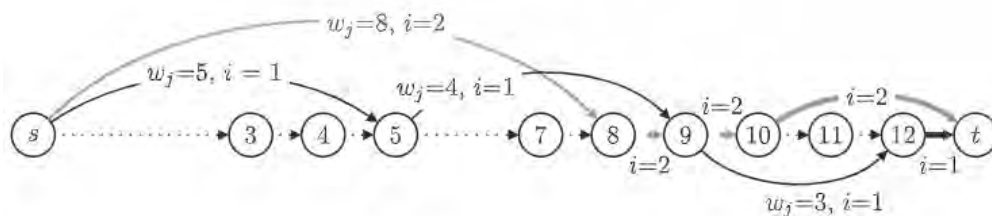
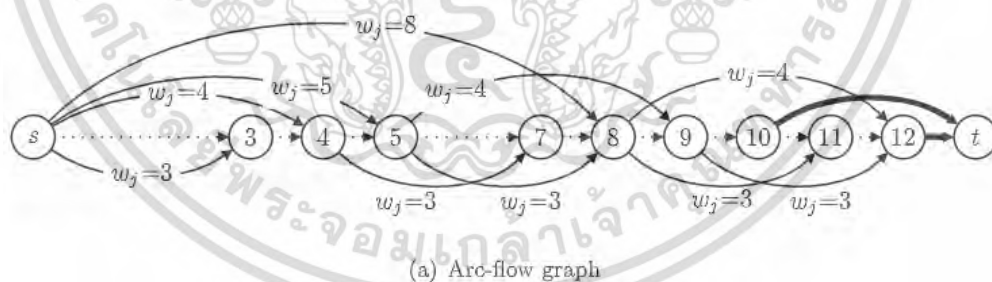
$A_l = \{(d, e, 0, 0), d < e\}$ คือเซตของการสูญเสียส่วนโค้ง

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการศึกษาเท่านั้น ไม่ให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนวิธีที่ 2.8 ขั้นตอนวิธีการสร้างรีเฟล็กมัลติกราฟสำหรับปัญหามัลติเพิลแนปแซค (Create-reflect-multigraph-MKP)

Algorithm: Create-reflect-multigraph-MKP

1. Input: m : number of knapsacks; C_i : knapsack capacity; n : number of items; w_j : item weight
2. $N = \emptyset$; $A_s = \emptyset$; $A_r = \emptyset$; $A_c = \emptyset$; $A_l = \emptyset$; $A = \emptyset$; $s = 0$; $C = \max(C_i)_{i=1,2,\dots,m}$
3. for $(l=1$ to $C/2)$ do $T[l]=0$;
4. $T[s]=1$; $N=N \cup \{s\}$;
5. for $(j=1$ to $n)$ do
6. for $(l=C/2-1$ down to $0)$ do
7. if $(T[l]=1)$ then
8. $A_s = A_s \cup \{l, l+w_j, j, 0\}$;
9. $T[l+w_j]=1$; $N=N \cup \{l+w_j\}$;
10. if $(l+w_j \leq C/2)$ then T
11. for $(i=1$ to $m)$ do
12. if $(l+w_j > C/2$ and $l \leq C_i - (l+w_j))$ then
13. $A_r = A_r \cup \{l, C_i - (l+w_j), j, i\}$; $N=N \cup \{C_i - (l+w_j)\}$;
14. end for i ;
15. end if;
16. end for l ;
17. end for j ;
18. for $(i=1$ to $m)$ do $N=N \cup \{C_i/2\}$;
19. for $(i=1$ to $m)$ do $A_c = A_c \cup \{(C_i/2, C_i/2, 0, i)\}$;
20. for $(l \in N)$ do $A_l = A_l \cup \{(l, l', 0, 0) : l' = \min(e \in N : e > l)\}$;
21. $A = A_s \cup A_r \cup A_c \cup A_l$;
22. return N, A ;

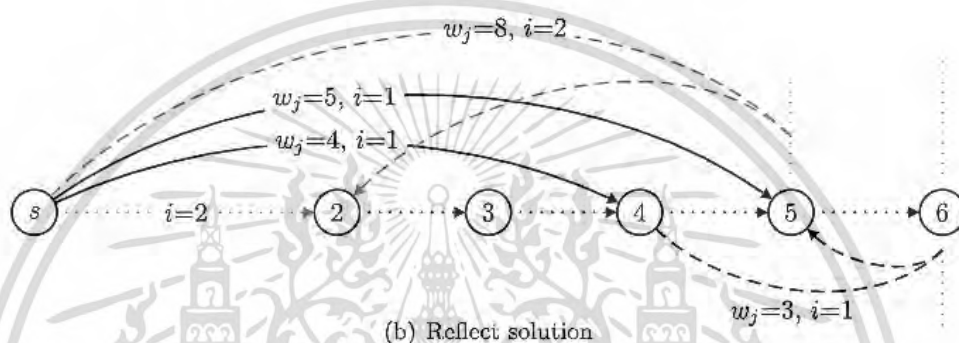
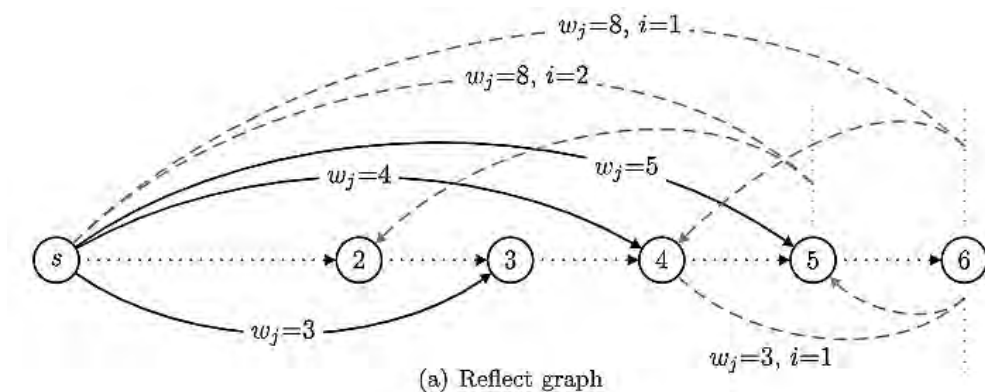


รูปที่ 2.18 แบบจำลองการไหลโมเดลอาร์คโฟลว์ (Arc-flow Graph and Optimal Solution)

สำหรับ 2 งบประมาณ 4 สิ่งของขั้นตอนวิธีดีเอสเอฟแอลเอแบบดัดแปลง (Modified-DSFLA)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ทางการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.19 แบบจำลองรีเฟล็กและการหาคำตอบที่ดีที่สุดต่อเนื่องจากรูปที่ 2.18 (Reflect Graph and Optimal Solution for the instant of Fig.2.18)

จากการศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการแก้ปัญหาถุงเป้ในบทที่ 2 นี้ สามารถแบ่งขั้นตอนวิธีสำหรับการแก้ปัญหาได้ 2 กลุ่ม คือ ขั้นตอนวิธีที่ได้คำตอบที่ดีที่สุด (Exact Algorithm) และขั้นตอนวิธีที่ได้คำตอบโดยประมาณ (Approximate Algorithm)

ตารางที่ 2.1 เปรียบเทียบประสิทธิภาพขั้นตอนวิธีที่ศึกษา

	Exact Algorithms			Approximate Algorithms		
	Basic-DP	Fast-DP	Complete-DP	GH	GDEE	Shuffled Frog
Total Profit (tp)	✓	✓	✓	✓	✓	✓
Total Weight (tw)	✓	✓	✓	✓	✓	✓
Solution set (solX)	✓	✗	✓	✓	✓	✓
Guarantee Opt.Sol.	✓	✓	✓	✗	✗	✗
Time Complexity	$O(nC)$	$O(nC)$	$O(knC)$	$O(n)$	$O(n)$	$O(mn^2)$
Space Utilization	nC	C	C	n	n	nm

สำหรับขั้นตอนวิธีที่ได้คำตอบโดยประมาณที่กล่าวถึง ได้แก่ ขั้นตอนวิธีกริดฮิวริสติก ขั้นตอนวิธีจิตอีอี และขั้นตอนวิธีซฟเฟิลฟรอก ซึ่งมีองค์ประกอบของผลลัพธ์ครบทั้งสามองค์ประกอบ จากขั้นตอนวิธีที่กล่าวถึงในบทนี้สามารถสรุปประสิทธิภาพของขั้นตอนวิธีดังกล่าวได้ดังตารางที่ 2.1 เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนวิธีที่ได้คำตอบที่ดีที่สุดที่ผู้วิจัยสนใจและกล่าวถึงในบทนี้ คือ ขั้นตอนวิธีโปรแกรมเชิงพลวัต ซึ่งแบ่งเป็นขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื่องต้นและแบบสมบูรณ์ที่สามารถหาผลลัพธ์ที่ประกอบด้วยค่า soltp, soltw และ solx ได้ และขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็วที่เน้นความเร็วในการประมวลผลแต่ได้ลดองค์ประกอบของผลลัพธ์เหลือเพียงค่า soltp นอกจากนี้ขั้นตอนวิธี โปรแกรมเชิงพลวัตแบบสมบูรณ์และแบบเร็วได้มีการพัฒนาขั้นตอนวิธีด้วยการใช้อาร์เรย์ขนาดหนึ่งมิติเพื่อลดขนาดหน่วยความจำที่ใช้ในการประมวลผล



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

ขั้นตอนวิธีที่เอสรัดักชั้นสำหรับปัญหาซิงเกิลแนปแซค (0/1-KP)

จากการศึกษาเพื่อออกแบบคิดค้นแนวทางการพัฒนาขั้นตอนวิธีเพื่อการแก้ปัญหาแนปแซค (Knapsack) อย่างต่อเนื่อง ในบทนี้ผู้วิจัยได้ให้ความสนใจเกี่ยวกับการออกแบบและพัฒนาขั้นตอนวิธีสำหรับแก้ปัญหาซิงเกิลแนปแซค (0/1-KP: 0/1-Knapsack) ที่สามารถหาคำตอบที่ดีที่สุด (Optimal Solution) ในเวลาโพลิโนเมียล (Polynomial Time) ด้วยการลดขนาดของปัญหาอย่างมีประสิทธิภาพ ร่วมกับการพัฒนาขั้นตอนวิธีโปรแกรมเชิงพลวัต ผู้วิจัยจึงได้ออกแบบขั้นตอนวิธีที่เอสรัดักชั้น (Time-Space Reduction Algorithm : $TS_{\text{Reduction}}$ Algorithm) ด้วยการคัดกรองแบบไม่เอนเอียง (Unbiased Filtering) ที่มีความซับซ้อนด้านเวลาที่ $O(n+C')$ ซึ่งแบ่งออกเป็น 3 ขั้นตอนใหญ่ ได้แก่ 1) ขั้นตอนวิธีอีวริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพเพื่อหาคำตอบตั้งต้น 2) ขั้นตอนการแบ่งกลุ่ม และการคัดกรองแบบไม่เอนเอียง ซึ่งประกอบด้วย 3 ขั้นตอนย่อย ได้แก่ 2.1) การปรับปรุงผลลัพธ์ให้ดีขึ้นด้วยขั้นตอนวิธีจีเอชพลัส 2.2) การกำหนดค่าน้ำหนักและค่านวนค่าความเหมาะสมแบบพลวัตเพื่อการแบ่งกลุ่มสมาชิก 2.3) การคัดกรองแบบไม่เอนเอียง และ 3) ขั้นตอนวิธีเอ็มดีพี (MDP) เพื่อหาคำตอบในส่วนที่เหลือ (n') ดังแสดงในขั้นตอนวิธีที่ 3.1

ขั้นตอนวิธีที่ 3.1 ขั้นตอนวิธีที่เอสรัดักชั้น ($TS_{\text{Reduction}}$ Algorithm)

Step 1: Apply the GH (greedy heuristic) method by sorting 3 features (P/W , P , W) for top 3 initial solutions in $O(n)$.

Note: Sorting (in each GH) relies on the major-minor keys.

- For P/W -decreasing sort, a major key is P/W and 2 minor keys are P & W .
- For P -decreasing sort, a major key is P and a minor key is W .
- For W -increasing sort, a major key is W and a minor key is P .

Step 2: Object classification and unbiased filtering in $O(n)$.

2.1 Improve 3 initial solutions by the GH^+ method.

2.2 Compute dynamic weight (dw) by integrating 3 solutions of GH_1^+ to GH_3^+ , where $dw = wx_1 + wx_2 + wx_3$; $wx_1 = 5$, $wx_2 = 3$, and $wx_3 = 1$ if ($x_i = 1$); otherwise $wx = 0$ (when $x_i = 0$).

2.3 Classify objects and perform filtering (Group 1 ($dw = 9$), Group 2 ($dw = 8, 6, 5$), Group 3 ($dw = 4, 3, 1$), and Group 4 ($dw = 0$)).

- Filtering in/out (to reduce $n' \leq 200$): Select worth objects ($x_i = 1$) with $dw = 9$ (selected by all 3-GH policies).
- Select other objects ($x_i = 1$) with $dw = 8, 6, 5$, except α (in Group 2) and do not select worst objects ($x_i = 0$) with $dw = 0$, except β (in Group 4).

Step 3: Apply the DP in $O(C')$ on remaining $n' = \alpha + |\text{Group3}| + \beta$.

ขั้นตอนวิธีที่เอสรัดักชั้นได้แนวคิดมาจากหลักการบริหารทรัพยากรมนุษย์หรือหลักการพัฒนาตนเองและสังคมที่ว่า “Put the Right Man in the Right Place at the Right Time” ซึ่งในงานวิจัยนี้เรียกว่า “หลักความถูกต้องสามประการ (Triple Right Commitment)” หมายถึงการพัฒนาองค์กรเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หรือทรัพยากรมนุษย์ให้สามารถปฏิบัติงานได้อย่างมีประสิทธิภาพสูงสุดด้วยความเหมาะสม 3 ประการ ได้แก่ การคัดสรรคนที่มีคุณภาพและเหมาะสม (Right Man) การจัดวางคนให้เหมาะกับตำแหน่งงาน หรือลักษณะงาน (Right Place) เพื่อกำหนดเวลาหรือใช้เวลาในการทำงานได้อย่างเหมาะสม (Right Time) ซึ่งเป็นการให้ความสำคัญกับทุกรายละเอียดตั้งแต่การคัดสรรอย่างใส่ใจกับองค์ประกอบที่เล็กที่สุดของวิธีการหรือกลุ่มงานเพื่อให้ได้คุณภาพ แล้วจึงจะจัดสรรองค์ประกอบเหล่านั้นให้สอดคล้องเหมาะสมกับรูปแบบงานเพื่อให้ได้งานที่มีประสิทธิภาพ รวมถึงการใช้ทรัพยากรที่มีอยู่จำกัดได้อย่างคุ้มค่ามากที่สุด โดยทั้งหมดกระทำภายใต้ระยะเวลาที่เหมาะสมซึ่งจะทำให้การทำงานเป็นไปได้อย่างเป็นระบบและมีประสิทธิภาพสูงสุดในเวลาที่รวดเร็วและเหมาะสม

จากแนวคิดดังกล่าว ผู้วิจัยจึงได้ออกแบบและพัฒนาขั้นตอนวิธีที่เอสรัดักชั้นสำหรับปัญหา จึงเกิดแผนผัง ซึ่งประกอบด้วย 3 ขั้นตอนใหญ่ ที่สอดคล้องกับหลักความถูกต้องสามประการ ดังนี้

- 1) ขั้นตอนวิธีฮิวริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพ เป็นการประเมินความเหมาะสมของสิ่งของที่จะเลือกลงถุงเป้ในระดับเบื้องต้น ซึ่งเทียบได้กับการคัดสรรบุคคลที่เหมาะสม (Right Man)
- 2) ขั้นตอนการแบ่งกลุ่มและการคัดกรองแบบไม่เอนเอียง เป็นการเพิ่มประสิทธิภาพการประเมินความเหมาะสมให้ละเอียดมากขึ้นและแบ่งกลุ่มสมาชิกออกเป็น 4 กลุ่ม ตามความเหมาะสม ได้แก่ กลุ่มที่เหมาะสมมากที่สุด กลุ่มที่เหมาะสมรองลงมา กลุ่มที่ไม่แน่ใจว่าควรเลือกหรือไม่เลือก และกลุ่มที่คาดว่าจะไม่ควรเลือกเป็นคำตอบ ซึ่งขั้นตอนนี้เทียบได้กับการจัดสรรคนให้เหมาะสมกับงาน (Right Man, Right Place)
- 3) ขั้นตอนวิธีโปรแกรมเอ็มดีพี เป็นการคัดเลือกสมาชิกในกลุ่มที่เหลือจากกระบวนการคัดกรอง โดยในขั้นตอนนี้จะสามารถเลือกสิ่งของที่เป็นคำตอบชุดสุดท้ายของปัญหา โดยจำกัดจำนวนสมาชิกของปัญหาไม่เกิน 200 ($n' \leq 200$) เพื่อลดเวลาในการประมวลผล ซึ่งขั้นตอนนี้เทียบได้กับการใช้เวลาอย่างมีประสิทธิภาพในการทำงาน (Right Man, Right Place, Right Time)

3.1 ขั้นตอนวิธีฮิวริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพ (Greedy Heuristic with Efficient Feature Sorting)

ขั้นตอนแรกของขั้นตอนวิธีที่เอสรัดักชั้น เป็นการหาคำตอบเริ่มต้นด้วยการประเมินความเหมาะสมของสิ่งของแต่ละชิ้นในระดับเบื้องต้น (Right Man) ด้วยขั้นตอนวิธีฮิวริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพ (GH with Efficient Feature Sorting) หรือขั้นตอนวิธีจีเอช (GH : Greedy Heuristic) ที่เป็นการเรียงลำดับปัจจัยหลักของปัญหาในหลายรูปแบบ (Multi-key Processing) ด้วยความซับซ้อนด้านเวลาที่ $O(n)$ ซึ่งมีทั้งหมด 3 กฎ ตามคุณลักษณะของสิ่งของ ได้แก่ GH1 เป็นการเรียงลำดับตามค่า P/W จากมากไปน้อย (P/W Ranking) เป็นหลัก GH2 เป็นการเรียงลำดับตามค่า P จากมากไปน้อย (P Ranking) เป็นหลัก และ GH3 เป็นการเรียงลำดับตามค่า W จากน้อยไปมาก (W Ranking) เป็นหลัก ซึ่งแต่ละกฎมีรายละเอียดดังนี้

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อใช้ในการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.2 การกำหนดค่าน้ำหนักและคำนวณค่าความเหมาะสมแบบพลวัตเพื่อการแบ่งกลุ่มสมาชิก (Compute Dynamic Weight to Classify Objects)

จากผลการเลือกสิ่งของด้วยขั้นตอนวิธีจีเอชพลัส (รูปที่ 3.2) พบว่าสิ่งของบางชิ้นมีผลการถูกเลือกตรงกันทั้ง 3 กฎ และบางชิ้นมีผลการถูกเลือกแตกต่างกัน ผู้วิจัยจึงมีแนวคิดในการประยุกต์ใช้หลักการอินเทอร์เซกชัน (Intersection) คือการตัดสินใจเลือกหรือไม่เลือกสิ่งของอย่างเป็นเอกฉันท์เพื่อการคัดกรองที่ไม่เอนเอียง นั่นคือ สิ่งของที่ควรถูกเลือกให้เป็นคำตอบจะถูกเลือกจากทั้ง 3 กฎ และสิ่งของที่ไม่ควรถูกเลือกให้เป็นคำตอบจะไม่ถูกเลือกจากทั้ง 3 กฎ เป็นต้น และเพื่อให้การคัดกรองถูกต้องแม่นยำมากยิ่งขึ้น จึงมีการกำหนดค่าน้ำหนักความสำคัญของแต่ละกฎ $GH1^+$ $GH2^+$ และ $GH3^+$ ด้วยค่า 5, 3, 1 ตามลำดับ เพื่าคำนวณค่าน้ำหนักรวมของสิ่งของแต่ละชิ้นซึ่งจะถูกเปลี่ยนแปลงตามการเลือกของแต่ละกฎ จึงเรียกรวมค่าน้ำหนักนี้ว่า ค่าน้ำหนักพลวัต (Dynamic Weight : dw) ดังนั้น ขั้นตอนการกำหนดค่าน้ำหนักและการคำนวณค่าความเหมาะสมแบบพลวัตเพื่อใช้ในการแบ่งกลุ่มสมาชิกนี้ จึงประกอบด้วย การกำหนดค่าน้ำหนักตามกฎของจีเอชพลัส และการคำนวณผลรวมของน้ำหนักเพื่อแสดงถึงค่าความเหมาะสมของแต่ละสมาชิก ซึ่งจากตัวอย่างรูปที่ 3.2 เมื่อกำหนดค่าน้ำหนักความสำคัญของแต่ละกฎและคำนวณค่าน้ำหนักรวมแล้ว จะได้ค่า dw ของสิ่งของแต่ละชิ้นดังแสดงในรูปที่ 3.3

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
W	6	5	8	10	8	11	4	11	3	3	8	8	7	10	11	10	4	7	8	9
P	43	55	45	42	55	39	32	38	45	31	37	40	36	35	54	34	54	37	50	38
P/W	7.2	11	5.6	4.2	6.9	3.5	8	3.5	15	10	4.6	5	5.1	3.5	4.9	3.4	14	5.3	6.3	4.2
$GH1^+$	5	5	5	5	5	0	5	0	5	5	5	5	5	0	5	0	5	5	5	0
$GH2^+$	3	3	3	3	3	3	0	0	3	3	3	3	0	0	3	0	3	3	3	0
$GH3^+$	1	1	1	0	1	0	1	0	1	1	1	1	1	0	1	0	1	1	1	1
dw	9	9	9	8	9	3	6	0	9	9	9	9	6	0	9	0	9	9	9	1

รูปที่ 3.3 ตัวอย่างการกำหนดค่าน้ำหนักและผลรวมค่าน้ำหนักพลวัต (Dynamic Weight : dw)

จากการกำหนดค่าน้ำหนักความสำคัญของแต่ละกฎจะทำให้ได้ค่า $dw = 9, 8, 6, 5, 4, 3, 1, 0$ ซึ่งสามารถแบ่งได้เป็น 4 กลุ่ม ได้แก่

กลุ่ม 1 คือ กลุ่มสิ่งของที่มีค่า $dw = 9$ เป็นสิ่งของที่ถูกเลือกด้วยกฎทั้ง 3 กฎของจีเอชพลัส

กลุ่ม 2 คือ กลุ่มสิ่งของที่มีค่า $dw = 8, 6, 5$ เป็นกลุ่มสิ่งของที่ถูกเลือกด้วยกฎ $GH1^+$ (ค่าน้ำหนัก = 5) ที่ใช้เกณฑ์ P/W เป็นหลัก ซึ่งเป็นเกณฑ์ที่ให้ความสำคัญกับองค์ประกอบของสิ่งของทั้งสองส่วน คือ มูลค่าและน้ำหนัก แต่ไม่ได้ถูกเลือกเป็นคำตอบด้วยกฎอื่นที่ให้ความสำคัญกับค่า P หรือ W อย่างใดอย่างหนึ่งเป็นหลัก ดังนั้น สิ่งของที่ถูกเลือกในกลุ่มนี้จึงมีโอกาที่จะเป็นคำตอบได้มาก

กลุ่ม 3 คือ กลุ่มสิ่งของที่มีค่า $dw = 4, 3, 1$ ซึ่งเป็นกลุ่มสิ่งของที่ไม่ได้ถูกเลือกด้วยกฎ $GH1^+$ (P/W ranking) แต่ถูกเลือกด้วยกฎ $GH2^+$ (P ranking) และ/หรือกฎ $GH3^+$ (W ranking) ซึ่งเป็นค่าที่

ไม่แน่นอนแต่อาจจะเป็นคำตอบของปัญหาได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กลุ่ม 4 คือ กลุ่มสิ่งของที่มีค่า $dw = 0$ ซึ่งเป็นสิ่งของที่ไม่ถูกเลือกด้วยกฎใด ๆ ซึ่งบ่งบอกได้ว่าสมาชิกกลุ่มนี้มีความเหมาะสมที่จะเป็นคำตอบน้อยหรือไม่เหมาะสมที่จะเป็นคำตอบ

จากการแบ่งกลุ่มสมาชิกออกเป็น 4 กลุ่มโดยใช้ค่า dw ทำให้เห็นถึงคุณสมบัติและความสัมพันธ์ของสมาชิกในแต่ละกลุ่ม ซึ่งสามารถสรุปคุณสมบัติของแต่ละกลุ่มได้ดังตารางที่ 3.2

ตารางที่ 3.2 การแบ่งกลุ่มสมาชิกตามค่าน้ำหนักพลวัต (Dynamic Weight : dw)

	dw	Selected decision	Meaning [if $(x_j = 1)$ $wx_1=5$, $wx_2=3$, $wx_3=1$; else $wx_j=0$]
group 1	9	3 of 3 (P/W, P, W)	3-GHs ⁺ commonly select the object [$dw=5+3+1$].
	8	2 of 3 (P/W, P)	Top 2-GHs ⁺ select the object [$dw = 5+3$].
group 2	6	2 of 3 (P/W, W)	Two GHs ⁺ select the object [$dw = 5+1$].
	5	1 of 3 (P/W)	Only top GH1 ⁺ (P/W) selects the object [$dw = 5$].
	4	2 of 3 (P, W)	Last 2-GHs ⁺ select the object [$dw=3+1$].
group 3	3	1 of 3 (P)	Only GH2 ⁺ (P) selects the object [$dw = 3$].
	1	1 of 3 (W)	Only GH3 ⁺ (W) selects the object [$dw = 1$].
group 4	0	0 of 3 (-)	None of policies selects the object [$dw = 0$].

3.2.3 การคัดกรองแบบไม่เอนเอียง (Unbiased Filtering)

ขั้นตอนการคัดกรองแบบไม่เอนเอียง เป็นขั้นตอนการเลือกหรือไม่เลือกสิ่งของในแต่ละกลุ่มที่ถูกแบ่งตามค่า dw ได้แก่ กลุ่ม 1 ($dw = 9$) กลุ่ม 2 ($dw = 8, 6, 5$) กลุ่ม 3 ($dw = 4, 3, 1$) และกลุ่ม 4 ($dw = 0$) โดยการคัดกรองแบบไม่เอนเอียงแบ่งออกเป็น 2 ระดับ ได้แก่ การคัดกรองระดับ 1 (Filtering 1) และการคัดกรองระดับ 2 (Filtering 2) ซึ่งมีรายละเอียดดังนี้

1) การคัดกรองระดับ 1 (Filtering 1)

การคัดกรองระดับ 1 เป็นการนำผลการเลือกหรือไม่เลือกสิ่งของแบบเอกฉันท์จากขั้นตอนวิธีจีเอชพลัสมาใช้ในการคัดกรองสิ่งของ โดยขั้นตอนนี้จะเลือกสิ่งของในกลุ่ม 1 ($dw = 9$) ให้เป็นคำตอบทั้งหมด ($x_j = 1$) เพราะเป็นกลุ่มที่ถูกพิจารณาจากทั้ง 3 กฎของขั้นตอนวิธีจีเอชพลัสว่ามีความเหมาะสมที่จะเป็นคำตอบ และสำหรับสิ่งของที่เหลือในกลุ่ม 2, 3 และ 4 ($n' = |g_2| + |g_3| + |g_4|$) ซึ่งมีจำนวนจำกัดไม่เกิน 200 ($n' \leq 200$) จะถูกนำไปประมวลผลเพื่อหาคำตอบชุดสุดท้ายด้วยขั้นตอนวิธีเอ็มดีพี (MDP : Modified Dynamic Programming) แต่หากสิ่งของที่เหลือมีจำนวนมากกว่า 200 ($n' > 200$) ซึ่งอาจเกิดขึ้นได้เมื่อสิ่งของในกลุ่ม 2 กลุ่ม 3 หรือกลุ่ม 4 มีจำนวนมาก สิ่งของเหล่านี้จะถูกประมวลผลด้วยการคัดกรองระดับ 2 เพื่อเลือกสิ่งของในกลุ่ม 2 และตัดทิ้งสิ่งของในกลุ่ม 4 ที่ทำให้เหลือจำนวนสมาชิกไม่เกิน 200 แล้วจึงนำไปหาคำตอบชุดสุดท้ายด้วยขั้นตอนวิธีเอ็มดีพี (MDP) ต่อไป

(สำหรับตัวอย่างที่ใช้ในการประกอบการอธิบายในงานวิจัยนี้ ใช้สำหรับแสดงกระบวนการคัดกรองทั้งระดับ 1 และระดับ 2)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

group 1	group 2	group 3	group 4
dw = 9	dw = 8,6,5	dw = 4,3,1	dw = 0
$[x_i = 1]$	dw = 8,6,5	dw = 4,3,1	dw = 0

$$n' = |g2| + |g3| + |g4| \leq 200$$

(ก)

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
W	6	5	8	10	8	11	4	11	3	3	8	8	7	10	11	10	4	7	8	9
P	43	55	45	42	55	39	32	38	45	31	37	40	36	35	54	34	54	37	50	38
P/W	7.2	11	5.6	4.2	6.9	3.5	8	3.5	15	10	4.6	5	5.1	3.5	4.9	3.4	14	5.3	6.3	4.2
GH1*	5	5	5	5	5	0	5	0	5	5	5	5	5	0	5	0	5	5	5	0
GH2*	3	3	3	3	3	3	0	0	3	3	3	3	0	0	3	0	3	3	3	0
GH3*	1	1	1	0	1	0	1	0	1	1	1	1	1	0	1	0	1	1	1	1
dw	9	9	9	8	9	3	6	0	9	9	9	9	6	0	9	0	9	9	9	1

X	0	1	2	4	8	9	10	11	14	16	17	18	3	5	6	7	12	13	15	19
W	6	5	8	8	3	3	8	8	11	4	7	8	10	11	4	11	7	10	10	9
P	43	55	45	55	45	31	37	40	54	54	37	50	42	39	32	38	36	35	34	38
P/W	7.2	11	5.6	6.9	15	10	4.6	5	4.9	14	5.3	6.3	4.2	3.5	8	3.5	5.1	3.5	3.4	4.22
GH1*	5	5	5	5	5	5	5	5	5	5	5	5	5	0	5	0	5	0	0	0
GH2*	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0	0	0	0	0	0
GH3*	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1	0	0	1
dw	9	9	9	9	9	9	9	9	9	9	9	9	8	3	6	0	6	0	0	1

tw= 79 tp= 546
solx= {0,1,2,4,8,9,10,11,14,16,17,18}

n'= 8 C'= 21

(ข)

รูปที่ 3.4 การคัดกรองระดับ 1 (Filtering 1) และตัวอย่าง

รูปที่ 3.4 (ข) แสดงตัวอย่างการคัดกรองระดับ 1 เพื่อเลือกสิ่งของที่มีค่า dw = 9 เป็นสมาชิกของคำตอบชุดแรก ดังนั้น ในตัวอย่างนี้จึงได้สิ่งของที่เป็นคำตอบชุดแรก คือ solx = {0, 1, 2, 4, 8, 9, 10, 11, 14, 16, 17, 18} โดยได้ค่า soltw = 79 และ soltp = 546

2) การคัดกรองระดับ 2 (Filtering 2)

การคัดกรองระดับ 2 เป็นการเลือกสิ่งของในกลุ่ม 2 (dw = 8,6,5) และตัดทิ้งสิ่งของในกลุ่ม 4 (dw = 0) โดยใช้การกำหนดขอบเขตความปลอดภัย (Safety Margin) เพื่อให้ได้ผลการคัดกรองที่แม่นยำยิ่งขึ้นด้วยค่า α ขอบเขตซ้าย (Left-margin) (สมการที่ 3.1) และค่า β ขอบเขตขวา (Right-margin) (สมการที่ 3.2) ซึ่งสิ่งของที่อยู่ในขอบเขตความไม่แน่นอนจำนวน $n' = \alpha + |g3| + \beta$ (สมการที่ 3.3) จะถูกนำไปประมวลผลเพื่อหาสิ่งของที่เป็นคำตอบชุดสุดท้ายด้วยขั้นตอนวิธีเอ็มดีพี (MDP) ต่อไป

$$\alpha = \min (70\%|g2|, 20) \quad \text{if } n \leq 30000 \quad (3.1)$$

$$= \min (100\%|g2|, 30) \quad \text{otherwise}$$

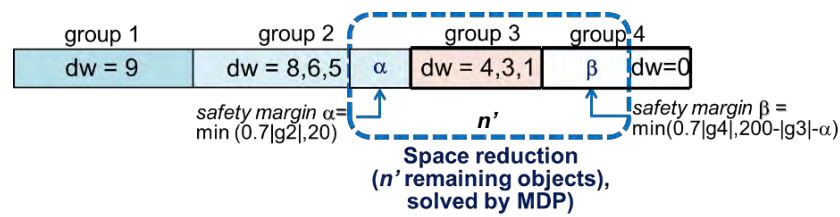
$$\beta = \min (70\%|g4|, 200-|g3|-\alpha) \quad \text{if } n \leq 30000 \quad (3.2)$$

$$= \min (100\%|g4|, 300-|g3|-\alpha) \quad \text{otherwise}$$

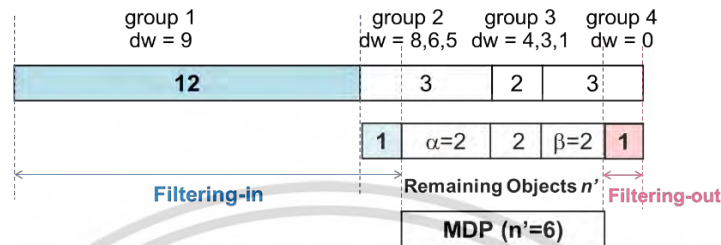
$$n' = \alpha + |g3| + \beta \leq 200 \quad \text{if } n \leq 30000 \quad (3.3)$$

$$\leq 300 \quad \text{otherwise}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



(ก)



(ข)

X	3	5	6	7	12	13	15	19
W	10	11	4	11	7	10	10	9
P	42	39	32	38	36	35	34	38
P/W	4.2	3.5	8	3.5	5.1	3.5	3.4	4.2
GH1 ⁺	5	0	5	0	5	0	0	0
GH2 ⁺	3	3	0	0	0	0	0	0
GH3 ⁺	0	0	1	0	1	0	0	1
dw	8	3	6	0	6	0	0	1



6	15	3	5	7	12	13	19
4	10	10	11	11	7	10	9
32	34	42	39	38	36	35	38
8	3.4	4.2	3.5	3.5	5.1	3.5	4.2
5	0	5	0	0	5	0	0
0	0	3	3	0	0	0	0
1	0	0	0	0	1	0	1
6	0	8	3	0	6	0	1

tw = 79+4 = 83

tp = 546+32 = 578

solx = solx U {6}

(ค)

รูปที่ 3.5 การคัดกรองระดับ 2 (Filtering 2) และตัวอย่าง

รูปที่ 3.5 แสดงแผนภาพและตัวอย่างการคัดกรองระดับ 2 โดยอาศัยค่า dw ในการแบ่งกลุ่ม และกำหนดขอบเขตความปลอดภัยด้วยค่า α และ β ร่วมกับการจัดเรียงข้อมูลแบบ P/W

รูปที่ 3.5 (ก) แสดงแผนภาพการกำหนดขอบเขตความปลอดภัยด้วยค่า α และ β โดยเลือกสมาชิกในกลุ่ม 2 ตามค่า P/W จากมากไปน้อย (หรือจากซ้ายไปขวา) จำนวน $|g2| - \alpha$ เพื่อนำไปเป็นคำตอบ และตัดสมาชิกในกลุ่ม 4 ตามค่า P/W จากน้อยไปมาก (จากขวาไปซ้าย) จำนวน $|g4| - \beta$ ทำให้เหลือสมาชิกส่วนกลางที่อยู่ในขอบเขตความปลอดภัย ซึ่งเป็นสมาชิกกลุ่มที่เหลือจากกระบวนการคัดกรอง จำนวนไม่เกิน 200 ($n' = \alpha + |g3| + \beta \leq 200$) ซึ่งจะถูกนำไปประมวลผลด้วยขั้นตอนวิธีเอ็มดีพี (MDP) ต่อไป

รูปที่ 3.5 (ข) แสดงตัวอย่างผลการคำนวณค่า α และ β ของขอบเขตความปลอดภัย (ต่อเนื่องจากตัวอย่างในรูปที่ 3.4 (ข)) ตามสมการที่ 3.1 และ 3.2 ตามลำดับ ซึ่งจากตัวอย่าง จะได้ค่า $\alpha = 2$ [$\min((0.7 \times 3), 20) = \min(2, 20) = 2$] และค่า $\beta = 2$ [$\min((0.7 \times 3), (200 - 2 - 2)) = \min(2, 196) = 2$] นั่นคือ สิ่งของส่วนซ้ายสุดในกลุ่ม 2 จะถูกเลือกจำนวน 1 ชิ้น ($|g2| - \alpha = 3 - 2 = 1$) และตัดทิ้งสิ่งของส่วนขวาสุดในกลุ่ม 4 จำนวน 1 ชิ้น ($|g4| - \beta = 3 - 2 = 1$) ดังนั้น จำนวนสมาชิกที่เหลือในขอบเขตความปลอดภัย ซึ่งจะถูกนำไปประมวลผลในขั้นตอนวิธีเอ็มดีพี คือ 6 ชิ้น ($\alpha + |g3| + \beta = 2$

+ 2 + 2 = 6)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 3.5 (ค) แสดงตัวอย่างการเลือกและไม่เลือกสิ่งของที่ได้จากการคำนวณในรูปที่ 3.5 (ข) ซึ่งสิ่งของส่วนซ้ายสุดในกลุ่ม 2 จะถูกเลือกจำนวน 1 ชิ้น นั่นคือ สิ่งของดัชนีที่ $x_j = 6$, $w_6 = 4$, $p_6 = 32$ ซึ่งมีค่า P/W มากที่สุดในกลุ่ม และตัดทิ้งสิ่งของส่วนขวาสุดในกลุ่ม 4 จำนวน 1 ชิ้น นั่นคือ สิ่งของดัชนีที่ $x_j = 15$, $w_{15} = 10$, $p_{15} = 34$ ซึ่งมีค่า P/W น้อยที่สุดในกลุ่ม ดังนั้น ในขั้นตอนนี้ จะทำให้ได้ผลลัพธ์โดยรวมเป็น $soltw = 83$, $soltp = 578$, $solx = \{0, 1, 2, 4, \underline{6}, 8, 9, 10, 11, 14, 16, 17, 18\}$ และสมาชิกที่เหลือจำนวน $n' = 6$ ประกอบด้วย $X' = \{3, 5, 7, 12, 13, 19\}$, $W' = \{10, 11, 11, 7, 10, 9\}$ และ $P' = \{42, 39, 38, 36, 35, 38\}$ ด้วยขนาดความจุของเป้ที่เหลือ $C' = 17$ ซึ่งจะถูกนำไปประมวลผลในขั้นตอนวิธีเอ็มดีพีต่อไป

3.3 ขั้นตอนวิธีเอ็มดีพี (MDP)

ขั้นตอนวิธีเอ็มดีพี (MDP : Modified-DP Algorithm) เป็นขั้นตอนสุดท้ายของขั้นตอนวิธีทีเอสรัดักชั้น สำหรับคัดเลือกสิ่งของในกลุ่มที่ไม่แน่นอนที่มีจำนวนจำกัดไม่เกิน 200 โดยขั้นตอนนี้ได้พัฒนากระบวนการมาจากขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ (Complete-DP) ซึ่งเป็นขั้นตอนวิธีหนึ่งที่สามารถหาคำตอบที่ดีที่สุดได้ ด้วยการใช้อาร์เรย์ 1 มิติในการประมวลผล โดยผู้วิจัยได้เพิ่มอาร์เรย์ 1 มิติ สำหรับเก็บชุดรหัสบิตที่แสดงถึงสถานะการเลือก (1) และไม่เลือก (0) ของสิ่งของแต่ละชิ้นที่ถูกประมวลผลในแต่ละรอบ (จำนวนและตำแหน่งของบิตแต่ละค่าจะสอดคล้องกับจำนวนและตำแหน่งของสิ่งของที่ถูกประมวลผล)

ในการประมวลผลแต่ละรอบ ค่าบิตแต่ละตำแหน่งจะถูกเปลี่ยนแปลงตามผลการประมวลผลแบบพลวัตซึ่งสอดคล้องกับค่า tp ใหม่ที่ได้ เมื่อการประมวลผลสิ้นสุดจะทำให้ได้ชุดรหัสบิตที่เป็นคำตอบซึ่งอยู่ในตำแหน่ง C' ที่สามารถนำไปประมวลผลการเลือก/ไม่เลือกสิ่งของแต่ละชิ้น เพื่อเพิ่มความสะดวกของผลลัพธ์ในส่วน $solx$ ให้ครบถ้วนมากขึ้น

C'	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$y=0\ tp$	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42
r	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
(Initial)																		
$y=1\ tp$	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42
r	00	00	00	00	00	00	00	00	00	00	10	10	10	10	10	10	10	10
$y=2\ tp$	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42
r	000	000	000	000	000	000	000	000	000	000	100	100	100	100	100	100	100	100
$y=3\ tp$	0	0	0	0	0	0	0	36	36	36	42	42	42	42	42	42	42	78
r	0000	0000	0000	0000	0000	0000	0000	0001	0001	0001	1000	1000	1000	1000	1000	1000	1000	1001
$y=4\ tp$	0	0	0	0	0	0	0	36	36	36	42	42	42	42	42	42	42	78
r	00000	00000	00000	00000	00000	00000	00000	00000	00000	00000	10000	10000	10000	10000	10000	10000	10000	10010
$y=5\ tp$	0	0	0	0	0	0	0	36	36	38	42	42	42	42	42	42	74	78
r	000000	000000	000000	000000	000000	000000	000000	000010	000010	000001	100000	100000	100000	100000	100000	100000	000101	100100

รูปที่ 3.6 ตัวอย่างขั้นตอนวิธีเอ็มดีพี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Step 1 : GH with Efficient-FS

GH1	X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19		
1.[P/W]	W	6	5	8	10	8	11	4	11	3	3	8	8	7	10	11	10	4	7	8	9		
2.[P]	P	43	55	45	42	55	39	32	38	45	31	37	40	36	35	54	34	54	37	50	38	tw	tp
3.[W]	P/W	7.2	11	5.6	4.2	6.9	3.5	8	3.5	15	10	4.6	5	5.1	3.5	4.9	3.4	14	5.3	6.3	4.2	99	652
	GH1	1	1	1	0	1	0	1	0	1	1	1	1	1	0	1	0	1	1	1	1		
GH2	X	1	4	16	14	18	8	2	0	3	11	5	19	7	17	10	12	13	15	6	9		
1.[P]	W	5	8	4	11	8	3	8	6	10	8	11	9	11	7	8	7	10	10	4	3	tw	tp
2.[W]	P	55	55	54	54	50	45	45	43	42	40	39	38	38	37	37	36	35	34	32	31	91	560
	GH2	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0		
GH3	X	8	9	16	6	1	0	17	12	4	18	2	11	10	19	3	13	15	14	5	7		
1.[W]	W	3	3	4	4	5	6	7	7	8	8	8	8	8	9	10	10	10	11	11	11	tw	tp
2.[P]	P	45	31	54	32	55	43	37	36	55	50	45	40	37	38	42	35	34	54	39	38	98	640
	GH3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0		

Step 2 : Object Classification and Unbiased Filtering**2.1 Improve Solutions by the GH+ Strategy**

GH1⁺	X	8	16	1	9	6	0	4	18	2	17	12	11	14	10	19	3	5	13	7	15		
	W	3	4	5	3	4	6	8	8	8	7	7	8	11	8	9	10	11	10	11	10		
	P	45	54	55	31	32	43	55	50	45	37	36	40	54	37	38	42	39	35	38	34	tw	tp
	P/W	15	14	11	10	8	7.2	6.9	6.3	5.6	5.3	5.1	5	4.9	4.6	4.2	4.2	3.5	3.5	3.5	3.4	100	656
	GH1 ⁺	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0		
GH2⁺	X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19		
	W	6	5	8	10	8	11	4	11	3	3	8	8	7	10	11	10	4	7	8	9	tw	tp
	P	43	55	45	42	55	39	32	38	45	31	37	40	36	35	54	34	54	37	50	38	100	627
	GH2 ⁺	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	0	1	1	1	0		
GH3⁺	X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19		
	W	6	5	8	10	8	11	4	11	3	3	8	8	7	10	11	10	4	7	8	9	tw	tp
	P	43	55	45	42	55	39	32	38	45	31	37	40	36	35	54	34	54	37	50	38	99	652
	GH3 ⁺	1	1	1	0	1	0	1	0	1	1	1	1	1	0	1	0	1	1	1	1		

2.2 Compute Dynamic Weighting to Classify Objects

	X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19		
	W	6	5	8	10	8	11	4	11	3	3	8	8	7	10	11	10	4	7	8	9		
	P	43	55	45	42	55	39	32	38	45	31	37	40	36	35	54	34	54	37	50	38		
	GH1 ⁺	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	5	0	5	5	0	
	GH2 ⁺	3	3	3	3	3	3	0	3	3	3	3	3	3	0	3	0	3	3	3	0		
	GH3 ⁺	1	1	1	0	1	0	1	0	1	1	1	1	1	0	1	0	1	1	1	1		
	dw	9	9	9	8	9	3	6	0	9	9	9	9	6	0	9	0	9	9	9	1		
Step 3 : MDP																							
	Y	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19		
	X	8	16	1	9	6	0	4	18	2	17	11	14	10	12	19	3	5	13	7	15		
	W	3	4	5	3	4	6	8	8	8	7	8	11	8	7	9	10	11	10	11	10		
	P	45	54	55	31	32	43	55	50	45	37	40	54	37	36	38	42	39	35	38	34		
	P/W	15	14	11	10	8	7.2	6.9	6.3	5.6	5.3	5	4.9	4.6	5.1	4.2	4.2	3.5	3.5	3.5	3.4		
	GH1 ⁺	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	5	0	0	0	0		
	GH2 ⁺	3	3	3	3	0	3	3	3	3	3	3	3	3	0	0	3	3	0	0	0		
	GH3 ⁺	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0		
	dw	9	9	9	9	6	9	9	9	9	9	9	9	9	6	1	8	3	0	0	0		
	OptX	1	1	1	1	1	1	1	1	1	1	1	1	1	?	?	?	?	?	?	?	tw	tp
																						83	578
	X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	tw	tp
	OptX	1	1	1	1	1	0	1	0	1	1	1	1	1	1	0	1	1	1	1	0	100	656

รูปที่ 3.7 ตัวอย่างการประมวลผลของขั้นตอนวิธีทีเอสรีดักชัน

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการศึกษาเท่านั้น เมื่อผู้ยืมได้เห็นว่าเอกสารนี้มีความจำเป็น
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 3.6 แสดงตัวอย่างการประมวลผลของขั้นตอนวิธีเอ็มดีพีสำหรับสิ่งของที่เหลือจากขั้นตอนการแบ่งกลุ่มและการคัดกรองแบบไม่เอนเอียง จำนวน $n' = 6$ ซึ่งประกอบด้วย $X' = \{3, 5, 7, 12, 13, 19\}$, $W' = \{10, 11, 11, 7, 10, 9\}$, $P' = \{42, 39, 38, 36, 35, 38\}$ และความจุถุงเป้ที่เหลือ $C' = 17$ โดยผลลัพธ์ที่ได้จากขั้นตอนนี้ คือ ค่าใน $tp[C'] = tp[17] = 78$ และรหัสบิตในอาร์เรย์ $r[C'] = r[17] = 100100$ หมายถึง สิ่งของดัชนีที่ 3 และ 12 เป็นสิ่งของที่ถูกเลือกให้เป็นคำตอบ ดังนั้น ผลลัพธ์โดยรวมของชุดปัญหานี้ คือ $solt_p = 656$, $solt_w = 100$ และ $solt_x = \{0, 1, 2, 3, 4, 6, 8, 9, 10, 11, 12, 14, 16, 17, 18\}$

รูปที่ 3.7 แสดงภาพรวมของตัวอย่างขั้นตอนวิธีทีเอสรีดักชันสำหรับปัญหาถุงเป้ขนาด $n = 20$, $C = 100$, $W = \{6, 5, 8, 10, 8, 11, 4, 11, 3, 3, 8, 8, 7, 10, 11, 10, 4, 7, 8, 9\}$ และ $P = \{43, 55, 45, 42, 55, 39, 32, 38, 45, 31, 37, 40, 36, 35, 54, 34, 54, 37, 50, 38\}$ ซึ่งประกอบด้วย 3 ขั้นตอนใหญ่ ได้แก่ 1) ขั้นตอนวิธีวิริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพ (GH with Efficient Feature Sorting) 2) ขั้นตอนการแบ่งกลุ่มและการคัดกรองแบบไม่เอนเอียง (Object Classification and Unbiased Filtering) ที่ประกอบด้วย 3 ขั้นตอนย่อย ได้แก่ 2.1) การปรับปรุงผลลัพธ์ให้ดีขึ้นด้วยขั้นตอนวิธีจีเอชพลัส (Improve Solutions by the GH⁺ Strategy) 2.2) การกำหนดค่าน้ำหนักและคำนวณค่าความเหมาะสมแบบพลวัตเพื่อการแบ่งกลุ่มสมาชิก (Compute Dynamic Weight to Classify Objects) และ 2.3) การคัดกรองแบบไม่เอนเอียง (Unbiased Filtering) ซึ่งในตัวอย่างนี้ จะแสดงกระบวนการคัดกรองทั้ง 2 ระดับ ได้แก่ การเลือกสิ่งของในกลุ่ม 1 ($dw = 9$) ทั้งหมด และเลือกสิ่งของในกลุ่ม 2 ($dw = 8, 6, 5$) เพิ่ม พร้อมนำสิ่งของบริเวณจุดวิกฤติ (α) ไปไว้ในกลุ่มที่ไม่แน่นอน และตัดสิ่งของในกลุ่ม 4 ($dw = 0$) ออก พร้อมนำสิ่งของบริเวณจุดวิกฤติ (β) ไปไว้ในกลุ่มที่ไม่แน่นอน ดังนั้น สมาชิกในกลุ่มที่ไม่แน่นอน คือ $n' = \alpha + |g_3| + \beta$ ซึ่งไม่เกิน 200 ($n' \leq 200$) ก่อนนำไปประมวลผลในขั้นตอนต่อไป และ 3) ขั้นตอนวิธีเอ็มดีพี (MDP) เป็นขั้นตอนที่หาคำตอบชุดสุดท้ายของปัญหาสำหรับสิ่งของที่เหลือจากการคัดกรอง ซึ่งสามารถหาคำตอบที่ดีที่สุดได้ ดังตัวอย่างที่ได้ผลลัพธ์ที่ได้ คือ $solt_p = 656$, $solt_w = 100$ และ $solt_x = \{0, 1, 2, 3, 4, 6, 8, 9, 10, 11, 12, 14, 16, 17, 18\}$ ซึ่งเป็นคำตอบที่ดีที่สุด (Optimal Solution) ของปัญหา

ตารางที่ 3.3 เปรียบเทียบข้อดีและข้อจำกัดของขั้นตอนวิธีที่ได้ศึกษากับขั้นตอนวิธีทีเอสรีดักชัน

Greedy Method	Frog SWARM Optimization	Complete DP	TSReduction
Sort feature (P/W , P or W) before select objects (fast)	mn initial solutions (m frog-memplexes)	Compute max total-profit for n objects on increasing C	Find efficient initial solutions (P/W -, P -, W - ordering)
-	Improve sols. (worst X_w + Mutation)	Repeat the process for finding each of k selected objects (Optimal X)	Improve 3 best solutions before filtering
-	Repeat until stable result		Apply DP on remaining n', C'
$O(n)$	$O(mn^2)$	$O(knC)$; $k < n, C \leq n^d, d \geq 1$	$O(n + C')$ where $C' \ll C$
Solutions are not optimal	Solutions are not optimal	Optimal solutions	Most solutions are optimal

เอกสารนี้เป็นเอกสารงานวิจัยสำหรับการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์อื่นใด

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 3.3 แสดงการเปรียบเทียบข้อดีของขั้นตอนวิธีที่เอสรีดักชันที่ได้นำเสนอในบทที่ 3 กับ ข้อดีและข้อจำกัดของขั้นตอนวิธีที่ได้ศึกษาในบทที่ 2 สำหรับการแก้ปัญหาเชิงเกิลแนปแซค ซึ่งสรุปได้ ดังนี้

- 1) ขั้นตอนวิธีกรีดีฮิวริสติกด้วยการเรียงลำดับแบบ P/W, P และ W ซึ่งสามารถประมวลผลได้ รวดเร็วด้วยความซับซ้อนด้านเวลาที่ $O(n)$ แต่คำตอบส่วนใหญ่ไม่เป็นคำตอบที่ดีที่สุด
- 2) ขั้นตอนวิธีการหาคำตอบที่เหมาะสมแบบกลุ่มด้วยวิธีซัพเฟิลพรอก ซึ่งเป็นขั้นตอนวิธีหนึ่งใน กลุ่มวิวิจยการหาคำตอบที่เหมาะสมแบบกลุ่ม (Swarm Optimization) โดยมีการค้นหาคำตอบ เริ่มต้นด้วยการแบ่งสมาชิกออกเป็นกลุ่ม และค้นหาคำตอบที่ดีที่สุดในแต่ละกลุ่มย่อย (Local Search) ด้วยการพัฒนาค่าความเหมาะสมของสมาชิกที่มีค่าน้อยที่สุดร่วมกับกระบวนการกลายพันธุ์ จากนั้น นำคำตอบของแต่ละกลุ่มย่อยมาเปรียบเทียบเพื่อหาผลลัพธ์รวม (Global Search) ขั้นตอนวิธีนี้มีความซับซ้อนด้านเวลาที่ $O(mn^2)$ ซึ่งได้คำตอบที่ดีขึ้น แต่ส่วนใหญ่ไม่สามารถหาคำตอบที่ดีที่สุดได้
- 3) ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ (Complete-DP) เป็นขั้นตอนวิธีที่สามารถหาคำตอบที่ดีที่สุดได้ โดยใช้พื้นที่เก็บข้อมูล 1 มิติ (C) และสามารถหาเซตดัชนีของสมาชิกที่เป็นคำตอบ (solx) จากการประมวลผลซ้ำอีก k รอบ ด้วยความซับซ้อนด้านเวลาที่ $O(knC)$ เมื่อ $k < n, C \leq n^d, d \geq 1$
- 4) ขั้นตอนวิธีที่เอสรีดักชันที่ได้นำเสนอ ประกอบด้วย การหาคำตอบเริ่มต้นด้วยขั้นตอนวิธีจีเอช โดยใช้การเรียงลำดับสมาชิกของปัญหาตามคุณลักษณะในหลายระดับ (GH with Multi-key) รวมถึง การปรับปรุงผลลัพธ์ให้ดีขึ้นด้วยขั้นตอนวิธีจีเอชพลัส (GH⁺) และคำนวณหาค่าความเหมาะสม (dw) เพื่อแบ่งกลุ่มสมาชิกเป็นกลุ่มที่เป็นคำตอบ (Select) กลุ่มที่ควรตัดทิ้ง (Unselect) และกลุ่มที่เหลือ จากกระบวนการคัดกรอง (Uncertain) ซึ่งกลุ่มนี้จะถูกประมวลผลด้วยขั้นตอนวิธีเอ็มดีพี (MDP) โดย ขั้นตอนวิธีนี้สามารถหาคำตอบที่ดีที่สุดได้เกือบทุกกรณี ด้วยความซับซ้อนด้านเวลาที่ $O(n+C')$ เมื่อ C' มีค่าน้อยกว่า C มาก ๆ ($C' \ll C$)

จากการเปรียบเทียบวิธีการแก้ปัญหาเชิงเกิลแนปแซคในตารางที่ 3.3 พบว่า ขั้นตอนวิธีกรีดีฮิวริสติกและขั้นตอนวิธีการหาคำตอบที่เหมาะสมแบบกลุ่มด้วยวิธีซัพเฟิลพรอก สามารถหาคำตอบที่ดีได้ แต่ไม่รับรองการได้คำตอบที่ดีที่สุด สำหรับขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์สามารถหาคำตอบที่ดีที่สุดได้ ในขณะที่ขั้นตอนวิธีที่เอสรีดักชันสามารถคำตอบที่ดีที่สุดได้เกือบทั้งหมดยกเว้น ข้อมูลพิเศษบางชุด

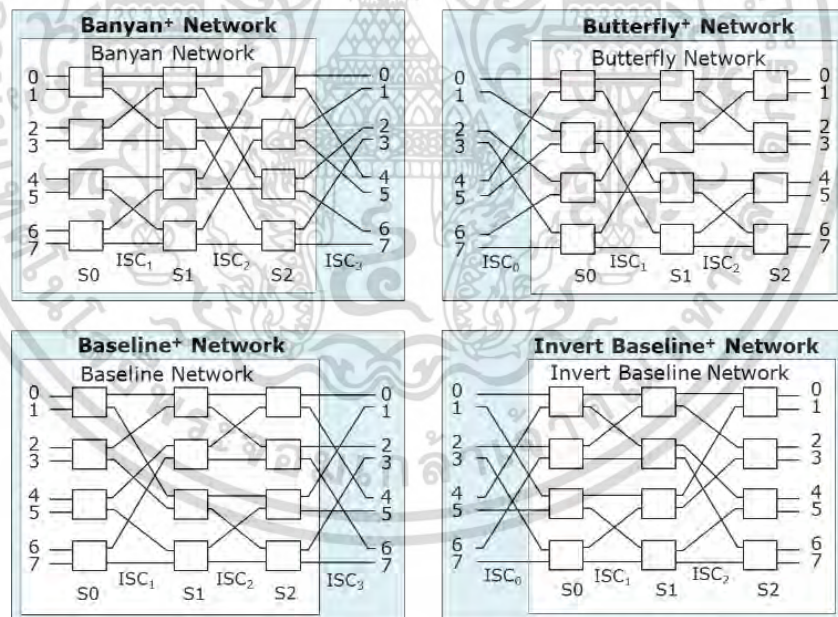
ด้านความซับซ้อนด้านเวลา ขั้นตอนวิธีที่ใช้เวลาประมวลผลน้อยที่สุด คือขั้นตอนวิธีกรีดีฮิวริสติก ซึ่งมีความซับซ้อนด้านเวลาที่ $O(n)$ รองลงมาคือขั้นตอนวิธีการหาคำตอบที่เหมาะสมแบบกลุ่มด้วยวิธีซัพเฟิลพรอกมีความซับซ้อนด้านเวลาที่ $O(mn^2)$ สำหรับขั้นตอนวิธี ที่หาคำตอบที่ดีที่สุดได้ ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ มีความซับซ้อนด้านเวลาที่ $O(knC)$ สำหรับขั้นตอนวิธีที่เอส รีดักชันที่

ได้นำเสนอในงานวิจัยนี้มีความซับซ้อนด้านเวลาที่ $O(n+C')$ นั้น ไม่นับญาติให้นำไปใช้ประโยชน์ด้านการคำนวณว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 ขั้นตอนวิธีทีเอสรีดักชันแบบขนาน (Parallel $TS_{Reduction}$ Algorithm)

จากขั้นตอนวิธีทีเอสรีดักชัน ($TS_{Reduction}$ Algorithm) ที่นำเสนอข้างต้น เพื่อเพิ่มประสิทธิภาพด้านเวลา ผู้วิจัยจึงได้ออกแบบขั้นตอนวิธีทีเอสรีดักชันแบบขนานบนระบบตามทฤษฎี PRAM (Parallel-RAM) เพื่อการติดต่อสื่อสารอย่างมีประสิทธิภาพ และสามารถปรับใช้บนระบบที่มีการติดต่อสื่อสารแบบ $MINs^+$ (Multistage Interconnection Networks⁺) [45] (รูปที่ 3.8) ซึ่งเป็นการติดต่อสื่อสารที่ยืดหยุ่นทำให้โปรเซสเซอร์สามารถประมวลผลได้พร้อมกันโดยไม่เกิดการชนกันของข้อมูลในหลากหลายฟังก์ชัน และประยุกต์ใช้การเรียงลำดับแบบโอบีเอส (OBS : Optimized Bitonic Sort) [54] (หัวข้อที่ 2.4)

เนื้อหาในบทนี้ เป็นการนำเสนอขั้นตอนวิธีทีเอสรีดักชันในแบบขนาน (p โปรเซสเซอร์) ด้วยหลักการเดียวกับกรณี $p = 1$ ได้แก่ 1) ขั้นตอนวิธีอีวิริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพในแบบขนาน (Parallel-GH) 2) ขั้นตอนการแบ่งกลุ่มและการคัดกรองแบบไม่เอนเอียงในแบบขนาน (Parallel-Classification and Filtering) และ 3) ขั้นตอนวิธีเอ็มดีพีแบบขนาน (Parallel-MDP) โดยแต่ละขั้นตอน เน้นการออกแบบให้มีการติดต่อสื่อสารระหว่างหน่วยประมวลผล (p) ที่มีประสิทธิภาพ ซึ่งทำให้ได้ความซับซ้อนด้านเวลาที่ $O(n/p \log_2 p)$



รูปที่ 3.8 ตัวอย่างการติดต่อสื่อสารของ $MINs^+$ สำหรับโปรเซสเซอร์ขนาด $n = 8$

ขั้นตอนวิธีที่ 3.3 ขั้นตอนวิธีทีเอสรีดักชันแบบขนาน (Parallel- $TS_{Reduction}$ Algorithm)

- Step 1: Parallel GH (pGH) algorithm for initial solutions by optimized Bitonic ($P/W, P, W$) sorting and parallel prefix-weight computing for selected/unselected objects, see Section 3.4.1.
- Step 2: Parallel classification and filtering for time-space reduction, see Section 3.4.2.
- Step 3: Parallel MDP algorithm on remaining n' , see Section 3.4.3.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.1 ขั้นตอนวิธีฮิวริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพในแบบขนาน (Parallel-GH)

การพัฒนาขั้นตอนวิธีฮิวริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพในแบบขนาน หรือขั้นตอนวิธีจีเอชแบบขนาน (Parallel-GH : pGH) ได้ประยุกต์ใช้การเรียงลำดับแบบโอบีเอส [54] ร่วมกับการหาผลรวมแบบพรีฟิกซ์ โดยจัดสรรสมาชิกให้แต่ละโพรเซสเซอร์ (Workload) ด้วยขนาด $w = n/p$ เพื่อให้แต่ละโพรเซสเซอร์สามารถประมวลผลในเวลาเท่ากันหรือใกล้เคียงกัน ด้วยการติดต่อสื่อสารระหว่าง p โพรเซสเซอร์ ที่มีประสิทธิภาพ $= O(\log_2 p)$

ขั้นตอนวิธีที่ 3.4 ขั้นตอนวิธีจีเอชแบบขนาน (Parallel-GH Algorithm)

Parallel Greedy Heuristic (pGH) Algorithm: $O(n/p \log_2 p)$

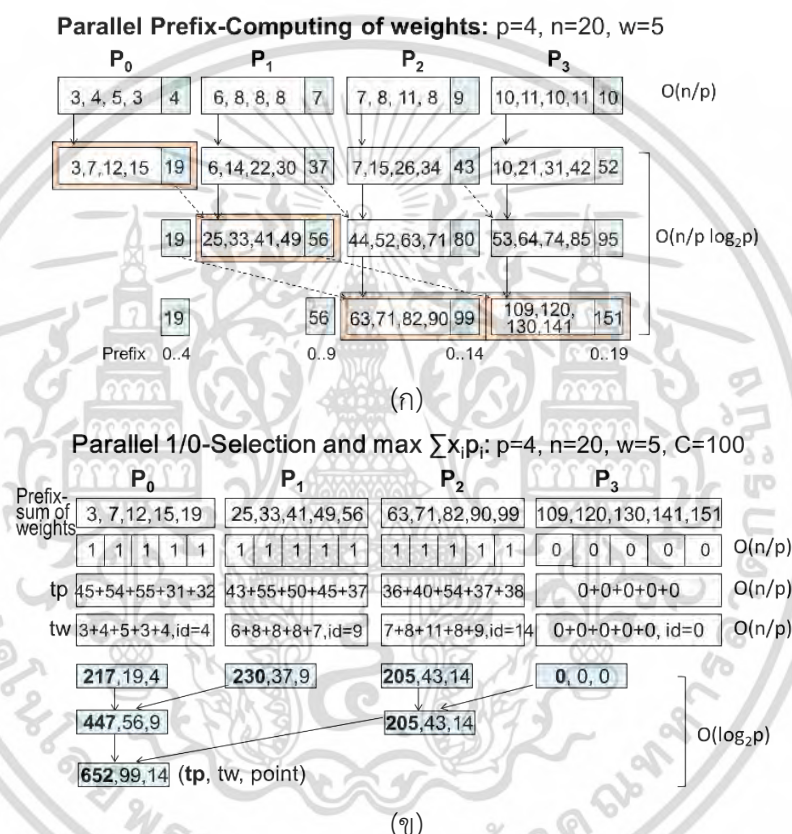
1. Parallel Sort P/W (or P , or W) by optimized Bitonic sorting
2. forall P_i ($i=0, 1, 2, \dots, p-1$) pardo // workload $w=n/p$ per PE
3. for $j=0$ to $n/p-1$ do // initial local prefix-sum
4. $l \leftarrow i(n/p)+j$
5. If ($j=0$) $W[l] \leftarrow w[l]$ else $W[l] \leftarrow w[l]; W[l] \leftarrow W[l]+W[l-1]$
6. end for j
7. end forall
8. for $k=0$ to $\log_2 p - 1$ do // communication ($\log_2 p$ steps)
9. forall P_i ($i=0, 1, 2, \dots, p-1$) pardo
10. for $j=0$ to $n/p-1$ do // local workload= n/p per PE
11. $l \leftarrow i(n/p)+j$
12. if ($l \geq (n/p)2^k$)
13. $W[l] \leftarrow W[l]+W[(i-2^k+1)n/p-1]$ // 1-data exchange
14. end if
15. end for j
16. end forall
17. end for k
-
18. Parallel SelectObjects: $O(n/p)$
19. forall P_i ($i=0, 1, 2, \dots, p-1$) pardo
20. for $j=0$ to $n/p-1$ do
21. $l \leftarrow i(n/p)+j$
22. If ($W[l] \leq C$) $solx[l] \leftarrow '1'$ else $solx[l] \leftarrow '0'$
23. end for j
24. end forall
-
25. ComputeTotalProfit: $O(\log_2 p)$
26. forall P_i ($i=0, 1, 2, \dots, p-1$) pardo
27. for $j=0$ to $n/p-1$ do // local $w=n/p$ per PE
28. $l \leftarrow i(n/p)+j$; $tp[i] \leftarrow tp[i]+(p[l]*solx[l]); tw[i] \leftarrow tw[i]+(w[l]*solx[l])$
29. If ($solx[l]='1'$) $id[i] \leftarrow l$
30. end for j
31. for $k=0$ to $\log_2 p - 1$ do // communication ($\log_2 p$)
32. if ($i\%2^k=0$ and $i+2^k < p$)
33. $tp[i] \leftarrow tp[i] + tp[i+2^k]$ // $tp[0]$: total profit
34. $tw[i] \leftarrow tw[i] + tw[i+2^k]$ // $tw[0]$: total weight
35. $id[i] \leftarrow \max(id[i], id[i+2^k])$ // $id[0]$: max id (or point)
36. end for k
37. end forall

นอกจากนี้เพื่อให้แต่ละโพรเซสเซอร์มีจำนวนสมาชิกที่ประมวลผลใกล้เคียงกัน จึงมีการปรับรูปแบบการคิดปริมาณงานให้มีความยืดหยุ่นตามขนาดของปัญหา ดังแสดงในสมการที่ 3.4 และ 3.5 เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้ในเพื่อการศึกษาเท่านั้น เมื่ออนุญาตให้เผยแพร่เป็นการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$I = i(n/p+1)+j \text{ and } w' = n/p+1 \quad ; \text{ if } (i < n\%p) \quad (3.4)$$

$$I = i(n/p)+(n\%p)+j \text{ and } w' = n/p \quad ; \text{ if } (i \geq n\%p) \quad (3.5)$$

รูปที่ 3.9 แสดงรูปแบบการติดต่อสื่อสารแบบขนานเพื่อคำนวณหาผลรวมแบบพรีฟิกซ์ โดยรูปที่ 3.9 (ก) แสดงรูปแบบการติดต่อสื่อสารเพื่อคำนวณหาค่า tw แบบพรีฟิกซ์ เมื่อ $p < n$ ด้วยความซับซ้อนด้านเวลาที่ $O(n/p \log_2 p)$ และรูปที่ 3.9 (ข) เป็นตัวอย่างการประยุกต์ขั้นตอนวิธีการหาผลรวมค่า tp แบบขนานร่วมกับขั้นตอนวิธีจีเอช (GH) ด้วยความซับซ้อนด้านเวลาที่ $O(n/p) + O(\log_2 p)$



รูปที่ 3.9 การติดต่อสื่อสารแบบขนานเพื่อการคำนวณค่า tw แบบพรีฟิกซ์ที่ $p < n$ เพื่อเลือก/ไม่เลือก ($x_j = 0/1$) และตัวอย่างการหาค่าผลรวม (tp และ tw) และตำแหน่งสุดท้ายในลำดับที่เลือก (point)

รูปที่ 3.9 (ข) แสดงการหาคำตอบของวิธีจีเอชแบบขนาน (pGH) สำหรับ $n = 20, C = 100, p = 4$ ซึ่งจะได้จำนวนสมาชิกในแต่ละโปรเซสเซอร์เป็น $w = 20/4 = 5$ การติดต่อสื่อสารเพื่อคำนวณหาผลลัพธ์ในรอบแรก แต่ละโปรเซสเซอร์จะส่งผลการคำนวณในแบบ $p_0 \leftarrow p_1$ และ $p_2 \leftarrow p_3$ รอบที่สองจะเป็นการส่งคำตอบแบบ $p_0 \leftarrow p_2$ ในแต่ละรอบจะได้ผลลัพธ์ที่ประกอบด้วยค่า tp , tw และชุดรหัสบิตที่แทนผลการเลือกหรือไม่เลือกสิ่งของตามลำดับของสิ่งของในแต่ละกลุ่ม การติดต่อสื่อสารรอบสุดท้ายจะได้ผลลัพธ์ที่ถูกประมวลผลด้วยโปรเซสเซอร์ p_0 ซึ่งประกอบด้วยค่า $soltp$ และค่า $soltw$ ที่ไม่เกินค่า C เอกสารนี้รวมถึงได้เลขดัชนีสิ่งของสุดท้ายที่ถูกเลือกเป็นคำตอบ (point) เพื่อจะนำไปประมวลผลในขั้นตอนถัดไป

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.2 ขั้นตอนการแบ่งกลุ่มและการคัดกรองแบบไม่เอนเอียงในแบบขนาน (Parallel-Classification and Filtering)

ขั้นตอนการแบ่งกลุ่มและคัดกรองแบบไม่เอนเอียงในแบบขนาน ซึ่งแสดงในขั้นตอนวิธีที่ 3.5 มีกระบวนการประมวลผลเช่นเดียวกับขั้นตอนวิธีที่ 3.2 ด้วยจำนวนสมาชิกในภาระงาน $w = n/p$ โดยไม่ต้องมีการติดต่อสื่อสารระหว่างโพรเซสเซอร์ซึ่งมีการจัดสรรสมาชิกให้แต่ละโพรเซสเซอร์อย่างเท่าเทียมกันตามสมการที่ 3.4 และ 3.5 เพื่อการประมวลผลที่สอดคล้องกับขั้นตอนวิธีจีเอชพลัสแบบขนาน (pGH^+) การแบ่งกลุ่ม และการคัดกรองแบบขนาน ด้วยความซับซ้อนด้านเวลา $O(n/p)$

ขั้นตอนวิธีที่ 3.5 ขั้นตอนวิธีจีเอชพลัสและการคัดกรองแบบขนาน (Parallel- GH^+ and Filtering Algorithm)

Improved Parallel Greedy Heuristic (pGH^+) and Filtering

1. apply Algorithm 3.2 to improve GH1-GH3 by $P_0 - P_2$ in $O(1)$
2. call ParallelAssignWeight ($c=5, wx[n]$) // $pGH1^+(P/W)$ $O(n/p)$
3. call ParallelAssignWeight ($c=3, wx[n]$) // $pGH2^+(P)$ $O(n/p)$
4. call ParallelAssignWeight ($c=1, wx[n]$) // $pGH3^+(W)$ $O(n/p)$
5. forall P_i ($i=0, 1, 2, \dots, p-1$) pardo
6. if ($i < n \% p$)
7. $w' \leftarrow n/p + 1$
8. else $w' \leftarrow n/p$
9. for $j=0$ to $w'-1$ do // workload w' per PE
10. If ($i < n \% p$)
11. $l \leftarrow i(w'+1) + j$
12. else
13. $l \leftarrow iw' + (n \% p) + j$
14. $dw[l] \leftarrow wx_1[l] + wx_2[l] + wx_3[l]$
15. assign group (1,2,3, or 4) to object l , according to $dw[l]$
16. end for j
17. end forall

18. ParallelAssignWeight($c, wx[n]$) // $wx_1=5; wx_2=3; wx_3=1$ of GH^+1-3
19. forall P_i ($i=0, 1, 2, \dots, p-1$) pardo
20. if ($i < n \% p$)
21. $w' \leftarrow n/p + 1$
22. else $w' \leftarrow n/p$
23. for $j=0$ to $w'-1$ do // workload w' per PE
24. If ($i < n \% p$)
25. $l \leftarrow i(w'+1) + j$
26. else
27. $l \leftarrow iw' + (n \% p) + j$
28. If ($solx[l] = '1'$)
29. $wx_{GH}[l] \leftarrow c$ // $c = 5, 3, \text{ or } 1$
30. else $wx_{GH}[l] \leftarrow 0$
31. end for j
32. end forall

Step 1

Step 1		P_0					P_1					P_2					P_3					tw	tp	
$[P/W]$	X	8	16	1	9	6	0	4	18	2	17	12	11	14	10	19	3	5	13	7	15			
	W	3	4	5	3	4	6	8	8	8	7	7	8	11	8	9	10	11	10	11	10			
	P	45	54	55	31	32	43	55	50	45	37	36	40	54	37	38	42	39	35	38	34			
pGH1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	99	652	
pGH1+	(by P_0)																0	1	0	0	0	0	100	656

		P_0					P_1					P_2					P_3						
[P]	X	1	4	14	16	18	2	8	0	3	11	5	7	19	10	17	12	13	15	6	9		
	W	5	8	11	4	8	8	3	6	10	8	11	11	9	8	7	7	10	10	4	3		
	P	55	55	54	54	50	45	45	43	42	40	39	38	38	37	37	36	35	34	32	31	tw	tp
	pGH2	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	96	560
	pGH2+ (by P_1)											0	0	1	1	0	0	0	0	1		100	627

		P_0					P_1					P_2					P_3							
[W]	X	8	9	6	16	1	0	12	17	2	4	10	11	18	19	3	13	15	5	7	14			
	W	3	3	4	4	5	6	7	7	8	8	8	8	8	9	10	10	10	11	11	11			
	P	45	31	32	54	55	43	36	37	45	55	37	40	50	38	42	35	34	39	38	54	tw	tp	
	pGH3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	98	640	
	pGH3+ (by P_2)																0	0	0	1	0	0	99	652

Step 2

ep 2		P_0					P_1					P_2					P_3				
X		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
W		6	5	8	10	8	11	4	11	3	3	8	8	7	10	11	10	4	7	8	9
P		43	55	45	42	55	39	32	38	45	31	37	40	36	35	54	34	54	37	50	38
P/W		7	11	6	4	7	4	8	3	15	10	5	5	5	4	5	3	14	5	6	4
pGH1 ⁺		5	5	5	5	5	0	5	0	5	5	5	5	5	0	5	0	5	5	5	0
pGH2 ⁺		3	3	3	3	3	3	0	0	3	3	3	3	0	0	3	0	3	3	3	0
pGH3 ⁺		1	1	1	0	1	0	1	0	1	1	1	1	0	1	0	1	1	1	1	1
dw		9	9	9	8	9	3	6	0	9	9	9	9	6	0	9	0	9	9	9	1

Step 3

p3

$pMDP(n'=6)$

Y	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
X	8	16	1	9	6	0	4	18	2	17	11	14	10	12	19	3	5	13	7	15
W	3	4	5	3	4	6	8	8	8	7	8	11	8	7	9	10	11	10	11	10
P	45	54	55	31	32	43	55	50	45	37	40	54	37	36	38	42	39	35	38	34
P/W	15	14	11	10	8	7.2	6.9	6.3	5.6	5.3	5	4.9	4.6	5.1	4.2	4.2	3.5	3.5	3.5	3.4
GH1 ⁺	5	5	5	5	5	5	5	5	5	5	5	5	5	5	0	5	0	0	0	0
GH2 ⁺	3	3	3	3	0	3	3	3	3	3	3	3	3	0	0	3	3	0	0	0
GH3 ⁺	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
dw	9	9	9	9	6	9	9	9	9	9	9	9	9	6	1	8	3	0	0	0
OptX	1	1	1	1	1	1	1	1	1	1	1	1	1	?	?	?	?	?	?	?

tw tp
83 578

X

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	tw	tp	
OptX	1	1	1	1	1	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	0	100	656

รูปที่ 3.10 ตัวอย่างการประมวลผลของขั้นตอนวิธีทีเอสรีดักชันแบบขนาน (Parallel-TS_{Reduction})

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในรูปที่ 3.10 เป็นการแสดงตัวอย่างภาพรวมของขั้นตอนวิธีที่เอสรีดักชันแบบขนาน Step 1 เป็นการแสดงตัวอย่างการจัดสรรข้อมูลให้แต่ละโพรเซสเซอร์ด้วยขนาด $w = 20/4 = 5$ ในขั้นตอนวิธีจีเอช (Parallel-GH : pGH) ซึ่งได้ประยุกต์ใช้การเรียงลำดับสิ่งของด้วยวิธีโอบีเอส (OBS) [54] และการหาผลรวมแบบพรีฟิกซ์ (Parallel Prefix-Sum) เพื่อหาคำตอบเริ่มต้น และขั้นตอนวิธีจีเอชพลัสแบบขนาน (Parallel-GH⁺ : pGH^+) โดยเริ่มจากเปลี่ยนค่าบิตสุดท้ายที่เป็นคำตอบ (point) ในแต่ละ $pGH1$, $pGH2$ และ $pGH3$ ให้เป็นบิต 0 (นำสิ่งของสุดท้ายที่เลือกออกจากถุงเป้) แล้วให้โพรเซสเซอร์ที่รับผิดชอบสมาชิกถัดไปปรับค่าคำตอบด้วยการประยุกต์ใช้ขั้นตอนวิธีการหาผลรวมพรีฟิกซ์แบบขนาน ซึ่งจะได้ผลลัพธ์ของ $pGH1^+$, $pGH2^+$ และ $pGH3^+$ และใน Step 2 เป็นการคัดกรองแบบขนาน (Parallel Classification and Filtering) โดยการกำหนดค่าน้ำหนักและคำนวณค่าน้ำหนักพลวัต (dw) และคัดกรองตามสมการที่ 3.2-3.3 และใน Step 3 เป็นการประมวลผลสมาชิกที่เหลือจากกระบวนการคัดกรองด้วยขั้นตอนวิธีเอ็มดีพีแบบขนาน ($pMDP$) ซึ่งทำให้ได้คำตอบที่ดีที่สุดสำหรับคำตอบสุดท้าย และทำให้ผลลัพธ์ที่ได้ทั้งหมดเป็นคำตอบที่ดีที่สุด (Optimal Solution)

3.4.3 ขั้นตอนวิธีเอ็มดีพีแบบขนาน (Parallel-MDP)

ขั้นตอนวิธีเอ็มดีพีแบบขนาน (Parallel-MDP : $pMDP$) สำหรับกลุ่มสมาชิกที่ไม่สามารถคัดกรองได้ มีกระบวนการเช่นเดียวกับขั้นตอนวิธีเอ็มดีพี (MDP) ที่นำเสนอในหัวข้อที่ 3.3 ด้วยการเพิ่มกระบวนการการจัดสรรจำนวนสมาชิกให้กับแต่ละโพรเซสเซอร์เพื่อการประมวลผลที่สอดคล้องกับขั้นตอนวิธีเอ็มดีพีแบบขนาน ที่มีการติดต่อสื่อสารระหว่างแต่ละโพรเซสเซอร์ ($p_i, p_j = O(1)$) ดังแสดงในขั้นตอนวิธีที่ 3.6

ขั้นตอนวิธีที่ 3.6 ขั้นตอนวิธีเอ็มดีพีแบบขนาน (Parallel-MDP Algorithm)

```

Parallel MDP Algorithm: return ( $C'$ , soltp', solx'):  $O(C'/p)$ 
1.  do                                     // repeat for  $k' = |selected\ objects|$ 
2.     $w \leftarrow C'/p$                      // w: workload (per PE)
3.    forall  $P_i$  ( $i=0$  to  $p-1$ ) pardo
4.      If ( $i < C'\%p$ )  $w' \leftarrow w+1$  else  $w' \leftarrow w$ 
5.      for  $d=0$  to  $w'-1$  do
6.        If ( $i < C'\%p$ )  $D \leftarrow iw' + d + 1$  else  $D \leftarrow iw' + C'\%p + d + 1$ 
7.         $tp[D] \leftarrow 0$ ;  $x[D] \leftarrow -1$ 
8.      end for  $d$ 
9.    end forall
10.   for  $j=0$  to  $n'-1$  do
11.     forall  $P_i$  ( $i=0$  to  $p-1$ ) pardo
12.       If ( $i < C'\%p$ )  $w' \leftarrow w+1$  else  $w' \leftarrow w$ 
13.       for  $d=w'-1$  down to  $0$  do
14.         if ( $i < C'\%p$ )  $D \leftarrow iw' + d + 1$  else  $D \leftarrow iw' + C'\%p + d + 1$ 
15.         if ( $(D \geq w_j) \& (tp[D-w_j] + p_j > tp[D])$ )  $x[D] \leftarrow j$ ;  $tp[D] \leftarrow tp[D-w_j] + p_j$ 
16.       end for  $d$ 
17.     end for all
18.   end for  $j$ 
19.    $r \leftarrow x[C']$ ;  $n' \leftarrow r$ ;  $C' \leftarrow C' - w_r$ ;  $solx[k] \leftarrow r$ ;  $k \leftarrow k+1$ 
20.   while  $C' > 0$ 
21.     soltp  $\leftarrow tp[C']$ 

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

		P_0					P_1					P_2				P_3				
$w_j=10$ ($x=3$)	d	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	soltp
$p_i=42$ ($Y=0$)	tp	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42
	r	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
$w_j=11$ ($x=5$)																				
$p_i=39$ ($Y=1$)	tp	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42
	r	00	00	00	00	00	00	00	00	00	00	10	10	10	10	10	10	10	10	
$w_j=11$ ($x=7$)																				
$p_i=38$ ($Y=2$)	tp	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42
	r	000	000	000	000	000	000	000	000	000	000	100	100	100	100	100	100	100	100	
$w_j=7$ ($x=12$)																				
$p_i=36$ ($Y=3$)	tp	0	0	0	0	0	0	0	36	36	36	42	42	42	42	42	42	42	78	78
	r	0000	0000	0000	0000	0000	0000	0000	0001	0001	0001	1000	1000	1000	1000	1000	1000	1000	1001	
$w_j=10$ ($x=13$)																				
$p_i=35$ ($Y=4$)	tp	0	0	0	0	0	0	0	36	36	36	42	42	42	42	42	42	42	78	78
	r	00000	00000	00000	00000	00000	00000	00000	00010	00010	00010	10000	10000	10000	10000	10000	10000	10000	10010	
$w_j=9$ ($x=19$)																				
$p_i=38$ ($Y=5$)	tp	0	0	0	0	0	0	0	36	36	38	42	42	42	42	42	42	74	78	78
	r	000000	000000	000000	000000	000000	000000	000000	000100	000100	000001	100000	100000	100000	100000	100000	100000	000101	100100	

รูปที่ 3.11 ตัวอย่างขั้นตอนวิธีเอ็มดีพีแบบขนาน (Parallel-MDP)

สำหรับรูปที่ 3.11 เป็นตัวอย่างแสดงการทำงานของขั้นตอนวิธีเอ็มดีพีแบบขนาน (pMDP) สำหรับข้อมูลที่เหลือจากขั้นตอนวิธีจีเอชพลัส (pGH^+) และการแบ่งกลุ่มและการคัดกรองแบบขนาน (Parallel Classification and Filtering) ที่เหลือสมาชิก $n' = 6$, $C' = 17$, $X' = \{3, 5, 7, 12, 13, 19\}$, $P' = \{42, 39, 38, 36, 35, 38\}$, $W' = \{10, 11, 11, 7, 10, 9\}$ ด้วยโปรเซสเซอร์ $p = 4$ และ $w = C'/p$

ตารางที่ 3.4 ความซับซ้อนด้านเวลาของขั้นตอนวิธีทีเอสรีดักชันแบบขนาน (Parallel-TS_{Reduction})

Functions	$p = 1$	$p = n$	$p < n$
Step 1: Parallel feature sorting Parallel object solution ($x_j=0/1$)	$O(n)$ $O(n)$	$O([\log_2 n]^2)$ $O(\log_2 n)$	$O(n/p \log_2 p)$ $O(n/p \log_2 p)$
Step 2: Parallel classification and filtering	$O(n)$	$O(1)$	$O(n/p)$
Step 3: Parallel Modified DP	$O(C')$	$O(C'/n)$ or $O(1)$ if $C' \leq n$	$O(C'/p)$
Total time complexity	$O(n+C')$	$O([\log_2 n]^2 + C'/n)$	$O(n/p \log_2 p + C'/p)$

ขั้นตอนวิธีทีเอสรีดักชันแบบขนาน (Parallel-TS_{Reduction} Algorithm) เป็นการพัฒนาแนวคิดต่อเนื่องจากขั้นตอนวิธีทีเอสรีดักชัน (TS_{Reduction} Algorithm) แบบอนุกรมที่นำเสนอในบทที่ 3 เพื่อให้สามารถประมวลผลบนระบบมัลติโพรเซสเซอร์ได้ โดยในขั้นตอนแรกเป็นการหาคำตอบเริ่มต้นด้วยขั้นตอนวิธีอีวิริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพในแบบขนานหรือขั้นตอนวิธีจีเอชพลัสแบบขนาน (pGH) ด้วยการเรียงลำดับหลายระดับซึ่งได้ใช้การเรียงลำดับแบบโอบีเอส (OBS) [54] เอกสารนี้ และขั้นตอนวิธีการรวมค่าพรีฟิกซ์แบบขนาน (Parallel Prefix-Sum) ขั้นตอนต่อมา เป็นการปรับปรุงไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลลัพธ์ด้วยขั้นตอนวิธีจีเอชพลัส (pGH^+) และกำหนดค่าน้ำหนักตามความสำคัญของกฎ การคำนวณค่าน้ำหนักพลวัต (dw) แบ่งกลุ่มและคัดกรองสมาชิกของปัญหาในแบบขนาน โดยใช้การจัดสรรจำนวนสมาชิกให้กับแต่ละโพรเซสเซอร์อย่างเท่าเทียมกันเพื่อลดเวลาในการประมวลผล ซึ่งได้นำไปใช้ในขั้นตอนวิธีเอ็มดีพีแบบขนาน ($pMDP$) เช่นกัน ซึ่งจากการพัฒนาขั้นตอนวิธีทีเอสรีดักชันแบบขนาน ทำให้ได้ความซับซ้อนด้านเวลาที่ $O(n+C')$ กรณีหนึ่งโพรเซสเซอร์ และมีความซับซ้อนด้านเวลาที่ $O([\log_2 n]^2 + C'/n)$ กรณีที่จำนวนโพรเซสเซอร์เท่ากับจำนวนสมาชิกของปัญหา ($p = n$) สำหรับกรณีที่โพรเซสเซอร์น้อยกว่าจำนวนสมาชิกของปัญหา ($p < n$) ซึ่งเป็นกรณีที่เป็นไปได้มากที่สุด จะมีความซับซ้อนด้านเวลาที่ $O(n/p \log_2 p + C'/p)$ ดังแสดงในตารางที่ 3.4



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

ขั้นตอนวิธีมัลติดีพีทีลิสต์ทีเอสอาร์สำหรับ ปัญหามัลติเฟิลแนปแซค (0/1-mKP)

ปัญหามัลติเฟิลแนปแซค (0/1-mKP: 0/1-Multiple Knapsack Problem) หรือปัญหาหลายถุงเป้ เป็นปัญหาถุงเป้ที่มีความซับซ้อนมากกว่าปัญหาซิงเกิลแนปแซค (0/1-KP) ด้วยจำนวนของถุงเป้ที่มีมากกว่าหนึ่งถุงเป้และมีหลายขนาดความจุ แต่ด้วยรูปแบบของปัญหาและการแก้ไขปัญหาที่สามารถนำไปประยุกต์ใช้ทั้งในการศึกษาพัฒนาต่อยอดงานวิจัยและนำไปใช้งานได้จริงในหลายศาสตร์ทำให้ปัญหาถุงเป้ประเภทนี้ได้รับการพัฒนาอย่างต่อเนื่อง สำหรับในบทนี้ ผู้วิจัยได้นำเสนอขั้นตอนวิธีมัลติดีพีทีลิสต์ทีเอสอาร์ (multi-DPT-List_{TSR} Algorithm) ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (Robust Unbiased Filtering) สำหรับปัญหามัลติเฟิลแนปแซค (0/1-mKP) โดยเริ่มจากการพัฒนาขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ (DPT-List_{TSR} Algorithm) สำหรับปัญหาซิงเกิลแนปแซค (0/1-KP) ที่ประมวลผลตามขั้นตอนวิธีโปรแกรมเชิงพลวัตแต่พัฒนาให้อยู่ในรูปแบบลิสต์เพื่อลดขนาดของหน่วยความจำและเพิ่มความเร็วในการประมวลผล แล้วจึงพัฒนาใช้กับปัญหามัลติเฟิลแนปแซครวมถึงนำเสนอรูปแบบการเรียงลำดับถุงเป้อย่างมีประสิทธิภาพและพัฒนากระบวนการคัดกรองจากที่เคยนำเสนอในบทที่ 3 ให้มีความแข็งแกร่ง (Robust) มากขึ้น เพื่อเพิ่มประสิทธิภาพในการคัดกรองสิ่งของให้มีความแม่นยำมากขึ้น สามารถคัดเลือกสิ่งของให้แต่ละถุงเป้ได้อย่างเหมาะสมและทำให้ได้มูลค่ารวมของปัญหาเป็นคำตอบที่ดีที่สุดหรือใกล้เคียงคำตอบที่ดีที่สุดได้

4.1 ขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์สำหรับปัญหาซิงเกิลแนปแซค (DPT-List_{TSR} Algorithm for the 0/1-KP)

ขั้นตอนวิธีหนึ่งที่สามารถหาคำตอบที่ดีที่สุดสำหรับปัญหาซิงเกิลแนปแซค (0/1-KP) ได้ คือ ขั้นตอนวิธีโปรแกรมเชิงพลวัต (DP: Dynamic Programming) ซึ่งประมวลผลด้วยอาร์เรย์ขนาด C (สำหรับขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็วและขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ที่ใช้อาร์เรย์ 1 มิติ) และอาร์เรย์ขนาด $n \times C$ (สำหรับขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื้องต้นที่ใช้อาร์เรย์ 2 มิติ) ซึ่งต้องใช้เนื้อที่มากในการประมวลผล โดยเฉพาะหากค่า n และ C มีค่ามาก ๆ ก็จะเป็นต้องต้องใช้เนื้อที่ในการประมวลผลมากตามไปด้วยซึ่งอาจทำให้ไม่สามารถประมวลผลหาคำตอบของปัญหาได้ ด้วยเหตุนี้ผู้วิจัยจึงมีแนวคิดที่จะออกแบบขั้นตอนภายใต้แนวคิดหลักการคำนวณของขั้นตอนวิธีโปรแกรมเชิงพลวัตที่สามารถหาคำตอบที่ดีที่สุดและใช้เนื้อที่ในประมวลผลน้อย ดังนั้น ในหัวข้อนี้จึงเสนอขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบแปรรูป ในชื่อ ขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ (DPT-List_{TSR}: DP transformation to List-based Time-Space Reduction Algorithm) สำหรับปัญหาซิงเกิลแนปแซค (0/1-KP) ที่ได้

เปลี่ยนรูปแบบการเก็บข้อมูลจากอาร์เรย์เป็นแบบลิสต์เพื่อลดขนาดข้อมูลที่ใช้ในการประมวลผล อีกทั้งยังเอกลิสต์เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้เผยแพร่หรือใช้ประโยชน์ในการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สามารถหาคำตอบที่ดีที่สุด โดยขั้นตอนหลักของขั้นตอนวิธีพีทีลิสต์ที่เอสอาร์ประกอบด้วย 2 ขั้นตอน ได้แก่ 1) ขั้นตอนพรีโพรเซสซิง (Pre-processing) เป็นขั้นตอนที่ใช้วิธีการ F-reduction (Forward Reduction) ในการเก็บ Original e-node ซึ่งเป็นโหนดที่มีค่า tp ใหม่โหนดแรกที่เกิดขึ้นของสิ่งของ j ณ น้ำหนัก d เท่านั้น ไว้ใน B-list โดยในขั้นตอนนี้จะใช้หลักการประมวลผลเดียวกับการโปรแกรมเชิงพลวัตแต่จะเก็บโหนดตั้งต้นและโหนดที่สำคัญเท่านั้น และ 2) ขั้นตอนการเอ็กซ์แทร็กกิง (X-tracking) เป็นขั้นตอนที่ใช้ในการแทร็กหา j ที่เป็นคำตอบ (solx) จาก Original e-node ที่เก็บใน B-list โดยใช้วิธีการ B-reduction (Backward Reduction)

สำหรับข้อมูลและฟังก์ชันที่สำคัญของขั้นตอนวิธีพีทีลิสต์ที่เอสอาร์ มีดังนี้

- e-node (Effective node) คือ โหนดตั้งต้นหรือโหนดแรกที่มีค่า tp ใหม่เกิดขึ้นของสิ่งของ j ในตำแหน่งน้ำหนักที่ d ซึ่งเป็นค่าแรกและมีค่าไม่ซ้ำกับโหนดอื่น
- Original e-node คือ e-node ของสิ่งของแต่ละ j ที่ผ่านการ F-reduction ซึ่งเป็นโหนดสำคัญของแต่ละ j แต่จะไม่ใช้โหนดสำคัญของสิ่งของขั้นต่อไป ($j+1$)
- F-List (Forward-List) คือลิสต์ที่เก็บ e-node หรือ Effective node ของสิ่งของแต่ละ j ณ น้ำหนัก d
- B-List (Backward-List) คือกระบวนการในขั้นตอนการ X-Tracking หลังจากเลือกแต่ละค่าของ $x_j = 1$ เพื่อลดบางโหนด(ที่ไม่จำเป็นต่อการแทร็กในส่วนที่เหลือ) เพื่อให้การแทร็กหาคำตอบในส่วนที่เหลือทำได้ง่ายขึ้น
- F-reduction (Forward Reduction) คือกระบวนการลดจำนวน e-node ใน F-list ที่มีคุณลักษณะเป็น Original e-node ไปเก็บใน B-list
- B-reduction (Backward Reduction) คือกระบวนการในการแทร็กหาค่า j หรือสิ่งของที่เป็นชุดคำตอบ (solx) ซึ่งจะใช้ในขั้นตอนการ X-Tracking

4.1.1 ขั้นตอนพรีโพรเซสซิง (Pre-processing)

ขั้นตอนการพรีโพรเซสซิง (Pre-processing) (ขั้นตอนวิธีที่ 4.1) ประกอบด้วย 2 ลิสต์ ได้แก่ 1) F-list $j-1$ เป็นลิสต์ที่ถูกสร้างขึ้นมาเป็นอันดับแรกที่จะมีเพียง e-node ตั้งต้น ($tp=0, d=0$) ก่อนเริ่มการประมวลผล และใช้เป็นลิสต์ตั้งต้นในการเปรียบเทียบค่าตามเงื่อนไขของโปรแกรมเชิงพลวัตเพื่อค้นหา e-node ใหม่ในระหว่างประมวลผลสิ่งของที่ $j=0$ ถึง $j=n-1$ และ 2) F-list j เป็นลิสต์ที่เก็บ e-node ของสิ่งของปัจจุบัน โดย e-node ในแต่ละ F-list j จะได้รับการประมวลผลและเปรียบเทียบสืบทอดมาจาก F-list ที่ $j-1$ ที่คัดลอกผลมาจาก F-list ก่อนหน้า และ e-node ของสิ่งของ j ปัจจุบันจะถูกเก็บเป็น Original e-node ใน B-list L_j เพื่อนำไปใช้ในการหา solx ในขั้นตอนการเอ็กซ์แทร็กกิงต่อไป สำหรับขั้นตอนขั้นตอนพรีโพรเซสซิงนี้สามารถลดเวลาและเนื้อที่ที่ใช้ในการประมวลผลในชุดข้อมูลที่หลากหลาย โดยได้ $O(n^2)$ ในกรณีที่ดีที่สุด (Best case) $O((n^2+nC)/2)$ ในกรณีเฉลี่ย (Average case) และ $O(nC)$ ในกรณีที่แย่ที่สุด (Worst case)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนวิธีที่ 4.1 ขั้นตอนพรีโพรเซสซิง (Pre-processing)

Pre-processing of DPT-List_{TSR}

1. create initial F-list j-1 (for j=0) with one e-node (tp=0,d=0);
2. for (j=0 to n-1) do // to fulfill F-list j of object j
3. initial-copy nodes (cn) of F-list j-1 to F-list j (while (cn.tp < p_j & cn.d < w_j));
4. e-node en = head(F-list j-1);
5. for (each e-node (en) in F-list j-1) do // from head to tail
6. inherit remaining e-node (en) to F-list j (in a proper location) if (it is worth);
7. compute d = en.d+w_j; tp = en.tp+p_j; n1 = en;
8. n1 = decode(n1,d); n2 = n1.next; TP = n1.tp (if n1.d > d < n2.d);
9. if (d ≤ C & tp > TP) add new e-node in F-list j and B-list j;
10. end for (F-list j-1);
11. end for j;
12. soltp = max (en.tp) of B-list j;

ขั้นตอนพรีโพรเซสซิง (Pre-processing) เป็นขั้นตอนการสร้าง สำหรับการสร้าง F-list j และ B-list L_j โดยเริ่มต้นจากการสร้าง F-list j-1 ตั้งต้น (Initial F-list j-1) ที่ประกอบด้วยโหนด tp=0, d=0) จากนั้น จะเป็นการสร้างโหนดในทั้งสองลิสต์ ซึ่งการประมวลผลมีขั้นตอนการสร้าง 2 ขั้นตอนหลัก ดังนี้

ขั้นตอนที่ 1 สร้าง F-list j-1 ตั้งต้นสำหรับใช้ประมวลผลในการสร้างโหนดใหม่ของแต่ละ j โดยคัดลอกโหนดใน F-list j-1 เฉพาะที่มีมูลค่าและน้ำหนักน้อยกว่าสิ่งของปัจจุบัน (cn.tp < p_j & cn.tw < p_j)

ขั้นตอนที่ 2 เพิ่ม e-node ใหม่ใน F-list j ปัจจุบัน โดยพิจารณาแต่ละโหนดจากต้นลิสต์ของ F-list j-1 จากนั้นคำนวณค่า d=en.d+w_j และ tp=en.tp+p_j แล้วพิจารณาว่าโหนดที่มี tp และ d ใหม่สมควรถูกเพิ่มใน F-list j ปัจจุบันหรือไม่ ซึ่งมีหลักการพิจารณา 2 ส่วนคือ d ≤ C และ tp > TP (ซึ่ง TP คือค่า tp ณ ตำแหน่ง d ใน F-list j-1) หากเป็นไปตามเงื่อนไข โหนดใหม่นี้จะถูกเพิ่มต่อท้ายลิสต์ใน F-list j ปัจจุบัน และเพิ่มในตำแหน่งต้นลิสต์ใน B-list L ปัจจุบัน แต่หากไม่เป็นไปตามเงื่อนไขใดเงื่อนไขหนึ่ง โหนดใหม่นี้ก็จะไม่ถูกเพิ่มในลิสต์ใดๆ

ในรูปที่ 4.1 แสดงภาพรวมและการเพิ่มโหนดในแต่ละ j ของขั้นตอนพรีโพรเซสซิง (Pre-processing) ด้วยชุดปัญหาขนาด n = 5, C = 18, P = {4, 10, 6, 9, 8} และ W = {8, 15, 4, 5, 12} โดยในรูปที่ 4.1(ก) แสดงภาพรวมเป้าหมายของขั้นตอนนี้ คือ ขั้นตอนนี้ต้องการปรับเปลี่ยนรูปแบบ (Transform) การเก็บค่าของขั้นตอนวิธีโปรแกรมเชิงพลวัต (รูปซ้าย) ที่ใช้อาร์เรย์ขนาด n×C ซึ่งหากเป็นชุดข้อมูลนี้ต้องใช้เนื้อที่ในการเก็บทั้งหมด 90 สมาชิก เป็นการเก็บแบบลิสต์ (รูปขวา) ซึ่งเก็บเพียงโหนดที่สำคัญของแต่ละ j เท่านั้น โดยจะเก็บไว้ใน B-list ดังในภาพซึ่งใช้ชุดข้อมูลเดียวกันจะเก็บเพียง 7 โหนดเท่านั้น รวมถึงได้คำตอบในส่วน soltp และ soltw ที่อยู่ในโหนดต้นลิสต์ของ L สุดท้าย ซึ่งก็คือโหนดสุดท้ายที่ถูกเพิ่มเข้ามาใน B-list นั่นเอง นอกจากนี้ B-list นี้ จะถูกนำไปใช้ในขั้นตอนการ

เอ็กซ์แทร็กกิ้ง (X-tracking) เพื่อหา solx ต่อไป เพื่อให้เกิดความสมบูรณ์ของผลลัพธ์ของปัญหา

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในรูปที่ 4.1(ข) แสดงตัวอย่างการคำนวณเพื่อสร้างโหนดใหม่และพิจารณาเพื่อเพิ่มโหนดใน F-list และ B-list สำหรับสิ่งของจำนวน $n=5$ ตั้งแต่ $j=0$ ถึง 4 โดยหลังจากสร้าง F-list $j-1$ ตั้งต้น (Initial F-list $j-1$) ที่ประกอบด้วย e-node ($tp=0, d=0$) แล้ว

ณ สิ่งของ $j=0$ ($w_0=8, p_0=4$) สร้าง F-list $j=0$ โดยคัดลอก e-node $(0, 0)$ จาก F-list $j-1$ ไปเก็บใน F-list $j=0$ เพราะ $cn.tp < p_0$ ($0 < 4$) และ $cn.tw < w_0$ ($0 < 8$) จากนั้นเป็นการสร้างและพิจารณาเพื่อเพิ่ม e-node ใน F-list $j=0$ และ original e-node ใน B-list L_0 โดยเริ่มที่ e-node $(0, 0)$ ซึ่งเป็นโหนดแรกของลิสต์ F-list $j-1$ สร้างโหนดใหม่ด้วยสูตร $d = en.d + w_0 = 0 + 8 = 8$ และ $tp = en.tp + p_0 = 0 + 4 = 4$ จึงได้โหนดใหม่ คือ $(4, 8)$ แล้วพิจารณาว่าควรจะนำโหนดใหม่นี้ เพิ่มเป็น e-node ใน F-list $j=0$ หรือไม่ ด้วยเงื่อนไข $d \leq C$ ($8 \leq 18$) (เงื่อนไขเป็นจริง) และเงื่อนไข $tp > TP$ ซึ่ง TP คือค่า tp ณ ตำแหน่งที่ d (8) ใน F-list $j-1$ จึงได้ $TP=0$ จึงทำให้เงื่อนไข $tp > TP$ ($4 > 0$) เป็นจริง ดังนั้น จึงเพิ่มโหนด $tp=4, d=8$ เป็น e-node ใหม่ใน F-list $j=0$ และเพิ่มเป็น original e-node ในตำแหน่งต้นโหนดของ B-list L_0

ณ สิ่งของ $j=1$ ($w_1=15, p_1=10$) สร้างลิสต์ (ตั้งต้น) F-list $j-1$ โดยคัดลอก e-node ทั้งหมด $((0, 0), (4, 8))$ จาก F-list $j=0$ แล้วจึงสร้างลิสต์ F-list $j=1$ โดยเริ่มจากการคัดลอกเฉพาะ e-node $(0, 0)$ และ $(4, 8)$ จาก F-list $j-1$ เพราะเป็น e-node ที่มี $cn.tp < p_1$ ($0, 4 < 10$) และ $cn.tw < w_1$ ($0 < 15$) จากนั้นสร้างโหนดและพิจารณาเพื่อเพิ่ม e-node ใน F-list $j=1$ และ original e-node ใน B-list L_1 โดยเริ่มที่ e-node $(0, 0)$ ซึ่งเป็นโหนดแรกของลิสต์ F-list $j-1$ โดยคำนวณเพื่อสร้างใหม่ด้วย $d = en.d + w_1 = 0 + 15 = 15$ และ $tp = en.tp + p_1 = 0 + 10 = 10$ ดังนั้น จึงได้โหนดใหม่ $(10, 15)$ ต่อมาพิจารณาว่าควรจะนำโหนดใหม่นี้เพิ่มใน F-list และ B-list หรือไม่ ด้วยเงื่อนไข $d \leq C$ ($15 \leq 18$) (เงื่อนไขเป็นจริง) และ $tp > TP$ โดย TP คือค่า tp ณ ตำแหน่ง d (15) ใน F-list $j-1$ ซึ่งมีค่า 4 จึงได้ $TP=4$ จึงทำให้เงื่อนไข $tp > TP$ ($10 > 4$) เป็นจริง ดังนั้น จึงเพิ่มโหนด $(10, 15)$ เป็น e-node ใหม่ในท้ายลิสต์ของ F-list $j=1$ และเพิ่มเป็น original e-node ในต้นลิสต์ของ B-list L_1 ถัดไป ณ e-node $(4, 8)$ ใน F-list $j-1$ สร้างโหนดใหม่ด้วย $d = en.d + w_1 = 8 + 15 = 23$ และ $tp = en.tp + p_1 = 8 + 10 = 18$ จึงได้โหนดใหม่เป็น $(18, 23)$ แต่เมื่อพิจารณาเงื่อนไข $d \leq C$ ($23 \leq 18$) ซึ่งเงื่อนไขเป็นเท็จ จึงทำให้โหนดนี้ไม่ถูกเพิ่มเป็น e-node ใน F-list $j=1$

ณ สิ่งของ $j=2$ ($w_2=4, p_2=6$) สร้างลิสต์ (ตั้งต้น) F-list $j-1$ โดยคัดลอก e-node ทั้งหมด $((0, 0), (4, 8), (10, 15))$ จาก F-list $j=1$ จากนั้นสร้างลิสต์ F-list $j=2$ โดยเริ่มจากการคัดลอกเฉพาะ e-node $(0, 0)$ จาก F-list $j-1$ เพราะเป็น e-node ที่มี $cn.tp < p_2$ ($0 < 6$) และ $cn.tw < w_1$ ($0 < 4$) ต่อมาสร้างโหนดและพิจารณาเพื่อเพิ่ม e-node ใน F-list $j=2$ และ original e-node ใน B-list L_2 โดยเริ่มที่ e-node $(0, 0)$ ของลิสต์ F-list $j-1$ โดยสร้างโหนดใหม่ด้วยสูตร $d = en.d + w_2 = 0 + 4 = 4$ และ $tp = en.tp + p_2 = 0 + 6 = 6$ จึงได้โหนดใหม่เป็น $(6, 4)$ แล้วพิจารณาว่าควรจะนำโหนดใหม่นี้เพิ่มใน F-list และ B-list หรือไม่ ด้วยเงื่อนไข $d \leq C$ ($4 \leq 18$) (เงื่อนไขเป็นจริง) และ $tp > TP$ ซึ่ง TP คือค่า tp ณ ตำแหน่ง d (6) ใน F-list $j-1$ ซึ่งมีค่า 0 จึงได้ $TP=0$ ทำให้เงื่อนไข $tp > TP$ ($6 > 0$) เป็นจริง ดังนั้น จึงเพิ่มโหนด $(6, 4)$ เป็น e-node ใหม่ในส่วนท้ายลิสต์ของ F-list $j=2$ และเพิ่มเป็น original e-node ต้นลิสต์ของ B-list L_2 ถัดไป ณ e-node $(4, 8)$ ใน F-list $j-1$ สร้างโหนดใหม่ด้วย $d = en.d + w_2 = 8 + 4 = 12$ และ $tp = en.tp + p_2 = 8 + 6 = 14$ จึงได้โหนดใหม่เป็น $(12, 12)$ แล้วพิจารณาว่าควรจะนำโหนดใหม่นี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้เพื่อการเรียนการสอนเท่านั้น เมื่ออนุญาตให้เผยแพร่ไปใช้ประโยชน์อื่นใด

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เพิ่มใน F-list และ B-list หรือไม่ ด้วยเงื่อนไข $d \leq C$ ($12 \leq 18$) (เงื่อนไขเป็นจริง) และ $tp > TP$ ซึ่ง TP คือค่า tp ณ ตำแหน่ง d (12) ใน F-list $j-1$ ซึ่งมีค่า 4 จึงได้ $TP=4$ จึงทำให้เงื่อนไข $tp > TP$ ($10 > 4$) เป็นจริง ดังนั้น จึงเพิ่ม (10, 12) เป็น e-node ใหม่ในท้ายลิสต์ของ F-list $j=2$ และเพิ่มเป็น original e-node ใน B-list L_2 ต่อไป ณ e-node (10, 15) ซึ่งเป็นโหนดสุดท้ายในลิสต์ F-list $j-1$ สร้างโหนดใหม่ $d = en.d + w_2 = 15 + 4 = 19$ และ $tp = en.tp + p_2 = 10 + 6 = 16$ จึงได้โหนดใหม่เป็น (16, 19) แต่เมื่อพิจารณาเงื่อนไข $d \leq C$ ($19 \leq 18$) แล้วไม่เป็นจริง จึงทำให้โหนดนี้ไม่ถูกเพิ่มเป็น e-node ใน F-list $j=2$

ณ สิ่งของ $j=3$ ($w_3=5$, $p_3=9$) สร้างลิสต์ (ตั้งต้น) F-list $j-1$ โดยคัดลอก e-node ทั้งหมด ((0, 0), (6, 4), (10, 12)) จาก F-list $j=2$ จากนั้นสร้างลิสต์ F-list $j=3$ โดยเริ่มจากการคัดลอกเฉพาะ e-node (0, 0) และ (6, 4) จาก F-list $j-1$ เพราะเป็น e-node ที่มี $cn.tp < p_3$ ($0, 6 < 9$) และ $cn.tw < w_3$ ($0, 4 < 5$) ต่อมาสร้างโหนดและพิจารณาเพื่อเติม e-node ใน F-list $j=3$ และ original e-node ใน B-list L_3 โดยเริ่ม ณ e-node (0, 0) ซึ่งเป็นโหนดแรกของลิสต์ F-list $j-1$ สร้างโหนดใหม่ด้วยสูตร $d = en.d + w_3 = 0 + 5 = 5$ และ $tp = en.tp + p_3 = 0 + 9 = 9$ จึงได้โหนดใหม่ (9, 5) แล้วพิจารณาว่าควรจะนำโหนดใหม่นี้เพิ่มใน F-list และ B-list หรือไม่ ด้วยเงื่อนไข $d \leq C$ ($5 \leq 18$) (เงื่อนไขเป็นจริง) และ $tp > TP$ ซึ่ง TP คือค่า tp ณ ตำแหน่ง d (5) ใน F-list $j-1$ ซึ่งมีค่า 6 จึงได้ $TP=6$ จึงทำให้เงื่อนไข $tp > TP$ ($9 > 6$) เป็นจริง ดังนั้น จึงเพิ่มโหนด (9, 5) เป็น e-node ใหม่ในท้ายลิสต์ของ F-list $j=3$ และเพิ่มเป็น original e-node ในต้นลิสต์ของ B-list L_3 ถัดไป ณ e-node (6, 4) ใน F-list $j-1$ สร้างโหนดใหม่ด้วย $d = en.d + w_3 = 4 + 5 = 9$ และ $tp = en.tp + p_3 = 6 + 9 = 15$ ได้โหนดใหม่ (15, 9) ซึ่งเมื่อพิจารณาเงื่อนไข $d \leq C$ ($9 \leq 18$) และ $tp > TP$ ($15 > 6$) แล้วเงื่อนไขเป็นจริงทั้งสองเงื่อนไข ดังนั้น โหนด (15, 9) จึงเพิ่มเป็น e-node ใหม่ในท้ายลิสต์ F-list $j=3$ และเพิ่มเป็น original e-node ในต้นลิสต์ของ B-list L_3 ถัดไป ณ e-node (10, 12) ใน F-list $j-1$ ซึ่งเป็น e-node สุดท้าย โดยได้โหนดใหม่คือ (19, 17) และเมื่อพิจารณาเงื่อนไข $d \leq C$ ($19 \leq 18$) และ $tp > TP$ ($19 > 10$) แล้วพบว่าเงื่อนไขเป็นจริง จึงเพิ่มโหนด (19, 17) เป็น e-node ใหม่ในท้ายลิสต์ของ F-list $j=3$ และเพิ่มเป็น original e-node ในต้นลิสต์ของ B-list L_3

ณ สิ่งของ $j=4$ ($w_4=12$, $p_4=8$) สร้างลิสต์ F-list $j-1$ โดยคัดลอก e-node ทั้งหมด ((0, 0), (6, 4), (9, 5), (15, 9), (19, 17)) จาก F-list $j=3$ จากนั้นสร้างลิสต์ F-list $j=4$ โดยเริ่มจากการคัดลอกเฉพาะ e-node (0, 0) และ (6, 4) จาก F-list $j-1$ เพราะเป็น e-node ที่มี $cn.tp < p_4$ ($0, 6 < 8$) และ $cn.tw < w_4$ ($0, 4 < 12$) ต่อมาสร้างโหนดและพิจารณาเพื่อเติม e-node ใน F-list $j=4$ และ original e-node ใน B-list L_4 เริ่มจาก ณ e-node (0, 0) ซึ่งเป็นโหนดแรกของลิสต์ F-list $j-1$ สร้างโหนดใหม่ซึ่งได้ (8, 12) จากนั้นพิจารณาเงื่อนไข $d \leq C$ ($12 \leq 18$) (เงื่อนไขเป็นจริง) และ $tp > TP$ ($8 > 15$) (แต่เงื่อนไขเป็นเท็จ) จึงไม่มี e-node และ original e-node เพิ่มใน F-list $j=4$ และ B-list L_4 ต่อมา ณ e-node (6, 4) สร้างโหนดใหม่ซึ่งได้ (14, 16) เมื่อพิจารณาเงื่อนไข $d \leq C$ ($16 \leq 18$) (เงื่อนไขเป็นจริง) และเงื่อนไข $tp > TP$ ($14 > 15$) (แต่เงื่อนไขเป็นเท็จ) จึงไม่มี e-node และ original e-node เพิ่มใน F-list $j=4$ และ B-list L_4 ถัดไป ณ e-node (9, 5) เป็น e-node ที่ต้องถูกถ่ายทอดไปยัง F-list j (Inherit Remaining e-node) ก่อนสร้างโหนดใหม่ซึ่งได้เป็น (17, 17) เมื่อพิจารณาเงื่อนไข $d \leq C$ ($17 \leq 18$) (เงื่อนไขเป็นจริง) และ $tp > TP$ ($17 > 19$) (แต่เงื่อนไขเป็นเท็จ) จึงไม่มี e-node และ original e-node เพิ่มใน F-list $j=4$ และ B-list L_4 ถัดไป ณ e-node (15, 9) เป็น e-node ที่ต้องถูกถ่ายทอดไปยัง F-list j ก่อนสร้าง

เอกสารนี้เป็นเอกสารสงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่อผู้ใช้ได้เห็นว่าเนื้อหาไม่เหมาะสมต่อการนำ

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โหนดใหม่ซึ่งได้เป็น (23, 21) แต่เมื่อพิจารณาเงื่อนไข $d \leq C$ ($21 \leq 18$) ซึ่งเงื่อนไขเป็นเท็จ จึงไม่มี e-node และ original e-node เพิ่มใน F-list $j=4$ และ B-list L_4 ณ e-node สุดท้ายใน F-list $j=1$ (19, 17) เป็น e-node ที่ต้องถูกถ่ายทอดไปยัง F-list j ก่อนได้โหนดใหม่เป็น (27, 29) แต่เมื่อพิจารณาเงื่อนไข $d \leq C$ ($27 \leq 18$) ซึ่งเงื่อนไขเป็นเท็จ จึงไม่มี e-node และ original e-node เพิ่มใน F-list $j=4$ และ B-list L_4

สำหรับฟังก์ชัน Inherit Remaining e-node ที่แสดงในขั้นตอนวิธีที่ 4.1 (บรรทัดที่ 6) ที่ได้แสดงอยู่ในตัวอย่างในรูปที่ 4.1 (ในกรณี $j=4$) จะมีการแสดงรายละเอียดของฟังก์ชันในหัวข้อ 5.2.1 (รูปที่ 5.5)

	5	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	
0							17																													
1							17												31																	
2			14				17						31													45										
3		15					17	29			32							46														60				
4		15					17	29			32						41	46											58			60				
5		15					17	29			32						41	46						48					58	62						
6		15					28	29			32	42				45	46						54	59				61	62			71		75		
7		15					28	29	30		32	42	43	44	45		46	47	57				59	60	61				69	74				75		
8		16					31				44	45	46			48	58	59	60	61		62	63	73			75	76		77				85		
9		16					31				44	45	46			49	58	59	60	62		63	64	73			76	77		78	79			80	85	
10		16			22	31				38	44	45	46	53			58	59	66	67	68			73	80	81		82	84		85	86	95			
11		16			22	31				38	44	45	46	53	55		58	59	66	68		69	70	77		80	82	83	90		91	92		95	97	
12		16		21	22	31				37	38	44	45	52	53	55		59	65	66	68		74	76	77	79	80	87	89	90		91	98		101	103
13		16		21	22	31				37	38	44	45	52	53	57		59	65	66	68		74	78	79	81		87	89	90	92	93	100	102	103	
14		16		21	22	31				37	38	44	45	52	53	57		59	65	66	68		74	78	79	81		87	89	90	92	93	100	102	103	

(ก)

	5	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40		
0							17																														
1																				31																	
2			14										31														45										
3		15					29				32							46														60					
4																	41																				
5																												58									
6					28							42				45						54	59					61			62			71		75	
7							30					43	44				47	57					60	61		61			69	74							
8		16			31				44	45	46			48	58	59	60	61		62	63	73					75	76		77					85		
9														49							62	63	64				76	77		78	79			80			
10					22				38					53					66	67	68					80	81		82	84		85	86	95			
11															55																					97	
12				21				37			52						59	65				68	69	70	77		82	83	90		91	92			98	101	103
13															57								74	76	79		87	89					92	93	100	102	103
14																																					

(ข)

รูปที่ 4.2 ตัวอย่างแสดงการลดจำนวนโหนดที่ใช้ใน F-list และ B-list สำหรับ $n=15$, $C=40$

จากขั้นตอนวิธีโปรแกรมเชิงข้อของขั้นตอนวิธีที่ 4.1 จะเห็นว่าหลักการของขั้นตอนวิธีนี้ คือการเลือกเก็บเพียงค่าที่สำคัญในการประมวลผลของการโปรแกรมเชิงพลวัตเท่านั้นโดยเก็บในรูปแบบลิสต์ ซึ่งสามารถลดจำนวนข้อมูลที่ใช้ในการประมวลผลได้ ในขณะที่ยังคงได้องค์ประกอบของผลลัพธ์ในส่วนของคุณค่า soltp และ soltw ซึ่งบรรจุอยู่ในโหนดสุดท้ายที่ถูกพิจารณาให้เป็น e-node ที่เป็นคำตอบที่ดีที่สุดได้ ด้วยพื้นฐานการคำนวณและพิจารณาเงื่อนไขตามหลักของการโปรแกรมเชิงพลวัต สำหรับรูปที่ 4.2 ตัวอย่างข้อมูลขนาด $n = 15$, $C = 40$, $P = \{17, 14, 14, 15, 12, 16, 13, 15, 16, 18, 22, 24, 21, 13, 11\}$ และ $W = \{11, 14, 7, 5, 10, 12, 5, 8, 5, 11, 9, 10, 8, 5, 6\}$ ที่แสดงให้เห็นขนาดอาร์เรย์ที่ต้องใช้ในการประมวลผลด้วยขั้นตอนวิธีโปรแกรมเชิงพลวัตที่มีทั้งหมด $25 \times 40 = 600$ รายการ ซึ่งด้วยขั้นตอนวิธีโปรแกรมเชิงข้อของจะถูกลดรูปแบบเป็น e-node 223 รายการ (รูปที่ 4.2 (ก)) และลดลงเหลือเฉพาะ original e-node เพียง 103 โหนดเท่านั้น (รูปที่ 4.2 (ข))

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อใช้ในการเรียนการสอนเท่านั้น ไม่อนุญาตให้เผยแพร่โดยไม่ได้รับอนุญาต

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.2 ขั้นตอนเอ็กซ์แทร็กกิ้ง (X-tracking)

ขั้นตอนเอ็กซ์แทร็กกิ้งของขั้นตอนวิธีดีฟิสต์เอสอาร์ (ขั้นตอนวิธีที่ 4.2) เป็นขั้นตอนที่ใช้สำหรับหาหมายเลขสิ่งของที่ถูกเลือก (x_j) ซึ่งเป็นองค์ประกอบของผลลัพธ์ส่วน $solx$ โดยใช้หลักการเลื่อนซ้าย-บน (Move Left-Up) เพื่อหา x_j ที่ถูกเลือกใน B-list โดยเริ่มตั้งแต่ B-list L_{n-1} ที่มี original e-node ล่าสุดที่เก็บค่า $soltp$ และ $soltw$ (original e-node ล่าสุดจะบรรจุใน B-list L_j ที่เป็น x_j) จากนั้น ค้นหา $x_j=1$ ถัดไปด้วย tp และ tw ที่ $tp = tp-p_j$ และ $tw = tw-w_j$ เลื่อนการค้นหาลงตามลูกศร (Back pointer) ในลิสต์ L ปัจจุบัน ไปใน L ถัดขึ้นไป เพื่อหาว่า L ลิสต์ใดที่มีโหนดในตำแหน่งต้นลิสต์เท่ากับ tp และ tw ล่าสุด หาก tw ของโหนดในตำแหน่งต้นลิสต์มีค่ามากกว่า tw ให้ข้ามลิสต์ L นั้นแล้วเลื่อนไปพิจารณาลิสต์ถัดขึ้นไป ทำเช่นนี้ จนกระทั่ง $tp=0, tw=0$ ซึ่งแสดงว่าได้ $x_j=1$ ครบ ซึ่งจะทำให้ได้คำตอบส่วน $solx$ ของปัญหา

หมายเหตุ หลักการ B-reduction ในขั้นตอนวิธีที่ 4.2 (บรรทัดที่ 4) จะใช้ในการลดบางโหนด (ใน B-list 0 – B-list $j-1$) ที่จะไม่มีส่วนเกี่ยวกับการเอ็กซ์แทร็กกิ้งเพื่อหาคำตอบที่เหลือหลังจากเลือก $x_j=1$ ทำให้สามารถแทร็กหา x ที่เหลือได้ง่ายขึ้น

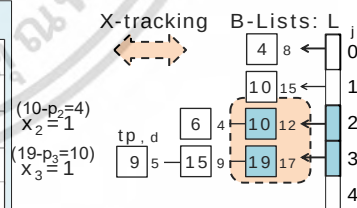
ขั้นตอนวิธีที่ 4.2 ขั้นตอนวิธีการเอ็กซ์แทร็กกิ้ง (X-tracking)

X-tracking (for $solx$) on the B-lists

1. start from B-list L_{n-1} up to L_j ($tp = soltp$); $tp = L_j.tp$; $tw = L_j.tw$;
2. while ($tp > 0$ & $j \geq 0$) do
3. $solx = solx \cup \{j\}$; // union a select object j ($x_j = 1$) with node- j (tp, d);
4. call B-reduction (delete e-node(tp, d) of L_0 to L_{j-1} if e-node. $d \geq$ node- $j.d$);
5. $tp = tp - p_j$; $tw = tw - w_j$; // update remaining (tp, tw);
6. $j = \text{moveLEFT-UP}(n, C, j, tp, tw, L)$; // move to next e-node($tp, d=tw$);
7. end while;

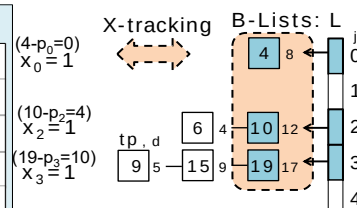
d \ j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	0	0	0	0	0	0	0	4	4	4	4	4	4	4	4	4	4	4
1	0	0	0	0	0	0	0	4	4	4	4	4	4	4	10	10	10	10
2	0	0	0	6	6	6	6	6	6	6	6	10	10	10	10	10	10	10
3	0	0	0	6	9	9	9	9	15	15	15	15	15	15	15	15	19	19
4	0	0	0	6	9	9	9	9	15	15	15	15	15	15	15	15	19	19

(ก)



d \ j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	0	0	0	0	0	0	0	4	4	4	4	4	4	4	4	4	4	4
1	0	0	0	0	0	0	0	4	4	4	4	4	4	4	10	10	10	10
2	0	0	0	6	6	6	6	6	6	6	6	10	10	10	10	10	10	10
3	0	0	0	6	9	9	9	9	15	15	15	15	15	15	15	15	19	19
4	0	0	0	6	9	9	9	9	15	15	15	15	15	15	15	15	19	19

(ข)

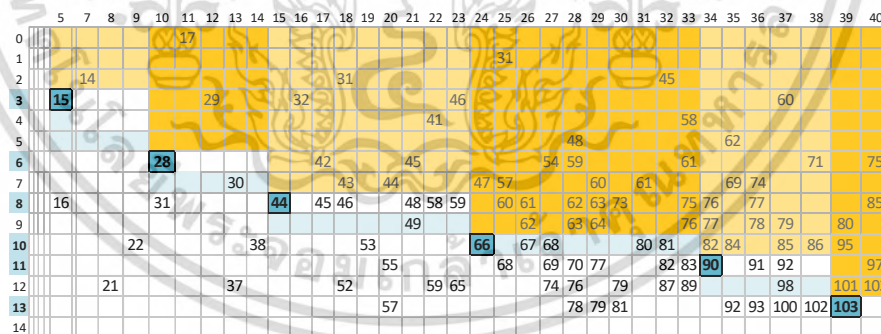


รูปที่ 4.3 ตัวอย่างการแทร็กข้อมูลในลิสต์ด้วยขั้นตอนวิธีการเอ็กซ์แทร็ก สำหรับ $n=5, C=18$

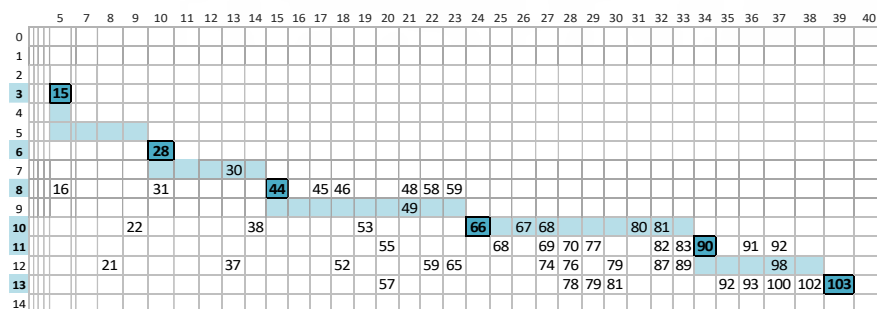
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่ออนุญาตให้นำไปเผยแพร่โดยไม่เสียค่าใช้จ่าย
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4.3 แสดงตัวอย่างเอ็กซ์แพร์กิ้งที่ประมวลผลต่อเนื่องจากตัวอย่าง (n=5, C=18) ในรูปที่ 4.1 โดยรูปฝั่งซ้ายเป็นตัวอย่างการแพร์กหา $x_j=1$ ด้วยขั้นตอนโปรแกรมเชิงพลวัตเปรียบเทียบกับขั้นตอนเอ็กซ์แพร์กิ้งของขั้นตอนวิธีดีพีที่ลิสต์ที่เอสอาร์ที่แสดงในรูปฝั่งขวา

กระบวนการทำงานของขั้นตอนเอ็กซ์แพร์กิ้งของขั้นตอนวิธีดีพีที่ลิสต์ที่เอสอาร์ (รูปที่ 4.3 (ฝั่งขวา)) เริ่มจาก B-list L_4 แต่ลิสต์นี้ไม่มีโหนดใดๆ จึงเลื่อนขึ้นไป ณ B-list L_3 โดยในลิสต์นี้มีโหนดต้นลิสต์เป็น (19, 17) ซึ่งเป็นค่า soltp และ soltw จึงได้หมายเลขสิ่งของที่เป็นคำตอบแรกเป็น $x_3=1$ (ได้ solx={3}) จากนั้นปรับค่า tp, tw ใหม่อด้วย $tp = tp - p_3 = 19 - 9 = 10$ และ $tw = tw - w_3 = 17 - 5 = 12$ นั่นคือโหนดถัดไปที่จะต้องค้นหาคือ (10, 12) ณ B-list L_2 มีโหนดต้นลิสต์ (10, 12) ซึ่งตรงกับ tp, tw ล่าสุด จึงได้ $x_2=1$ (ได้ solx={3, 2}) (พบที่ node-j = (10,12)) ฟังก์ชัน B-reduction จะลบบางโหนดออกจาก B-lists $L_0 - L_1$ ก่อน (ถ้ามี) ณ B-list L_1 มีโหนดต้นลิสต์ (10, 15) ที่ถูกลบด้วยเงื่อนไข node.d (15) > node-j.d (12) ก่อนที่จะแพร์กผ่านไป จากนั้นปรับค่า tp, tw ใหม่อด้วย $tp = tp - p_2 = 10 - 6 = 4$ และ $tw = tw - w_2 = 12 - 4 = 8$ นั่นคือโหนดถัดไปที่จะต้องค้นหาคือ (4, 8) ณ B-list L_1 มีโหนดต้นลิสต์ (10, 15) แต่ด้วยเงื่อนไข node.d $\geq$ tw (15 $\geq$ 8) แสดงว่า ในลิสต์นี้ไม่มีโหนดใดที่เป็นคำตอบ ดังนั้น x_1 จึงไม่ใช่สิ่งของที่ถูกเลือก ($x_1 \neq 1$) นั่นคือไม่ได้เป็นสิ่งของส่วนหนึ่งของคำตอบถัดไป ณ B-list L_0 มีโหนดต้นลิสต์ (4, 8) ซึ่งมีค่าตรงกับ tp, tw ล่าสุด (4, 8) จึงได้ $x_0=1$ (ได้ solx={3, 2, 1}) จากนั้นปรับค่า tp, tw ใหม่อด้วย $tp = tp - p_0 = 4 - 4 = 0$ และ $tw = tw - w_0 = 8 - 8 = 0$ นั่นคือ $tp=0, tw=0$ จึงสิ้นสุดการเอ็กซ์แพร์กิ้ง ดังนั้น จึงได้ solx={3, 2, 1} ซึ่งการเอ็กซ์แพร์กิ้งของขั้นตอนวิธีดีพีที่ลิสต์ที่เอสอาร์ที่อธิบายข้างต้นมีกระบวนการเช่นเดียวกับการแพร์กค่า x_j ของขั้นตอนวิธีโปรแกรมเชิงพลวัต (รูปที่ 4.3 (ฝั่งซ้าย))



(ก)



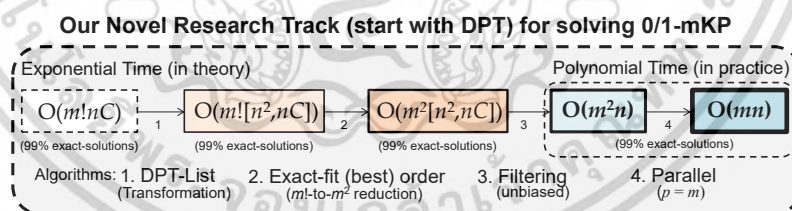
(ข)

รูปที่ 4.4 ตัวอย่างการลดจำนวนข้อมูลด้วย B-reduction ของปัญหขนาด n=15, C=40
เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อใช้ในการศึกษาเท่านั้น ไม่อนุญาตให้นำไปเผยแพร่
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

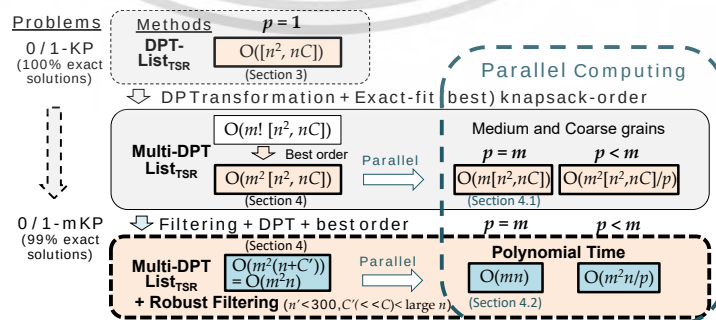
รูปที่ 4.4 แสดงตัวอย่างการลดขนาดของข้อมูลในการแก้หาค่า solx ด้วยการเอ็กแพริกิ้งของขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์พร้อมด้วย B-reduction ใน B-list ซึ่งเป็นขั้นตอนต่อเนื่องจากตัวอย่างในรูปที่ 4.2 ซึ่งเป็นตัวอย่างชุดปัญหาขนาด $n=15$, $C=40$ ซึ่งหลังจากสิ่งของที่ 13 ถูกเลือก ($x_{13}=1$) จะมี original e-node จำนวน 7 โหนด ในช่วง $j=0$ ถึง 12 และ $d=39$ ถึง 40 ((80,39), (95,39), (101,39), (75,40), (85,40), (97,40), (103,40)) ใน B-list ที่ถูกลบทิ้ง ต่อมา หลังจากสิ่งของหมายเลข 11 ถูกเลือก ($x_{11}=1$) จะมี original e-node จำนวน 14 โหนด ในช่วง $j=0$ ถึง 10 และ $d=34$ ถึง 38 ((76,34), (77,34), (82,34), (62,35), (69,35), (84,35), (74,36), (77,36), (78,36), (60,37), (79,37), (85,37), (71,38), (86,38)) ที่อยู่ใน B-list ถูกลบทิ้ง ต่อมาเมื่อ $x_{10}=1$ จะมี original e-node จำนวน 21 โหนด (ในช่วง $j=0$ ถึง 9 และ $d=24$ ถึง 33 ที่อยู่ใน B-list) ถูกลบทิ้ง ต่อมา เมื่อ $x_8=1$ จะมี original e-node จำนวน 8 โหนด (ในช่วง $j=0$ ถึง 7 และ $d=15$ ถึง 23 ที่อยู่ใน B-list) ถูกลบทิ้ง ต่อมา เมื่อ $x_6=1$ จะมี original e-node จำนวน 2 โหนดที่ถูกลบทิ้ง และสุดท้ายเมื่อ $x_3=1$ จะมี original e-node จำนวน 1 โหนดที่ถูกลบทิ้ง ดังนั้นเมื่อสิ้นสุดกระบวนการเอ็กแพริกิ้งจะได้ $\text{solx} = \{13, 11, 10, 8, 6, 3\}$

4.2 ขั้นตอนวิธีมัลติดีพีทีลิสต์ทีเอสอาร์ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่งสำหรับปัญหามัลติเฟิลแนปแซค (multi-DPT-List_{TSR} plus Robust Filtering for 0/1-mKP)

สำหรับการแก้ปัญหามัลติเฟิลแนปแซค (0/1-mKP) ผู้วิจัยได้นำเสนอแนวทางขั้นตอนวิธีแบบใหม่ที่พัฒนาขึ้นในเชิงทฤษฎีสู่การปฏิบัติ (รูปที่ 4.5 (ก)) ที่เริ่มจากการพัฒนาขั้นตอนวิธีภายใต้หลักของขั้นตอนวิธีโปรแกรมเชิงพลวัตด้วยเวลา $O(m!nC)$ สู่ขั้นตอนวิธีทางปฏิบัติด้วยเวลา $O(m^2n)$ ซึ่งใช้วิธีการคัดกรองที่ไม่เอนเอียงแบบแข็งแกร่งโดยที่ยังคงได้คำตอบที่ดีที่สุดที่ 99%



(ก)



(ข)

รูปที่ 4.5 แนวทางการพัฒนางานวิจัยสำหรับปัญหามัลติเฟิลแนปแซค และภาพรวมของขั้นตอน

วิธีมัลติดีพีทีลิสต์ทีเอสอาร์และรูปแบบสำหรับการประมวลผลแบบขนาน

เอกสารนี้เป็นเอกสารทบทวนวิสัยทัศน์สำหรับการแข่งขันเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การแก้ปัญหาหมัลดิเพิลแนปแซค (0/1-mKP) ที่นำเสนอ ประกอบด้วยวิธีที่มีประสิทธิภาพ 3 ส่วน ได้แก่ 1) การประยุกต์ขั้นตอนวิธีดีพีที่ลิสต์ที่เอสอาร์สำหรับปัญหาหมัลดิเพิลแนปแซค 2) การเรียงถุงเป้แบบเหมาะสมพอดีที่มีรูปแบบการจัดเรียงลดลงจาก $m!$ เป็น m^2 ในขณะที่ได้เป้าหมายที่ดีเหมือน การเรียงถุงเป้จำนวน $m!$ 3) การคัดกรองแบบไม่เอนเอียงที่แข็งแกร่งที่มีประสิทธิภาพ ในรูปที่ 4.5 (ข) ได้นำเสนอรูปแบบการลดขนาดที่ใช้สำหรับการประมวลผลแบบขนานด้วยกับจำนวนหน่วยประมวลผล หลายระดับทั้งในแบบ Medium-grain และ Coarse-grain ที่มีหน่วยประมวลผลไม่มากกว่าจำนวน ถุงเป้ในปัญหา ($p \leq m$) ดังนั้น ในหัวข้อนี้จึงเป็นการนำเสนอ 1) ขั้นตอนวิธีหมัลดิเพิลแนปแซคที่เอสอาร์ที่สามารถหาคำตอบที่ดีที่สุดได้ 99% สำหรับปัญหาหมัลดิเพิลแนปแซคที่ประมวลผลในเวลา $O(m^2[n^2, nC])$ จากนั้นเป็นเวลา $O(m[n^2, nC])$ สำหรับการประมวลผลแบบขนานที่มีหน่วยประมวลผลจำนวน $p=m$ 2) การคัดกรองแบบไม่เอนเอียงที่แข็งแกร่งที่ถูกรวมกับขั้นตอนวิธีหมัลดิเพิลแนปแซคที่เอสอาร์ซึ่งสามารถประมวลผลในเวลา $O(m^2(n + C'))$ หรือ $O(m^2n)$ ด้วย $C' (<< C)$ ซึ่งน้อยกว่าปัญหาที่ n จำนวนมาก และเวลา $O(mn)$ ในการประมวลผลแบบขนานที่มีหน่วยประมวลผลจำนวน $p=m$

4.2.1 ขั้นตอนวิธีดีพีที่ลิสต์ที่เอสอาร์สำหรับปัญหาหมัลดิเพิลแนปแซค (Efficient multi-DPT-List_{TSR} for solving 0/1-mKP)

การศึกษาปัญหาหมัลดิเพิลแนปแซคเป็นปัญหาหนึ่งที่ยากมากในกลุ่มปัญหาแนปแซค ซึ่งเป็นปัญหาที่ยากในการหาคำตอบที่ดีที่สุด (Optimal Solution) สำหรับในการศึกษาปัญหานี้ ผู้วิจัยได้เริ่มด้วยการออกแบบคิดค้นพัฒนาขั้นตอนวิธีภายใต้พื้นฐานของขั้นตอนวิธีโปรแกรมเชิงพลวัตสำหรับ ปัญหาซิงเกิลแนปแซคเนื่องจากเป็นขั้นตอนวิธีหนึ่งที่สามารถหาคำตอบที่ดีที่สุดได้ จึงได้ขั้นตอนวิธีดีพีที่ ลิสต์ที่เอสอาร์สำหรับปัญหาซิงเกิลแนปแซค (หัวข้อที่ 4.1) ซึ่งสามารถนำมาใช้ในการหาคำตอบแต่ละ ถุงเป้ (Internal Solution) ของปัญหาหมัลดิเพิลแนปแซค นอกจากนี้ ด้วยปัญหานี้ที่มีหลายถุงเป้จึงได้ นำการเรียงถุงเป้มาใช้ ซึ่งหากมีการเรียงถุงเป้จำนวน $m!$ รูปแบบ จะทำให้ได้คำตอบโดยรวม (External Solution) เป็นคำตอบที่ดีที่สุดไม่น้อยกว่า 99% ในเวลา $O(m! [n^2, nC])$ แต่หากปัญหามีขนาดใหญ่ จะทำให้ใช้เวลาในการประมวลผลมากขึ้นตามขนาดปัญหา ดังนั้น ผู้วิจัยจึงได้เสนอรูปแบบการเรียงถุงเป้ 2 แบบ คือ 1) การเรียงถุงเป้แบบ 9 อันดับสูงสุด ที่ใช้เวลา $O(m[n^2, nC])$ ซึ่งเหมาะสมกับชุดข้อมูล ปกติ (Regular Dataset) และ 2) การเรียงถุงเป้แบบเหมาะสมพอดี ที่ใช้เวลา $O(m^2[n^2, nC])$ ซึ่ง เหมาะสมกับชุดข้อมูลไม่ปกติ (Irregular Dataset)

1. การเรียงถุงเป้แบบเก้าอันดับสูงสุด (Top Nine Knapsacks Orders for Regular Datasets)

การเรียงถุงเป้แบบเก้าอันดับเป็นส่วนหนึ่งที่ใช้ในขั้นตอนวิธีหมัลดิเพิลแนปแซคที่เอสอาร์ ที่เหมาะสมกับข้อมูลปกติ ในแต่ละการเรียงถุงเป้จะพิจารณาทั้งความจุและหมายเลขถุงเป้ ตัวอย่างเช่น ในชุดปัญหามีถุงเป้จำนวน $m=5$, $C=\{66, 26, 80, 96, 70\}$, $C_i=\{1, 2, 3, 4, 5\}$ ซึ่งจะมีรูปแบบการ เรียงถุงเป้ 9 รูปแบบ คือ เรียงจากน้อยไปมาก (Increasing) เรียงจากมากไปน้อย (Decreasing)

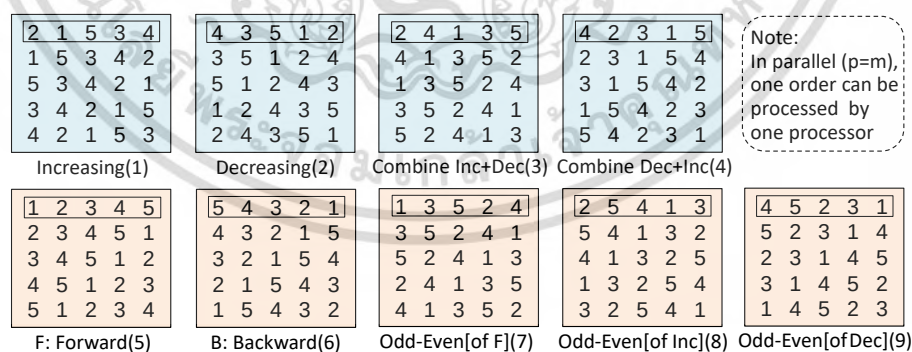
เอกสารนี้เป็นเอกสารลิขสิทธิ์สำหรับการใช้ในเพื่อการศึกษาเท่านั้น เมื่อเผยแพร่ให้คนอื่นใช้โดยไม่ได้รับอนุญาตถือว่าผิดกฎหมาย
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เรียงแบบน้อยไปมากสลับกับแบบมากไปน้อย (Combining Inc-Dec) เรียงแบบมากไปน้อยสลับกับแบบน้อยไปมาก (Combining Dec-Dec) เรียงตามหมายเลขถุงเป้แบบก้าวหน้า (Forward (F)) เรียงตามหมายเลขถุงเป้แบบถอยหลัง (Backward (B)) เรียงเลขคี่เลขคู่สลับกันตามหมายเลขถุงเป้แบบก้าวหน้า (Odd-Even (of F)) การเรียงถุงเป้สลับเลขคี่เลขคู่ตามน้ำหนักจากน้อยไปมาก (Odd-Even (of Inc)) และ การเรียงถุงเป้สลับเลขคี่เลขคู่ตามน้ำหนักจากมากไปน้อย (Odd-Even (of Dec)) ดังแสดงในรูปที่ 4.6 นอกจากนี้ ได้ปรับปรุงรูปแบบการเรียงถุงเป้ด้วยการสับเปลี่ยนแบบลาตินสแควร์ (Latin Square Design: LS) ด้วย m รูปแบบ ซึ่งสามารถได้คำตอบที่ดีที่สุดได้ 99% โดยในทางปฏิบัติการเรียงถุงเป้เพียงส่วนหนึ่งคือเพียง 9 รูปแบบของการเรียงถุงเป้ที่สามารถรักษาเวลาการประมวลผลได้ที่ $O(m[n^2, nC])$ สำหรับ $m > 9$

ขั้นตอนวิธีที่ 4.3 ขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ด้วยการจัดเรียงถุงเป้หนึ่งในเก้ารูปแบบ (Multi-DPT-List_{TSR} for one order Algorithm)

Multi-DPT-List_{TSR} for one order

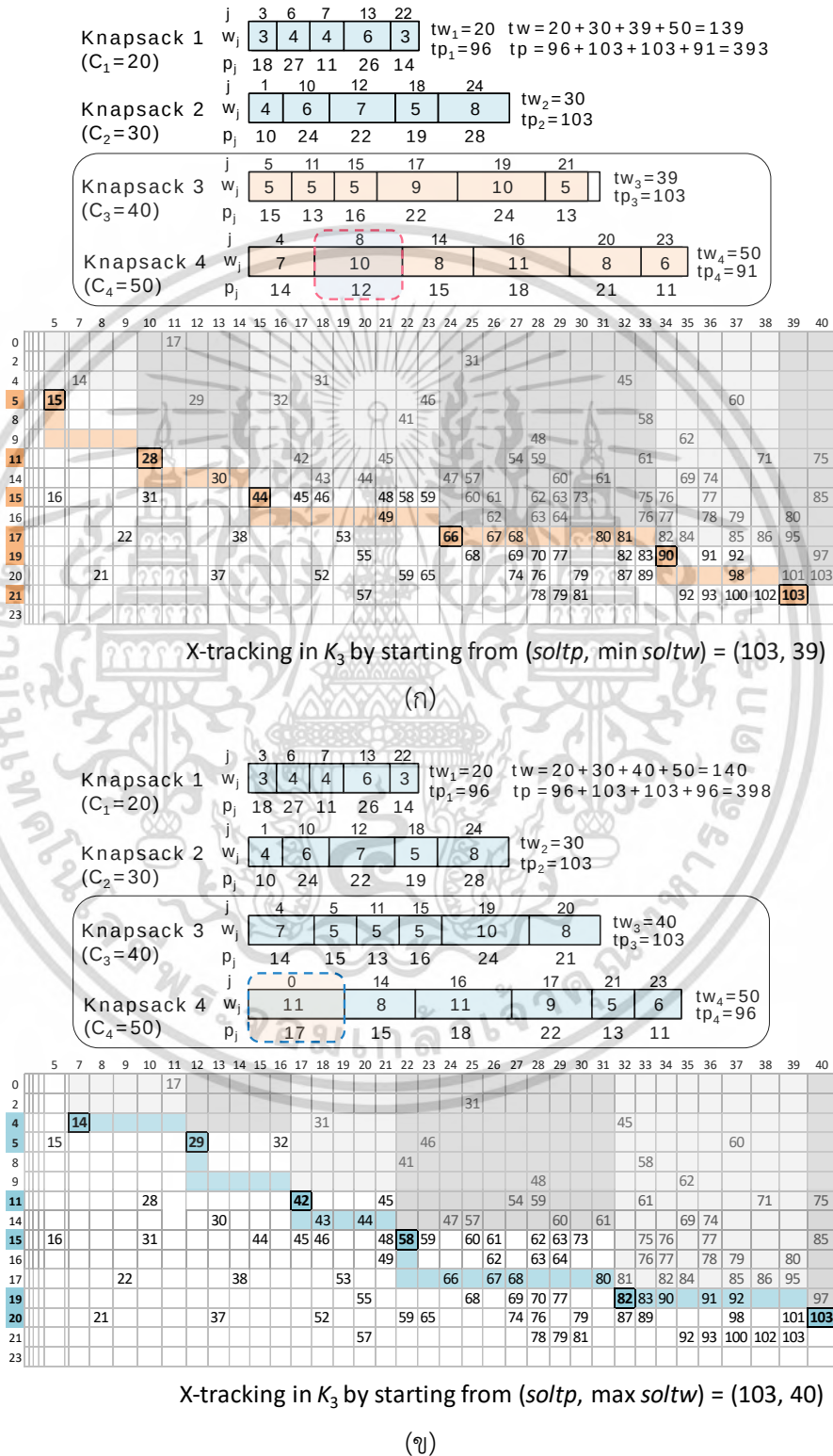
1. $n^* = n$;
2. for ($i=1$ to m) do
3. apply DPT-List_{TSR} (n^* objects) on knapsack i (K_i);
4. call Algorithm 7; // preprocessing (of DPT-List)
5. if ($i < m$) start X-tracking at ($\max tp$, $\max tw$);
6. else ($i = m$) start X-tracking at ($\max tp$, $\min tw$);
7. call Algorithm 8; // X-tracking (of DPT-List)
8. Total profit = Total profit + $\max tp$;
9. update n^* (exclude k_i selected objects of knapsack i);
10. end for i ;
11. return Total profit.



รูปที่ 4.6 ตัวอย่างการเรียงถุงเป้แบบเก้าอันดับสูงสุดร่วมกับลาตินสแควร์สำหรับปัญหา 5 ถุงเป้

ขั้นตอนวิธีที่ 4.3 เป็นการแก้ไขปัญหามัลติเพิลแนปแซคสำหรับการจัดเรียงหนึ่งในเก้ารูปแบบ นอกจากนี้ในบางกรณีมีหลาย soltw ที่มี soltp (ที่ได้สิ่งของที่ถูกละเลือก (Xs) แตกต่างกัน) นั่นคือ คำตอบที่ได้ไม่ได้มีเพียงหนึ่งค่า (Nonunique solution Xs) ในแต่ละถุงเป้ สำหรับปัญหาหนึ่งถุงเป้ การเอ็กซ์แทร็กก็จะต้องเริ่มการแทร็กที่ ($soltp$, $\min soltw$) หรือ ($soltp$, $\max soltw$) สำหรับ Xs ที่แตกต่างกัน เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับในปัญหามัลติเฟิลแนปแซค ถุงเป้ที่ $i (\leq m-1)$ ควรเริ่มการเอ็กซ์แทร็กกิ้งที่ $(soltp, \max soltw)$ ซึ่งจะทำให้ถุงเป้ที่เหลือได้ผลลัพธ์ที่ดี ในขณะที่ถุงเป้สุดท้ายจะต้องเริ่มเอ็กซ์แทร็กกิ้งที่ $(soltp, \min soltw)$ (หมายเหตุ ค่า $soltp$ และ $soltw$ ของแต่ละถุงเป้ได้มาจากขั้นตอนพรีโพรเซสซิง)



ตัวอย่างชุดปัญหา $n = 25$, $m = 4$, $C = (20, 30, 40, 50)$, $P = \{17, 10, 14, 18, 14, 15, 27, 11, 12, 16, 24, 13, 22, 26, 15, 16, 18, 22, 19, 24, 21, 13, 14, 11, 28\}$ และ $W = \{11, 4, 14, 3, 7, 5, 4, 4, 10, 12, 6, 5, 7, 6, 8, 5, 11, 9, 5, 10, 8, 5, 3, 6, 8\}$ ในถุงเป้ที่ K_1 ถึง K_2 มีเพียงผลลัพธ์เดียว (Unique Result) แต่ทำให้เกิดผลลัพธ์หลายค่า (Nonunique Result) ในถุงเป้ที่ K_3 ($C_3 = 40$, $n^* = 15$, $j = \{0, 2, 4, 5, 8, 9, 11, 14, 15, 16, 17, 19, 20, 21, 23\}$) ดังนั้น ในการเอ็กซ์แพทริกกิ่งของถุงเป้ที่ K_3 จะแสดงการเปรียบเทียบให้เห็นผลลัพธ์ที่แตกต่างของแนวคิดการเอ็กซ์แพทริกกิ่งทั้ง 2 แบบคือ ถ้าเริ่มจาก ($solt_p$, $\min solt_w$) จะทำให้ได้ผลลัพธ์รวมเป็น 393 ($solt_p = 393$) แต่ถ้าเริ่มจาก ($solt_p$, $\max solt_w$) จะทำให้ได้ผลลัพธ์รวม 398 ($solt_p = 398$) ดังแสดงในรูปที่ 4.7

รูปที่ 4.7 (ก) เป็นการเอ็กซ์แพทริกกิ่งในถุงเป้ K_3 ที่เริ่มจาก ($solt_p$, $\min solt_w$) คือ (103, 39) ซึ่งมี $\min solt_w = 39$ ทำให้ผลลัพธ์โดยรวม $solt_p=393$ แต่ในรูปที่ 4.7 (ข) เป็นการเอ็กซ์แพทริกกิ่งในถุงเป้ K_3 ที่เริ่มจาก ($solt_p$, $\max solt_w$) ซึ่งทำให้ผลลัพธ์โดยรวม $solt_p=398$ ซึ่งเริ่มเอ็กซ์แพทริกกิ่งจากผลลัพธ์ที่ $solt_p=103$, $solt_w=40$ ซึ่งมี $\max solt_w = 40$ จึงนำไปสู่ผลลัพธ์ของ K_4 ที่เลือกสิ่งของที่ $j = 0$ ($w_0 = 11$, $p_0 = 17$) แทนที่การเลือกสิ่งของที่ $j = 8$ ($w_8 = 10$, $p_8 = 12$) สำหรับแนวคิดในการประมวลผลแบบขนานที่ $p=m$ สามารถกำหนดการเรียงถุงเป้ต่อหนึ่งหน่วยประมวลผลซึ่งมีการสับเปลี่ยนมากที่สุดจำนวน m รูปแบบ ซึ่งการคำนวณจะสามารถประมวลผลพร้อมกันในเวลาเดียวกันก่อนที่จะเลือกคำตอบที่ดีที่สุด

2. การเรียงถุงเป้แบบเหมาะสมพอดี (The Exact-fit (Best) Knapsack Order for Regular and Irregular Datasets)

สำหรับชุดข้อมูลที่ผิดปกติ หากต้องการได้คำตอบที่ดีที่สุดอย่างน้อย 99% จะต้องใช้การเรียงถุงเป้จำนวน $m!$ รูปแบบซึ่งใช้เวลา $O(m! [n, nC])$ ที่ m จำนวนน้อย แต่หากต้องการลดเวลาในขณะที่ยังรักษาความแม่นยำของผลลัพธ์ที่ได้ในปัญหาที่มีจำนวนถุงเป้ (m) จำนวนมาก ผู้วิจัยจึงเสนอการเรียงถุงเป้แบบเหมาะสมพอดี (Exact-fit (Best) Knapsack-order) ดังแสดงในขั้นตอนวิธีที่ 4.4 ซึ่งใช้เวลา $O(m^2[n^2, nC])$ ที่ส่งผลดีทั้งในแต่ละถุงเป้ (Internal Effect) และผลกระทบต่อถุงเป้อื่น (External Effect) ซึ่งทำให้ผลลัพธ์ภาพรวมเป็นคำตอบที่ดีที่สุดด้วย

การประมวลผลภาพรวม จะใช้ขั้นตอนวิธีดีฟิทีลิสต์ที่เอสอาร์ที่ใช้กับปัญหาซิงเกิลแนปแซคเพื่อหาผลลัพธ์ของแต่ละถุงเป้ก่อน แล้วจึงเลือกหมายเลขถุงเป้ (K_i) ที่จะเอ็กซ์แพทริกกิ่งด้วยการเรียงถุงเป้แบบเหมาะสมพอดี ที่ $\text{Best exact-fit}_i = \min(\text{Fit}_i)$ และ $\text{Fit}_i = C_i - TW_i$ โดย $\forall i \leq m$ ดังตัวอย่างในรูปที่ 4.8 ที่แสดงตัวอย่างการเรียงถุงเป้แบบเหมาะสมพอดีกับชุดปัญหา $m = 5$, $C = (66, 26, 80, 96, 70)$, $n = 33$, $P = \{18, 44, 7, 21, 22, 29, 42, 24, 36, 17, 13, 23, 12, 25, 15, 41, 15, 19, 33, 5, 8, 18, 28, 25, 12, 30, 19, 14, 48, 25, 16, 23, 25\}$, และ $W = \{6, 12, 16, 12, 14, 14, 5, 12, 12, 15, 10, 17, 14, 9, 19, 5, 7, 12, 8, 14, 14, 15, 14, 12, 7, 6, 13, 15, 10, 14, 8, 12, 10\}$ ซึ่งได้ลำดับถุงเป้เป็น (2, 4, 1, 5, 3) ที่ทำให้ได้คำตอบโดยรวมที่ดีที่สุด ($solt_p = 726$) นอกจากนี้ เมื่อมีข้อมูลบางชุดที่มีค่า $\min(\text{Fit}_i)$ ใน K_i ถึง K_{i^*} ที่เท่ากัน ก็สามารถตัดสินใจเลือก K_i ที่เหมาะสมได้ ด้วยเงื่อนไขที่แสดงใน Step 2 ของขั้นตอนวิธีที่ 4.4

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนวิธีที่ 4.4 ขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์โดยใช้กลยุทธ์การเรียงถุงเป้ที่ดีที่สุด
(multi-DPT-List_{TSR} with exact-fit (best) order))

- Step 1: apply DPT-List_{TSR} for (TP_i, TW_i) on each of m knapsacks
in $O(m[n^2, nC])$ and $O([n^2, nC])$ in parallel ($p = m$).
- Step 2: select best $K_i = \min(\text{Fit}_i)$; $\text{Fit}_i = C_i - TW_i$ ($i = 1, 2, \dots, m$).
if (there are equal $\min(\text{Fit}_i)$ s), select best $K_i = \min(C_i)$;
if (there are equal $\min(C_i)$ s), select best $K_i = \max(TP_i)$;
- Step 3: update unselected $n^* = n - k$ and $m' = m - 1$.
- Step 4: repeat step 1 – 3 on n^* and m' until $m' = 1$.

Note: In Step 2, for critical $\min(\text{Fit}_i)$ each of 3-policies are applied.

Policy 1: (see Step 2 in Algorithm 12), the most effective policy.

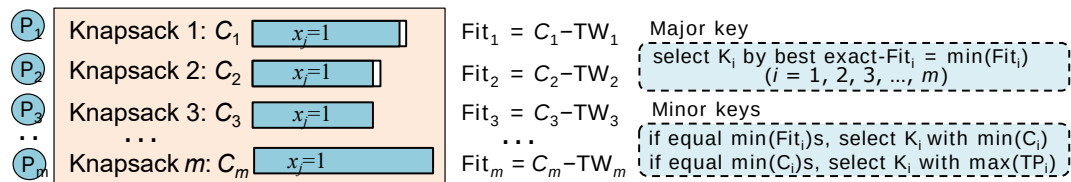
Policy 2: if (there are equal $\min(\text{Fit}_i)$ s), select best $K_i = \max(TP_i/TW_i)$;
if (there are equal $\max(TP_i/TW_i)$ s), select best $K_i = \min(C_i)$;
if (there are equal $\min(C_i)$ s), select best $K_i = \max(TP_i)$;

Policy 3: if (there are equal $\min(\text{Fit}_i)$ s), select best $K_i = \max(TP_i)$;
if (there are equal $\max(TP_i)$ s), select best $K_i = \min(C_i)$;

ขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ที่ใช้การเรียงถุงเป้แบบเหมาะสมพอดี เริ่มจากการประมวลผลด้วยขั้นตอนวิธีดีพีที่ลิสต์ที่เอสอาร์กับสิ่งของทั้งหมดกับแต่ละถุงเป้ แล้วเลือกถุงเป้ที่เหลือผลต่างน้ำหนักของถุงเป้ที่น้อยที่สุด ($\text{Fit} = C_i - TW_i$) แต่หากมีถุงเป้ที่มีผลต่างน้ำหนักของถุงเป้ที่น้อยที่สุดเท่ากันหลายถุงเป้ให้เลือกถุงเป้ที่ C_i น้อยที่สุด แต่หากยังมีถุงเป้ที่ C_i น้อยที่สุดเท่ากันหลายถุงเป้ให้เลือกถุงเป้ที่ TP_i มากที่สุด เมื่อเลือกถุงเป้ในแต่ละรอบแล้วให้ปรับค่า n และ m ที่เหลือ แล้ววนรอบทำเช่นเดิมอีกครั้งจนได้คำตอบทุกถุงเป้ ดังแสดงในรูปที่ 4.8 (ก)

สำหรับรูปที่ 4.8 (ข) เป็นการแสดงตัวอย่างขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ที่ใช้การเรียงถุงเป้แบบเหมาะสมพอดีที่ได้ลำดับถุงเป้เป็น 2, 4, 1, 5, 3 ด้วยจำนวน $m(m+1)/2 = 15$ รอบ เริ่มจากใช้ขั้นตอนวิธีดีพีที่ลิสต์ที่เอสอาร์กับสิ่งของ 33 ชิ้น ทั้ง 4 ถุงเป้ ซึ่งจะได้ให้ค่า TP และ TW ของแต่ละถุงเป้ (ซึ่งจะนำมาหาค่า Fit เพื่อเลือกถุงเป้ที่จะแทรกหาสิ่งของสำหรับบรรจุลงถุงเป้นั้นในขั้นต่อไป) ในรอบแรก (1st select K_i) จะเห็นว่ามีถุงเป้ที่ K_2 และ K_3 ที่มีค่า $\text{Fit} = C_i - TW_i = 0$ เท่ากัน แต่ถุงเป้ K_2 มีความจุน้อยกว่า K_3 ($(C_2 < C_3) = (26 < 80)$) ดังนั้นจึงเลือกถุงเป้ K_2 (นั่นคือ เลือกบรรจุเต็มในถุงเป้ที่เล็กกว่าก่อน) (ขณะนี้เหลือถุงเป้ K_1, K_3, K_4, K_5) ดังนั้น ในรอบแรก จึงได้ $\sum TP_{i(i=2)} = 61$ ต่อมาในรอบที่สอง (2nd select K_i) ถุงเป้ที่ถูกเลือกคือ K_4 เพราะมีค่า Fit ที่น้อยที่สุด ($=0$) จากทั้ง 4 ถุงเป้ที่เหลือ จึงได้ $\sum TP_{i(i=2,4)} = 161 + 261 = 422$ ในรอบที่สาม (3rd select K_i) มีถุงเป้ที่มี $\min \text{Fit}$ เท่ากัน 2 ถุงเป้ คือ K_1 กับ K_3 ในที่นี้จึงเลือก K_1 เพราะมีความจุน้อยกว่า K_3 ($(C_1 < C_3) = (66 < 80)$) รอบนี้จึงได้ TP รวม ได้ $\sum TP_{i(i=2,4,1)} = 161 + 261 + 126 = 548$ ในรอบที่สี่ (4th select K_i) ถุงเป้ที่ถูกเลือกคือ K_5 เพราะมีค่า Fit ที่น้อยที่สุดในรอบนี้จึงได้ TP รวมที่ $\sum TP_{i(i=2,4,1,5)} = 61 + 261 + 126 + 97 = 645$ ในรอบที่ห้า (5th select K_i) ซึ่งเหลือถุงเป้เดียวคือ K_3 จึงได้ TP รวมทั้งหมดจากถุงเป้ 5 ถุงเป้เป็น $\sum TP_{i(i=2,4,1,5,3)} = 161 + 261 + 126 + 97 + 81 = 726$

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Note: call $DPT-List_{TSR}(n, C_i)$ to find the (exact) TP_i and TW_i (in each knapsack)

(ก)

1st selected K_i (in the exact-fit (best) order) by $p=m$ (in parallel), $m=5$

K1: $C_1 = 66$	$[0, 1, 6, 8, 15, 18, 23, 28]$	$TP_1=292, TW_1=64$ ($Fit_1=C_1-TW_1=2$)	equal minFits select K2 ($TP_2=161$) ($Fit_2=0$) $C_2=26$ ($Fit_3=0$) $C_3=80$
K2: $C_2 = 26$	$[6, 15, 23, 28]$	$TP_2=161, TW_2=26$ ($Fit_2=C_2-TW_2=0$)	
K3: $C_3 = 80$	$[0, 1, 6, 8, 13, 15, 16, 18, 25, 28]$	$TP_3=332, TW_3=80$ ($Fit_3=C_3-TW_3=0$)	
K4: $C_4 = 96$	$[0, 1, 6, 8, 13, 15, 18, 23, 25, 28]$	$TP_4=367, TW_4=95$ ($Fit_4=C_4-TW_4=1$)	
K5: $C_5 = 70$	$[1, 6, 8, 13, 15, 18, 25, 28]$	$TP_5=299, TW_5=67$ ($Fit_5=C_5-TW_5=3$)	

2nd selected K_i (in the exact-fit (best) order) by $p=m'=m-1$ (in parallel), $m'=4$

K1: $C_1 = 66$	$[0, 1, 8, 13, 18, 30, 32]$	$TP_1=197, TW_1= 65$ (Fit ₁ = $C_1-TW_1=1$)	select K4 (Fit ₄ =0) (TP ₄ =261)
K3: $C_3 = 80$	$[0, 1, 5, 8, 13, 18, 30, 32]$	$TP_3=226, TW_3= 79$ (Fit ₃ = $C_3-TW_3=1$)	
K4: $C_4 = 96$	$[0, 1, 7, 8, 13, 16, 18, 23, 30, 32]$	$TP_4=261, TW_4= 96$ (Fit ₄ = $C_4-TW_4=0$)	
K5: $C_5 = 70$	$[0, 1, 8, 13, 18, 23, 32]$	$TP_5=206, TW_5= 69$ (Fit ₅ = $C_5-TW_5=1$)	

3rd selected K_i (in the exact-fit (best) order) by $p=m'=m-2$ (in parallel), $m'=3$

K1: $C_1 = 66$	$[3, 5, 22, 29, 31]$	$TP_1=126, TW_1=66$ ($Fit_1=C_1-TW_1=0$)	equal minFits select K1 ($TP_1=126$) ($Fit_1=0$) $C_2=66$ ($Fit_3=0$) $C_3=80$
K3: $C_3 = 80$	$[3, 4, 5, 22, 29, 31]$	$TP_3=148, TW_3=80$ ($Fit_3=C_3-TW_3=0$)	
K5: $C_5 = 70$	$[4, 5, 22, 29, 31]$	$TP_5=127, TW_5=68$ ($Fit_5=C_5-TW_5=2$)	

4th selected K_i (in the exact-fit (best) order) by $p=m'=m-3$ (in parallel), $m'=2$

K3: $C_3 = 80$	$[4, 11, 17, 21, 24, 26]$	$TP_3=113, TW_3=78$ ($Fit_3=C_3-TW_3=2$)	select K5 ($TP_5=97$) ($Fit_5=0$)
K5: $C_5 = 70$	$[4, 10, 12, 17, 24, 26]$	$TP_5=97, TW_5=70$ ($Fit_5=C_5-TW_5=0$)	

5th selected K_i (in the exact-fit (best) order) by $p=1$, $m'=1$

K3: $C_3 = 80$	$[9, 11, 14, 20, 21]$	$TP_3=81, TW_3=80$ ($Fit_3=C_3-TW_3=0$)	select K3 ($TP_3=81$) ($Fit_3=0$)
----------------	-----------------------	---	--

$$\sum TP_i (i=1,2,3,4,5) = 161+261+126+97+81=726$$

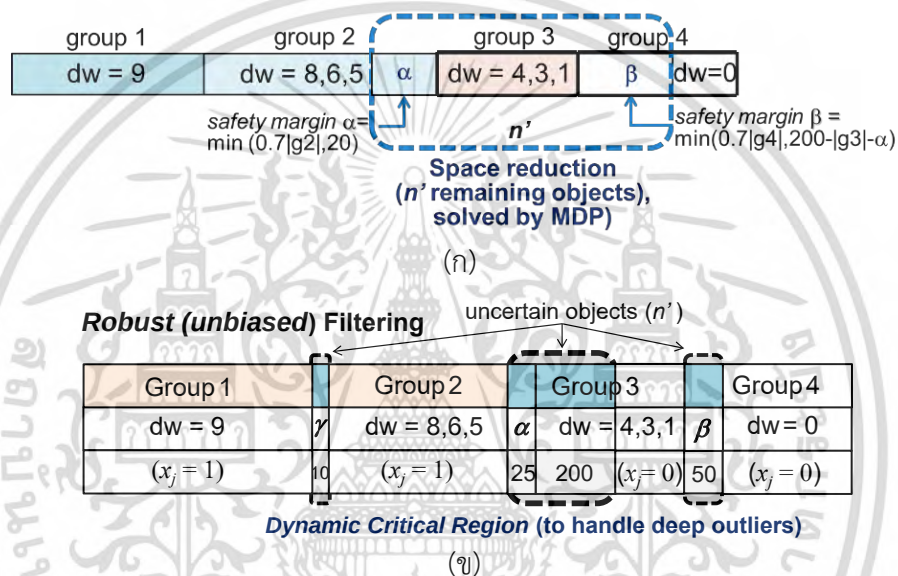
(ข)

รูปที่ 4.8 หลักการเรียงถุงเป้แบบเหมาะสมพอดีและตัวอย่างของวิธีการเลือกถุงเป้ตามรูปแบบการเรียงถุงเป้แบบเหมาะสมพอดี

4.2.2 การคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (Efficient Robust Unbiased Filtering)

การเพิ่มประสิทธิภาพในการแก้ปัญหาการคัดกรองแบบไม่เอนเอียงที่ได้นำเสนอในบทที่ 3 มาใช้พัฒนาเวลาโพลิโนเมียล ผู้วิจัยได้นำกระบวนการคัดกรองแบบไม่เอนเอียงที่ได้นำเสนอในบทที่ 3 มาใช้พัฒนาพร้อมกับขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ที่ใช้การเรียงถุงเป้แบบเหมาะสมพอดี ซึ่งทำให้ใช้เวลาในการประมวลผลเป็น $O(m^2(n + C'))$ หรือ $O(m^2n)$ และ $C' \ll C$ ($C \ll \text{large } n$) โดยที่ยังคงได้คำตอบที่ถูกต้อง 99% ซึ่งจากกระบวนการคัดกรองของขั้นตอนวิธี $TS_{Reduction}$ ที่นำเสนอในบทที่ 3 ได้มีการกำหนดเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

พารามิเตอร์จาก α และ β เพื่อใช้ในการกำหนดขอบเขตข้อมูลกลุ่มที่ไม่สามารถตัดสินใจได้หรือเรียกว่า ช่วงข้อมูลวิกฤติ (รูปที่ 4.9 (ก)) เพื่อใช้ในปัญหาซึ่งเกิดแบบแซค แต่ในการเพิ่มประสิทธิภาพเพื่อเพิ่มความแม่นยำในการคัดกรองสำหรับปัญหามัลติเพิลแซค ผู้วิจัยศึกษาข้อมูลสำหรับปัญหาและพบว่า มีสิ่งของบางส่วนในกลุ่ม 1 ที่อาจจะไม่เหมาะสมกับถุงเป้ปัจจุบันที่ถูกประมวลผลจึงได้เพิ่มขอบเขตเป็น γ, α, β (รูปที่ 4.9 (ข)) เพื่อให้เข้าถึงข้อมูลวิกฤติในแต่ละกลุ่มข้อมูลมากขึ้น นั่นคือพารามิเตอร์ γ และ α เป็นช่วงวิกฤติของสิ่งของในกลุ่ม 1 ($dw = 9$) และ 2 ($dw = 8, 6, 5$) และ พารามิเตอร์ β เป็นช่วงวิกฤติของสิ่งของในกลุ่ม 4 ($dw = 0$) ซึ่งสิ่งของที่ไม่สามารถตัดสินใจได้เกือบทั้งหมดจะอยู่กลุ่มที่ 3 ($dw=4, 3, 1$) ดังสมการที่ 4.1-4.5 และพัฒนาในขั้นตอนวิธีที่ 4.5



รูปที่ 4.9 แผนภาพแสดงการคัดกรองด้วยวิธีการคัดกรองแบบไม่เอนเอียง (บทที่ 3) และการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง

รูปที่ 4.9 แผนภาพแสดงรูปแบบการคัดกรองสิ่งของด้วยกระบวนการคัดกรองแบบไม่เอนเอียง (บทที่ 3) เปรียบเทียบกับรูปแบบการคัดกรองสิ่งของด้วยกระบวนการคัดกรองแบบแข็งแกร่ง โดยในรูปที่ 4.9 (ก) เป็นรูปแบบการคัดกรองแบบไม่เอนเอียง (Unbiased Filtering) ที่นำเสนอในบทที่ 3 ซึ่งเป็นการเลือกสิ่งของในกลุ่มที่ 3 ($dw = 4, 3, 1$) ทั้งหมด และขยายเนื้อที่วิกฤติไปในกลุ่ม 2 ($dw = 8, 6, 5$) ด้วยค่า α (สมการที่ 3.1) และกลุ่ม 4 ($dw = 0$) ด้วยค่า β (สมการที่ 3.2) เพื่อให้สิ่งของที่จะนำไปประมวลผลด้วยขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบดัดแปลง (MDP) มีจำนวนไม่เกิน 200 (สมการที่ 3.3) รูปที่ 4.9 (ข) แสดงถึงรูปแบบการปรับเนื้อที่วิกฤติไปในทุกกลุ่ม (Group 1 – Group 4) ด้วยกระบวนการคัดกรองแบบแข็งแกร่ง (Robust Filtering) ซึ่งมีการคัดกรองสิ่งของในทุกกลุ่ม โดยมีค่า γ สำหรับกลุ่ม 1 : $dw = 9$ ค่า α สำหรับกลุ่ม 2 : $dw = 8, 6, 5$ ค่า U สำหรับกลุ่ม 3 : $dw = 4, 3, 1$ และค่า β สำหรับกลุ่ม 4 : $dw = 0$ (ดังแสดงในสมการที่ 4.2 – 4.5) สำหรับปัญหาขนาดใหญ่ที่มีจำนวนสิ่งของมากจะมีการกำหนดสิ่งของที่เหลือ $n' < 300$ นั่นคือ $10(\gamma) + 25(\alpha) +$ เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$50(\beta) + 200(U) = 285$ ซึ่งจากสมการที่ 4.1 จะเห็นว่า ค่า n' จะมีค่าน้อยกว่า 285 สาเหตุมาจากปัญหาที่มี C ขนาดใหญ่จะทำให้มีจำนวนสิ่งของที่ถูกละทิ้งลงถุงเป้ ($x_j=1$) จำนวนมาก และสำหรับถุงเป้ที่มี C ขนาดเล็กจะมีสิ่งของที่ถูกละทิ้งที่ไม่ถูกละทิ้ง ($x_j=0$) จำนวนมาก ดังนั้น จากรูปแบบการคัดกรองแบบเข้มแข็งนี้ จะเห็นว่าค่า n' , γ , α , U และ β เปลี่ยนไปตามค่าคุณสมบัติของสิ่งของและความจุถุงเป้ที่ประมวลผลในรอบนั้น ๆ จึงเรียกช่วงวิกฤติที่ปรับเปลี่ยนตามองค์ประกอบของการหาคำตอบสำหรับแต่ละถุงเป้ ว่า ช่วงวิกฤติแบบพลวัต (Dynamic Critical Region)

$$n' = \gamma + \alpha + U + \beta < 300 \quad (4.1)$$

$$\gamma = \min(10, 0.15 \times |g1|) \text{ or } \max \gamma = 10 \quad (4.2)$$

$$\alpha = \min(25, 0.85 \times |g2|) \text{ or } \max \alpha = 25 \quad (4.3)$$

$$\beta = \min(50, 0.70 \times |g4|) \text{ or } \max \beta = 50 \quad (4.4)$$

$$U = \min(200, |g3|) \text{ or } \max U = 200 \quad (4.5)$$

การคัดกรองที่มีประสิทธิภาพในขั้นตอนวิธีที่ 4.5 (บรรทัดที่ 3) จะถูกประมวลผลในแต่ละถุงเป้ก่อนก่อนที่จะประมวลผลด้วยขั้นตอนวิธี DPT-List_{TSR} ที่ n' และ C' (โดยที่ n' คือสิ่งของที่เหลือหลังจากได้คัดกรองในอันดับแรก (Filtering-in) แล้วในแต่ละถุงเป้ และ n^* คือ สิ่งของที่เหลือทั้งหมดที่จะนำไปประมวลสำหรับถุงเป้ถัดไป) ดังแสดงตัวอย่างในรูปที่ 4.10

ขั้นตอนวิธีที่ 4.5 ขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เฮอร์วาร์ดร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (Multi-DPT-List_{TSR} plus Robust Unbiased Filtering)

Multi-DPT-List_{TSR} + Robust Unbiased Filtering

1. $n^* = n$;
2. for ($i=1$ to m) do
3. do object classification and filtering (for Filt- tp) on n^* ;
4. apply DPT-List_{TSR} ($n' < 300$) on knapsack i (C');
5. call Algorithm 7 (preprocessing on n' , C');
6. if ($i < m$) start = (max tp , max tw) else start = (max tp , min tw);
7. call Algorithm 8 (X-tracking on n');
8. Total profit = Total profit + Filt- tp + max tp ;
9. update n^* (exclude k_i selected objects of knapsack i);
10. end for i ;
11. return Total profit.

รูปที่ 4.10 (ก) แสดงค่าน้ำหนักพลวัต (dw) สำหรับปัญหาขนาด $n = 25$ และจำนวนถุงเป้ $m = 2$ ($C = \{30, 40\}$) ซึ่งนำไปใช้ในการแบ่งกลุ่มสิ่งของดังแสดงรูปที่ 4.10 (ข) ด้วยการคัดกรองแบบเข้มแข็ง โดยในการประมวลผลสำหรับถุงเป้หมายเลข 1 ($K_1: C_1 = 30$) สิ่งของทุกชิ้น (n) จะถูกแบ่ง

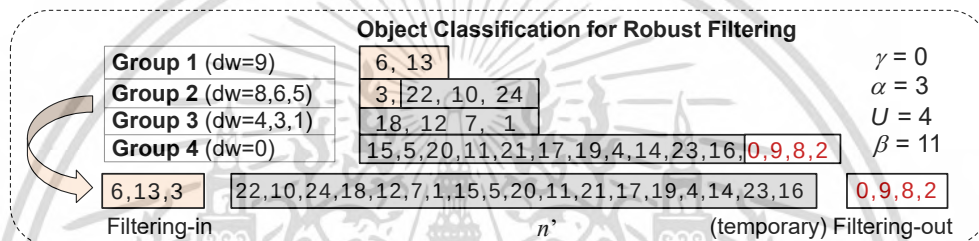
ออกเป็น 4 กลุ่ม ตามค่า dw และในแต่ละกลุ่มมีการเรียงลำดับสิ่งของตามค่า P/W จากมากไปน้อย เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปเผยแพร่โดยไม่ได้รับอนุญาตจากเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังนั้น จะได้ลำดับสิ่งของในแต่ละกลุ่มดังนี้ กลุ่ม 1 ($dw=9$) ประกอบด้วย {6, 13} กลุ่ม 2 ($dw=8,6,5$) ประกอบด้วย {3, 22, 10, 24} กลุ่ม 3 ($dw=4,3,1$) ประกอบด้วย {18, 12, 7, 1} และ กลุ่ม 4 ($dw=0$) ประกอบด้วย {15, 5, 20, 11, 21, 17, 19, 4, 14, 23, 16, 0, 9, 8, 2} เมื่อคัดกรองสิ่งของในแต่ละกลุ่มตามสมการที่ 4.2 - 4.5 จะทำให้ได้ค่า $\gamma = 0$, $\alpha = 3$, $U = 4$, $\beta = 11$ ตามลำดับ

3 GH⁺ solutions and dw (dynamic weight) for filtering

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
P	17	10	14	18	14	15	27	11	12	16	24	13	22	26	15	16	18	22	19	24	21	13	14	11	28
W	11	4	14	3	7	5	4	4	10	12	6	5	7	6	8	5	11	9	5	10	8	5	3	6	8
P/W	1.5	2.5	1	6	2	3	6.8	2.8	1.2	1.3	4	2.6	3.1	4	2	3.2	1.6	2	4	2.4	2.6	2.6	4.7	1.8	3.5
GH1 ⁺	0	0	0	5	0	0	5	0	0	0	5	0	0	5	0	0	0	0	0	0	0	5	0	5	0
GH2 ⁺	0	0	0	0	0	0	3	0	0	0	0	0	0	3	3	0	0	0	3	0	0	0	0	0	3
GH3 ⁺	0	1	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0
dw	0	1	0	6	0	0	9	1	0	0	5	0	3	9	0	0	0	0	4	0	0	0	6	0	8

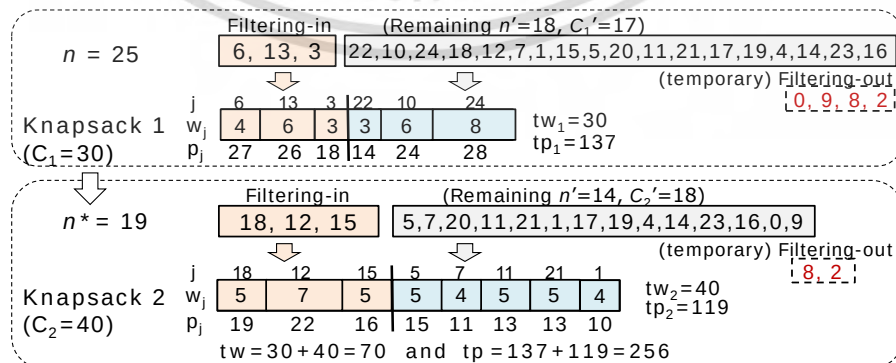
(ก)



(ข)

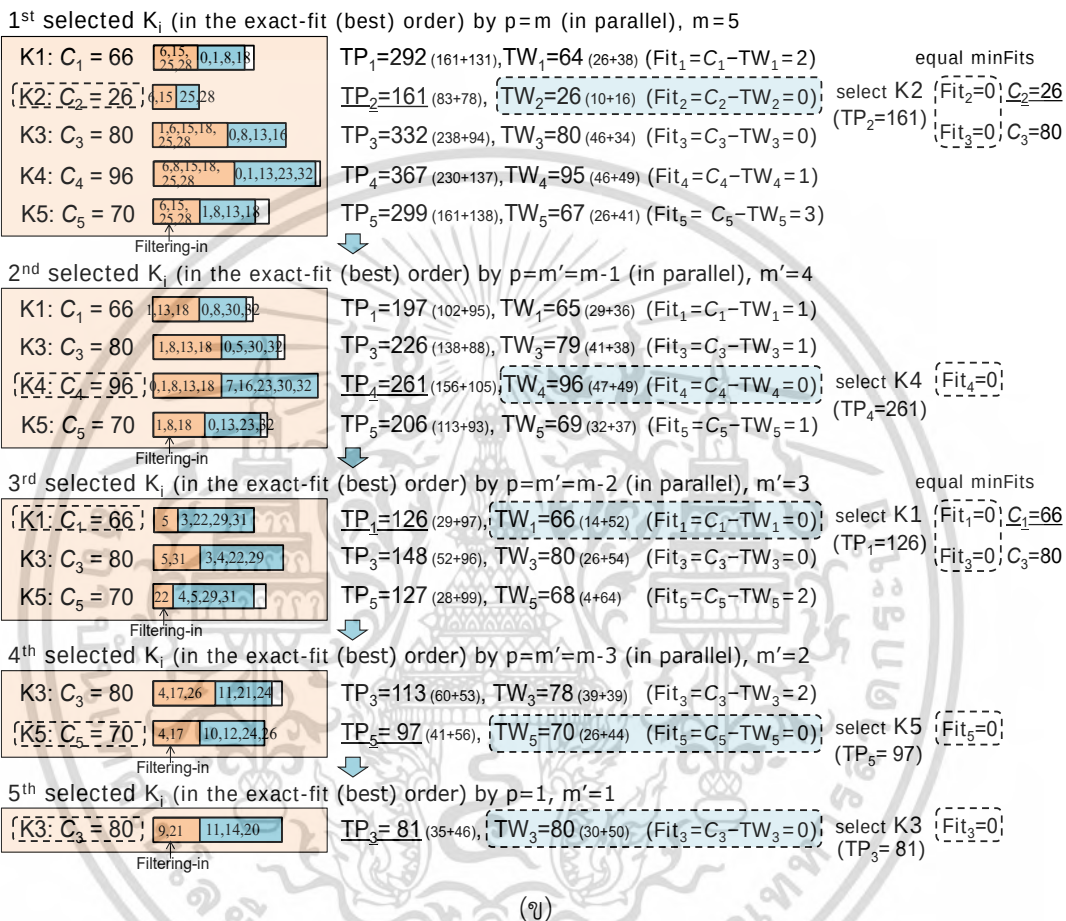
รูปที่ 4.10 ตัวอย่างการแบ่งกลุ่มข้อมูลของปัญหาขนาด $n = 25$, $m = 2$, $C = (30, 40)$ ในถุงเป้ หมายเลข K_1

รูปที่ 4.11 จากการจัดกลุ่มจะทำให้ได้สิ่งของที่ถูกเลือกโดยเอกฉันท์ Filtering-in = {6, 13, 3} ซึ่งจะเป็นสิ่งของที่ถูกเลือกเป็นส่วนแรกสำหรับถุงเป้ K_1 สำหรับสิ่งของและน้ำหนักที่เหลือ $n' = 18$ (ไม่รวมสิ่งของส่วน Filtering-out = {0, 9, 8, 2}), $C_1' = 17$ จะถูกประมวลผลด้วยขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์ (DPT-List_{TSR}) ซึ่งทำให้ได้สิ่งของที่ถูกเลือกเพิ่มเติม คือสิ่งของหมายเลข 22, 10, 24 ดังนั้นผลลัพธ์สำหรับถุงเป้ที่ 1 (K_1) คือ $\text{sol}x_1 = \{6, 13, 3, 22, 10, 24\}$, $\text{tp}_1 = 137$, $\text{tw}_1 = 30$



รูปที่ 4.11 ตัวอย่างการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่งและขั้นตอนดีพีทีลิสต์ทีเอสอาร์สำหรับ n' ใน K_1 และ K_2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



จากกระบวนการแบ่งกลุ่มและกระบวนการดังแสดงในรูปที่ 4.12 ที่เป็นการแสดงกระบวนการคัดกรองแบบแข็งแกร่งสำหรับปัญหาขนาด $n = 25$ $m = 2$ และ $C = (30, 40)$ ที่มีการแบ่งกลุ่มเป็น 4 กลุ่ม และมีการจัดเรียงสิ่งของในแต่ละกลุ่มด้วยค่า P/W จากมากไปน้อยแล้ว ผลที่ได้จากขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่งโดยใช้การเรียงลำดับถุงเบ้จากน้อยไปมาก (Multi-DPT-List_{TSR} + Robust Unbiased Filtering (Increasing Order)) ทำให้ได้มูลค่าโดยรวมทั้งสิ้น (Total Profit) ที่ 256 ซึ่งเป็นคำตอบที่ดีที่สุด (Optimal Solution) ของชุดข้อมูลนี้ สำหรับชุดข้อมูลไม่ปกติ การคัดกรองแบบไม่เอนเอียงสามารถเลือกสิ่งของให้กับแต่ละถุงเบ้เป็นเอกสารที่ส่งหนังสือหรือการเขียนเพื่อการศึกษาค้นคว้า ไม่นับญาติเกินไปใช้ประโยชน์ด้านการพาณิชย์ใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในแต่ละเบื้องต้นได้ด้วย ขั้นตอนวิธีมัลติดีพีทีลิสต์ทีเอสอาร์ (Multi-DPT-List_{TSR}) ด้วยการจัดเรียงถุงเป้ด้วยกระบวนการจัดเรียงถุงเป้ที่ดีที่สุด (The Exact-fit (best) Knapsack Order สำหรับสิ่งของที่เหลือ (n') ซึ่งทำให้ได้คำตอบที่ดีพอ ดังแสดงในรูปที่ 4.10 (ก) แสดงการจัดเรียงถุงเป้ที่ดีที่สุด (The Exact-fit) รูปที่ 4.10 (ข) แสดงการคัดกรองแบบแข็งแกร่งร่วมกับการการจัดเรียงถุงเป้แบบดีที่สุด (Best Knapsack Order) ซึ่งได้การจัดเรียงถุงเป้ลำดับ 2, 4, 1, 5 และ 3 ตามลำดับกับข้อมูลขนาด $n=33$ $m=5$ และ $C=\{66, 26, 80, 96, 70\}$ ซึ่งได้ Total Profit ของแต่ละถุงเป็น $\{126, 161, 81, 261, 97\} = 726$ ซึ่งตรงกับที่คำตอบที่ดีที่สุด (Exact Solution)



บทที่ 5

การวิเคราะห์ความถูกต้องแม่นยำและความซับซ้อนด้านเวลา

การศึกษา ออกแบบและพัฒนาขั้นตอนวิธีสำหรับการแก้ปัญหาซิงเกิลแนปแซค (0/1-KP) และ มัลติเพิลแนปแซค (0/1-mKP) ในวิทยานิพนธ์นี้ ได้แบ่งการนำเสนอเป็น 2 ส่วน คือ ส่วนของขั้นตอนวิธี ทีเอสรีดักชัน (TS_{Reduction} Algorithm) เพื่อการลดขนาดของปัญหาและเวลาประมวลผลสำหรับการ แก้ปัญหาซิงเกิลแนปแซค (0/1-KP) ที่ได้นำเสนอในบทที่ 3 ซึ่งประกอบด้วย 3 ขั้นตอนหลัก ได้แก่ 1) ขั้นตอนวิธีฮิวริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพ (GH with Feature Sorting) 2) ขั้นตอนการแบ่งกลุ่มและการคัดกรองแบบไม่เอนเอียง (Object Classification and Unbiased Filtering) และ 3) ขั้นตอนวิธีเอ็มดีพี (MDP) และส่วนของขั้นตอนวิธีมัลติดีพีทีลิสต์ทีเอสอาร์สำหรับ ปัญหา มัลติเพิลแนปแซค (0/1-mKP) ที่นำเสนอในบทที่ 4 เป็นการพัฒนาระบบการคัดกรองให้มีความยืดหยุ่นและแม่นยำมากขึ้น เรียกว่า ระบบการคัดกรองแบบแข็งแกร่ง (Robust Unbiased Filtering) รวมถึงได้ปรับเปลี่ยนขั้นตอนวิธีโปรแกรมเชิงพลวัตเป็นขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ (DPT-List_{TSR} Algorithm) สำหรับปัญหาซิงเกิลแนปแซค และพัฒนาต่อยอดเป็นขั้นตอนวิธีมัลติดีพีทีลิสต์ทีเอสอาร์ (Multi-DPT-List_{TSR} Algorithm) เพื่อปัญหา มัลติเพิลแนปแซค สำหรับบทนี้ เป็นการนำเสนอการวิเคราะห์ความถูกต้องแม่นยำของขั้นตอนวิธีและความซับซ้อนด้านเวลาของขั้นตอนวิธีที่นำเสนอซึ่งได้ศึกษาและทดลองกับชุดข้อมูลที่หลากหลาย โดยคาดหวังถูกต้องของผลลัพธ์ที่ 100% สำหรับข้อมูลปกติ (Regular Datasets) และ 99% สำหรับข้อมูลผิดปกติ (Irregular Datasets)

5.1 ขั้นตอนวิธีทีเอสรีดักชันแบบขนาน (TS_{Reduction} Algorithm)

5.1.1 ความแม่นยำของขั้นตอนวิธีทีเอสรีดักชัน (Precision of TS_{Reduction} Algorithm)

วิธีการแก้ปัญหาแนปแซค มีหลักการพื้นฐานด้วยการเลือกสิ่งของที่มีคุณสมบัติเหมาะสมที่สุดบรรจุลงถุงเป้ โดยมีวัตถุประสงค์เพื่อให้ได้มูลค่ารวมมากที่สุดด้วยขนาดน้ำหนักรวมที่ไม่เกินความจุสูงสุดถุงเป้ ($\sum_{i=1}^n p_i x_i, \sum_{i=1}^n w_i x_i \leq C$) ซึ่งขั้นตอนวิธีที่ได้รับการยอมรับและหาคำตอบที่ดีที่สุดได้ (Optimal Solution) คือขั้นตอนวิธีโปรแกรมเชิงพลวัต แต่ขั้นตอนวิธีนี้มีข้อจำกัดในด้านเวลาประมวลผล โดยหากสมาชิกของปัญหามีจำนวนมากอาจจะส่งผลให้ใช้เวลาประมวลผลนานเกินไป (หรือไม่สามารถหาคำตอบในเวลาที่เหมาะสม) ในขณะที่ขั้นตอนวิธีกริดฮิวริสติกซึ่งเป็นขั้นตอนวิธีที่สามารถหาคำตอบได้โดยประมาณ (Approximate Solution) ซึ่งอาจจะไม่ได้คำตอบที่ดีที่สุดเสมอไปโดยเฉพาะชุดข้อมูลพิเศษ (ดังตัวอย่างชุดข้อมูลในรูปที่ 5.1 และ 5.2) แต่ประมวลผลได้อย่างรวดเร็วแม้ว่าสมาชิกของ

ปัญหามีจำนวนมากก็ตาม ซึ่งเป็นประโยชน์กับปัญหามีขนาดใหญ่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	X	0	1	2	3	4
	W	12	7	11	8	9
	P	24	13	23	15	16
	P/W	2	1.9	2.1	1.9	1.8

$\xrightarrow{\text{P/W ranking}}$

	Y	0	1	2	3	4
	X	2	0	3	1	4
	W	11	12	8	7	9
	P	23	24	15	13	16
	P/W	2.1	2	1.9	1.9	1.8

รูปที่ 5.1 ตัวอย่างชุดข้อมูลพิเศษขนาด $n = 5$ ที่ไม่สามารถหาคำตอบที่ดีที่สุดได้ด้วยวิธี GH (P/W)

Y	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
X	2	4	6	15	1	23	10	3	0	14	31	34	18	32	9	12	7	30	29	22	13	26	20	24	17	21	19	5	8	11	33	25	16	27	28
W	74	70	75	77	71	72	74	75	71	75	74	86	84	75	74	75	84	75	93	84	96	98	83	100	95	93	101	103	93	92	98	91	102	98	101
P	506	471	502	504	462	463	471	476	450	470	454	496	484	429	420	425	463	411	504	450	505	505	427	503	471	460	499	491	443	438	463	416	434	414	426
P/W	6.8	6.7	6.7	6.5	6.5	6.4	6.4	6.3	6.3	6.3	6.1	5.8	5.8	5.7	5.7	5.5	5.5	5.4	5.4	5.3	5.2	5.1	5	5	4.9	4.9	4.8	4.8	4.8	4.7	4.6	4.3	4.2	4.2	

รูปที่ 5.2 ตัวอย่างชุดข้อมูลพิเศษขนาด $n = 35$ ที่ไม่สามารถหาคำตอบที่ดีที่สุดได้ด้วยวิธี GH (P/W)

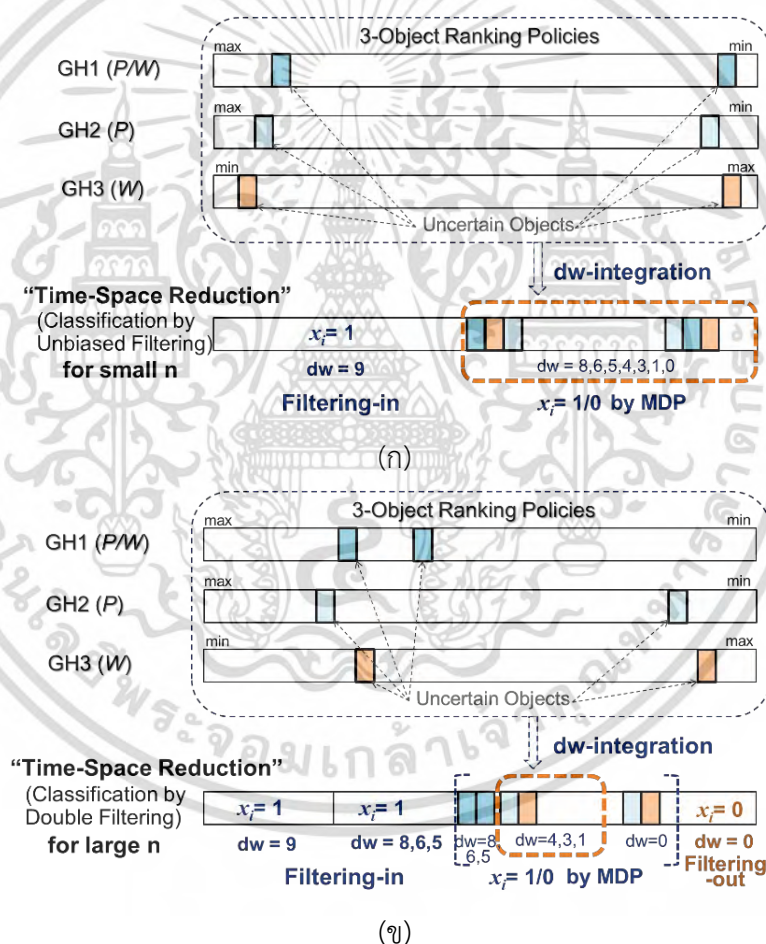
ตัวอย่างชุดข้อมูลมาตรฐานในรูปที่ 5.1 เป็นชุดข้อมูลสำหรับการทดลองขนาด $n = 5$, $C = 26$ ซึ่งหากประมวลผลด้วยขั้นตอนวิธีกริดอีวิริสติกจะทำให้ได้ผลลัพธ์ที่ $\text{solt}_p = 47$ และ $\text{solt}_w = 23$ ซึ่งไม่ใช่คำตอบที่ดีที่สุด แต่เมื่อประมวลผลด้วยขั้นตอนวิธีทีเอสรีดักชัน (TS_{Reduction} Algorithm) จะได้ผลลัพธ์ที่ $\text{solt}_p = 51$ และ $\text{solt}_w = 26$ โดยสิ่งของดัชนีที่ 0 จะถูกเลือกเป็นคำตอบในขั้นตอนที่ 2 เนื่องจากเป็นสิ่งของที่ถูกเลือกด้วยกฎ GH1 และ GH2 เท่านั้น เช่นเดียวกับชุดตัวอย่างปัญหาในรูปที่ 5.2 ($n = 35$) สิ่งของที่ $X_j = \{0, 7, 9, 12, 22, 30, 32\}$ ถูกจัดเรียงไว้ในส่วนกลางของกลุ่มสมาชิก ซึ่งหากเลือกสิ่งของบรรจุลงไปด้วยขั้นตอนวิธีกริดอีวิริสติกจะทำให้ไม่ได้คำตอบที่ดีที่สุดเช่นกัน แต่ชุดปัญหานี้สามารถหาคำตอบที่ดีที่สุดได้ด้วยขั้นตอนวิธีทีเอสรีดักชันที่มีการคัดกรองอย่างละเอียดและมีประสิทธิภาพ ตั้งแต่การคัดเลือกสิ่งของเริ่มต้นด้วยขั้นตอนวิธีจีเอช (GH) การปรับปรุงผลลัพธ์ให้ดีขึ้นด้วยขั้นตอนวิธีจีเอชพลัส (GH⁺) การกำหนดค่าน้ำหนักอย่างเหมาะสมของแต่ละกฎ และการรวมค่าน้ำหนักพลวัต (dw) เพื่อให้ทราบค่าน้ำหนักความเหมาะสมของสิ่งของแต่ละชิ้น ทำให้สามารถแบ่งกลุ่มสมาชิกได้ตามความเหมาะสม ได้แก่ กลุ่ม 1 ($dw = 9$) เป็นกลุ่มที่มีความเหมาะสมเป็นคำตอบมากที่สุด กลุ่ม 2 ($dw = 8, 6, 5$) เป็นกลุ่มที่มีความเหมาะสมรองลงมา ซึ่งถูกเลือกด้วยกฎ GH1⁺ แต่อาจจะไม่ถูกเลือกด้วยกฎอื่น กลุ่ม 3 ($dw = 4, 3, 1$) เป็นกลุ่มที่อาจจะเป็นคำตอบ เพราะไม่ถูกเลือกด้วยกฎ GH1⁺ แต่ถูกเลือกด้วยกฎอื่น และกลุ่ม 4 ($dw = 0$) เป็นกลุ่มที่ไม่ถูกเลือกด้วยกฎใด ๆ โดยการแบ่งกลุ่มนี้ จะถูกนำไปใช้คัดกรองในกระบวนการคัดกรองอย่างเป็นธรรมชาติ (หรือไม่เอนเอียง)

การวิเคราะห์ความถูกต้อง ผู้วิจัยได้ศึกษาธรรมชาติของชุดข้อมูลทั้งแบบมาตรฐานและแบบสุ่มพบว่า ข้อมูลบางชุดเป็นชุดข้อมูลพิเศษที่อาจมีค่าบางอย่างผิดปกติ (Outlier) นั่นคือ มีค่า P หรือค่า W เอนเอียงไปด้านใดด้านหนึ่งมากเกินไปทำให้ไม่สามารถถูกเลือกเป็นคำตอบได้ด้วยวิธีทั่วไป เช่น สิ่งของบางชิ้นที่มีค่า P/W ในระดับกลางแต่มีค่า P และค่า W มาก หรือสิ่งของบางชิ้นมีค่า P/W ในระดับกลางแต่มีค่า P และค่า W น้อย เป็นต้น ซึ่งเมื่อเรียงลำดับด้วยกฎจีเอชแล้วพบว่าข้อมูลมีลักษณะการกระจายตำแหน่ง ดังรูปที่ 5.3

รูปที่ 5.3 แผนภาพแสดงความสามารถคัดกรองและคัดแยกสมาชิกบริเวณจุดวิกฤติ (α และ β)

ของปัญหาขนาดเล็กและปัญหาขนาดใหญ่ โดยแทนสมาชิกที่ไม่สามารถคัดกรองได้ด้วยแถบสี ซึ่งเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

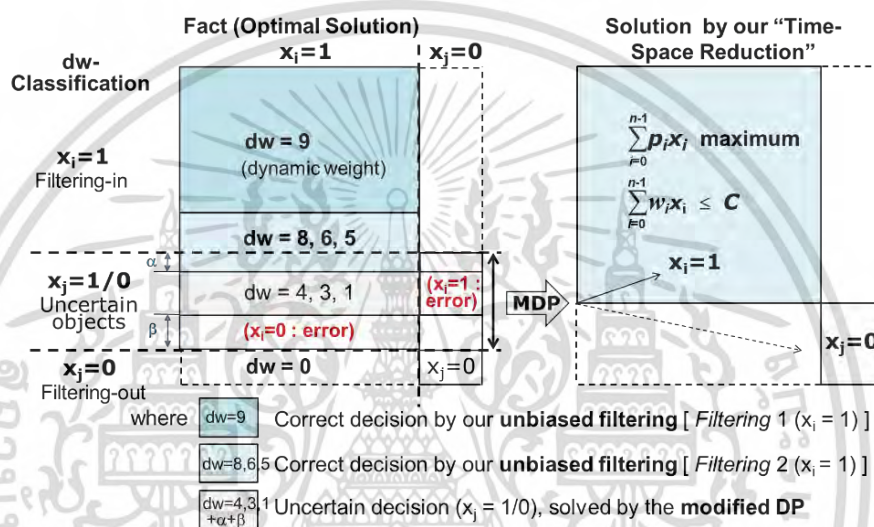
เกิดขึ้นได้หลังจากขั้นตอนวิธีจีเอช ทั้ง 3 กฎ ซึ่งในรูปที่ 5.3 (ก) เป็นแผนภาพแสดงความเป็นไปได้ของตำแหน่งและการจัดกลุ่มสิ่งของที่ไม่สามารถคัดกรองได้สำหรับปัญหาขนาดเล็ก (Small n) ซึ่งจะเห็นว่าผลการเรียงลำดับข้อมูลทั้ง 3 กฎของขั้นตอนวิธีจีเอชอาจจะมีสิ่งของบางชิ้นที่อาจจะเป็นคำตอบ (1) หรือไม่คำตอบ (0) อยู่ในตำแหน่งส่วนต้นหรือท้ายของลำดับ ซึ่งหากใช้ขั้นตอนวิธีจีเอชอย่างเดียว อาจจะได้คำตอบที่ดีที่สุด จึงได้ปรับปรุงผลลัพธ์ด้วยขั้นตอนวิธีจีเอชพลัส (GH⁺) แล้วจึงกำหนดค่าน้ำหนักและคำนวณค่าน้ำหนักความเหมาะสมของสิ่งของในขั้นตอนการคัดกรองระดับ 1 (Filtering 1) ทำให้ได้กลุ่มของสมาชิกที่เหมาะสม ($dw = 9$) และกลุ่มของสมาชิกที่คัดกรองไม่ได้ ($dw = 8, 6, 5, 4, 3, 1, 0$) โดยกลุ่มสมาชิกที่คัดกรองไม่ได้จะจำกัดจำนวนไม่เกิน 200 ($n' \leq 200$) เพื่อส่งไปประมวลผลอย่างละเอียดด้วยโปรแกรมเชิงพลวัตแบบดัดแปลง ซึ่งสามารถทำให้ได้คำตอบที่ดีที่สุดและรวดเร็ว



รูปที่ 5.3 แผนภาพแสดงความสามารถคัดแยกสิ่งของบริเวณจุดวิกฤติ (α และ β) มาไว้ในกลุ่มที่ไม่แน่นอนสำหรับปัญหาขนาดเล็กและปัญหาขนาดใหญ่

รูปที่ 5.3 (ข) เป็นแผนภาพแสดงความเป็นไปได้ของตำแหน่งสิ่งของที่ไม่สามารถคัดกรองด้วยวิธีการคัดกรองและจัดกลุ่มได้สำหรับปัญหาขนาดใหญ่ (Large n) จากรูปจะเห็นว่าผลการเอกสาร์นี้เป็นเอกสาร์ที่ส่งวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสาร์ทุกครั้งที่มีการนำไปใช้

เรียงลำดับของขั้นตอนวิธีจีเอช อาจมีสมาชิกที่ไม่สามารถคัดกรองได้กระจายอยู่ทั้งในส่วนต้น กลาง และท้ายของสมาชิกทั้งหมด ซึ่งเมื่อได้ปรับปรุงการเลือกด้วยขั้นตอนวิธีจีเอชพลัสแล้ว ตำแหน่งสิ่งของ มีรูปแบบการรวมกลุ่มที่ดีขึ้นแต่ยังคงมีบางสิ่งของที่ยังคงกระจายอยู่ในส่วนต้น กลางและท้าย จึงเพิ่ม การคัดกรองระดับ 2 (Filtering 2) เพื่อคัดกรองสมาชิกในกลุ่ม 2 ($dw = 8, 6, 5$) และกลุ่ม 4 ($dw = 0$) เพื่อให้เหลือสมาชิกที่ไม่แน่นอนจำนวนไม่เกิน 200 ($n' \leq 200$) โดยสมาชิกที่อยู่ในขอบเขตความไม่แน่นอนจะขยายให้ครอบคลุมค่า α และ β ที่อยู่ในบริเวณจุดวิกฤติ ซึ่งจะถูกนำไปประมวลผลด้วย ขั้นตอนวิธีเอ็มดีพีต่อไป



รูปที่ 5.4 แผนภาพแสดงการคัดกรองแบบไม่เอนเอียง (Unbiased Filtering) และการจัดการสมาชิกที่เป็นค่าผิดปกติ (Outlier Handling) ด้วยขั้นตอนวิธีทีเอสรีดักชัน (TS_{Reduction})

จากการศึกษาวิธีการคัดเลือกล้างสำหรับปัญหาแนปแซค ด้วยขั้นตอนวิธีที่มีในปัจจุบัน พบข้อสังเกตได้ 2 ประการ สำหรับค่าวิกฤติของบางสมาชิกที่อาจจะให้เกิดข้อผิดพลาดในการเลือก (เช่น GH1, GH2, GH3 เป็นต้น) (ดังแสดงในรูปที่ 5.4) คือ

- 1) กรณีที่สมาชิกบางตัวถูกเลือก ($x_j = 1$) หรือมีค่า $dw = 4, 3, 1$ (เมื่อเทียบกับขั้นตอนวิธีทีเอสรีดักชัน) แสดงว่าสมาชิกเหล่านั้นถูกเลือกด้วยการเรียงลำดับตามค่า P หรือค่า W แต่ไม่ได้ถูกเลือกด้วยการเรียงลำดับด้วยค่า P/W (ซึ่งเป็นการจัดเรียงที่สามารถได้คำตอบที่ใกล้เคียงกับคำตอบที่ดีที่สุด) แต่ในคำตอบที่ดีที่สุดไม่ได้เลือก x_j นี้ ($x_j = 0$) ดังนั้น คำตอบที่ได้ด้วยวิธีเหล่านี้จึงไม่ได้คำตอบที่ดีที่สุด
- 2) กรณีที่สมาชิกบางตัวไม่ถูกเลือก ($x_j = 0$) ให้เป็นคำตอบด้วยกฎใด ๆ ของขั้นตอนวิธีจีเอช ($dw = 0$) แต่ถูกเลือกให้เป็นคำตอบที่ดีที่สุด ดังนั้นคำตอบที่ได้จากวิธีเหล่านี้จึงไม่ใช่คำตอบที่ดีที่สุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนวิธีที่เอสรัดักชั้นที่นำเสนอในงานวิจัยนี้ สามารถแก้ปัญหาข้อผิดพลาดข้างต้นได้ด้วย การกำหนดค่าน้ำหนัก และการรวมค่าน้ำหนักตามผลที่ได้จากกฎของจีเอชพลัส (GH⁺) ร่วมกับการกำหนดขอบเขตที่ครอบคลุมค่าวิกฤติ α และ β (ตามสมการที่ 3.1-3.3) นอกจากนี้ ในการคัดกรองสมาชิกที่มีค่าผิดปกติ (Outlier) ซึ่งกระจายอยู่ในกลุ่ม 2 ($dw = 8, 6, 5$) และกลุ่ม 4 ($dw = 0$) จะถูกจัดอยู่ในกลุ่มสมาชิกที่ไม่แน่นอนหรือวิกฤติ เพื่อนำไปหาคำตอบอย่างละเอียดด้วยขั้นตอนวิธีเอ็มดีพี

ในการคิดค้นและพัฒนาขั้นตอนวิธีที่เอสรัดักชั้นสำหรับการแก้ปัญหาแนปแซค ผู้วิจัยได้ศึกษาธรรมชาติของข้อมูลสำหรับปัญหาถุงเป้เพื่อพัฒนาวิธีการแก้ไขปัญหของชุดข้อมูลในการทดลองทั้งหมด ซึ่งประกอบด้วยชุดข้อมูลมาตรฐาน [18][27] และชุดข้อมูลที่สร้างจากการสุ่ม พบว่าสมาชิกในชุดข้อมูลทั้งหมดมีคุณสมบัติที่สามารถแบ่งออกได้เป็น 3 กลุ่ม ได้แก่

- 1) กลุ่มสมาชิกที่มีมูลค่ามาก (Hight Profit) และมีน้ำหนักเบา (Light Weight) ซึ่งจัดอยู่ในกลุ่มที่ 1 ($dw = 9$) เป็นกลุ่มสมาชิกที่ควรเลือกเป็นคำตอบ
- 2) กลุ่มสมาชิกที่มีน้ำหนักมาก (Heavy Weight) แต่มีมูลค่าน้อย (Low Profit) ซึ่งส่วนใหญ่จะอยู่ในกลุ่ม 4 ($dw = 0$) เป็นกลุ่มสมาชิกที่อาจจะไม่ถูกเลือกเป็นคำตอบ (ถ้าสมาชิกนั้น ๆ มีน้ำหนักมากกว่ามูลค่ามาก) หรืออาจถูกเลือกเป็นคำตอบได้ (ถ้าสมาชิกนั้น ๆ มีน้ำหนักมากกว่ามูลค่าไม่มาก)
- 3) กลุ่มสมาชิกที่มีสัดส่วนของมูลค่าต่อน้ำหนักอยู่ในลำดับกลาง ๆ ของชุดข้อมูล (Medium P and W) ซึ่งมีค่า $dw = 8, 6, 5, 4, 3, 1$ ซึ่งสามารถพิจารณาเพิ่มเติมได้ ดังนี้
 - 3.1) กลุ่มสมาชิกที่สัดส่วนของมูลค่าต่อน้ำหนักค่อนข้างน้อย พบว่าเป็นสมาชิกที่มีค่า $dw = 6$ แสดงว่ามีการเอนเอียงไปทางค่าของน้ำหนักสิ่งของ (W-biased)
 - 3.2) กลุ่มสมาชิกที่สัดส่วนของมูลค่าต่อน้ำหนักค่อนข้างมาก พบว่าเป็นสมาชิกที่มีค่า $dw = 8$ แสดงว่ามีการเอนเอียงไปทางมูลค่าของสิ่งของ (P-biased)

ซึ่งสมาชิกที่ $dw = 8$ มีโอกาสได้เลือกเป็นคำตอบได้มากกว่า $dw = 6$ แต่ก็มีโอกาสที่จะไม่ได้เลือกเป็นคำตอบได้เช่นกัน ดังนั้น บางสมาชิกตรงบริเวณจุดวิกฤติของทั้งสองกลุ่มนี้ (α และ β) จึงควรถูกพิจารณาอย่างละเอียดด้วยขั้นตอนวิธีเอ็มดีพี

5.1.2 ความซับซ้อนด้านเวลาของขั้นตอนวิธีที่เอสรัดักชั้น (Time Complexity of TS_{Reduction} Algorithm)

การวิเคราะห์ความซับซ้อนด้านเวลาของขั้นตอนวิธีที่เอสรัดักชั้น (TS_{Reduction} Algorithm) ได้เปรียบเทียบกับขั้นตอนวิธีกรีดีฮิวริสติก (GH) และขั้นตอนวิธีโปรแกรมเชิงพลวัต (DP) ด้วยการเปรียบเทียบองค์ประกอบของผลลัพธ์ ความถูกต้องของคำตอบ และความซับซ้อนด้านเวลาในแบบอนุกรม (ตารางที่ 5.1) รวมถึงกระบวนการของแต่ละขั้นตอนวิธี จุดแข็งและจุดอ่อน (ตารางที่ 5.2)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

นอกจากนี้ ได้แสดงการเปรียบเทียบความถูกต้องของคำตอบและความซับซ้อนด้านเวลาของขั้นตอนวิธีทีเอสรีดักชันแบบขนาน (Parallel-TS_{Reduction}) กับขั้นตอนวิธีกรีดฮีริสติกแบบขนาน (Parallel-GH) และขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบขนาน (Parallel DP) (ดังแสดงในตารางที่ 5.3)

ตารางที่ 5.1 แสดงการเปรียบเทียบองค์ประกอบของผลลัพธ์สำหรับปัญหาแนปแซค ซึ่งประกอบด้วยค่า soltp, soltw และ solx รวมถึงความถูกต้องของคำตอบ (Solution) และความซับซ้อนด้านเวลา (Time Complexity) ของขั้นตอนวิธีทีเอสรีดักชัน (TS_{Reduction} Algorithm) ขั้นตอนวิธีกรีดฮีริสติก (GH) และขั้นตอนวิธีโปรแกรมเชิงพลวัต (DP) พบว่าผลลัพธ์ที่ได้จากขั้นตอนวิธีกรีดฮีริสติกทั้ง 3 รูปแบบ มีความสมบูรณ์ด้านองค์ประกอบของผลลัพธ์ครบทั้ง 3 ด้าน (soltp, soltw, solx) ด้วยความซับซ้อนด้านเวลาที่ $O(n)$ แต่ไม่สามารถได้คำตอบที่ดีที่สุดในทุกชุดข้อมูล ในขณะที่ขั้นตอนวิธีโปรแกรมเชิงพลวัตเป็นวิธีที่สามารถหาคำตอบที่ดีที่สุดได้กับทุกชุดข้อมูล โดยขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์เน้นความครบถ้วนของผลลัพธ์ซึ่งมีความซับซ้อนด้านเวลาที่ $O(knC)$ แต่ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็วมุ่งเน้นความเร็วในการประมวลผล โดยการลดขั้นตอนด้านการค้นหาชุดรายการสมาชิกที่เป็นคำตอบ (solx) โดยมีความซับซ้อนด้านเวลาที่ $O(nC)$ สำหรับผลลัพธ์ที่ได้จากขั้นตอนวิธีทีเอสรีดักชันมีความซับซ้อนด้านเวลาที่ $O(n+C')$ และมีองค์ประกอบของผลลัพธ์ครบทั้ง 3 ด้าน ซึ่งคำตอบที่ได้ส่วนใหญ่เป็นคำตอบที่ดีที่สุด

ตารางที่ 5.1 เปรียบเทียบความสมบูรณ์ของผลลัพธ์ ประสิทธิภาพด้านความถูกต้องและความซับซ้อนด้านเวลาของขั้นตอนวิธีทีเอสรีดักชัน (TS_{Reduction}) ขั้นตอนวิธีกรีดฮีริสติก (Greedy Heuristic : GH) และขั้นตอนวิธีโปรแกรมเชิงพลวัต (Dynamic Programming : DP)

Algorithms (Sequence)	Completeness of Solutions			Performance	
	Total Weight (soltw)	Total Profit (soltp)	Sol.X (solx)	Solution	Time Complexity
GH1(P/W)	✓	✓	✓	n/a Opt.	$O(n)$
GH2(P)	✓	✓	✓	n/a Opt.	$O(n)$
GH3(W)	✓	✓	✓	n/a Opt.	$O(n)$
Fast DP	✓	✓	✗	Optimal	$O(nC)$
Complete DP	✓	✓	✓	Optimal	$O(knC)$
Proposed TS _{Reduction}	✓	✓	✓	Most sols. are Opt.	$O(n+C')$

นอกจากนี้ได้มีการเปรียบเทียบกระบวนการในแต่ละขั้นตอนวิธี จุดแข็ง และจุดอ่อน ทั้ง 3 ขั้นตอนวิธีข้างต้น (ดังแสดงในตารางที่ 5.2) ซึ่งขั้นตอนวิธีกรีดฮีริสติกได้ใช้การเรียงลำดับแบบเรดิคซ์ (Radix Sort) ซึ่งทำให้ใช้เวลาในการประมวลผลรวดเร็ว $O(n)$ แต่ไม่รับรองการได้คำตอบที่ดีที่สุด สำหรับขั้นตอนวิธีโปรแกรมเชิงพลวัตทั้ง 3 แบบ เป็นขั้นตอนวิธีที่รับรองการได้คำตอบที่ดีที่สุด โดยเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื่องต้นได้ใช้อาร์เรย์ 2 มิติ ขนาด nC ในการประมวลผลด้วยความซับซ้อนด้านเวลาที่ $O(nC)$ โดยมีการค้นหารายการของสมาชิกในชุดคำตอบร่วมด้วย แต่วิธีนี้ไม่สามารถประมวลผลอาร์เรย์ขนาดใหญ่ได้หากปริมาณข้อมูลมีขนาดใหญ่ (C) และจำนวนสมาชิกมาก (n) จึงได้มีการพัฒนาขั้นตอนวิธี การโปรแกรมเชิงพลวัตแบบเร็วที่ใช้พื้นที่ในการประมวลผลด้วยการใช้อาร์เรย์หนึ่งมิติ ขนาด C และลดการค้นหารายการสมาชิกของคำตอบ (solx) เน้นเฉพาะการหามูลค่ารวม (solt) จึงทำให้มีความซับซ้อนด้านเวลาที่ $O(nC)$ เท่าเดิม และด้วยแนวคิดดังกล่าวจึงได้มีการพัฒนาต่อในขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ โดยใช้อาร์เรย์ 1 มิติ จำนวน 2 อาร์เรย์ในการประมวลผลและเก็บดัชนีสิ่งของที่เป็นคำตอบเพื่อเพิ่มความสมบูรณ์ของผลลัพธ์ โดยวิธีนี้ใช้การประมวลผลซ้ำตามขนาดความจุและจำนวนสิ่งของที่ลดลงในแต่ละรอบที่ได้สมาชิกของคำตอบ ทำให้ใช้เวลาประมวลผลที่นานกว่าเป็น $O(knC)$

สำหรับขั้นตอนวิธีที่เอสตีคชันที่ผู้วิจัยนำเสนอในงานวิจัยนี้ ประกอบด้วย 3 ขั้นตอนใหญ่ ได้แก่ 1) ขั้นตอนการหาคำตอบเริ่มต้นด้วยขั้นตอนวิธีฮิวริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพด้วยการเรียงลำดับตามคุณสมบัติของสิ่งของหลายระดับอย่างเหมาะสมและการเลือกสิ่งของด้วยกฎของขั้นตอนวิธีจีเอช (GH) 2) ขั้นตอนการแบ่งกลุ่มและคัดกรองแบบไม่เอนเอียงเพื่อลดขนาดของปัญหา โดยอาศัยการแบ่งกลุ่มสิ่งของตามค่าความเหมาะสม (dw) ซึ่งได้มาจากการปรับปรุงผลลัพธ์ในขั้นตอนที่ 1 ด้วยกฎของขั้นตอนวิธีจีเอชพลัส (GH+) และการกำหนดค่าน้ำหนักตามกฎอย่างเหมาะสม ทำให้สามารถคัดกรองสิ่งของในแต่ละกลุ่มได้อย่างมีประสิทธิภาพด้วยกระบวนการคัดกรองที่ไม่เอนเอียง โดยสามารถเลือกสิ่งของที่เป็นคำตอบและสิ่งของที่ไม่ใช่คำตอบ สำหรับสิ่งของที่เหลือจากการคัดกรองซึ่งถูกลดจำนวนให้เหลือขนาดจำกัด จะถูกนำไปประมวลผลในขั้นตอนต่อไป 3) ขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบดัดแปลง (MDP) เป็นขั้นตอนวิธีที่รับรองการหาคำตอบที่ดีที่สุด จึงสามารถหาสมาชิกคำตอบชุดสุดท้ายได้อย่างถูกต้อง โดยขั้นตอนนี้ได้พัฒนามาจากขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ที่ใช้อาร์เรย์หนึ่งมิติ และได้เพิ่มอาร์เรย์หนึ่งมิติสำหรับเก็บรหัสบิตของคำตอบเลือก (1) และไม่เลือก (0) ของสิ่งของแต่ละชิ้น ทำให้ประหยัดพื้นที่ที่ใช้ประมวลผลเมื่อเทียบกับการโปรแกรมเชิงพลวัตแบบเบื่องต้นที่ใช้อาร์เรย์ 2 มิติ อีกทั้ง เมื่อการประมวลผลเสร็จสิ้นจะทำให้ได้สมาชิกของคำตอบชุดสุดท้ายและได้องค์ประกอบของผลลัพธ์สมบูรณ์ (solt, soltw, solx) ด้วยความซับซ้อนด้านเวลาที่ $O(n+C')$

ตารางที่ 5.3 แสดงการเปรียบเทียบความซับซ้อนด้านเวลาของทั้ง 3 ขั้นตอนวิธีข้างต้นด้วยการประมวลผลแบบขนานสำหรับขั้นตอนวิธีที่ได้ความสมบูรณ์ของผลลัพธ์ครบถ้วน (Complete-DP) ขั้นตอนวิธีกรีดีฮิวริสติก (GH) มีความซับซ้อนด้านเวลาที่ $O[(\log_2 n)^2]$ ในกรณีที่จำนวนโปรเซสเซอร์มีจำนวนเท่ากับจำนวนสมาชิกของปัญหา ($p = n$) และ $O(n/p \log_2 p)$ ในกรณีที่จำนวนโปรเซสเซอร์มีจำนวนน้อยกว่าจำนวนสมาชิกของปัญหา ($p < n$) สำหรับขั้นตอนวิธีโปรแกรมเชิงพลวัตได้พิจารณาจากขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบสมบูรณ์ (Complete-DP) ซึ่งมีความซับซ้อนด้านเวลาที่ $O(kC)$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อ $p = n$ และ $O(knC/p)$ เมื่อ $p < n$ ในขณะที่ขั้นตอนวิธีที่เอสรัดักชันมีความซับซ้อนด้านเวลาที่ $O([\log_2 n]^2 + C'/n)$ เมื่อ $p = n$ และ $O(n/p \log_2 p + C'/p)$ เมื่อ $p < n$ ซึ่งมีผลมาจากการแบ่งงานอย่างสมดุลให้กับแต่ละโพรเซสเซอร์ร่วมกับการติดต่อสื่อสารอย่างมีประสิทธิภาพและการเรียงลำดับด้วยขั้นตอนวิธีโอบีเอส [54] การหาผลรวมแบบพรีฟิฟซีในขั้นตอนวิธีฮิวริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพ และขั้นตอนจีเอชพลัส (GH⁺) รวมถึงขั้นตอนการแบ่งกลุ่มสมาชิก การคัดกรอง และการประมวลผลด้วยขั้นตอนวิธีเอ็มดีพี (MDP)

ตารางที่ 5.2 เปรียบเทียบวิธีการ จุดแข็ง และจุดอ่อนของขั้นตอนวิธี

Algorithms	Methodology	Strength	Weakness
GH1(P/W)	Best ranking by combining P/W	- Fast finding initial solutions in $O(n)$.	- Not guarantee the optimal solutions.
GH2(P)	Ranking by P (profit)	- Fast finding initial solutions in $O(n)$.	- Not guarantee the optimal solutions.
GH3(W)	Raking by W (weight)	- Fast finding initial solutions in $O(n)$.	- Not guarantee the optimal solutions.
Basic DP	(Simple) Dynamic Programming (using 2-D array).	- Found the optimal solutions in $O(nC)$. - Get the full solution ($solx$, $soltw$).	- Use 2-D array ($n \times C$) that cannot be simply implemented for large n and massive $C \leq n^2$.
Fast DP	(Improved) Dynamic Programming (on single array).	- Found the optimal solutions in $O(nC)$. - Save space with a single 1-D array.	- Compute only total profit ($soltw$) and total weight ($soltw$).
Complete DP	(Completed) Dynamic Programming of the fast DP.	- Found the optimal solutions. - Save space with a single 1-D array. - Get the full solution ($solx$, $soltw$).	- After compute total profit ($soltw$), repeat process for the full solution ($solx$, $soltw$) in $O(knC)$, where $k = \text{selected objects} < n$.
Our Time-space Reduction (TS _{Reduction})	Integration of 1. the (fast) GH (P/W , P , W), 2. our unbiased filtering (to reduce n), and 3. the (optimal) DP on small n' .	- Found good solutions in $O(n+C')$ and most of solutions are optimal. - Save space with a single array. - Get the full solution ($solx$, $soltw$). - Combine the fast GH on large n and optimal DP on small n' .	-

ตารางที่ 5.3 เปรียบเทียบความซับซ้อนด้านเวลาของขั้นตอนวิธีแบบขนาน

Algorithms	Optimal Solutions	Time Complexity	
		$p = n$	$p < n$
Parallel GH (P/W)	n/a Opt.	$O([\log_2 n]^2)$	$O(n/p \log_2 p)$
Parallel Complete-DP	Optimal	$O(kC)$	$O(knC/p)$
Parallel TS _{Reduction}	Mots sols. are Opt.	$O([\log_2 n]^2 + C'/n)$	$O(n/p \log_2 p + C'/p)$

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2 ขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ (DPT-List_{TSR} Algorithm)

การแก้ปัญหาหมัดฟิลแนปแซค (ในทางทฤษฎี) ยังมีการศึกษาวิจัยที่มุ่งเน้นการหาขั้นตอนวิธีที่ดีที่สุด อาทิ ขั้นตอนวิธีบรานซ์แอนด์บาวนด์ และขั้นตอนวิธีไฮเอมดีพี [16] ซึ่งได้ผลลัพธ์ที่มีประสิทธิภาพมากแต่อาจจะสามารถนำไปใช้ได้กับปัญหาที่มีข้อมูลขนาดไม่มาก ($n \leq 500$)

สำหรับปัญหาที่มีขนาดข้อมูลมากๆ (ในทางปฏิบัติ) ก็มีหลากหลายขั้นตอนวิธีอิวิริสติก [43,44,59,65] ที่สามารถนำมาใช้ในการแก้ปัญหาได้ อาทิ ขั้นตอนวิธีทางพันธุกรรม การคำนวณแบบควอนตัม หรือขั้นตอนวิธีแบบผสมผสาน เป็นต้น ซึ่งขั้นตอนวิธีเหล่านี้สามารถนำไปประยุกต์เพื่อแก้ปัญหาหมัดฟิลแนปแซคสำหรับผลลัพธ์แต่ละถุงได้ เช่น สามารถประยุกต์ขั้นตอนวิธี QDGWO [59] สำหรับทั้งปัญหาซึ่งเกิดแนปแซคและหมัดฟิลแนปแซค แต่อย่างไรก็ตามขั้นตอนวิธีอิวิริสติกยังไม่สามารถรับรองได้ว่าจะได้ผลลัพธ์ที่เป็นคำตอบที่ดีที่สุดได้โดยเฉพาะในชุดข้อมูลที่ไม่ปกติและหากชุดข้อมูลมีขนาดใหญ่อาจจะทำให้ยิ่งยากต่อการได้มาของคำตอบที่ดีที่สุด

5.2.1 ความถูกต้องของขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ (Correctness of the DPT-List_{TSR} Algorithm)

ขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ที่นำเสนอในวิทยานิพนธ์นี้ เป็นการมุ่งเน้นการปรับเปลี่ยนรูปแบบการประมวลผลในรูปแบบอาร์เรย์ขนาด $n \times C$ ที่ใช้ในขั้นตอนวิธีโปรแกรมเชิงพลวัต (ซึ่งเป็นวิธีหนึ่งที่สามารถหาคำตอบที่ดีที่สุดได้) ให้เป็นรูปแบบลิสต์ที่เก็บเฉพาะข้อมูลสำคัญที่นำไปสู่ผลลัพธ์แต่ละองค์ประกอบของปัญหา (soltp, soltw, solx) โดยมีจุดประสงค์เพื่อลดขนาดของข้อมูลในหน่วยความจำที่ใช้ในการประมวลผลรวมถึงเพิ่มความเร็วในการประมวลผลให้มีประสิทธิภาพมากขึ้น อีกทั้งยังสามารถหาคำตอบที่ดีที่สุดได้ การแปลงรูปที่นำเสนอได้ใช้ลิสต์สำหรับเก็บข้อมูล 2 ชนิด ได้แก่ F-list และ B-list โดยการประมวลผลสิ่งของแต่ละชั้นจะมี F-list เพียง F-list $j-1$ ที่เก็บ e-node สำรองมาจาก F-list ก่อนหน้าและ F-list j ที่ต้องหา e-node (tp, d) ใหม่ที่มีมูลค่า (tp) ใหม่ที่ดีกว่า ณ น้ำหนัก (d) เดิม ดังนั้น การประมวลผลของ F-list j จึงต้องอาศัยการสืบทอดมาจาก F-list $j-1$ เท่านั้น ซึ่งเมื่อได้ e-node ใหม่ แสดงว่าเป็นโหนดที่มีมูลค่ามากที่สุดในน้ำหนักที่ไม่เกินความจุ ก็จะเก็บ e-node นั้นเป็น original e-node ใน B-list เพื่อนำไปแทรกหาหมายเลขสิ่งของ (x_j) ที่เป็นคำตอบต่อไป ซึ่งขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ที่นำเสนอนี้ สามารถแก้ไขปัญหาซึ่งเกิดแนปแซค (0/1-KP) ได้ในเวลา $O(n^2, nC)$ ในรูปแบบของลิสต์ อีกทั้งยังสามารถหาคำตอบที่ดีที่สุดได้เหมือนขั้นตอนวิธีโปรแกรมเชิงพลวัต ก่อนที่จะนำไปประยุกต์ใช้กับปัญหาหมัดฟิลแนปแซคต่อไป

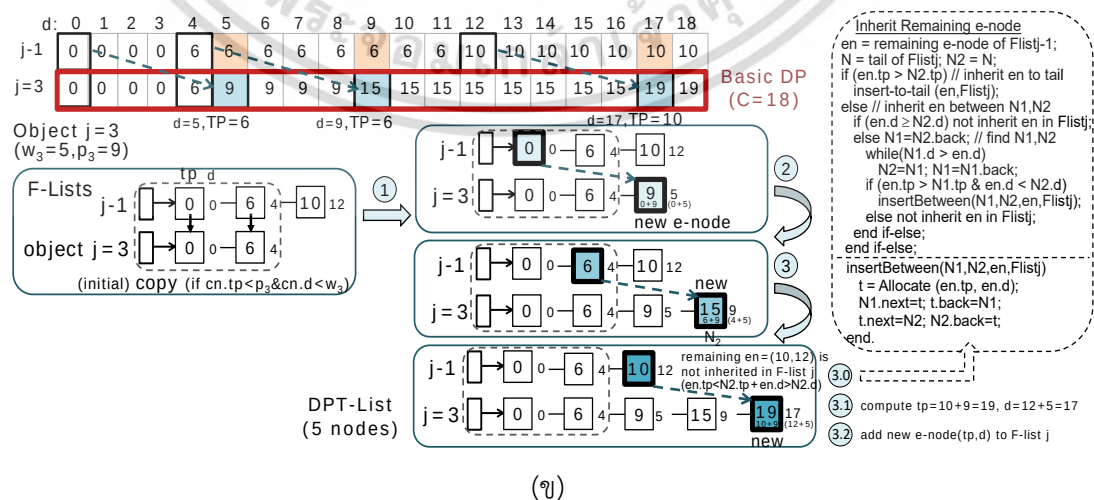
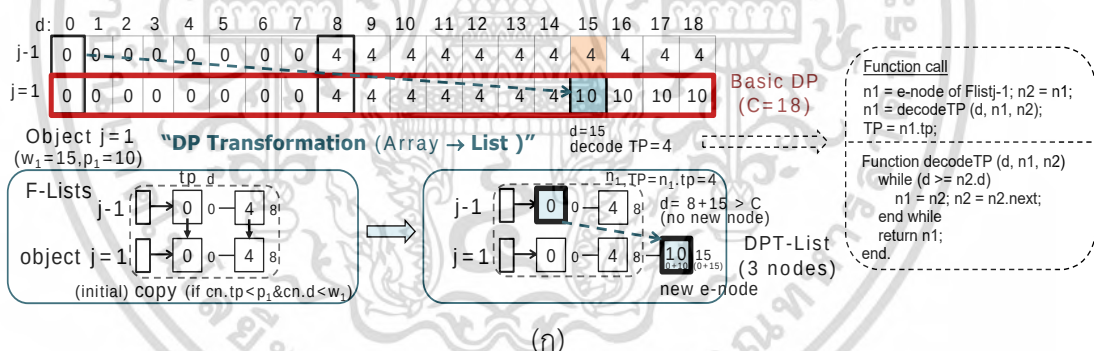
รูปที่ 5.5 แสดงความถูกต้องแม่นยำของการแปลงรูปของขั้นตอนวิธีโปรแกรมเชิงพลวัตที่ได้แสดงการทำงานในรูปที่ 4.1 ซึ่งเป็นการสร้าง e-node ใน F-list โดยในที่นี้ได้แสดงตัวอย่างของสิ่งของ $j=1$ ($p_1 = 10, w_1 = 15$) ในรูปที่ 5.5 (ก) หลังจากได้คัดลอก 2 e-node คือโหนด (0, 0) และ (4, 8) ($cn = (tp, d) = (0, 0), (4, 8)$) จาก F-list $j-1$ ไว้ใน F-list $j=1$ เพราะมีค่า $cn.tp < p_1$ และ $cn.d < w_1$ จากนั้นจึงเป็นการพิจารณาเพื่อเพิ่มโหนดใหม่ใน F-list $j=1$ ซึ่งเมื่อพิจารณาจาก e-node (0, 0) ใน F-list $j-1$

จะทำให้ได้ e-node ใหม่คือ (10, 15) เพราะ $d (=15)$ ที่มีค่าน้อยกว่า $C (=18)$ และ $tp (=10)$ ที่มีค่ามากกว่า $p_1 (=10)$ ทั้งหมดนี้แสดงให้เห็นว่าการแปลงรูปของขั้นตอนวิธีโปรแกรมเชิงพลวัตเป็นรูปแบบลิสต์นั้นเป็นเอกสารถูกต้องและสามารถนำมาใช้ในการแก้ปัญหาหมัดฟิลแนปแซคได้ โดยไม่ต้องกังวลถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มากกว่า TP (=4 ที่ เป็นค่า tp ในตำแหน่งที่ d=15 ใน F-list j-1) และเมื่อพิจารณาโหนด e-node (4, 8) ใน F-list j-1 ได้ค่า $d = 8 + 15 = 23$ ซึ่งมากกว่า C จึงไม่มี e-node เพิ่ม

สำหรับในรูปที่ 5.5 (ข) เป็นการแสดงตัวอย่างการสร้าง e-node ของสิ่งของที่ $j=3$ ($p_3 = 9$, $w_3 = 5$) ใน F-list j-1 โดยหลังจากที่คัดลอก e-node ที่มีน้ำหนักและมูลค่าน้อยกว่าสิ่งของปัจจุบันจาก F-list j-1 ไปไว้ใน F-list j=3 ซึ่งได้ทั้ง 2 e-node นั่นคือ (0, 0) และ (6, 4) แล้ว จากนั้นจะเป็นการพิจารณาโหนดใน F-list j-1 เพื่อเพิ่มโหนดให้กับ F-list j=3 โดยเริ่มจากโหนด (0, 0) ที่ได้ e-node ใหม่คือ (9, 5) ต่อมาเมื่อพิจารณาโหนด (6, 4) ทำให้ได้ e-node ใหม่ คือ (15, 9) (มาจาก $(6+9, 4+5)$) ต่อมาเมื่อพิจารณาโหนด (10, 12) (ถึงแม้ โหนดนี้ไม่ได้ถูกนำมาใส่ใน F-list j=3 ไม่ถูกสืบทอดเพราะมีโหนด (15, 9) ใน F-list j ที่ดีกว่า เพราะมีมูลค่า ($tp=15, d=9$) สูงกว่า ($tp=10, d=12$)) แต่ยังคงใช้สืบทอดมาเพื่อพิจารณาในการสร้าง e-node ใหม่สำหรับสิ่งของปัจจุบัน ซึ่งได้โหนดใหม่คือ $(10+9, 12+5) = (19, 17)$

ในการเพิ่ม e-node ใหม่แต่ละโหนดของแต่ละสิ่งของ e-node เหล่านั้นจะถูกเพิ่มเป็น original e-node ใน B-List ตั้งแต่ L_0 ถึง L_{n-1} ด้วย เพื่อใช้ในการแทรกด้วยกระบวนการเอ็กแทรกกิ้งหาผลลัพธ์ส่วน solx ดังได้แสดงตัวอย่างในรูปที่ 4.1 และรูปที่ 4.3 ซึ่งได้ผลลัพธ์เช่นเดียวกับขั้นตอนวิธีโปรแกรมเชิงพลวัต



รูปที่ 5.5 ตัวอย่างการสร้าง e-node ใน F-list j

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2.2 ความซับซ้อนของขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์ (Complexity Analysis of the DPT-List_{TSR} Algorithm)

ความซับซ้อนด้านเวลาของขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์สำหรับปัญหา 0/1-KP อยู่ที่ $O([n^2, nC])$ ด้วยการลดความซ้ำซ้อนและเนื้อที่ในการประมวลผล ดังในรูปที่ 5.6 ที่แสดงการลดความซ้ำซ้อนแบบ F (forward) และ B (backward) ของขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์ซึ่งความซับซ้อนด้านเวลาขึ้นอยู่กับ e-node ของ F-list_{j-1} และ F-list_j โดยค่าตั้งต้นของ F-list ประกอบด้วย e-node ค่าเดียว ($tp = 0, d = 0$) ณ สิ่งของที่ $j=0$ มี e-node จำนวนไม่เกิน 2 โหนด ณ สิ่งของที่ $j=1$ มี e-node จำนวนไม่เกิน 4 โหนด และในสิ่งของอื่นๆ ก็เช่นกันที่มีจำนวน e-node น้อยกว่า C ซึ่งความซับซ้อนด้านเวลาของขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์จะนำเสนอทั้งในกรณีที่ดีที่สุด (Best case) กรณีเฉลี่ย (Average case) และกรณีที่แย่ที่สุด (Worse case) เช่นในรูปที่ 4.2 เป็นตัวอย่างกรณีที่ดีที่สุดกับปัญหาขนาด $n = 5$, e-nodes = 19 ($1+2+3+3+5+5$) และ original e-nodes = 7 เมื่อเทียบกับขนาด $n \times C$ ซึ่งเท่ากับ 90 โหนด รูปที่ 4.3 เป็นอีกตัวอย่างในกรณีที่แย่ที่สุดกับปัญหาขนาด $n = 15$, e-nodes = 223 และ original e-nodes = 103 เมื่อเปรียบเทียบกับขนาด $n \times C$ ซึ่งเท่ากับ 600 โหนด จากข้างต้นจึงสามารถสรุปความซับซ้อนในกรณีต่างๆ ดังนี้

กรณีที่ดีที่สุด (Best case) มีจำนวนขั้นตอนทั้งหมดสำหรับ n ลิสต์ จะมีความซับซ้อนด้านเวลาที่ $1 + 2 + 3 + \dots + (j+1) + \dots + n$ หรือประมาณ $n(n+1)/2 = O(n^2)$ กรณีรองลงมา เป็น $1 + 2 + 4 + \dots + 2(j+1) + \dots + 2n$ หรือประมาณ $n(n+1) = O(n^2)$

กรณีที่แย่ที่สุด (Worst case) ซึ่งมีโอกาสเกิดขึ้นได้น้อย จะมีความซับซ้อนด้านเวลาที่ $1 + 2 + (\leq 4) + (\leq 8) + \dots + (\leq 2 \times 2^j) + \dots + (< 2^n < C) \times (n \times C)/2$ ซึ่งน้อยกว่า $n \times C$ นั่นคือเป็น $O(nC)$ โดยปกติ กรณีที่แย่ที่สุดมีโอกาสยากมากที่จะเกิดขึ้นเนื่องจาก e-node ของแต่ละสิ่งของจะน้อยกว่า 2×2^j ($< C$) ขึ้นอยู่กับ 3 เงื่อนไข คือ 1) $\text{node.tp} + p_j < \text{tp}[d]$ (ไม่มีการปรับปรุงค่า tp) 2) $\text{node.d} + w_j > C$ (สิ่งของ j ไม่สามารถบรรจุในถุงเป้) 3) ในขั้นสองไม่สืบทอดบางโหนดจาก F-list $j-1$ (ที่เหลือจากขั้นแรก) มายัง F-list j

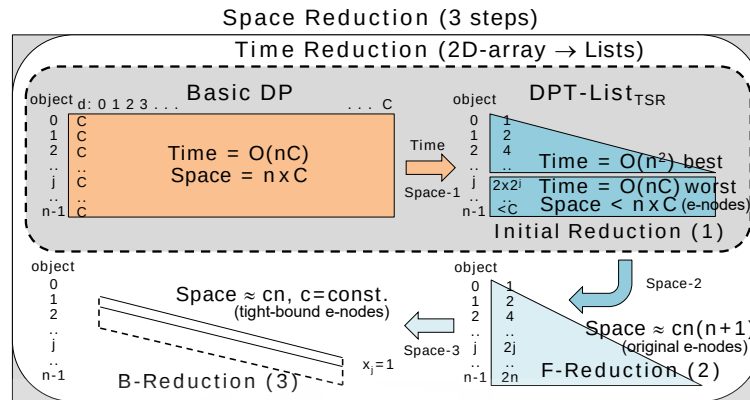
กรณีเฉลี่ย (Average cases) ซึ่งได้แสดงตัวอย่างในรูปที่ 4.10 ที่ $n=15$ e-node=223 และ original e-nodes=103 เมื่อเปรียบเทียบกับขนาดโหนดที่ต้องใช้ทั้งหมดจริงที่ 600 โหนด ซึ่งความซับซ้อนด้านเวลาโดยเฉลี่ยขึ้นอยู่กับชุดข้อมูลและค่า C นั่นคือ ประมาณช่วง (กรณีที่ดีที่สุด + กรณีที่แย่ที่สุด)/2 หรือประมาณ $nC/2$ ตัวอย่างเช่น น้ำหนัก w_j ของสิ่งของ j อยู่ระหว่าง $1 \leq w_j \leq C$ ดังนั้น ค่าเฉลี่ยของ w_j จะเป็น C/n เช่น $C = n^d$ สำหรับกรณีอื่นที่ $w_j \geq C/n$ อาจจะไม่มีการเพิ่ม e-node ก็ได้เนื่องจาก $\text{node.d} + w_j > C$

สำหรับการวิเคราะห์ขนาดหน่วยความจำในการประมวลผล (Space Complexity) ในรูปที่ 5.6 ได้แสดงการลดขนาดหน่วยความจำ 3 ระดับ ได้แก่ 1) e-node 2) original e-node และ 3) tight bound ของ e-node ดังในรูปที่ 4.4 ที่ $n=5$ และรูปที่ 4.5 ที่ $n=15$ ซึ่งในผลการทดลอง (แสดงในตารางที่ 6.9) ได้ทำการตรวจสอบกับข้อมูลขนาด $n \leq 3000$ ซึ่งเมื่อผ่านกระบวนการ F-reduction จำนวนของ Original e-node อยู่ที่ประมาณ n^2 และเมื่อผ่านกระบวนการ B-reduction จำนวนของ

Original e-node เหลืออยู่น้อยกว่า n^2

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

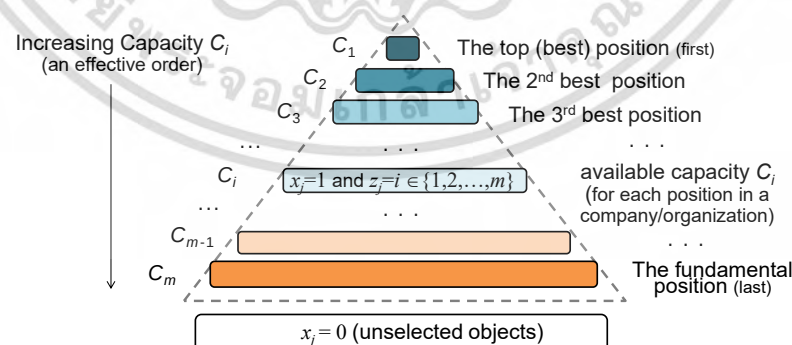
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.6 ภาพรวมของการลดขนาดข้อมูลด้วย F-reduction และ B-reduction ของขั้นตอนวิธีดีฟิทีลิสต์ทีเอสอาร์

5.2.3 ความแม่นยำของการเรียงถุงเป้ที่เหมาะสมพอดี (Exact Precision (as m! Orders) of the Exact-fit (Best) Knapsack Order)

ในทางทฤษฎี การหาคำตอบที่ดีที่สุดของปัญหา 0/1-mKP สามารถประมวลผลได้โดยวิธีโบราณซ์แอนด์บาวนด์ (Branch-and-Bound) แต่ต้องใช้เวลาประมวลผลที่ $O((m+1)^n)$ สำหรับในทางปฏิบัติ ผู้วิจัยได้นำเสนอขั้นตอนวิธีดีฟิทีลิสต์ทีเอสอาร์ (บทที่ 4) สำหรับหาคำตอบที่ดีที่สุดของแต่ละถุงเป้ได้ในเวลา $O([n^2, nC])$ โดยได้คำตอบที่ดีที่สุดที่ 99% ซึ่งในขั้นตอนวิธีที่นำเสนอกลยุทธ์การจัดเรียงถุงเป้ 2 กลยุทธ์ คือ 1. การเรียงถุงเป้ 9 อันดับแรกที่มีประสิทธิภาพ (The (Fast) Top 9 Knapsacks Orders Strategy) ที่มีความซับซ้อนด้านเวลาที่ $O(m[n^2, nC])$ สำหรับชุดข้อมูลปกติ ซึ่งได้มีการศึกษาค้นคว้าตั้งแต่ปี 2018-2019 และ 2. การเรียงถุงเป้ที่มีประสิทธิภาพ (The Exact-fit (Best) Order) ที่มีความซับซ้อนด้านเวลาที่ $O(m^2[n^2, nC])$ สำหรับชุดข้อมูลที่ไม่ปกติ ซึ่งผู้วิจัยได้ทำการศึกษาและพัฒนาตั้งแต่ปี 2020-2021 เพื่อแข่งขันกับขั้นตอนวิธีที่เคยได้คิดค้นขึ้น



รูปที่ 5.7 แผนภาพพีรามิดแสดงการเรียงลำดับความจุของถุงเป้จากน้อยไปมาก กรณีปัญหาถุงเป้แบบหลายมิติ

สำหรับชุดข้อมูลปกติ การจัดเรียงที่มีประสิทธิภาพมากที่สุด (สำหรับปัญหาที่ถุงเป้ที่มีขนาดไม่ใหญ่) จะเป็นการเรียงขนาดถุงเป้จากน้อยไปมาก ดังแสดงในรูปที่ 5.7 เป็นการแสดงแนวคิดของเอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อใช้ในการศึกษาเท่านั้น ไม่อนุญาตให้นำไปเผยแพร่โดยไม่ได้รับอนุญาต หรือทำซ้ำโดยไม่ได้รับอนุญาต หากต้องการนำเอกสารนี้ไปใช้ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การจัดเรียงถุงเป้ในรูปของตำแหน่งงานและลำดับตำแหน่งที่ต้องถูกกำหนดบุคคล โดยตำแหน่งบนสุดในรูปเปรียบได้กับตำแหน่งด้านการบริหารซึ่งมีจำนวนบุคลากรน้อยที่สุดในองค์กร แต่เป็นกลุ่มคนที่ต้องมีความรู้ความสามารถสูงสุดและต้องรับผิดชอบมากที่สุด ซึ่งเป็นตำแหน่งที่ต้องถูกกำหนดบุคคลที่เหมาะสมที่สุดก่อนเป็นอันดับแรก ลำดับรองลงมาคือกลุ่มคนในระดับบริหารรองลงมาซึ่งมีจำนวนมากกว่าระดับบนสุดซึ่งเป็นตำแหน่งรองลงมาที่ถูกกำหนดบุคคล โดยกลุ่มคนในระดับนี้จะต้องมีความรู้ความสามารถและความรับผิดชอบรองลงมาจากระดับบนสุด ในแต่ละระดับจะมีจำนวนคนที่มากขึ้นตามลำดับจนกระทั่งถึงขั้นที่เป็นตำแหน่งงานทั่วไปที่เป็นกลุ่มคนที่มีจำนวนมากที่สุดในองค์กร ซึ่งต้องมีความรู้ความสามารถและความรับผิดชอบเหมาะสมกับงานและองค์กร ส่วนสิ่งของที่ไม่ได้ถูกเลือกเปรียบเสมือนบุคคลที่ไม่ได้รับการคัดเลือกให้ปฏิบัติงานในองค์กรซึ่งเป็นคนที่มีความรู้ความสามารถแต่อาจจะไม่เหมาะสมกับงานและองค์กร โดยสิ่งของเหล่านี้อาจจะเหมาะสมกับงานและองค์กรอื่นได้ ซึ่งเปรียบได้กับสิ่งของเหล่านี้อาจจะเหมาะสมสำหรับถุงเป้อื่น ในแต่ละการจัดเรียงถุงเป้ได้มีการเพิ่มประสิทธิภาพด้วยการเรียงแบบสับเปลี่ยน ซึ่งได้พัฒนาภายใต้แนวคิดลาตินสแควร์ (LS) ที่มีการสับเปลี่ยนจำนวน m ซึ่งผู้วิจัยได้เลือกการจัดเรียงของลาตินสแควร์มาเพียง 9 รูปแบบ เพื่อจำกัดเวลาในการประมวลผลให้อยู่ในขอบเขต $O(m[n^2, nC])$ ดังแสดงในตารางที่ 5.4 ที่มีจำนวนถุงเป้อยู่ระหว่าง $5 \leq m \leq 100$

ตารางที่ 5.4 เปรียบเทียบประสิทธิภาพการจัดเรียงถุงเป้สำหรับปัญหา 0/1-mKP

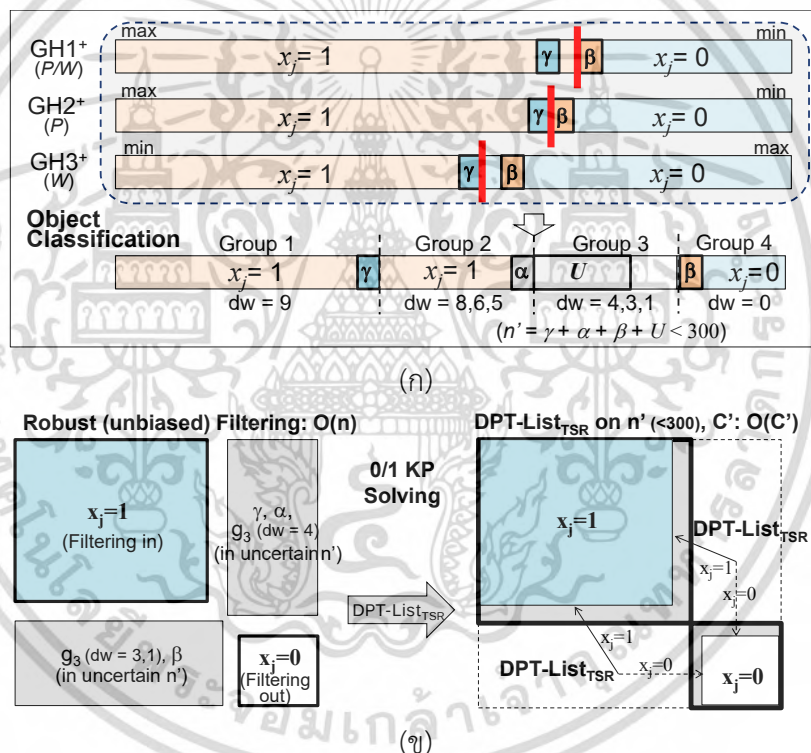
Multi-DPT-List + effective knapsack orders					
m knapsacks	All orders ($m!$)	Exact-fit/Best ($m(m+1)/2$)	Top (9)	Partial LS min($9m, 9 \times 9$)	Full LS ($9m$)
5	120	15	9	45	45
6	720	21	9	54	54
7	5040	28	9	63	63
8	40320	36	9	72	72
9	326880	45	9	81	81
10	3268800	55	9	81	90
20	20 !	210	9	81	180
50	50 !	1275	9	81	450
100	100 !	5050	9	81	900

สำหรับชุดข้อมูลที่ผิดปกติ จำเป็นต้องใช้การเรียงถุงเป้ที่เหมาะสมพอที่จะทำให้ได้คำตอบที่ดีที่สุดเหมือนการจัดเรียงขนาด $m!$ ผู้วิจัยได้ใช้กลยุทธ์การเรียงถุงเป้ที่เหมาะสมพอสำหรับวิธีมัลติดีพีทีลิสต์ทีเอสอาร์ (multi-DPT-List_{TSR}) ด้วยความซับซ้อนด้านเวลาที่ $O(m^2[n^2, nC])$ ซึ่งขั้นตอนการเรียงถุงวิธินี้มีทั้งหมด $m(m+1)/2$ ขั้น โดยผลลัพธ์ที่ได้เหมือนกับการจัดเรียงถุงทุกกรณีที่มี $m!$ ขั้น ซึ่งการเรียงเหมาะสมพอที่นี่ ถุงเป้ที่ K_i ที่ดีที่สุดจะถูกเลือกโดยค่า $\min(\text{Fit})$ ที่ดีที่สุด ซึ่ง $\text{Fit}_i = C_i - \text{TW}_i$ รวมถึง TP_i และ TW_i ที่ดีที่สุดที่ได้จากการประมวลผลด้วยขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ในแต่ละถุงเป้นอกจากนี้ ในกรณีที่มีค่า $\min(\text{Fit})$ เท่ากันก็ยังมีเงื่อนไขในการพิจารณาเพื่อเลือก K_i ที่ดีที่สุด นำไปสู่ผลลัพธ์ที่มีความถูกต้องถึง 99% ซึ่งร่วมกับประสิทธิภาพของการเรียงถุงเป้ที่เหมาะสมพอ

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของโรงเรียนเพื่อการศึกษาเท่านั้น เมื่อผู้ใดนำเอกสารนี้ไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2.4 ความแม่นยำของการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (High Exact Precision of the Robust Unbiased Filtering)

ในทางปฏิบัติ การแก้ปัญหาซิงเกิลแนปแซค (0/1-KP) ด้วยขั้นตอนวิธีดีพีที่ลิสต์ที่เอสอาร์ ที่ทำงานร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (Robust Unbiased Filtering) สามารถทำได้ในเวลา $O(n+C')$ หรือ $O(n)$ ที่ $C' \ll C$ และแก้ปัญหาแมลติเพิลแนปแซค (0/1-mKP) ได้ในเวลา $O(m^2n)$ การคัดกรองแบบไม่เอนเอียงที่แข็งแกร่งเป็นการจัดกลุ่มสิ่งของด้วยค่าน้ำหนัก (dw = Dynamic Weight) โดยมาจากการบูรณาการมาจากการเรียงสิ่งของ 3 รูปแบบ (P/W, P, W) ดังนั้น สิ่งของที่มีค่าน้ำหนักดีที่สุด ($dw=9$) คือสิ่งของที่ได้รับการยอมรับตรงกันทั้ง 3 รูปแบบ จึงเป็นสิ่งของที่ควรถูกเลือกให้เป็นคำตอบ ($x_j=1$) ในขณะที่สิ่งของที่มีค่าน้ำหนักแย่ที่สุด ($dw=0$) คือสิ่งของที่ไม่ได้รับการยอมรับจากทั้ง 3 รูปแบบ จึงเป็นสิ่งของที่ควรตัดทิ้ง



รูปที่ 5.8 ความแม่นยำของการคัดกรองแบบแข็งแกร่งสำหรับปัญหาซิงเกิลแนปแซค (0/1-KP)

การคัดกรองแบบไม่เอนเอียงในขั้นตอนวิธีทีเอสรีดักชัน ($TS_{Reduction}$) ที่ใช้ในการแก้ไขปัญหาลิงเกิลแนปแซคมีการกำหนดขอบเขตด้วยค่า α และ β ซึ่งใช้ในการแบ่งขอบเขตสำหรับข้อมูลพิเศษที่ไม่สามารถหาคำตอบได้ หรือเรียกว่า Outlier โดยจะถูกนำไปประมวลผลด้วยวิธีเอ็มดีพี (MDP) ต่อไป ซึ่งด้วยวิธีนี้ทำให้ได้คำตอบที่ดีที่สุดอย่างน้อย 99% การศึกษาในครั้งนี้พบว่า การคัดกรองแบบไม่เอนเอียงที่แข็งแกร่งที่เพิงพารามิเตอร์ γ ในกลุ่ม 1 ($dw=9$) สามารถควบคุมกลุ่ม Deep Outlier ซึ่งอาจจะเป็นสิ่งของบางชิ้นที่อยู่ใกล้ท้ายของกลุ่ม $dw=9$ นอกจากนี้ อาจบางชิ้นที่อยู่อันดับต้นของกลุ่ม $dw=0$ เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(รูปที่ 5.8 (ก)) จึงมีการเพิ่มค่า γ เพื่อยืดหยุ่นขอบเขตวิกฤติให้เหมาะสมกับข้อมูลมากขึ้น ดังในรูปที่ 5.8 (ข) แสดงสิ่งของที่ไม่สามารถพิจารณาได้ชัดเจนใน γ และ β ซึ่งวิธีการนี้ช่วยการกำหนดขอบเขตวิกฤติสำหรับสิ่งของที่ไม่สามารถพิจารณาได้ในแต่ละกลุ่มมีความยืดหยุ่นให้เหมาะสมกับชุดข้อมูลดังในรูปที่ 5.8 (ค)



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6

ผลการทดลอง

วิทยานิพนธ์นี้เป็นการนำเสนอขั้นตอนวิธีสำหรับปัญหาแนปแซคทั้งเชิงเกิลแนปแซค (0/1-KP) ที่มีหนึ่งถุงเป้ในชุดปัญหาและปัญหามัลติเพิลแนปแซค (0/1-mKP) ที่มีหลายถุงเป้ในชุดปัญหาที่มีจุดประสงค์ในการหาคำตอบที่ดีที่สุด (Optimal Solution) ให้ได้มากที่สุด (อย่างน้อย 99%) ในเวลาโพลิโนเมียล (Polynomial Time) ซึ่งขั้นตอนวิธีที่ได้นำเสนอมีเป็น 2 ส่วน ได้แก่ 1) ขั้นตอนวิธีทีเอสรีดักชัน ($TS_{Reduction}$) (บทที่ 3) สำหรับปัญหาเชิงเกิลแนปแซคโดยมุ่งเน้นกระบวนการคัดกรองแบบไม่เอนเอียง (Unbiased Filtering) และโปรแกรมเชิงพลวัตที่ได้ดัดแปลงขึ้น (MDP) และ 2) ขั้นตอนวิธีมัลติดีฟิสิสต์เอสอาร์ (Multi-DPT-List_{TSR}) (บทที่ 4) เป็นการพัฒนาต่อยอดเพื่อแก้ปัญหามัลติเพิลแนปแซคซึ่งมีความซับซ้อนมากกว่าปัญหาเชิงเกิลแนปแซค โดยในการพัฒนาต่อยอดนี้ได้เปลี่ยนแปลงรูปแบบการเก็บข้อมูลที่ใช้ในการประมวลผลตามรูปแบบของโปรแกรมเชิงพลวัตที่มีขนาด $n \times C$ ให้อยู่ในรูปแบบลิสต์ที่เก็บเฉพาะข้อมูลสำคัญที่ทำให้ได้องค์ประกอบของคำตอบของปัญหา (soltp, soltw, solx) และพัฒนากระบวนการคัดกรองแบบไม่เอนเอียงให้มีความยืดหยุ่นเหมาะสมกับปัญหามากขึ้น รวมถึง ด้วยรูปแบบปัญหาที่มีหลายถุงเป้ ผู้วิจัยจึงนำเสนอการเรียงถุงเป้ที่เหมาะสมพอดีเพื่อเพิ่มประสิทธิภาพเพื่อให้ได้คำตอบที่ดีที่สุดให้ได้มากที่สุด สำหรับในบทนี้ เป็นการนำเสนอผลการทดลองของขั้นตอนวิธีที่ได้นำเสนอ ได้แก่ 6.1 ผลการทดลองของขั้นตอนวิธีทีเอสรีดักชัน (บทที่ 3) 6.2 ผลการทดลองของขั้นตอนวิธีดีฟิสิสต์ทีเอสอาร์และมัลติดีฟิสิสต์ทีเอสอาร์ (บทที่ 4)

6.1 ผลการทดลองของขั้นตอนวิธีทีเอสรีดักชัน (Experimental Result of $TS_{Reduction}$ Algorithm)

การวัดประสิทธิภาพของขั้นตอนวิธีทีเอสรีดักชันที่นำเสนอในบทที่ 3 ได้ทำการทดลองเพื่อเปรียบเทียบความถูกต้องของผลลัพธ์ของขั้นตอนวิธีทีเอสรีดักชัน ($TS_{Reduction}$) ที่นำเสนอ ขั้นตอนวิธีกริดีฮิวริสติก (GH: Greedy Heuristic) ที่เร็ว และขั้นตอนวิธีโปรแกรมเชิงพลวัต (DP: Dynamic Programming) ซึ่งสามารถได้คำตอบที่ดีที่สุด รวมถึงได้เปรียบเทียบเวลาประมวลผลของขั้นตอนวิธีทีเอสรีดักชัน ขั้นตอนวิธีกริดีฮิวริสติก กับขั้นตอนวิธีโปรแกรมเชิงพลวัต โดยชุดข้อมูลที่ใช้ในการทดลองในงานวิจัยนี้แบ่งออกเป็น 3 ชุดข้อมูล ได้แก่

1) ชุดข้อมูลมาตรฐาน (ชุดที่ 1)

ชุดข้อมูลมาตรฐานนี้ได้จากแหล่งข้อมูล [18] ประกอบด้วยข้อมูล 3 ชุดข้อมูล ดังนี้

ชุดข้อมูลที่ 1 ($n = 5, C = 26$)

$$W = \{12, 7, 11, 8, 9\}$$

$$P = \{24, 13, 23, 15, 16\}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชุดข้อมูลที่ 2 ($n = 10, C = 165$)

$W = \{23, 31, 29, 44, 53, 38, 63, 85, 89, 82\}$

$P = \{92, 57, 49, 68, 60, 43, 67, 84, 87, 72\}$

ชุดข้อมูลที่ 3 ($n = 15, C = 750$)

$W = \{70, 73, 77, 80, 82, 87, 90, 94, 98, 106, 110, 113, 115, 118, 120\}$

$P = \{135, 139, 149, 150, 156, 163, 173, 184, 192, 201, 210, 214, 221, 229, 240\}$

2) ชุดข้อมูลมาตรฐาน (ชุดที่ 2)

ชุดข้อมูลมาตรฐานนี้ได้จากแหล่งข้อมูล [27] ประกอบด้วยข้อมูล 5 ชุดข้อมูล ซึ่งแสดงในตารางที่ 6.1 รายการชุดข้อมูลที่ 14, 18, 22, 23, 26, 31

3) ชุดข้อมูลจากการสุ่ม (ชุดที่ 3)

ชุดข้อมูลนี้เป็นชุดข้อมูลที่ได้จากการสุ่มหลายขนาดปัญหา ($5 \leq n \leq 30000$) เพื่อใช้ในการทดลองสำหรับขั้นตอนวิธีแบบอนุกรม (ดังแสดงในตารางที่ 6.2, 6.3, 6.4 และรูปที่ 6.1) สำหรับขั้นตอนวิธีแบบขนานได้สุ่มชุดข้อมูลเพิ่มเติมด้วยขนาด $n = 500000$ (ดังแสดงในตารางที่ 6.5) ด้วยการจำลองหน่วยประมวลผลขนาด $p = 2, 4, 6, \dots, 4096$ PEs ดังแสดงประสิทธิภาพในรูปที่ 6.2

ตารางที่ 6.1 ผลการทดลองส่วนมูลค่ารวมสำหรับชุดข้อมูลมาตรฐาน

Datasets: n	0/1-KP Algorithms				
	GH1(P/W)	GH2(P)	GH3(W)	Fast DP	TS _{Reduction}
Dataset #1:5	47	47	44	51	51
Dataset #2:10	266	179	309	309	309
Dataset #3:15	1302	1315	1249	1458	1458
Dataset #14:100	10592	9788	8166	10785	10785
Dataset #18:500	81254	75326	72106	81522	81522
Dataset #22:900	135320	124014	127496	135632	135632
Dataset #23:1000	82996	82512	82996	83022	83022
Dataset #26:4000	75544	45586	59874	75606	75606
Dataset #31:9000	84723	79944	60504	84897	84897

ตารางที่ 6.2 ผลการทดลองส่วนมูลค่ารวมสำหรับชุดข้อมูลแบบสุ่ม

Datasets	0/1-KP Algorithms				
	GH1(P/W)	GH2(P)	GH3(W)	Fast DP	TS _{Reduction}
$n = 20$	614	560	590	621	621
$n = 35$	6638	5975	6410	6768	6768
$n = 50$	7907	7734	6833	7907	7907
$n = 200$	26086	24899	21674	26091	26091
$n = 400$	41220	41266	32500	41404	41404
$n = 600$	71886	68929	69375	71968	71968
$n = 1000$	82996	42512	82996	83022	83022
$n = 2000$	376785	281510	343765	376926	376926
$n = 3000$	84462	64261	75635	84493	84493
$n = 4000$	75544	45586	59782	75606	75606
$n = 5000$	440412	344114	417261	440653	440653
$n = 6000$	412266	360074	366835	412465	412465
$n = 7000$	212589	154512	175217	212816	212816
$n = 8000$	288179	144014	272014	288228	288228
$n = 9000$	84723	79768	60176	84897	84897
$n = 10000$	211869	157430	196542	211929	211929
$n = 20000$	480954	306778	433961	480974	480974
$n = 30000$	1553299	1208633	1510874	1553414	1553414

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นเพื่อใช้ในการศึกษาวิจัยเท่านั้น ไม่สามารถนำออกจำหน่ายหรือทำซ้ำโดยไม่ได้รับอนุญาต

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 6.1 และ 6.2 แสดงการเปรียบเทียบผลการทดลองด้านมูลค่ารวม (soltp) ของขั้นตอนวิธีที่เอสรัดักชัน ($TS_{Reduction}$) กับขั้นตอนวิธีกริดิวิริสติก (GH) และขั้นตอนวิธีโปรแกรมเชิงพลวัต แบบเร็ว (Fast-DP) ซึ่งประกอบด้วยชุดข้อมูลมาตรฐานและชุดข้อมูลแบบสุ่มที่มีทั้งค่าข้อมูลแบบปกติและไม่ปกติ โดยในงานวิจัยนี้ได้ทำการทดลองกับชุดข้อมูลขนาด $5 \leq n \leq 30000$ พบว่า ขั้นตอนวิธีที่เอสรัดักชัน ได้คำตอบที่ดีที่สุดทุกกรณีสำหรับชุดข้อมูลมาตรฐานและประมาณ 99% สำหรับชุดข้อมูลสุ่ม

ตารางที่ 6.3 ผลการทดลองส่วนน้ำหนักรวมสำหรับชุดข้อมูลแบบสุ่ม

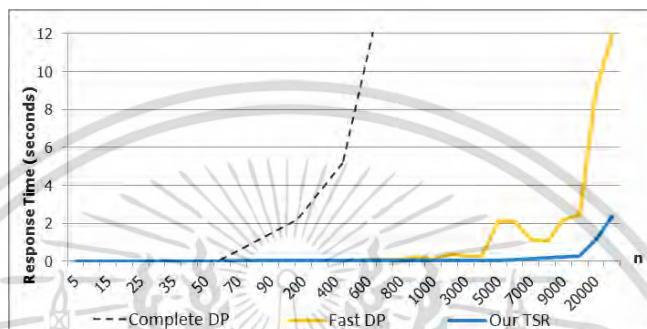
Datasets	C	0/1-KP Algorithms				
		GH1(P/W)	GH2(P)	GH3(W)	Fast DP	$TS_{Reduction}$
$n = 20$	200	191	195	186	200	200
$n = 35$	1100	1053	1062	1030	1096	1096
$n = 50$	1500	1496	1475	1442	1496	1496
$n = 200$	6582	6577	6539	6522	6582	6582
$n = 400$	15800	15709	15747	15712	15800	15800
$n = 600$	25520	25478	25477	25461	25520	25520
$n = 1000$	12822	12796	12812	12796	12822	12822
$n = 2000$	25200	25182	25176	25164	25200	25200
$n = 3000$	10750	10745	10732	10739	10750	10750
$n = 4000$	11350	11338	11337	11345	11350	11350
$n = 5000$	65000	64953	64975	64981	65000	65000
$n = 6000$	56500	56464	54691	56500	56500	56500
$n = 7000$	27060	27024	27060	27057	27060	27060
$n = 8000$	21840	21835	21769	21829	21840	21840
$n = 9000$	42200	42108	42128	42198	42200	42200
$n = 10000$	35975	35962	35970	35938	35975	35975
$n = 20000$	75200	75196	75197	75151	75200	75200
$n = 30000$	102500	102490	102498	102477	102500	102500

ตารางที่ 6.4 เวลาประมวลผลของขั้นตอนวิธีที่เอสรัดักชัน ขั้นตอนวิธีกริดิวิริสติก และขั้นตอนวิธีโปรแกรมเชิงพลวัต

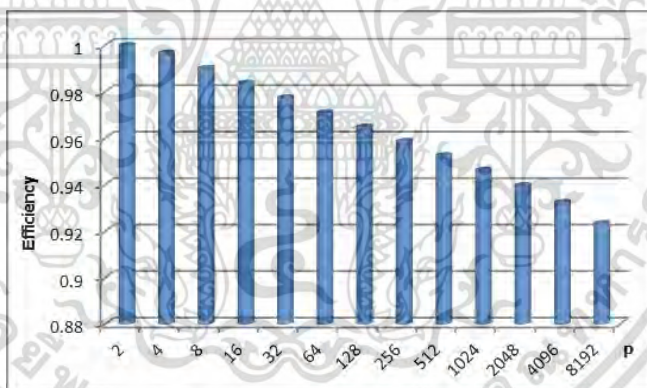
n	GH (P/W)	Fast DP (soltp, soltw)	Complete DP (soltp, soltw, solx)	Proposed $TS_{Reduction}$
500	0.002	0.051	5.091	0.015
1000	0.007	0.089	33.16	0.031
2000	0.013	0.374	10.76	0.015
3000	0.018	0.283	43.33	0.031
4000	0.040	0.267	30.36	0.053
5000	0.043	2.111	222.45	0.084
6000	0.077	2.077	102.04	0.115
7000	0.100	1.118	101.99	0.153
8000	0.115	1.036	314.03	0.200
9000	0.146	2.180	196.38	0.246
10000	0.181	2.440	2261.65	0.284
20000	0.697	9.028	3522.29	1.130
30000	1.551	11.952	58675.84	2.409

นอกจากนี้ ผู้วิจัยได้เปรียบเทียบประสิทธิภาพด้านความเร็วในการประมวลผลของขั้นตอนวิธีที่นำเสนอกับขั้นตอนวิธีอื่น ในตารางที่ 6.4 และรูปที่ 6.1 สำหรับการประมวลผลแบบอนุกรม จะเห็นเอกสารนี้เป็นเอกสารทงสวนวสสำหรับการใช้งานเพื่อการศีกษาเท่านั้น ไมออนุญาตให้นำไปใชประโยชน์ดานการคาไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ว่าความเร็วในการประมวลผลของขั้นตอนวิธีที่เอสรีดักชันมีความเร็วใกล้เคียงกับขั้นตอนวิธีกรีดี ฮิวริสติก และเร็วกว่าขั้นตอนวิธีโปรแกรมเชิงพลวัต แบบเร็วและแบบสมบูรณ์มาก สำหรับเวลาที่ใช้ในการประมวลผลแบบขนาน ซึ่งแสดงในตารางที่ 6.2 และรูปที่ 6.5 เป็นการทดลองความเร็วสำหรับชุดข้อมูลขนาด $n = 500000$ ด้วยการเปรียบเทียบความเร็วในการประมวลผลตามจำนวนหน่วยประมวลผลพบว่า ค่า Speedup (S_p) $\rightarrow p$ และ Efficiency (E_p) $\rightarrow 1$ เนื่องจากการติดต่อสื่อสารที่มีประสิทธิภาพ ($O(\log_2 p)$) ในขั้นตอนแรก และ $O(1)$ ในขั้นตอนที่ 2 และ 3



รูปที่ 6.1 แผนภาพแสดงเวลาประมวลผลของขั้นตอนวิธีที่เอสรีดักชัน ($TS_{\text{Reduction}}$) กับขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเร็ว (Fast-DP) และแบบสมบูรณ์ (Complete-DP)



รูปที่ 6.2 แผนภาพแสดงประสิทธิภาพด้านเวลาของขั้นตอนวิธีที่เอสรีดักชันแบบขนาน (Parallel- $TS_{\text{Reduction}}$) ขนาด $n = 5000000$ ด้วยจำนวนหน่วยประมวลผลที่ $p \leq 4096$

ตารางที่ 6.5 ประสิทธิภาพด้านความเร็วสำหรับชุดข้อมูลขนาด $n = 5000000$ ด้วยจำนวนหน่วยประมวลผล ที่ $p \leq 4096$

Coarse-Grained Processing			Medium-Grained Processing		
p	S_p	E_p	p	S_p	E_p
2	2.00	1.0000	128	123.51	0.9649
4	3.99	0.9967	256	245.44	0.9587
8	7.92	0.9902	512	487.71	0.9526
16	15.74	0.9838	1024	968.98	0.9463
32	31.28	0.9774	2048	1924.46	0.9397
64	62.15	0.9711	4096	3818.93	0.9324

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.2 ผลการทดลองของขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์และมัลติดีพีทีลิสต์ทีเอสอาร์

6.2.1 ผลการทดลองของขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (Results of the DPT-List_{TSR} and Robust Unbiased Filtering)

การประเมินประสิทธิภาพของการคัดกรองแบบแข็งแกร่งในขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์ (DPT-List_{TSR}) สำหรับปัญหาซิงเกิลแนปแซค (0/1-KP) ผู้วิจัยได้สร้างชุดข้อมูลแบบสุ่มในแบบ Dynamic Seeds ที่ไม่ระบุรูปแบบการกระจายการสุ่มเพื่อให้ได้ค่ามูลค่า (Profit) และน้ำหนัก (Weight) ของสิ่งของจำนวน n ชิ้น ($n \leq 10000$) ในการทดลอง พบว่า ขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (DPT-List_{TSR} + Robust Unbiased Filtering) สามารถหาคำตอบที่ดีที่สุดในแต่ละชุดข้อมูลได้ ในขณะที่เมื่อใช้ขั้นตอนวิธีทีเอสอาร์ดักชันที่ใช้การคัดกรองแบบไม่เอนเอียงไม่สามารถหาคำตอบที่ดีที่สุดได้บางส่วน (223 ชุดข้อมูลจาก 10000 ชุดข้อมูล)

ตารางที่ 6.6 ผลการทดลองสำหรับชุดข้อมูลไม่ปกติ (จากชุดข้อมูลบางส่วนใน 223 จาก 10000)

n	C	Total Weight (<i>soltw</i>)			Total Profit (<i>soltp</i>)		
		DPT-List (Opt.)	DPT + rFilter	TSR + uFilter	DPT-List (Opt.)	DPT + rFilter	TSR + uFilter
12	96	96	96	92	282	282	280
14	112	111	111	109	365	365	362
21	168	168	168	167	500	500	498
26	208	208	208	203	637	637	636
39	312	312	312	312	868	868	866
45	360	360	360	360	1177	1177	1172
73	510	510	510	509	1640	1640	1639
80	559	559	559	558	1712	1712	1711
81	566	566	566	566	1888	1888	1886
143	1000	1000	1000	1000	3084	3084	3082
147	1028	1028	1028	1028	3250	3250	3239
155	1084	1084	1084	1083	3440	3440	3437
166	1161	1161	1161	1161	3617	3617	3616
182	1273	1273	1273	1273	3967	3967	3966
197	1378	1378	1378	1378	4534	4534	4533
199	1392	1392	1392	1391	4561	4561	4560
247	1481	1481	1481	1480	4822	4822	4821
276	1655	1655	1655	1655	5446	5446	5445
286	1715	1715	1715	1715	5889	5889	5888
316	1895	1895	1895	1895	6266	6266	6257
329	1973	1973	1973	1973	6484	6484	6469
360	2159	2159	2159	2159	6985	6985	6984
385	2309	2309	2309	2309	7710	7710	7709

ตารางที่ 6.7 ความแม่นยำของขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (DPT-List_{TSR} + Robust Unbiased Filtering) เมื่อ ($n \leq 10000$)

#datasets	DPT-List + robust unbiased filtering			TSR + unbiased filtering [11]		
	#not opt	#opt sols	precision	#not opt	#opt sols	precision
$5 \leq n \leq 100$	0	95	99.99%	9	86	90.5%
$5 \leq n \leq 200$	0	195	99.99%	16	179	92.0%
$5 \leq n \leq 500$	0	495	99.99%	23	472	95.4%
$5 \leq n \leq 2000$	0	1995	99.99%	28	1967	98.6%
$5 \leq n \leq 5000$	0	4995	99.99%	109	4886	97.8%
$5 \leq n \leq 10000$	0	9995	99.99%	223	9772	97.8%

เอกสารนี้เป็นเอกสารร่างสำหรับการใช้ภายในเท่านั้น ไม่ควรเผยแพร่สู่สาธารณะโดยไม่ได้รับอนุญาต

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 6.8 ประสิทธิภาพการลดเนื้อที่เก็บข้อมูลของขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์ (DPT-List_{TSR})

n	$n \times C$ (full space)	e-nodes (1. initial reduction)		original e-nodes (2. F-reduction)		tight-bound e-nodes (3. B-reduction)	
5	90	13	86%	7	92%	6	93%
15	600	223	63%	103	83%	50	92%
50	17450	6559	62%	2465	86%	1917	89%
100	69900	35852	49%	15263	78%	10849	84%
200	239800	150518	37%	66883	72%	35932	85%
500	1250000	915303	27%	374729	70%	173786	86%
1000	5000000	3832827	23%	1566414	69%	692691	86%
1500	11250000	8669932	23%	3364525	70%	1581133	86%
2000	11998000	10322643	14%	3293641	73%	1115040	91%
2500	18747500	16071768	14%	5338435	72%	1752052	91%
3000	26997000	23225460	14%	7497627	72%	2620552	90%

ตารางที่ 6.6 แสดงผลการทดลองที่ในของ 23 รายการแรกจาก 223 รายการเป็นชุดข้อมูลผิดปกติที่ $n = 12, 14, 21, 26, 39, \dots, 385$ จาก $n \leq 10000$ ซึ่งไม่สามารถหาคำตอบที่ดีที่สุดได้ด้วยวิธีทีเอสรีดักชัน (TS_{Reduction}) แต่สามารถหาได้ด้วยวิธีพีทีลิสต์ทีเอสอาร์ (DPT-List_{TSR})

ตารางที่ 6.7 แสดงประสิทธิภาพด้านความถูกต้องแม่นยำของขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์ ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (DPT-List_{TSR}+ Robust Unbiased Filtering) เทียบกับขั้นตอนวิธีทีเอสรีดักชัน (TS_{Reduction}) ที่ใช้หลักการ Unbiased Filtering ซึ่งพบว่า ขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง สามารถหาคำตอบที่ดีที่สุดได้ประมาณ 99% จากกลุ่มชุดข้อมูลตั้งแต่ $n = 5$ ถึง $n = 10000$ ชุดปัญหา โดยในการคำนวณค่าความแม่นยำของการหาคำตอบที่ดีที่สุดของขั้นตอนวิธี คำนวณจากสัดส่วนของจำนวนชุดปัญหาที่หาคำตอบที่ดีที่สุดได้ต่อจำนวนชุดข้อมูลนั้น ๆ เช่น จากชุดข้อมูลจำนวน $5 \leq n \leq 100$ ที่ถูกประมวลด้วยขั้นตอนวิธีพีทีลิสต์ทีเอสอาร์ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่งสามารถหาคำตอบที่ดีที่สุดได้ 95 ชุดข้อมูล คิดเป็น $(95 \times 100 / 95) = 99.99\%$ แต่ใช้ขั้นตอนวิธีทีเอสรีดักชันได้คำตอบที่ดีที่สุด 86 ชุดข้อมูล จึงได้ค่าความแม่นยำโดยประมาณที่ร้อยละ 90.5 (หรือ $86 \times 100 / 95$) นั่นคือ ประสิทธิภาพด้านความถูกต้องแม่นยำของขั้นตอนวิธีที่ได้คำตอบที่ดีที่สุด 99% หมายถึง ในการทดลองเพื่อหาคำตอบจากชุดข้อมูล 100 ชุด ได้คำตอบที่ดีที่สุด 99 ชุด

ตารางที่ 6.8 แสดงประสิทธิภาพของการลดเนื้อที่เก็บข้อมูลซึ่งขั้นตอนวิธี DPT-List_{TSR} สามารถหาคำตอบที่ดีที่สุดได้จำนวน $n \leq 3000$ ภายในเวลาน้อยกว่า 1 นาที ซึ่งกระบวนการลดเนื้อที่เก็บข้อมูลด้วย F-reduction สามารถลดเนื้อที่เก็บข้อมูลลงได้ถึง 69% ถึง 92% และ B-reduction สามารถลดเนื้อที่เก็บข้อมูลได้ 84% ถึง 93%

6.2.2 ผลการทดลองของขั้นตอนวิธีมัลติพีทีลิสต์ทีเอสอาร์ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (Results of the Multi-DPT-List_{TSR} and Robust Unbiased Filtering)

ในการแก้ปัญหาหมัดติเฟิลแนปแซคซึ่งมีหลายถุงเป้ สามารถแก้ปัญหาได้ด้วยขั้นตอนวิธีมัลติพีทีลิสต์ทีเอสอาร์ร่วมกับการเรียงถุงเป้ที่เหมาะสม อาทิ การเรียงที่ดีที่สุด 9 อันดับแรก การเรียงด้วยลาตินสแควร์ และการเรียงถุงเป้ที่เหมาะสมที่สุด ซึ่งสามารถประเมินด้วยผลลัพธ์ที่ได้เปรียบเทียบกับเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กับผลลัพธ์ที่ดีที่สุด (Optimal Solution) แต่ในการปฏิบัติ สิ่งที่สำคัญของขั้นตอนวิธีคือเวลาในการประมวลผลที่ต้องรวดเร็วและได้ผลลัพธ์ที่ถูกต้อง ซึ่งในวิทยานิพนธ์นี้ได้นำเสนอขั้นตอนวิธีมัลติดีฟิสิลิสต์ที่เอสอาร์ที่ใช้ร่วมกับการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่งซึ่งทำให้ได้ผลลัพธ์ที่มีประสิทธิภาพ ด้วยการทดลองกับชุดข้อมูลแบบสุ่มขนาด $n \leq 10000$ และ $m \leq 100$ ด้วยขนาดความจุของหน่วยที่หลากหลาย (เช่น $C_i \pm 10$, $C_i \pm 15$, $C_i \pm 20$) รวมถึงข้อมูลมาตรฐาน [16] ที่ได้ผ่านการเปรียบเทียบกับผลลัพธ์ที่เป็นคำตอบที่ดีที่สุดแล้ว

ในการประเมินประสิทธิภาพ มุ่งเน้นที่ 1. ความถูกต้องของผลลัพธ์ ซึ่งสนใจการเรียงถูกเป้ที่ดีที่สุดที่ทำให้ได้คำตอบที่ดีที่สุด 99% ในเชิงทฤษฎี และ 2. ความเร็ว ซึ่งในได้นำกระบวนการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่งมาใช้ทำให้ได้ความเร็วในการประมวลผลเป็นโพลิโนเมียลทั้งในด้านทฤษฎีและปฏิบัติ ในการทดลองได้ขั้นตอนวิธีและกระบวนการที่นำเสนอ ได้แก่ ขั้นตอนวิธีมัลติดีฟิสิลิสต์ที่เอสอาร์ร่วมกับการเรียงถูกเป้ที่เหมาะสมที่สุด และขั้นตอนวิธีมัลติดีฟิสิลิสต์ที่เอสอาร์ร่วมกับการคัดกรองที่นำเสนอ สำหรับในทางปฏิบัติซึ่งเน้นเวลาโพลิโนเมียลได้ใช้ขั้นตอนวิธีมัลติดีฟิสิลิสต์ที่เอสอาร์ร่วมกับการคัดกรองที่นำเสนอเปรียบเทียบกับ multi-GH⁺ ซึ่งเป็นขั้นตอนวิธีฮิวริสติกที่มีประสิทธิภาพ

การทดลองด้วยการเขียนโปรแกรมได้แบ่งออก 3 กลุ่มวิธี ดังนี้

กลุ่มวิธีทางทฤษฎี

1.1 Multi-DPT-List_{TSR} (exact-fit best order) เวลาที่ $O(m^2[n^2, nC])$

กลุ่มวิธีทั้งทางทฤษฎีและปฏิบัติ

2.1 Fast multi-DPT-List_{TSR} + filtering (exact-fit best order) เวลาที่ $O(m^2n)$

2.2 Fast multi-DPT-List_{TSR} + filtering (top 9 orders+ partial LSs) เวลาที่ $O(mn)$

2.3 Fast multi-DPT-List_{TSR} + filtering (top 9 orders) เวลาที่ $O(mn)$

กลุ่มวิธีทางปฏิบัติ

3.1 Quick multi-GH (P/W rank) (top 9 orders) เวลาที่ $O(mn)$

3.2 Improved multi-GH⁺ (P/W) (top 9 orders) เวลาที่ $O(mn)$

3.3 Improved multi-GH⁺ (P/W) (top 9 + full LS) เวลาที่ $O(m^2n)$

อันดับแรก ผู้วิจัยได้ประเมินประสิทธิภาพของขั้นตอนวิธีมัลติดีฟิสิลิสต์ที่เอสอาร์ (Multi-DPT-List_{TSR}) และขั้นตอนวิธีมัลติดีฟิสิลิสต์ที่เอสอาร์แบบเร็วที่ใช้การคัดกรองเข้ามารวม (Fast multi-DPT-List_{TSR} + Filtering) ด้วยการจัดเรียงถูกเป้จำนวน $m!$ (สำหรับปัญหาที่ $m = 2, 3, 4$) ซึ่งสามารถหาคำตอบที่ดีที่สุดได้เกือบทุกชุดข้อมูล (แสดงในคอลัมน์ที่ 3 – 4 ตารางที่ 6.9 – 6.11) สำหรับชุดข้อมูลปกติขั้นตอนวิธีมัลติดีฟิสิลิสต์ที่เอสอาร์แบบเร็วที่ใช้การคัดกรองเข้ามารวม (Fast multi-DPT-List_{TSR} + Filtering) สามารถหาคำตอบที่ดีที่สุดได้เกือบทุกชุดข้อมูล สำหรับในกรณีที่ $m > 4$ ผู้วิจัยได้ตรวจสอบผลกระทบของการกรองที่มีประสิทธิภาพร่วมกับการเรียงถูกเป้แบบ 9 อันดับที่ดีที่สุดและลาดินสแควร์บางส่วน (คอลัมน์ 4–5 ตารางที่ 8–9) สำหรับชุดข้อมูลปกติ ($n \leq 10000$ และ $m \leq 100$) ขั้นตอนวิธีมัลติดีฟิสิลิสต์ที่เอสอาร์ที่ใช้การคัดกรองเข้ามารวมสามารถได้คำตอบที่ดีที่สุดหรือได้คำตอบที่เหมาะสมที่สุดอย่างน้อย 99%

อันดับสอง เป้าหมายในการวิจัยนี้คือ การเปรียบเทียบวิธีการที่มีเวลาในการประมวลผลเป็นโพลิโนเมียล ($O(m^2n)$) ด้วยการตรวจสอบผลจากการจัดเรียงแบบ 9 อันดับที่ดีที่สุด (คอลัมน์ที่ 4 – 7 เอกสารนี้เป็นเอกสารต้นฉบับสำหรับการแข่งขันในเวทีวิชาการ เมื่อผู้เขียนได้เห็นว่าประโยชน์ในการศึกษาไม่จำกัดเฉพาะกลุ่มนี้ อีกทั้งยังมีให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในตารางที่ 6.12 – 6.13) ในกรณีที่ $m \leq 100$ and $n \leq 10000$ ผลลัพธ์ของขั้นตอนวิธีมัลติดีฟิสิสต์ที่เอสอาร์ที่ใช้การคัดกรองเข้ามาร่วม (mDPT-List + filtering) (คอลัมน์ที่ 5) ได้นำมาเปรียบเทียบกับขั้นตอนวิธีที่เร็วซึ่งในที่นี้ใช้ขั้นตอนวิธีมัลติกรีดีฮิวริสติกพลัส (mGH⁺) ซึ่งเป็นวิธีที่ประมวลผลเร็วที่เรียงสิ่งของแบบ P/W (P/W rank) (คอลัมน์ที่ 6-7) ใช้เวลาในการประมวลผลจริงน้อยกว่า 1 นาที ในกรณีชุดข้อมูลปกติ ขั้นตอนวิธีมัลติดีฟิสิสต์ที่เอสอาร์ร่วมกับการคัดกรอง (mDPT-List + filtering) (คอลัมน์ที่ 5) สามารถหาคำตอบที่ดีที่สุดได้เกือบทุกชุดข้อมูล ในขณะที่ผลลัพธ์ของของวิธี mGH⁺ (คอลัมน์ที่ 6) และวิธีการที่ปรับปรุงด้วยลาติสสแควร์ในการจัดเรียงแบบ 9 อันดับที่ดีที่สุด (คอลัมน์ที่ 7) ไม่ใกล้เคียงกับคำตอบที่ดีที่สุดโดยเฉพาะเมื่อปัญหานั้นมีจำนวนสูงไปจำนวนมาก ($m > 10$)

หมายเหตุ: ขั้นตอนวิธีกรีดีฮิวริสติกเป็นขั้นตอนวิธีที่ถูกนำไปใช้มากในการพัฒนาขั้นตอนวิธีต่างๆ รวมถึงในการแก้ปัญหาเชิงเส้นแบบแซดที่ได้ออกมาและนำเสนอในบทที่ 3 เช่นกัน ก็สามารถหาคำตอบเริ่มต้น (Initial Solutions) ที่ดี ซึ่งได้พัฒนาเป็นขั้นตอนวิธีกรีดีฮิวริสติกพลัส (GH⁺) เพื่อเพิ่มประสิทธิภาพในกระบวนการคัดกรองแบบไม่เอนเอียง ดังนั้น ในการศึกษา ผู้วิจัยจึงได้พัฒนาขั้นตอนวิธีมัลติกรีดีฮิวริสติกพลัส (mGH⁺) กับลาติสสแควร์ของการเรียงสูงไปแบบ 9 อันดับที่ดีที่สุดเพื่อจำลองกระบวนการวิวัฒนาการของการเพิ่มประสิทธิภาพขั้นตอนวิธีเชิงพันธุกรรม (GA) หรือการประมวลผลแบบกลุ่ม (Swarm) ในการเปรียบเทียบนี้

ตารางที่ 6.9 ผลลัพธ์ส่วน Total Profit ของชุดข้อมูลที่มีความจุของสูงไปขนาด $C \pm 10$ ($m = 2$)

$m = 2$	Optimal	mDPT-L $m! O(m[n^2, nC])$	mDPT-L+Filter $m! O(mn)$	mGH $m! O(mn)$	mGH ⁺ $m! O(mn)$
n	orders	$m!=2$	$m!=2$	$m!=2$	$m!=2$
15	315	315	315	308	310
20	420	420	420	393	420
30	800	800	800	790	796
40	1050	1050	1050	1022	1047
50	1019	1019	1019	1006	1013
100	2359	2359	2359	2313	2357
200	3878	3878	3878	3860	3870
300	6202	6202	6202	6171	6196
400	7686	7686	7686	7654	7683
500	9074	9074	9074	9045	9072
1000	18038	18038	18038	18002	18031

ตารางที่ 6.10 ผลลัพธ์ส่วน Total Profit ของชุดข้อมูลที่มีความจุของสูงไปขนาด $C \pm 15$ ($m = 3$)

$m = 3$	Optimal	mDPT-L $m! O(m[n^2, nC])$	mDPT-L+Filter $m! O(mn)$	mGH $m! O(mn)$	mGH ⁺ $m! O(mn)$
n	orders	$m!=6$	$m!=6$	$m!=6$	$m!=6$
15	327	327	327	320	322
20	466	466	466	454	466
30	840	840	840	829	839
40	1103	1103	1103	1090	1099
50	1067	1067	1067	1031	1062
100	2427	2427	2427	2390	2426
200	3949	3949	3949	3908	3943
500	9150	9150	9150	9133	9148
1000	18112	18112	18112	18088	18110
2000	25547	25547	25547	25524	25544

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 6.11 ผลลัพธ์ส่วน Total Profit ของชุดข้อมูลที่มีความจุของถุงเป็ขนาด $C \pm 20$ ($m = 4$)

$m = 4$	Optimal	mDPT-L $m! O(m[n^2, nC])$	mDPT-L+Filter $m! O(mn)$	mGH $m! O(mn)$	mGH+ $m! O(mn)$
n	orders	$m!=24$	$m!=24$	$m!=24$	$m!=24$
20	495	495	495	490	490
30	884	884	884	884	884
40	1200	1200	1200	1187	1192
50	1137	1137	1136	1121	1136
60	1504	1504	1504	1472	1499
100	2548	2548	2548	2534	2546
200	4098	4098	4098	4071	4088
500	9318	9318	9318	9297	9314
1000	18284	18284	18284	18249	18280
2000	25760	25760	25760	25735	25752
3000	38941	38941	38941	38899	38930

ตารางที่ 6.12 ผลลัพธ์ของชุดข้อมูล $C \pm 20$ สำหรับ $n = 1000$ ถึง 5000 ที่ $m = 6$ ถึง 50

m	Optimal	mDPT-L $O(m^2[n^2, nC])$	mDPT-L+LS Filter $O(m^2n)$	mDPT-L + Filter $O(mn)$	mGH+ $O(mn)$	mGH++LS $O(m^2n)$
$n=1000$	orders	Best	9 m	9	9	9 m
6	18541	18541	18541	18541	18535	18535
7	18703	18703	18703	18703	18684	18693
8	18889	18889	18889	18889	18879	18884
9	19079	19079	19079	19079	19068	19071
$n=2000$	orders	Best	9x9	9	9	9m
12	27769	27769	27769	27769	27742	27753
13	28154	28154	28154	28154	28127	28127
14	28547	28547	28547	28547	28498	28517
15	28948	28948	28948	28948	28900	28935
$n=5000$	orders	Best	9x9	9	9	9m
20	54736	54736	54736	54736	54680	54695
30	62496	62496	62496	62496	62410	62450
40	72417	72417	72417	72417	72323	72331
50	84051	84051	84051	84051	83943	83963

ตารางที่ 6.13 ผลลัพธ์ของชุดข้อมูล $C \pm 10, 20$ สำหรับ $n = 9000$ ที่ $m = 40$ ถึง 90

m	Optimal	mDPT-L $O(m^2[n^2, nC])$	mDPT-L+LS Filter $O(m^2n)$	mDPT-L + Filter $O(mn)$	mGH+ $O(mn)$	mGH++LS $O(m^2n)$
$C_i \pm 10$	orders	Best	9x9	9	9	9 m
40	53669	53669	53669	53669	53494	53520
50	54803	54803	54803	54803	54538	54655
60	58614	58614	58614	58614	58425	58450
70	64018	64018	64018	64018	63689	63784
$C_i \pm 20$	orders	Best	9x9	9	9	9m
40	85380	85380	85380	85380	85203	85286
50	100859	100859	100859	100859	100683	100724
60	118729	118729	118729	118729	118525	118616
70	138195	138195	138195	138195	137963	138038
80	158983	158983	158983	158983	158722	158857
90	180054	180054	180054	180054	179873	179981

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 6.14 ผลลัพธ์ส่วน Total Profit ของชุดข้อมูลผิดปกติ ที่ $m = 3$ ถึง 25 และ $n \leq 10000$

$n \leq 10000$	Optimal	mDPT-L $O(m^2[n^2, nC])$	mDPT-L+LS Filter $O(m^2n)$	mDPT-L + Filter $O(mn)$	mGH+ $O(mn)$	mGH+LS $O(m^2n)$
$m : n$	orders	$m!$	$m!$	$m!$	$m!$	$m!$
3 : 51	1318	1318	1317	1317	1315	1315
3 : 73	1727	1727	1725	1725	1725	1725
4 : 33	752	752	747	747	747	747
4 : 49	1264	1264	1263	1263	1254	1254
4 : 50	1137	1137	1136	1136	1136	1136
4 : 65	1565	1565	1563	1563	1563	1563
$m : n$	orders	Best	$9m$	9	9	$9m$
5 : 89	2366	2366	2365	2365	2359	2359
7 : 77	1834	1834	1833	1833	1829	1830
7 : 138	3263	3263	3262	3262	3256	3256
7 : 148	3780	3780	3799	3799	3773	3777

ตารางที่ 6.15 จำนวนชุดข้อมูลที่ไม่ได้คำตอบที่ดีที่สุดสำหรับ $n \leq 10000$ ต่อ m (ที่ $m < 100$)

$n \leq 10000$	mDPT-L: $O(m^2[n^2, nC])$		mDPT-L + Filtering: $O(m^2n)$		
m	Best	$9m$ (LS)	Best	$9m$ (LS)	Top 9
5	0	0	0	1	2
6	0	0	0	1	3
7	0	1	0	3	6
8 - 14	0	0	0	2	1.6 (ave. per m .)
15 - 19	0	0	0	0	1.6 (ave. per m .)
20 - 24	0	0	0	0	2.6 (ave. per m .)
25 - 29	0	0	0	0	4.8 (ave. per m .)
30 - 34	0	0	0	0	5.4 (ave. per m .)
35 - 39	0	0	0	0	13.8 (ave. per m .)
40 - 44	0	0	0	0	31.4 (ave. per m .)
45 - 49	0	0	0	0	34.8 (ave. per m .)
50 - 54	0	0	0	0	69.2 (ave. per m .)

สำหรับผลการเปรียบเทียบประสิทธิภาพ คำตอบที่ดีที่สุด (Optimal Solutions) ของปัญหามัลติเพิลแนปแซค (คอลัมน์ที่ 2 ในตารางที่ 6.9 – 6.14) สามารถคำนวณได้โดยตรงจากผลรวมความจุของถุงเป้ทั้งหมดเป็นถุงเป้เดียวที่มีขนาดใหญ่ ในกรณีที่มีถุงเป้ขนาดใหญ่ ($C = \sum_{j=1}^m c_j$) ด้วยขั้นตอนวิธีดีฟิทีลิสต์ที่เอซอร์ทั้งชุดข้อมูลปกติและไม่ปกติเกือบทั้งหมด

ในการวิเคราะห์ กรณี $m=2$ ที่ $n \leq 10000$ ด้วยวิธีมัลติดีฟิทีลิสต์ที่เอซอร์ (mDPT-List + filtering) ที่ใช้การเรียงถุงเป้จำนวน $m!$ (ตารางที่ 6.9) ได้ผลลัพธ์ที่เป็นคำตอบที่ดีที่สุดทั้งหมดเป็น 100% สำหรับที่ $m=3, 4$ ($n \leq 10000$) สามารถได้ผลลัพธ์ที่เป็นคำตอบที่ดีที่สุด 99% (ตารางที่ 6.10 และ 6.11) จากนั้น ยังพบว่าในกรณีของข้อมูลผิดปกติ การเรียงถุงเป้แบบ 9 อันดับที่ดีที่สุดไม่สามารถเพียงพอในการหาคำตอบที่ดีที่สุดได้ โดยเฉพาะเมื่อ $m \geq 5$ ผู้วิจัยจึงได้ใช้การจัดเรียงแบบลาติสสแควร์แบบ 9 อันดับแรก ซึ่งแสดงในตารางที่ 6.12 – 6.14

นอกจากนี้ ในตารางที่ 6.14 - 6.15 เป็นการรายงานเกี่ยวกับการศึกษาชุดข้อมูลผิดปกติที่พบระหว่างการประมวลผล ($n \leq 10000$) ซึ่งการเรียงถุงเป้แบบ 9 อันดับแรกไม่สามารถหาคำตอบที่ดีเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ที่สุดได้ สำหรับปัญหาที่มี $m \geq 5$ ได้ศึกษาและทดลองแต่ละ $n \leq 10000$ ($m \leq 100$) และพบว่าจำนวนของชุดข้อมูลผิดปกติเพิ่มขึ้นเมื่อ m เพิ่มขึ้น (ดังแสดงในคอลัมน์ที่ 6 ตารางที่ 6.15) ดังนั้นการเรียงถุงที่เหมาะสมพอดีที่นำเสนอสามารถประยุกต์เพื่อแก้ปัญหานี้ได้

หลังจากได้ศึกษาประสิทธิภาพและเปรียบเทียบกับข้อมูลขนาด $n \leq 10000$ พบว่า ขั้นตอนวิธีมัลติดีทีฟิสิสที่เอสอาร์ร่วมกับการคัดกรองแบบเหมาะสมพอดี (mDPT-List + filtering (the exact-fit (best) order: $O(m^2n)$)) สามารถหาคำตอบที่ดีที่สุดสำหรับชุดข้อมูลผิดปกติได้อย่างน้อย 99% เช่นเดียวกับขั้นตอนวิธีมัลติดีทีฟิสิสที่เอสอาร์ ($O(m^2[n^2, nC])$) ดังแสดงจำนวนของผลลัพธ์ที่ไม่ใช่คำตอบที่ดีที่สุด (Non-optimal Solutions) ที่แสดงเป็น 0% ในคอลัมน์ที่ 4 ตารางที่ 6.15

สุดท้าย ได้ทำการทดลองพิเศษเพื่อประเมินประสิทธิภาพของขั้นตอนวิธีมัลติดีทีฟิสิสที่เอสอาร์ด้วยชุดข้อมูลมาตรฐาน [16] ซึ่งเผยแพร่ใน <http://or.dei.unibo.it/library> (ตารางที่ 6.16 - 6.17) ซึ่งแสดงให้เห็นผลลัพธ์ของขั้นตอนวิธีมัลติดีทีฟิสิสที่เอสอาร์ที่นำเสนอร่วมกับการจัดเรียงถุงที่ดีที่สุด ($O(m^2[n^2, nC])$) และขั้นตอนวิธีมัลติดีทีฟิสิสที่เอสอาร์ที่เร็วด้วยการคัดกรองที่มีประสิทธิภาพร่วมกับการจัดเรียงถุงที่ดีที่สุด $O(m^2n)$ เปรียบเทียบกับผลลัพธ์ที่เป็นคำตอบที่ดีที่สุด (Optima Solution)

สำหรับกรณีของ 10 ชุดข้อมูล ($n:m = 100:10$) [16] ได้แสดงผลลัพธ์ในตารางที่ 6.16 ซึ่งแสดงให้เห็นว่าขั้นตอนวิธีมัลติดีทีฟิสิสที่เอสอาร์ทั้งที่ใช้การคัดกรองร่วมด้วยและไม่ได้ใช้การคัดกรองร่วมด้วยและทั้งการเรียงถุงแบบดีที่สุดและแบบ 9 อันดับที่ดีที่สุด สามารถหาคำตอบที่ดีที่สุดได้ (คอลัมน์ที่ 4-7) ในขณะที่ผลของขั้นตอนวิธีที่เน้นความเร็วอย่างมัลติกริดิวิริสติกพลัสที่ใช้การเรียงถุงเป้ 9 รูปแบบไม่สามารถหาคำตอบที่ดีที่สุดได้

สำหรับกรณีของ 20 ชุดข้อมูล ($n:m = 200:20, 300:30, 500:50$) [16] ได้แสดงผลลัพธ์ในตารางที่ 6.17 ซึ่งคำตอบที่ดีที่สุดส่วนใหญ่กับข้อมูลวิฤติเหล่านี้ อาจเกิดการบรรจุเกิน (Over Packing) ในถุงเป้ที่มีขนาดความจุมาก ($C^s = \sum_{i=1}^m C_i$) ดังในรูปที่ 6.3 (ก)) แสดงตัวอย่างการบรรจุเกินขนาดเมื่อสิ่งของบางชิ้นในชุดข้อมูล (เช่น สิ่งของบางชิ้นมีค่าคุณสมบัติ p_j/w_j ใกล้กับสิ่งของที่มีค่า p_j, w_j แต่มีค่า w_j มาก ซึ่งไม่สามารถบรรจุลงถุงเป้ใดที่มีความจุ C_i ได้ ยกเว้นถุงเป้ที่มีความจุรวม C^s ที่ยังมีเนื้อที่เหลือ (ซึ่งเป็นเนื้อที่เหลือสมมติเพราะเป็นของทุกถุงรวมกัน) ทำให้มีขนาดใหญ่พอที่จะบรรจุสิ่งของชิ้นนั้นได้

ตารางที่ 6.16 ผลลัพธ์ของชุดข้อมูลมาตรฐานขนาด $n = 100$ และ $m = 10$

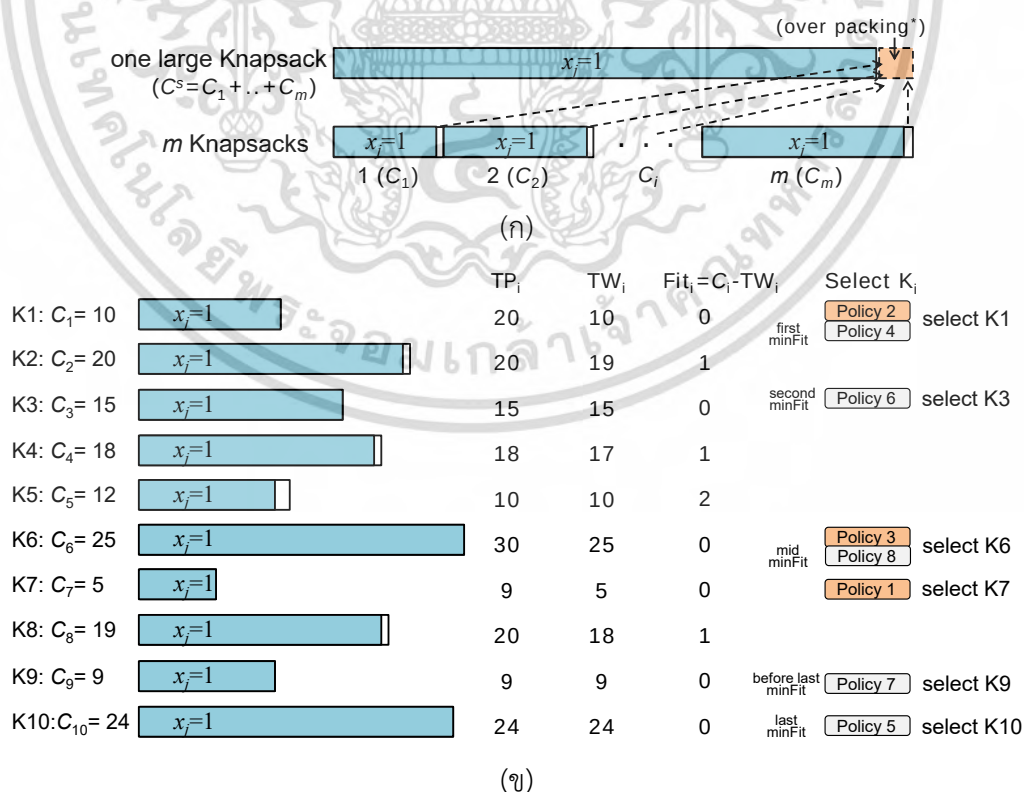
Research approach		Theory		Theory + Practice			Practice
		mDP	mDPT-L	mDPT-L + Filtering : $O(m^2n)$			mGH ⁺
$n : m$	Optimal	Best	Best	Best	9m	9	9m
100:10-1	26797	26797	26797	26797	26797	26797	26763
100:10-2	24116	24116	24116	24116	24116	24116	24093
100:10-3	25828	25828	25828	25828	25828	25828	25812
100:10-4	24004	24004	24004	24004	24004	24004	23977
100:10-5	23958	23958	23958	23958	23958	23958	23933
100:10-6	24650	24650	24650	24650	24650	24650	24614
100:10-7	23911	23911	23911	23911	23911	23911	23886
100:10-8	26612	26612	26612	26612	26612	26612	26579
100:10-9	24588	24588	24588	24588	24588	24588	24565
100:10-10	24617	24617	24617	24617	24617	24617	24591

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้เผยแพร่ไปใช้ประโยชน์ทางการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 6.17 ผลลัพธ์ของชุดข้อมูลมาตรฐานขนาด $200 \leq n \leq 500$ และ $20 \leq m \leq 50$

Research approach		Theory			Theory + Practice		Practice
	Over Packing*		mDPT-L $O(m^2[n^2, nC])$		mDPT-L+Filter $O(m^2n)$		mGH* $O(m^2n)$
$n: m$	Optimal	Best of this study	Best +extra	Best	Best	9 m	9 m
200:20-1	80260*	80205	80205	80163	80196	80121	79606
200:20-2	80171*	80122	80122	80122	80121	80069	79488
200:20-3	79101*	79083	79083	79061	79083	79041	78561
200:20-4	76264*	76208	76208	76174	76174	76149	75823
200:20-5	79619	79619	79619	79581	79581	79515	78886
200:20-6	76749*	76711	76711	76711	76711	76612	76203
200:20-7	76543*	76474	76474	76429	76474	76402	75959
300:30-1	121806*	121756	121742	121742	121756	121654	120842
300:30-2	119877*	119828	119828	119795	119828	119743	118938
300:30-3	119806*	119762	119762	119756	119749	119684	118937
300:30-4	115567*	115556	115529	115516	115556	115434	114767
300:30-5	117204*	117175	117175	117160	117168	117065	116350
300:30-6	118516*	118493	118493	118493	118450	118386	117737
300:30-7	115793*	115752	115752	115706	115693	115641	115093
300:30-8	123664*	123624	123624	123620	123620	123552	122570
500:50-1	205672*	205645	205645	205645	205645	205488	204132
500:50-2	199868*	199781	199775	199775	199781	199681	198462
500:50-3	202321*	202286	202286	202277	202277	202164	201102
500:50-4	136669*	136657	136657	136653	136652	136595	135409
500:50-5	135806*	135796	135795	135795	135796	135736	134793



รูปที่ 6.3 การบรรจุเกิน (Over Packing) ของปัญหาซิงเกิลแนปแซค (0/1-KP) ในชุดข้อมูลวิกฤติ เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในทางปฏิบัติสำหรับปัญหาที่มีข้อมูลมาก ขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ร่วมกับการคัดกรองและการเรียงถุงเป้แบบ 9 อันดับแรกที่ดีที่สุด ก็เพียงพอสำหรับการหาผลลัพธ์สำหรับชุดข้อมูลปกติ (ตารางที่ 6.9-6.13 และ 6.16) แต่สำหรับในชุดข้อมูลผิดปกติ (ตารางที่ 6.15) ขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ร่วมกับการคัดกรองและแบบเหมาะสมพอดี (ซึ่งเป็นการเรียงลำดับถุงเป้ที่ดีที่สุด (Best order)) สามารถหาคำตอบได้ดีที่สุดได้ 99% คล้ายกับผลของการใช้ขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ที่ใช้การเรียงถุงเป้แบบเหมาะสมพอดี ดังนั้น ในทางปฏิบัติ ทั้งชุดข้อมูลปกติและชุดข้อมูลผิดปกติ ขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ร่วมกับการคัดกรองและแบบเหมาะสมพอดี ก็เพียงพอต่อการประมวลผลเพื่อให้ได้ผลลัพธ์ที่ดีที่สุดประมาณ 99% ในเวลาโพลิโนเมียล ($O(m^2n)$) นอกจากนี้ ผู้วิจัยยังได้ทำการศึกษาเกี่ยวกับชุดข้อมูลมาตรฐานซึ่งมีข้อมูลวิกฤติหรือชุดข้อมูลมาตรฐานพิเศษซึ่งได้ทำการทดลองเพื่อปรับปรุงให้ได้ผลลัพธ์ที่ดีขึ้น (คอลัมน์ 4 ตารางที่ 6.17) ซึ่งจะแสดงในหัวข้อถัดไป

6.2.3 ผลการทดลองเพิ่มเติมและการปรับปรุงเพิ่มเติมเกี่ยวกับชุดข้อมูลพิเศษ

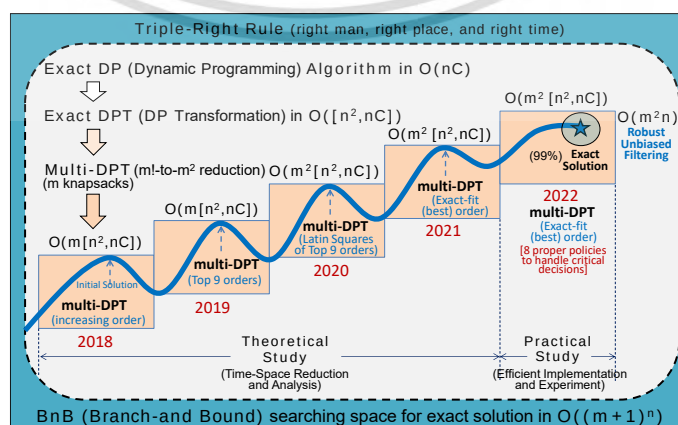
ในการปรับปรุงผลลัพธ์ (กรณีชุดข้อมูลพิเศษ) ผู้วิจัยได้มุ่งเน้นไปที่การตัดสินใจที่สำคัญทั้งหมด ผู้วิจัยจึงต้องการหาเงื่อนไขการตัดสินใจเหมาะสม เช่น 1) คำตอบที่ดีที่สุดในแต่ละถุงเป้มี X_s หลายชุดที่ไม่ซ้ำกัน (ในการเอ็กซ์แทรกกิ้ง ในหัวข้อ 4.2.1) ดังตัวอย่างที่เกิดในรูปที่ 4.7 และ 2) ค่า $\min F_{it_i}$ ที่เท่ากันในหลายถุงเป้ (ในการเรียงลำดับถุงเป้แบบเหมาะสมพอดี (Exact-fit (Best) Order) ดังตัวอย่างในรูปที่ 4.8 ซึ่งต้องใช้กฎที่เหมาะสมในการจัดการสิ่งเหล่านี้ โดยปกติถ้ามีเพียงค่า $\min(F_{it_i})$, $F_{it_i} = C_i - TW_i$ เพียงหนึ่งค่า จะเลือกถุงเป้ K_i ที่ดีที่สุดในการจัดเรียงที่ดีที่สุด แต่มีบ่อยครั้งที่เกิดเหตุการณ์ที่ค่า $\min(F_{it_i})$ เท่ากันในหลายถุงเป้ แต่จะต้องมีเพียงถุงเป้เดียวเท่านั้นที่ถูกเลือก (ในลำดับที่ดีที่สุด ณ เวลานั้น) ซึ่งการตัดสินใจด้วยเงื่อนไขนี้อาจจะทำให้ได้คำตอบในถุงเป้ไม่เป็นคำตอบที่ดีที่สุด เพราะการที่ไม่เลือกถุงเป้อื่นที่มีค่า $\min(F_{it_i})$ เท่ากัน ซึ่งอาจจะได้คำตอบที่ดีกว่า (เหมาะสมที่สุด/ดีที่สุด)

ในการจัดการปัญหาเช่นนี้ ผู้วิจัยจึงใช้กฎ 3 กฎ ในขั้นตอนวิธีที่ 4.4 และนำผลลัพธ์ที่ดีที่สุด(ที่ได้จากแต่ละกฎ) มาเปรียบเทียบเพื่อเลือกคำตอบที่ดีที่สุดเป็นคำตอบสุดท้าย แต่อย่างไรก็ตาม เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด (ที่มีค่าเพิ่มมากขึ้น) สำหรับชุดปัญหาที่มีความพิเศษเหล่านี้ จึงได้เพิ่มกฎอีก 5 ข้อ ในขั้นตอนวิธีที่ 4.4 (Step 2) เพื่อให้ครอบคลุมการตัดสินใจในกรณีที่เจอข้อมูลวิกฤติ ซึ่งแสดงในขั้นตอนวิธีที่ 4.4 (Step 2) เพื่อให้ครอบคลุมการตัดสินใจในกรณีที่เจอข้อมูลวิกฤติ ดังแสดงได้ในรูปที่ รูปที่ 6.3 (ข) เช่น เลือก K_i ด้วยค่า $\min(F_{it_i})$ ที่อันดับแรก อันดับสุดท้าย อันดับที่สอง ก่อนลำดับสุดท้าย และค่า $\min(F_{it_i})$ กลาง ในการทดลอง พบว่ามีผลลัพธ์ที่ได้ขึ้นสำหรับชุดข้อมูลพิเศษดังแสดงค่ามูลค่ารวมในคอลัมน์ที่ 4 ในตารางที่ 6.17 ซึ่งแสดงภาพตัวอย่าง (รูปที่ 6.3 (ข)) ที่มีถุงเป้ $m = 10$, $C_i = (10, 20, 15, 18, 12, 25, 5, 19, 9, 24)$ และได้ใช้กฎทั้ง 8 กฎในการตัดสินใจ (ในรูปจะเห็นว่ามี 6 ถุงเป้ที่มีค่า $\min(F_{it_i}) = 0$ (K_i , $i = 1, 3, 6, 7, 9, 10$)) ซึ่งต้องเลือกถุงเป้ที่เหมาะสมที่สุดด้วยการเรียงลำดับถุงเป้ที่ดีที่สุดร่วมกับกฎแต่ละข้อ ซึ่งจะได้ 8 ผลลัพธ์ที่ดีที่สุดของแต่ละกฎ และเปรียบเทียบเพื่อเลือกคำตอบที่ดีที่สุดเป็นคำตอบสุดท้าย นั้นแสดงว่า การใช้กฎที่ตรงกับปัญหาที่พบเพิ่มมากขึ้น (8 กฎ) ในเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนวิธีที่ 4.4 ที่นำเสนอร่วมกับขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์สามารถช่วยให้เลือกบรรจุสิ่งของลงถุงเป้ที่เหมาะสมได้มากขึ้น โดยเฉพาะในกรณีที่พบข้อมูลไม่ปกติชนิดพิเศษเพื่อให้มูลรวมที่ดีที่สุดนั้นแสดงว่า การใช้กฎในขั้นตอนวิธีที่ 4.4 ที่นำเสนอร่วมกับขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์สามารถช่วยให้เลือกบรรจุสิ่งของลงถุงเป้ที่เหมาะสมได้

ในการเปรียบเทียบผลของชุดข้อมูลพิเศษ 20 ชุดข้อมูลในตารางที่ 6.17 ที่แสดงผลลัพธ์ของขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ร่วมกับการคัดกรอง (mDPT-List + filtering) (คอลัมน์ที่ 6) จะเห็นว่า ได้ผลลัพธ์ที่ดีที่สุดจำนวน 9 จาก 20 รายการ เทียบกับผลลัพธ์ขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ที่ไม่ใช้การคัดกรอง (คอลัมน์ที่ 4) ได้ผลลัพธ์ที่ดีที่สุดจำนวน 4 จาก 20 รายการ เมื่อเทียบกับผลลัพธ์ขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ที่ใช้กฎทั้งแปดที่เพิ่มเติมขึ้นมาทำให้ได้ผลลัพธ์ที่ดีที่สุดจำนวน 16 จาก 20 รายการ ในขณะที่อีก 4 รายการที่เหลือเป็นคำตอบที่ดีที่สุดที่ได้จากขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์เมื่อมีการคัดกรองที่ไม่เอนเอียง เนื่องจากในบางชุดข้อมูล ผลลัพธ์ที่ดีกว่าของ 4 รายการนี้ เป็นผลจากการคัดกรองโดยตรง (ก่อนที่จะเลือกถุงเป้) ซึ่งช่วยลดผลกระทบในการแทรกหาค่า X_s ของขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ที่มีค่า n' น้อย ($n' < 300$) ในแต่ละถุงเป้ หมายเหตุ ในข้อมูลปกติ การแทรกหาค่า X ในแต่ละถุงเป้จะมีค่า X เพียงชุดเดียว (Unique X) ดังนั้นผลลัพธ์ของขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ทั้งที่ใช้การคัดกรองร่วมและไม่ใช้การคัดกรอง จะได้ผลลัพธ์ที่ดีที่สุดชุดเดียวกัน

ในทางปฏิบัติ เวลาที่ใช้ในการประมวลผล (Response Time) ของทั้งสามรูปแบบของขั้นตอนวิธีมัลติดีพีที่ลิสต์ที่เอสอาร์ (basic mDP, mDPT-List, and mDPT-List + filtering) ได้เปรียบเทียบในการทดลองนี้ภายใต้ประสิทธิภาพของผลลัพธ์ที่ 99% สำหรับในทางทฤษฎี ความซับซ้อนด้านเวลาของทั้งสามรูปแบบที่ทดลองได้แก่ 1. $O(m!nC)$ สำหรับ basic mDP ($m!$ orders) 2. $O(m^2[n^2, nC])$ สำหรับ mDPT-List (the best order) และ 3. $O(m^2n)$ สำหรับ the fast mDPT-List + filtering (the best order) เนื่องจากประสิทธิภาพของการคัดกรองจึงทำให้เวลาที่ใช้ในการประมวลผลของ fast mDPT-List + filtering ที่ $n = 20000$ และ $m = 20$ น้อยกว่า 1 วินาที ในส่วนของ the best mDPT-List ที่ $n = 20000$ ใช้เวลาประมาณ 10 นาที และสำหรับ basic mDP ที่ $n = 20000$ ใช้เวลาในการประมวลผลมากกว่าหนึ่งชั่วโมงและนานขึ้นเมื่อ n มากขึ้น



รูปที่ 6.4 การปรับปรุงแนวทาง multi-DPT-List_{TSR} สำหรับปัญหามัลติเพิลแนปแนค (0/1-mKP)
เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อใช้ในการศึกษาเท่านั้น เมื่อผู้ใดได้เห็นเอกสารฉบับนี้แล้ว
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากนี้ เพื่อช่วยในการวิเคราะห์ผลการเปรียบเทียบสำหรับการแก้ปัญหาหมัดติเฟิลแนปแซคที่มีชุดข้อมูลพิเศษที่ได้ออกแบบขั้นตอนวิธีภายใต้กฎความถูกต้องทั้งสาม (Triple-right : Right man, Right place, Right time) จึงได้แสดงแนวทางการพัฒนาในรูปที่ 6.4 ที่นำไปสู่การพัฒนาขั้นตอนวิธีหมัดติเฟิลที่ลิสต์ที่เอสอาร์ที่ทำให้ได้ความถูกต้องของผลลัพธ์ที่ 99% ในเวลาที่เร็ว

ในทางทฤษฎีแล้ว ขั้นตอนวิธีบรานซ์แอนด์บาวนด์ (BnB) สามารถค้นหาคำตอบที่เหมาะสมที่สุดหรือดีที่สุดสำหรับปัญหาหมัดติเฟิลแนปแซคแต่ใช้เวลาเป็นเอ็กซ์โพเนนเชียลที่ $O((m+1)n)$ ซึ่งหากเทียบกับกฎความถูกต้องทั้งสามจะได้เพียง Right-man และ Right-place ในทางปฏิบัติ ขั้นตอนหมัดติเฟิลที่วิธีวิริสติกพลัส (mGH+) ที่ใช้รูปแบบ 9 อันดับแรกของการเรียงถุงเป้ด้วยลาดินสแควร์สามารถค้นหาคำตอบที่ดีได้ในเวลาโพลิโนเมียลแต่ผลลัพธ์เหล่านั้นอาจจะไม่ได้เป็นคำตอบที่ดีที่สุด ซึ่งเป็นไปตามกฎ Right-time เท่านั้น

ในการศึกษาที่แสดงแนวทางในรูปที่ 6.4 ผู้วิจัยได้พัฒนาขั้นตอนวิธีที่ลิสต์ที่เอสอาร์ (ในเชิงทฤษฎี) และพัฒนาการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (เพื่อทางปฏิบัติ) ซึ่งทำให้ได้ผลลัพธ์ที่ตามกฎความถูกต้องทั้งสามอย่างที่ตั้งใจ คือ ได้สิ่งของที่ควรถูกเลือก (Right-man) ใส่ในถุงเป้ที่เหมาะสม (Right-place) ด้วยเวลาที่เป็นไปได้ (Right-time) ผลงานที่ได้คือการลดการเรียงถุงเป้จาก $m!$ เป็น m^2 และลดขนาดของเนื้อที่และลดเวลาที่ใช้ในการประมวลผลลงได้มาก

ตารางที่ 6.18 การเปรียบเทียบขั้นตอนวิธี mDPT-List_{TSR} ที่นำเสนอและขั้นตอนวิธี HyMKP [16]

For regular and irregular datasets ($n \leq 10000$, $m \leq 100$)		
Theory + Practice	Our multi-DPT-List + filtering (with the exact-fit best order) could find most optimal solutions ($\geq 99\%$) in efficient response time (< 1 sec. per n); see confirmed results in Tables 5–12.	$O(m^2n)$
Theory	Mathematical HyMKP [10] can work (in τ secs.) with Algorithm 6 (reflect multi-graph MKP with decreasing n weights (w_i)) like the basic DP for each of m knapsacks. Then, that initial solution can be improved by the knapsack-based decomposition in v iterations to find the optimal solution (if any) in τ secs. However, no available results for $n > 500$ in that study.	$O(mnC)$
Practice	Multi-GH⁺ (with Latin squares of top nine orders) could find the feasible solutions in efficient time (< 1 sec.) but they are not optimal (see the last column in Tables 5–10). Note that LSs of top 9 orders could emulate 9 m iterations (evolutions) in the GA/swarm optimization with using the good GH ⁺ results (near optimal in each knapsack) in their initial solutions.	$O(m^2n)$
For critical and special benchmark datasets [10] ($n \leq 500$)		
Theory	Partial BnB (in HyMKP) [10]: The existing BnB (MULKNAP program) could find most optimal solutions ($\geq 99.9\%$) in τ secs for $n \leq 500$.	$O((m+1)^n)$
Theory + Practice	Our multi-DPT-List + filtering (with the exact-fit best order): For critical datasets in 0/1-mKP applications, we can adopt the MULKNAP program [10] for $n \leq 500$ in our filtering approach to achieve 99.9% exact solutions and for $n > 500$ we can apply our efficient multi-DPT-List + filtering.	$O(m^2n)$

เอกสารนี้เป็นเอกสารสงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 6.18 เป็นการนำเสนอการเปรียบเทียบขั้นตอนวิธีมัลติดีพีทีลิสต์ร่วมกับการคัดกรอง (รูปที่ 6.4) กับขั้นตอนวิธีไฮเอ็มเคพี [16] และมัลติกรีดอีวีรีสติกพลัส (mGH⁺) ในกรณีที่ n มีปริมาณมาก ($n \leq 10000$ ในการทดลอง) ขั้นตอนวิธีที่ผู้วิจัยได้เสนอสามารถหาคำตอบที่ดีที่สุดได้ 99% ในเวลา $O(m^2n)$ (ดังแสดงในตารางที่ 6.9-6.16) ในขณะที่ผลของมัลติกรีดอีวีรีสติกพลัสไม่สามารถหาคำตอบที่ดีที่สุดได้ ส่วนในการทดลองของขั้นตอนวิธีไฮเอ็มเคพีได้นำเสนอคำตอบ $n \leq 500$ เท่านั้น เมื่อ $n > 500$ ขั้นตอนวิธีไฮเอ็มเคพีอาจจะไม่สามารถหาคำตอบที่เหมาะสมภายในเวลา τ secs เนื่องจากขั้นตอนวิธีไฮเอ็มเคพีได้มีการใช้ของขั้นตอนวิธีบรานซ์แอนด์บาวนด์บางส่วนที่ชื่อว่า MULKNAP มาผสมในขั้นตอนวิธี ซึ่งขั้นตอนวิธีบรานซ์แอนด์บาวนด์ใช้เวลาเพิ่มขึ้นเมื่อ n เพิ่มขึ้น นอกจากนี้ในการ reflect multi-graph MKP ในแต่ละถุงเป้ที่เรียงสิ่งของตามน้ำหนักจากน้อยไปมาก ด้วยหลักการวนรอบซ้ำและเงื่อนไขคล้ายกับขั้นตอนวิธีโปรแกรมเชิงพลวัตแบบเบื้องต้น (ซึ่งอาจจะได้ผลลัพธ์ที่เทียบเท่ากับขั้นตอนวิธีมัลติดีพีทีลิสต์ที่เอสอาร์ที่ใช้การเรียงถุงเป้แบบเก้าอันดับสูงสุดสำหรับชุดข้อมูลปกติ) สำหรับชุดข้อมูลที่ผิดปกติ กระบวนการ v-round ในขั้นตอนวิธีไฮเอ็มเคพี (ขั้นตอนวิธีที่ 2.7) สามารถช่วยเพิ่มประสิทธิภาพของผลลัพธ์ตั้งต้น แต่อย่างไรก็ตาม กระบวนการในขั้นตอนวิธีที่ 2.8 อาจจะใช้เวลาจนเพื่อให้ได้คำตอบที่ดีที่สุดถึง 99% เนื่องจากมีความซับซ้อนที่ $O(mnC)$ ในขณะที่ขั้นตอนวิธีมัลติดีพีทีลิสต์ร่วมกับการคัดกรองที่เหมาะสมที่สุดสามารถหาคำตอบที่ดีที่สุดได้ 99% เหมือนกับการใช้ $m!$ รอบ ด้วยเวลาเพียง $O(m^2n)$ ส่วนในการประมวลผลชุดข้อมูลพิเศษสำหรับข้อมูลขนาด $n \leq 500$ ขั้นตอนวิธีไฮเอ็มเคพีสามารถหาคำตอบที่ดีที่สุดได้ถึง 99.9% ด้วยบรานซ์แอนด์บาวนด์บางส่วน (partial BnB) (ในโปรแกรม MULKNAP) ในเวลา τ secs ดังนั้นในการแก้ปัญหา มัลติเฟิลแนปแซค สำหรับชุดข้อมูลวิกฤติหรือชุดข้อมูลพิเศษ ถ้า $n \leq 500$ เราสามารถเพิ่มการประยุกต์ใช้โปรแกรม MULKNAP ในขั้นตอนวิธีที่นำเสนอในงานวิจัย เพื่อให้ได้คำตอบที่ดีที่สุดที่ 99%

นอกจากนี้ ผู้วิจัยได้เสนอแนะแนวทางการจัดการในกรณีที่ต้องประมวลผลเพื่อหาคำตอบของชุดข้อมูลวิกฤติ (ตารางที่ 6.17) ด้วยการประมวลผลแบบขนานที่นำไปสู่ผลลัพธ์ที่ดีที่สุด (Global Best Result) ที่เร็วและแม่นยำขึ้นที่ได้จากสองแนวทาง โดยแนวทางแรกเป็นผลลัพธ์ของขั้นตอนวิธีมัลติดีพีทีลิสต์ที่เอสอาร์ในแบบขนาน ($O(m[n^2, nC])$) (คอลัมน์ที่ 4) และแนวทางที่สองเป็นผลลัพธ์ของขั้นตอนวิธีมัลติดีพีทีลิสต์ที่เอสอาร์ในแบบขนานร่วมกับการคัดกรอง ($O(mn)$) (คอลัมน์ที่ 6)

ในทางปฏิบัติ ที่ใช้ประยุกต์ปัญหา มัลติเฟิลแนปแซค (กรณี n มาก) ไปใช้นั้น หากเป็นงานที่ให้ ความสำคัญกับเวลาในการประมวลผล (ทั้งข้อมูลปกติและข้อมูลไม่ปกติ) ขั้นตอนวิธีมัลติดีพีทีลิสต์ที่เอสอาร์ ร่วมกับการคัดกรองแบบไม่เอนเอียงสามารถหาคำตอบที่ดีที่สุดเพียงพอต่อการนำไปใช้ (99% Optimal Solutions) ได้ในเวลา $O(m^2n)$ หรือ $O(mn)$ ในการประมวลผลแบบขนานที่ $p = m$ สำหรับกรณี 1% ของผลลัพธ์ที่ไม่สามารถหาคำตอบที่ดีที่สุด เนื่องมาจากการเกิด Nonunique Solution ของ X_s ซึ่งปกติผลลัพธ์ส่วนใหญ่จะเป็น Unique Solution X แต่สามารถเกิดขึ้นได้ในบางชุดข้อมูลที่ไม่ปกติทำให้มีผลต่อผลลัพธ์โดยรวมของปัญหา เช่น รูปที่ 4.7(ก) ผลลัพธ์ในถุงเป้ K_1, K_2 เป็น unique X แต่ K_3

ไม่ Unique ดังนั้นทำให้เกิดผลกระทบต่อทางเลือกสิ่งของในถุงเป้ K_4 ด้วยทำให้ไม่ได้คำตอบที่ดีที่สุด เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7

สรุปผลการวิจัยและข้อเสนอแนะ

7.1 สรุปผลการวิจัย

งานวิจัยฉบับนี้ได้นำเสนอขั้นตอนสำหรับการแก้ปัญหาซึ่งเกิดแบบแซค (0/1-KP) และปัญหามัลติเพิลแซคที่มีความซับซ้อนมากขึ้น โดยมีเป้าหมายทั้งในด้านของความต้องการของผลลัพธ์ที่ได้คำตอบที่ดีที่สุด (Optimal Solution) อย่างน้อย 99% ในเวลาโพลิโนเมียล ภายใต้แนวคิดของหลักความถูกต้องสามประการ (Triple-right: Right-man, Right-place, Right-time) ซึ่งได้แบ่งการคิดค้นออกแบบพัฒนาขั้นตอนวิธีออกเป็น 2 ช่วง คือ ช่วงแรกเป็นการคิดค้นพัฒนาขั้นตอนวิธีสำหรับแก้ปัญหาซึ่งเกิดแบบแซค และช่วงที่สองเป็นการพัฒนาต่อยอดสำหรับแก้ปัญหามัลติเพิลแซค

ช่วงที่หนึ่ง (บทที่ 3) เป็นการคิดค้นออกแบบขั้นตอนวิธีสำหรับแก้ปัญหาซึ่งเกิดแบบแซคที่มีถุงเป้เดียวในชุดปัญหา ซึ่งได้ขั้นตอนวิธีในชื่อ ขั้นตอนวิธีทีเอสรัดักชัน ($TS_{Reduction}$ Algorithm) ที่ประกอบด้วย 3 ขั้นตอนใหญ่ คือ 1) ขั้นตอนวิธีอีวีริสติกแบบเรียงลำดับคุณลักษณะอย่างมีประสิทธิภาพ (GH with Efficient Feature Sorting) โดยใช้การเรียงลำดับหลายระดับเพื่อหาคำตอบเริ่มต้น (Initial Solution) ซึ่งสอดคล้องกับคุณสมบัติ Right Man 2) ขั้นตอนการแบ่งกลุ่มและการคัดกรองแบบไม่เอนเอียง (Object Classification and Unbiased Filtering) เป็นการแบ่งสมาชิกของปัญหาออกเป็นกลุ่มอย่างเหมาะสมและคัดกรองสมาชิกในแต่ละกลุ่มเพื่อเป็นสมาชิกที่เป็นคำตอบ (Select) สมาชิกที่ควรตัดทิ้ง (Unselect) และสมาชิกที่ไม่สามารถตัดสินใจได้ (Uncertain) ซึ่งสอดคล้องกับคุณสมบัติ Right Man, Right Place 3) ขั้นตอนวิธีเอ็มดีพี (MDP) เป็นการปรับปรุงภายใต้การโปรแกรมเชิงพลวัตเป็นขั้นตอนการค้นหาสมาชิกชุดสุดท้ายของคำตอบสำหรับสมาชิกที่คัดกรองไม่ได้ (n') และขนาดความจุที่เหลือของถุงเป้ (C') ซึ่งทำให้ใช้เวลาประมวลผลเร็วขึ้น สอดคล้องกับคุณสมบัติ Right Man, Right Place, Right Time

ช่วงที่สอง (บทที่ 4) เป็นการคิดค้นและพัฒนาต่อยอดจากขั้นตอนวิธีทีเอสรัดักชันสำหรับแก้ปัญหามัลติเพิลแซค ซึ่งได้ขั้นตอนวิธีในชื่อดีพีทีลิสต์เอสอาร์ (สำหรับปัญหาซึ่งเกิดแบบแซค) และมัลติเพิลแซค (สำหรับปัญหามัลติเพิลแซค) โดยเริ่มจากการออกแบบการแปลงรูปแบบของการโปรแกรมเชิงพลวัตให้อยู่ในรูปของลิสต์ที่เก็บเพียงข้อมูลสำคัญในการประมวลผลที่จะนำไปสู่ผลลัพธ์เท่านั้นเพื่อลดขนาดเนื้อที่และลดเวลาที่ใช้ในการประมวลผลและเป็นการนำเสนอแนวคิดในรูปแบบใหม่ ซึ่งในวิทยานิพนธ์นี้ เรียกว่า DP Transformation เพื่อแก้ปัญหาซึ่งเกิดแบบแซคเพราะมีเพียงหนึ่งถุงเป้ในชุดปัญหา โดยให้ชื่อขั้นตอนวิธีว่า ขั้นตอนวิธีดีพีทีลิสต์ทีเอสอาร์ (DPT-List_{TSR} Algorithm) ต่อมาเพื่อการแก้ปัญหามัลติเพิลแซคที่มีหลายถุงเป้ จึงได้นำเสนอการเรียงถุงเป้ที่เหมาะสมหรือเรียกว่าการเรียงถุงเป้ที่ดีที่สุด (The Exact-fit (best) Knapsack-order) ซึ่งสามารถลดเวลาในการเรียงถุงเป้จาก $m!$ เป็น m^2 รูปแบบ ในขั้นตอนวิธีมัลติดีพีทีลิสต์ทีเอสอาร์ (multi-DPT-List_{TSR})

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซึ่งทำให้ได้คำตอบที่ดีที่สุดที่ 99% ในเวลา $O(m^2[n^2, nC])$ ซึ่งเร็วกว่าเดิม $O(m![n^2, nC])$ จากนั้น เพื่อลดเวลาในการประมวลผล จึงได้ปรับปรุงกระบวนการคัดกรองที่นำเสนอในขั้นตอนวิธีทีเอสรีดักชันให้มีความยืดหยุ่นและแม่นยำมากขึ้นด้วยการขยายพื้นที่วิกฤติในทุกกลุ่ม โดยกระบวนการคัดกรองที่พัฒนานี้ชื่อ กระบวนการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (Unbiased Robust Filtering) ซึ่งเมื่อนำมาใช้ร่วมกับขั้นตอนวิธีมัลติดีฟิสิกส์ทีเอสอาร์แล้วสามารถลดเวลาการประมวลผลได้เป็น $O(m^2n)$ ซึ่งเร็วกว่าเดิม $O(mnC)$ โดยได้คำตอบที่ดีที่สุด (Optimal Solution) ที่ 99%

7.2 ข้อเสนอแนะและแนวทางการศึกษา

ข้อเสนอแนะในการพัฒนาต่อยอดขั้นตอนวิธีที่ผู้วิจัยได้นำเสนอในวิทยานิพนธ์นี้ ซึ่งได้แก่ กระบวนการคัดกรองแบบไม่เอนเอียง (Unbiased Filtering) กระบวนการคัดกรองแบบไม่เอนเอียงที่แข็งแกร่ง (Robust Unbiased Filtering) ขั้นตอนวิธีทีเอสรีดักชัน ($TS_{Reduction}$) และดีฟิสิกส์ทีเอสอาร์ ($DPT-List_{TSR}$) รวมถึงการเรียงถุงเป้แบบเหมาะสมพอดี (The Exact-fit (Best) Knapsack Order) ในปัญหาอื่นๆ จะต้องพิจารณาปัจจัยของปัญหาและองค์ประกอบของผลลัพธ์ในปัญหานั้นๆ ตัวอย่างเช่น ปัญหาการเดินทางของพนักงานขาย (Traveling Salesman Problem : TSP) เป็นปัญหาที่ต้องการหาระยะทางที่สั้นที่สุดหรือเหมาะสมที่สุด โดยเริ่มต้นเดินทางจากจุดเริ่มต้นผ่านทุกจุดที่กำหนดแล้วกลับมายังจุดเริ่มต้น ดังนั้นปัจจัยของปัญหาจึงประกอบด้วยระยะทางและจำนวนต่างๆ ที่จะต้องแวะ ซึ่งปัญหาที่มีความซับซ้อนมากขึ้นคือปัญหาการจัดเส้นทางขนส่ง (Vehicle Routing Problem : VRP) ซึ่งมีปัจจัยเพิ่มขึ้น คือ คลังพัสดุ (Depot) และพาหนะที่มีมากกว่า 1 พาหนะ (Fleet) ดังนั้น ในการประยุกต์ขั้นตอนวิธีที่นำเสนอจึงอาจจะต้องมีการปรับรูปแบบหรือนำขั้นตอนวิธีการอื่นมารวมเพื่อให้ได้ผลลัพธ์ตรงตามเป้าหมายของปัญหานั้นๆ

บรรณานุกรม

- [1] จีรพร วีระพันธุ์. 2555. **ขั้นตอนวิธีทางคอมพิวเตอร์ (Computer Algorithms)**. กรุงเทพฯ : ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.
- [2] จีรพร วีระพันธุ์. 2555. **ขั้นตอนวิธีแบบขนาน (Parallel Algorithms)**. กรุงเทพฯ : ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.
- [3] Ahmad, S.J. Reddy, V.S.K. Damodaram, A. and Krishna, P.R. 2015. "Delay Optimization Using Knapsack Algorithm for Multimedia Traffic Over MANET." *Expert Syst. Appl.* 42(20) : 6819-6827.
- [4] Aigner, M. and Ziegler, G.M. 2013. **Proofs from the Book**. Berlin: Springer-Verlag.
- [5] Aisopos, F. Tserpes, K. and Varvarigou, T. 2013. "Resource Management in Software as a Service Using the Knapsack Problem Model." *Int. J. Prod. Econ.* 141(2) : 465-477.
- [6] Anbiyaa, D. Setyawan, E.B. and Purba, H.H. 2019. "Multiple Knapsack Problem for Racking Selection Model." pp. 1-7. In **Conference Series: Materials Science and Engineering**. 508. Jakarta : IOP.
- [7] Bansal, J.C. and Deep, K. 2012. "A Modified Binary Particle Swarm Optimization for Knapsack Problems." *Appl. Math. Comput.* 218(22) : 11042-11061.
- [8] Bas, E. 2011. "A Capital Budgeting Problem for Preventing Workplace Mobbing by Using Analysis Hierarchy Process and Fuzzy 0-1 Bidimensional Knapsack Model." *Expert Syst. Appl.* 38(10) : 12415-12422.
- [9] Basak, A. 2021. "Latin and Euler Square." *IISER Berhampur : Episteme (Science Magazine)*. 2(2) : 15-17.
- [10] Bhattacharjee, K.K. and Sarmah, S.P. 2014. "Shuffled Frog Leaping Algorithm and Its Application to 0/1 Knapsack Problem." *Appl. Soft Comput.* 19 : 252-263.
- [11] Buayen, P. and Werapun, J. 2018. "Parallel time-space Reduction by Unbiased Filtering for Solving the 0/1-Knapsack Problem." *J. Parallel Distrib. Comput.* 122 : 195-208.
- [12] Calvin, J.M. and Leung, J.Y.-T. 2003. "Average-case Analysis of a Greedy Algorithm for the 0/1 Knapsack Problem." *Oper. Res. Lett.* 31(3) : 202-210.

- [13] Camargo, V. C.B. Mattioli, L. and Toledo, F. M.B. 2012. "A Knapsack Problem as a Tool to Solve the Production Planning Problem in Small Foundries." *Comput. Oper. Res.* 39(1) : 86-92.
- [14] Changdar, C., Mahapatra, G.S. and Pal, R.K. 2013. "An Ant Colony Optimization Approach for Binary Knapsack Problem under Fuzziness." *Appl. Math. Comput.* 223(15) : 243-253.
- [15] Cunha, A.S. Bahiense, L. Lucena, A. and Souza, C.C. 2010. "A New Lagrangian Based Branch and Bound Algorithm for the 0-1 Knapsack Problem." *Electron. Notes Discrete Math.* 36 : 623-630.
- [16] Dell'Amico, M. Delorme M. Iori, M. and Martello, S. 2019. "Mathematical Models and Decomposition Methods for the Multiple Knapsack Problem." *Eur. J. Oper. Res.* 274(3) : 886-899.
- [17] Eusuff, M. Lansey, K. and Pasha, F. 2007. "Shuffled Frog-Leaping Algorithm: A Memetic Meta-Heuristic for Discrete Optimization." *Eng. Optim.* 38(2) : 129-154.
- [18] Free Software Foundation. 2014. **Data for the 01 Knapsack Problem.** [Online]. Available : http://people.sc.fsu.edu/~jburkardt/datasets/knapsack_01/knapsack_01.html
- [19] Fukunaga, A.S. 2011. "A Branch-and-Bound Algorithm for Hard Multiple Knapsack Problems." *Ann. Oper. Res.* 184 : 97-119.
- [20] Fukunaga, A.S. and Korf, R.E. 2007. "Bin Completion Algorithms for Multi-Container Packing, Knapsack, and Covering problems." *J. Artif. Intell. Res.* 28 : 393-429.
- [21] Gao, C. Lu, G. Yao, X. and Li, J. 2016. "An Iterative Pseudo-gap Enumeration Approach for the Multidimensional Multiple-choice Knapsack Problem." *Eur. J. Oper. Res.* 260(1) : 1-11.
- [22] Gao, L. 2005. **Latin Square in Experimental Design.** United States : Michigan State University.
- [23] Garey, M.R. and Johnson, D.S. 1979. **Computers and Intractability: A Guide to the Theory of NP-Completeness.** San Francisco : W.H. Freeman and Company.
- [24] Gupta, M. 2013. "A Fast and Efficient Genetic Algorithm to Solve 0-1 Knapsack Problem." *Int. j. digit. appl. contemp. res.* (1)6.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- [25] Haddar, B. Khemakhem, M. Hanafi, S. and Wilbaut, C. 2016. “A Hybrid Quantum Particle Swarm Optimization for the Multidimensional Knapsack Problem.” *Eng. Appl. Artif. Intell.* 55 : 1-13.
- [26] Henkel, C.V. Bäck, T. Kok, J.N. Rozenberg, G. and Spaink H.P. 2007. “DNA Computing of Solution to Knapsack Problems.” *Biosyst.* 88(1-2) : 156-162.
- [27] Hifi, M. Library of instances. 2005. **CERMSEM**. [Online]. Available : <ftp://cermse.univ-paris1.fr/pub/CERMSEM/hifi/KSP>.
- [28] Jiménez-González, D. Navarro, J.J. and Larriba-Pey, J.-L. 2002. “The Effect of Local Sort on Parallel Sorting Algorithms.” 360-367. In **IEEE Proceedings of Euromicro Workshop on Parallel, Distributed and Network-based Processing**. Spain : IEEE.
- [29] Jolai, F. Rezaee, M.J. Rabbani, M. Razmi, J. and Fattahi, P. 2007. “Exact Algorithm for Bi-Objective 0-1 Knapsack Problem.” *Appl. Math. Comput.* 194(2) : 544-551.
- [30] Kellerer, H., Pferschy, U. and Pisinger, D. 2004. **Knapsack Problem**. New York : Springer-Verlag.
- [31] Kong, X. Gao, L. Ouyang, H. and Li, S. 2015. “A Simplified Binary Harmony Search Algorithm for Large Scale 0-1 Knapsack Problem.” *Expert Syst. Appl.* 42(12) : 5337-5355.
- [32] Kuchta, D. Rynca, R. Skorupka, D. and Duchaczek, A. 2019. “The Use of the Multiple Knapsack Problem in Strategic Management of a Private Polish University: Case Study.” *Int. J. Educ. Manag.* 33(2) : 335-358
- [33] Li, X. and Liu, T. 2010. “On Exponential Time Lower Bound of Knapsack under Backtracking.” *Theor. Comput. Sci.* 411(16-18) : 1883-1888.
- [34] Lim, T.Y. Al-Betar, M.A. and Khader, A.T. 2016. “Taming the 0/1 Knapsack Problem with Monogamous Pairs Genetic Algorithm.” *Expert Syst. Appl.* 54 : 241-250.
- [35] Lv, J. Wang, X. Huang, M. Cheng, H. and Li, F. 2016. “Solving 0-1 Knapsack Problem by Greedy Degree and Expectation Efficiency.” *Appl. Soft Comput.* 41 : 94-103.
- [36] Martello, S. and Monaci, M. 2020. “Algorithmic Approaches to the Multiple Knapsack Assignment Problem.” *Omega* 90 : 102004.

- [37] Martello, S. and Toth, P. 1990. **Knapsack Problems: Algorithms and Computer Implementation**. Chichester : J. Wiley&Sons.
- [38] Maus, J.S. 2019. “Applying the Multiple Multidimensional Knapsack Assignment Problem to a Cargo Allocation and Transportation Problem with Stochastic Demand.” United States : Air Force Institute of Technology.
- [39] Meng, T. and Pan, Q.-K. 2017. “An Improved Fruit Fly Optimization Algorithm for Solving the Multidimensional Knapsack Problem.” *Appl. Soft Comput.* 50 : 79-93.
- [40] Merkle, R.C. and Hellman, M.E. 1978. “Hiding Information and Signatures in Trapdoor Knapsacks.” *IEEE Trans. Inform. Theory.* 24(5) : 525-530.
- [41] Monaci, M., Pferschy, U. and Serafini, P. 2013. “Exact Solution of the Robust Knapsack Problem.” *Comput. Oper. Res.* 40(11) : 2625-2631.
- [42] Montgomery, D.C. 2012. **Design and Analysis of Experiments (8th Edition)**. Courier Westford : J. Wiley&Sons.
- [43] Moosavian, N. 2015. “Soccer League Competition Algorithm for Solving Knapsack Problems.” *Swarm Evol. Comput.* 20 : 14-22.
- [44] Pavithr, R.S. and Gursaran 2016. “Quantum Inspired Social Evolution (QSE) Algorithm for 0-1 Knapsack Problem.” *Swarm Evol. Comput.* 29 : 33-46.
- [45] Petagon, R. and Werapun, J. 2016. “Embedding the Optimal All-to-All Personalized Exchange on Multistage Interconnection Networks⁺.” *J. Parallel Distrib. Comput.* 88 : 15-30.
- [46] Qui, X. and Kern, W. 2015. “Improved Approximation Algorithm for a Bilevel Knapsack Problem.” *Theor. Comput. Sci.* 595 : 120-129.
- [47] Rong, A. and Figueira, J.R. 2014. “Dynamic Programming Algorithm for The Bi-Objective Integer Knapsack Problem.” *Eur. J. Oper. Res.* 236(1) : 85-99.
- [48] Rong, A. Figueira, J.R. and Klamroth, K. 2012. “Dynamic Programming Based Algorithms for the Discounted {0-1} Knapsack Problem.” *Appl. Math. Comput.* 218(12) : 6921-6933.
- [49] Rong, A. Figueira, J.R. and Pato, M.V. 2011. “A Two-State Reduction Based Dynamic Programming Algorithm for the Bi-Objective 0-1 Knapsack Problem.” *Comput. Math. Appl.* 62(8) : 2913-2930.

- [50] Rooderkerk, R.P. and Heerde, H.J. 2016. “Robust Optimization of the 0-1 Knapsack Problem: Balancing Risk and Return in Assortment Optimization.” *Eur. J. Oper. Res.* 250(3) : 842-854.
- [51] Sachdeva, C. and Goel, S. 2014. “An Improved Approach for Solving 0-1 Knapsack Problem in Polynomial Time Using Genetic Algorithms.” in **IEEE International Conference on Recent Advances and Innovations in Engineering**. India : IEEE.
- [52] Simon, J. Apte, A. and Regnier, E. 2017. “An Application of the Multiple Knapsack Problem : The Self-sufficient Marine.” *Eur. J. Oper. Res.* 256(3) : 868-876.
- [53] Tang, J. Zhang, R. Wang, P. Zhao, Z. Fan, L. and Liu, X. 2020. “A Discrete Shuffled Frog-Leaping Algorithm to Identify Influential Nodes for Influence Maximization in Social Networks.” *Knowl.-Based Syst.* 187 : 104833.
- [54] Thanakulwarapas, T. and Werapun, J. 2015. “An Optimized Bitonic Sorting Strategy with Midpoint-Based Dynamic Communication.” *J. Parallel Distrib. Comput.* 84 : 37-50.
- [55] Truong, T.K. Li, K. and Xu, Y. 2013. “Chemical Reaction Optimization with Greedy Strategy for the 0-1 Knapsack Problem.” *Appl. Soft Comput.* 13(4) : 1774-1780.
- [56] Vanderster, D.C. Dimopoulos, N.J. Parra-Hernandez, R. and Sobie, R.J. 2009. “Resource Allocation on Computations Grids Using a Utility Model and the Knapsack Problem.” *Future Gener. Comput. Syst.* 25(1) : 35-50.
- [57] Wang, E. Yang, Y. and Wu, J. 2015. “A Knapsack-based Buffer Management Strategy for Delay-tolerant Networks.” *J. Parallel Distrib. Comput.* 86 : 1-15.
- [58] Wang, L. Yang, R. Ni, H. Ye, W. Fei, M. and Pardalos, P.M. 2015. “A Human Learning Optimization Algorithm and Its Application to Multi-Dimensional Knapsack Problems.” *Appl. Soft Comput.* 34 : 736-743.
- [59] Wang, Y. and Wang, W. 2021. “Quantum-inspired differential evolution with gray wolf optimizer for 0-1 knapsack problem.” *Mathematics* 9(11) : 1233.

- [60] Wedashwara, W. Mabu, S. Obayashi, M. and Kuremoto, T. 2016. "Combination of Genetic Network Programming and Knapsack Problem to Support Record Clustering on Distributed Databases." *Expert Syst. Appl.* 46 : 15-23.
- [61] Zhang, J. 2011. "Comparative Study of Several Intelligent Algorithms for Knapsack Problem." *Procedia Environ. Sci.* 11(Part A) : 163-168.
- [62] Zhang, X. Huang, S. Hu, Y. Zhang, Y. Mahadevan, S. and Deng, Y. 2013. "Solving 0-1 Knapsack Problem Based on Amoeboid Organism Algorithm." *Appl. Math. Comput.* 219(19) : 9959-9970.
- [63] Zhang, X. Wu, C. Li, J. Wang, X. Yang, Z. Lee, J.-M. and Jung, K.-H. 2016. "Binary Artificial Algae Algorithm for Multidimensional Knapsack Problems." *Appl. Soft Comput.* 43 : 583-595.
- [64] Zhao, J.F. Huang, T.L. Pang, F. and Liu, Y. 2009. "Genetic Algorithm Based on Greedy Strategy in the 0-1 Knapsack Problem." in **IEEE 3rd International Conference on Genetic and Evolutionary Computing.** China : IEEE.
- [65] Zhou, Y. Chen, X. and Zhou, G. 2016. "An Improved Monkey Algorithm for 0-1 Knapsack Problem." *Appl. Soft Comput.* 38 : 817-830.
- [66] Zou, D. Gao, L. Li, S. and Wu, J. 2011. "Solving 0-1 Knapsack Problem by Novel Global Harmony Search Algorithm." *Appl. Soft Comput.* 11(2) : 1556-1564.



ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Parallel time–space reduction by unbiased filtering for solving the 0/1-Knapsack problem

Patcharin Buayen*, Jeeraporn Werapun

Department of Computer Science, Faculty of Science, King Mongkut's Institute of Technology, Ladkrabang, Bangkok 10520, Thailand

HIGHLIGHTS

- Parallel time–space reduction is proposed for solving the 0/1 Knapsack problem.
- Unbiased filtering and dynamic weighting are study keys in this NP-hard problem.
- Right man, right place, and right time are our commitment in polynomial time.
- Correctness of solutions were verified by simulation study and benchmark datasets.
- In experiment, most of our solutions are optimal in efficient time and speedup.

ARTICLE INFO

Article history:

Received 26 September 2017

Received in revised form 27 June 2018

Accepted 5 August 2018

Available online 21 August 2018

Keywords:

Parallel time–space reduction for solving

0/1-KP

NP-hard problems and optimization

Unbiased filtering for polynomial-time reduction

0/1-KP feature sorting

Dynamic programming

ABSTRACT

The 0/1-Knapsack problem (0/1-KP) is one of popular NP-hard problems since its optimal solutions are meaningful to data-science computing in real-world applications (i.e., maximum profit of product planning-and-loading, minimum encryption of reliable information-transmission, etc.). Theoretically, the optimal solutions of the 0/1-KP can be solved by the dynamic programming (DP) but in non-polynomial time, which is not reasonable for big data n and massive limited-capacity C . Practically, the hybrid swarm-optimization could improve the 0/1-KP solutions in polynomial time but best fitness-solutions may not be optimal. Therefore, this paper proposes the parallel time–space reduction for solving the 0/1-KP in polynomial time. Our innovation and contribution focus on achieving 1. the fast processing as the FS (feature sorting) and the optimal solutions close to the DP by our unbiased-filtering and 2. the optimized time-complexity $O(n/p \log_2 p + C'/p)$ by our parallel time–space reduction. In this new approach, we integrate the FS method (for the efficient initial solutions) and the optimal DP (on the remaining critical-objects n' and knapsack-capacity C'), incorporated with the proposed unbiased-filtering. Finally, performance is evaluated by simulation study on a variety of problem-sizes. In experiment, most of our solutions are optimal as comparable to the optimal solutions of the DP.

© 2018 Elsevier Inc. All rights reserved.

1. Introduction

The optimal solutions of many real-world applications are NP-hard problems, which cannot be solved in polynomial time. Recently, many researchers have intently focused on finding the optimal solutions of the NP-hard problems in efficient time, especially for the real-time data-science computing under the big-data era. The 0/1 knapsack problem (0/1-KP) is one of challenging combinatorial NP-hard problems [12,21], which can be applied to the reliable real-time processing. The 0/1-KP is formally defined as follows: Given an object set N (of n objects) with a limited capacity C , where each object i consists of the profit p_i and weight

w_i [17]. The objective of the 0/1-KP is to select the most valuable objects kept into the knapsack, leading to the requirement that the maximum total-profit and the total weight of all selected objects not to exceed a limited capacity C , as expressed in expressions (1.1)–(1.3).

$$\text{Maximize } \sum_{i=1}^n p_i x_i \quad (1.1)$$

$$\text{Subject to } \sum_{i=1}^n w_i x_i \leq C \quad (1.2)$$

$$x_i \in \{0, 1\} \quad (1.3)$$

Expression (1.1) denotes the KP objective that the total profit from the selected objects must be maximized. Expression (1.2) indicates the constraint that the total weight of the selected objects cannot

* Corresponding author.

E-mail addresses: 56605013@kmitl.ac.th (P. Buayen), jeeraporn.we@kmitl.ac.th (J. Werapun).

exceed the knapsack capacity C . Lastly, expression (1.3) defines the binary (0/1) decision for selecting an object to put into the knapsack, where $x_i = 1$ (select the object i) and $x_i = 0$ (not select the object i).

The 0/1-KP is a popular NP-hard problem since its solutions have been widely applied to many real-world applications. For instance, they were applied in furnace loading and production planning [7], in risk balancing and assortment optimization [31], and also in storage capacity limitation [16]. The fuzzy 0/1-dimensional knapsack model was also applied to solve the capital budgeting [4]. In addition, the KP model was applied efficiently in resource allocation [2,35]. Moreover, the research in security network could apply the KP model for the construction of trapdoor KP and the encryption of information and signature for secure transmission [22] as well as other applications in network systems [1,36,37].

However, the performance of such applications depends on the precision of the 0/1-KP solving, and hence the efficient NP-hard solving still has been researched for the optimal solutions in reasonable time. Currently, there exist two main approaches for solving the 0/1-KP with different advantages, which are either the fast computing or the optimal solutions. In theoretical study, optimal solutions can be solved by the dynamic programming (DP), the branch-and-bound (BnB), and the backtracking algorithms but in non-polynomial time. In practice, many heuristic approaches focus on the fast processing (in polynomial time) to find the approximate solutions (i.e., feature-sorting (FS) of many heuristic algorithms, genetic algorithms (GA), swarm optimization, etc.).

In the studying of optimal solutions, the new branch-and-bound (BnB) algorithm [9] was introduced in 2010 to solve the faster exact solution. In 2011, the backward reduced-state DP space (BRDS) [29] was proposed for multi-objective 0/1-KP. Later, the BRDS was improved for the bi-objective integer 0/1-KP in 2014 [30] as well as the robust-KP [23] in 2013. In addition, the DP could be applied to solve the discounted 0/1-KP (DKP) [28], as the extension of the classical 0/1-KP. However, the optimal solutions could be achieved in reasonable time for limited $n < 1000$ objects.

For large n , inexact solutions of the fast heuristic algorithms have been researched. One of such algorithms was the decreasing density greedy (DDG) for the 0/1-KP in average-case performance [6] and in 2015 [27] a tightly approximate algorithm was used for the 0/1-KP solving.

The genetic algorithm (GA) is a crucial approach of the evolutionary computation that has been widely researched to improve the 0/1-KP solutions [13,19,32,39] during 2013–2016, especially for a variety of applied sciences. One was an emulated DNA model [14] by molecular biological searching technique, which also focused on the fast parallel searching. In 2011, a novel global harmony search (NGHS) [42] was effectively worked for large-scale instances of the 0/1-KP solving. The simplified binary harmony search (SBHS) [18], an extension of harmony search, was introduced in 2015 for tackling KP efficiently. A chemical reaction optimization (CRO) [34] was also solved by the 0/1-KP that was inspired by the nature of chemical reaction.

Another widespread research of the evolutionary computation is the swarm-optimization. Such an algorithm was parodied from behavior of many creatures (such as ant, bird, frog, monkey, etc.), including activities and attitude of human. In 2011, the improved solutions of many intelligent algorithms (i.e., the ant colony algorithm) [38] were studied and compared to the simulated annealing algorithm and taboo search method for the 0/1-KP solving. Later in 2013, a novel ant colony optimization [8] was proposed to solve the 0/1-KP by focusing on the fuzzy in nature of the profit and weight, where the fuzzy possibility and necessity approaches were used to obtain the better decision. For the swarm-based research, a modified binary particle swarm optimization (MBPSO) [3] was introduced in 2012 for the particular 0/1-KP and the

multidimensional-KP. In 2014, a modified discrete shuffled frog leaping (MDSFL) algorithm [5] was proposed to solve the 0/1-KP that parodied from behavior of food sourcing of frogs. In 2015, a soccer league competition (SLC) algorithm [24] was a new meta-heuristic optimization technique for solving the 0/1-KP that parodied from the soccer competition to be used successfully to tackle that optimization problem.

In fact, many 0/1-KP applications, especially real-time processing, rely on the exact solutions of the fast and correct decision. Currently, such solutions can be solved by the DP algorithm for small n only. For large n , the fast heuristic methods offer only feasible (inexact) solutions that cause a risk of errors and reduce the precision of the real-time processing. Thus, it is inspiring to find the fast 0/1-KP solutions with least errors for those applications.

In this paper, we propose the parallel time–space reduction for solving the 0/1-KP, focusing on 1. unbiased filtering for the time–space reduction to solve the 0/1-KP in polynomial time and 2. the parallel time–space reduction for the optimized time-complexity $O(n/p \log_2 p + C'/p)$. Our innovation is the integration of the fast FS (feature sorting) and the optimal DP (dynamic programming), incorporated with unbiased filtering (for the efficient time–space reduction). Our key contribution relies on the triple-right commitment (right man, right place, and right time) that applies the right strategies in the right places (i.e., the proper sequence of functional steps) with the right time (i.e., the FS algorithm for initial solutions on large n plus a massive capacity C and the DP algorithm for exact solutions on small remaining n' and a reduced capacity C'). In the algorithm analysis, we are able to contribute the fast processing as the FS and the optimal solutions close to the DP. Finally, in our experiment most of our solutions confirm the optimal solutions as those of the DP.

The main parts of this paper are organized as follows: Section 2 demonstrates the related work in the 0/1-KP, including the feature sorting (FS) of efficient heuristic algorithms for the fast initial solutions, the dynamic programming (DP) for the optimal solutions, and the best fitness-results of swarm-optimization. Section 3 proposes the time–space reduction by unbiased filtering for solving the 0/1-KP and Section 4 presents the efficient parallel time–space reduction. Sections 5 and 6 provide algorithm analysis and experimental results on a number of datasets ($n \leq 30000$). For $n > 30000$, the speedup and efficiency of the parallel 0/1-KP solving have been investigated. Finally, Section 7 discusses the conclusion of this study and our future work.

2. Related work

Recently, the hybrid-computing approach has been focused to solve the better 0/1-KP solutions in polynomial time, such as the GA combined with the greedy algorithm [40]. In 2016, a binary version of monkey algorithm [41] was introduced by using the greedy algorithm to strengthen in the local search ability to avoid falling into the local optimal solution. In 2016, a quantum inspired social evolution (QSE) algorithm [25] was another hybrid approach of human interactions and the biased quantum computing for the 0/1-KP solving. Lastly, in 2016 the fast 0/1-KP solving, based on the greedy strategy and the dynamic expectation efficiency (GDEE) algorithm [20] was proposed to classify the group of objects for improving the solutions.

In this 0/1-KP study, there are three outstanding approaches, related to our new time–space reduction algorithm (in Section 3), which are the fast FS (feature sorting) of the heuristic algorithm for large n , reviewed in Section 2.1 and the optimal DP (dynamic programming) for small n (in Section 2.2). In practice, Section 2.3 shows the 0/1-KP solving by the swarm-optimization.

X	1	2	3	4	5	6
W	10	11	11	8	10	9
P	42	39	38	37	35	38

n = 6 C = 18

Greedy Heuristic 1 (GH1)
(Decreasing order by P/W)

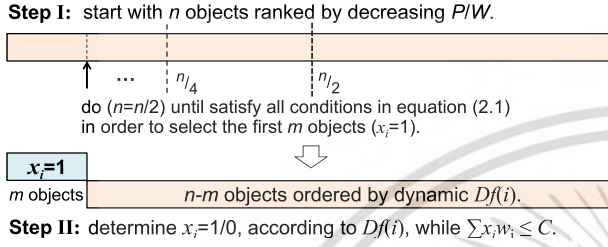
Y	1	2	3	4	5	6
X	4	6	1	2	5	3
W	8	9	10	11	10	11
P	37	38	42	39	35	38
P/W	4.6	4.22	4.2	3.55	3.5	3.45
	1	1	0	0	0	0
tw1	= 17					
tp1	= 75					
Opt. Sol.	= Not Optimal					

Greedy Heuristic 2 (GH2)
(Decreasing order by P)

Y	1	2	3	4	5	6
X	1	2	6	3	4	5
W	10	11	9	11	8	10
P	42	39	38	38	37	35
	1	0	0	0	0	0
tw2	= 10					
tp2	= 42					
Opt. Sol.	= Not Optimal					

Greedy Heuristic 3 (GH3)
(Increasing order by W)

Y	1	2	3	4	5	6
X	4	6	1	5	2	3
W	8	9	10	10	11	11
P	37	38	42	35	39	38
	1	1	0	0	0	0
tw3	= 17					
tp3	= 75					
Opt. Sol.	= Not Optimal					

Fig. 1. An example of 0/1-KP ($n = 6$) solved by the greedy strategies.Fig. 2. Initial region ($x_i = 1$) and uncertain region of GDEE for 0/1-KP.

2.1. 0/1-KP greedy heuristic algorithms

The 0/1-KP greedy heuristic (GH) algorithm is a well-known FS method since it can find the approximate solutions very quickly. Recently, the GH strategy has been applied to find the initial solutions in the hybrid optimization algorithms. The fast solution of the GH strategy is assembled by ranking objects (i.e., $O(n)$ by the Radix sort) with respected to decreasing P/W , or decreasing P (profit), or increasing W (weight) before selecting objects into the knapsack, according to expressions (1.1)–(1.3). For example, given n objects ($n = 6$), a capacity $C = 18$, a set of profit $P = \{42, 39, 38, 37, 35, 38\}$, and a set of weight $W = \{10, 11, 11, 8, 10, 9\}$. In this example, none of those GHs (see Fig. 1) can find the optimal solution (shaded $x_i = \{1, 4\}$ with max total-profit = 79).

In 2016, the greedy method with dynamic expectation efficiency (GDEE) [20] and parallel GDEE algorithms were proposed to improve 0/1-KP solutions. The GDEE strategy applied P/W -ranking to partition data into initial selected region and uncertain region (see Fig. 2). In step I, the first region (m objects) was found by dividing objects from $n = n/2$ until satisfy all conditions of Eq. (2.1). In step II, the remaining ($n-m$) objects were ordered by the dynamic $Df(i) = f(i) \cdot (Ar_{i-1}B)/(A-C)$ before selecting, where $f(i)$ was the expectation efficiency of item i ($m+1 \leq i \leq n$), $f(i) = r_i/r_{i-1}$, $r_i = (cap - \sum_{k=1}^i w_k)P_i/w_i$ and $r_{i-1} = (cap - \sum_{k=1}^{i-1} w_k)P_{i-1}/w_{i-1}$, $A = cap - \sum_{k=1}^m w_k - \sum_{k=m+1}^{i-1} w_k x_k$, $B = (n-i+1)p_i$, $C = (n-i+1)w_i$. That fast GDEE yielded the improved results similar to the MDSFL swarm-optimization [5] with the greedy method for the initial solutions, see detail in Section 2.3. However, most of their solutions were not optimal.

$$a_1 \wedge a_2 \leq \gamma, a_1^+ \wedge a_2^+ > \gamma, b_1 \wedge b_2 < \delta, b_1^+ \wedge b_2^+ \geq \delta \quad (2.1)$$

$$\text{where } a_1 = \frac{\sum_{i=1}^j w_i}{Gw}, a_2 = \frac{\sum_{i=1}^j p_i}{Gtp}, a_1^+ = \frac{\sum_{i=1}^{j+1} w_i}{Gw}, a_2^+ = \frac{\sum_{i=1}^{j+1} p_i}{Gtp}$$

$$b_1 = \frac{\sum_{i=1}^{n/2} w_i + \sum_{i=n/2+1}^j w_i}{Gw}, b_2 = \frac{\sum_{i=1}^{n/2} p_i + \sum_{i=n/2+1}^j p_i}{Gtp}$$

$$b_1^+ = \frac{\sum_{i=1}^{n/2} w_i + \sum_{i=n/2+1}^{j+1} w_i}{Gw}, b_2^+ = \frac{\sum_{i=1}^{n/2} p_i + \sum_{i=n/2+1}^{j+1} p_i}{Gtp}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาค้นคว้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

d	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
j=1	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42
j=2	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42
j=3	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42
j=4	0	0	0	0	0	0	0	0	0	37	37	42	42	42	42	42	42	42	79
j=5	0	0	0	0	0	0	0	0	37	37	42	42	42	42	42	42	42	42	79
j=6	0	0	0	0	0	0	0	37	38	42	42	42	42	42	42	42	42	75	79

Fig. 3. An example ($n = 6$) of n iterations of $tp[C]$ results by the fast DP.

2.2. 0/1-KP dynamic programming algorithms

The optimal solutions of the 0/1-KP could be solved by the dynamic programming (DP) algorithm in non-polynomial time $O(knC)$, according to a production of a large problem size (n), a number of selected objects ($k < n$), and a massive limited-capacity C , where $C \leq n^d$ and $d \geq 1$.

For the 0/1-KP solving, let tp (total profit) be an array for the total profit of each iteration, $soltp$ represent a maximum total-profit, $soltw$ refer to a total weight, and $solx$ be a solution set X (an index set of k selected objects).

Algorithm 2.1: The fast DP using a single array in $O(nC)$.

```

Algorithm fast DP: find only max  $tp$  (total profit) or  $soltp$ .
1. for  $d=0$  to  $C$  do
2.    $tp[d] \leftarrow 0$ 
3. for  $j=1$  to  $n$  do
4.   for  $d=C$  down to  $w_j$  do
5.     if  $tp[d-w_j]+p_j > tp[d]$  then
6.        $tp[d] \leftarrow tp[d-w_j]+p_j$ 
7.   end for  $d$ 
8. end for  $j$ 
9.  $soltp \leftarrow tp[C]$ 

```

Algorithm 2.2: The complete DP using a single array in $O(knC)$.

```

Algorithm complete DP: return ( $soltp, solx$ ).
1.  $solx \leftarrow \emptyset$ ;  $C' \leftarrow C$ ;  $n' \leftarrow n$ 
2. do // repeat for  $k = |\text{selected objects}|$ 
3.   for  $d=0$  to  $C'$  do
4.      $tp[d] \leftarrow 0$ ;  $x[d] \leftarrow -1$  // initialized arrays ( $tp, x$ )
5.     for  $j=1$  to  $n'$  do
6.       for  $d=C'$  down to  $w_j$  do
7.         if  $tp[d-w_j]+p_j > tp[d]$  then
8.            $x[d] \leftarrow j$ ;  $tp[d] \leftarrow tp[d-w_j]+p_j$ 
9.       end for  $d$ 
10.    end for  $j$ 
11.     $r \leftarrow x[C']$ 
12.     $solx \leftarrow solx \cup \{r\}$ ;  $k \leftarrow k+1$ 
13.     $n' \leftarrow n'-1$ ;  $C' \leftarrow C'-w_r$ 
14.  while  $C' > 0$ 
15.   $soltp \leftarrow tp[C]$ 

```

While the basic DP algorithm uses a 2-D array ($n \times C$) for solving the complete solution ($soltp, soltw, solx$) in $O(nC)$, the efficient DP algorithms presented in this section are Algorithm 2.1, the fast DP ($O(nC)$) using a 1-D array ($tp[C]$) for the total profit ($soltp$) only, and Algorithm 2.2, the complete DP ($O(knC)$) using a 1-D array for the complete result ($soltp, soltw, solx$).

The advantage of the fast DP algorithm is fast to find the $soltp$ (the maximum total-profit) in $O(nC)$ but it does not cover the $solx$ (solution set X). Fig. 3 illustrates the profit result ($soltp$) of the fast DP for a dataset in Fig. 1. For each object j in iteration j , the $tp[d]$ from $d = 0$ to w_j-1 are not changed, and hence only $d = w_j$ to C are computed (for $tp[d-w_j] + p_j$) and updated (in new $tp[d]$ if it is better than old $tp[d]$). Finally, the maximum total-profit ($soltp$) will be in the array $tp[C]$ after finishing the last iteration ($n = 6$) and $C = 18$.

In Algorithm 2.2, the complete DP computes the full solution of the DP, including the total profit ($soltp$), the total weight ($soltw$), the solution set X ($solx$), and the efficient process on a single array $tp[C]$. In that case, the repeating process is required for finding all of the selected objects ($x_i = 1$) in the $solx$ in $O(knC)$, where the process

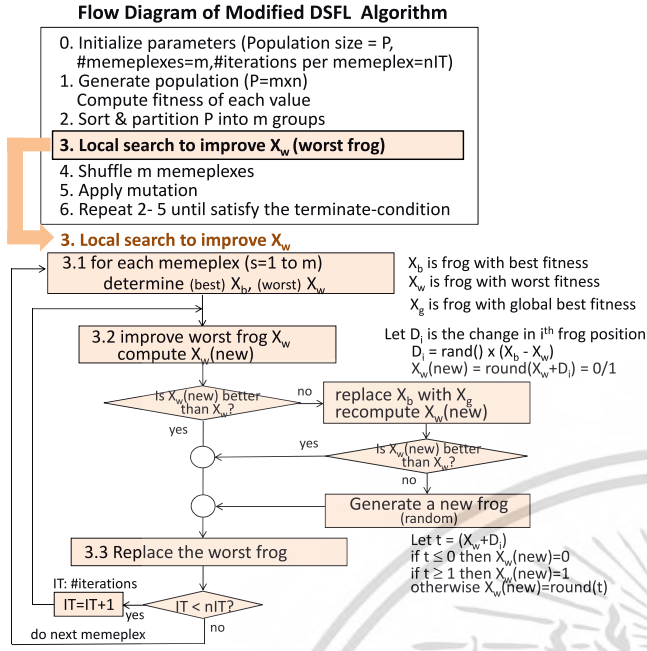


Fig. 4. A flow diagram of the modified DSFL algorithm.

Table 1

Advantages and limitation of 0/1-KP approaches (in Section 2) and our new approach, called time-space reduction (in Section 3).

Greedy method	Frog SWARM optimization	Complete DP	Our time-space reduction
Sort feature (P/W , P or W) before select objects (fast)	mn initial solutions (m frog-memeplexes)	Compute max total-profit for n objects on increasing C	Find efficient initial solutions (P/W -, P -, W -ordering)
–	Improve sols. (worst X_w + Mutation)	Repeat the process for finding each of k selected objects	Improve 3 best solutions before filtering
–	Repeat until stable result	(Optimal X)	Apply DP on remaining n', C'
$O(n)$	$O(mn^2)$	$O(knC)$; $k < n$, $C \leq n^d$, $d \geq 1$	$O(n + C')$ where $C' \ll C$
Solutions are not optimal	Solutions are not optimal	Optimal solutions	Most solutions are optimal

of the first iteration is the same as that of the fast DP with a single array $tp[C]$, except an array r for holding a temporary selected x_i (i.e., the first x_i is $x_4 = 1$). Then, the repeated process is continue to find others x_j while $C' > 0$ and the final result $\text{solx} = \{4, 1\}$.

2.3. Shuffled-frog swarm-optimization for the 0/1-KP

The modified discrete shuffled frog leaving (MDSFL) algorithm [5] was presented in 2014, see the flow diagram in Fig. 4. That study included two important operations: 1. the local search of PSO (particle swarm optimization) and 2. the competitive mixing of information of the shuffled complex evolution. The advantages of the MDSFL were the fast convergence of the average fitness and the local optimal solving by the mutation. In experiment, the average fitness-results of the MDSEL outperformed those of the existing DSFL in 2003. However, most of the best fitness solutions by the MDSFL were not optimal.

Table 1 shows the advantages of existing 0/1-KP methods, which are the fast heuristic method by feature-sorting in $O(n)$, the recent meta-heuristic swarm-optimization in $O(mn^2)$ in practical study, and the complete DP for the optimal solutions in $O(knC)$ in theoretical study, where a knapsack capacity $C \leq n^d$ and $d \geq 1$.

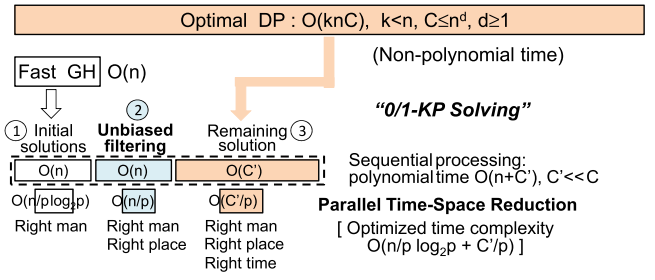


Fig. 5. The innovation and contribution of our unbiased filtering for parallel time-space reduction to solve the 0/1-KP in polynomial time.

Next, in Section 3 we propose a new approach of 0/1-KP solving, called “time-space reduction” in $O(n + C')$, focusing on unbiased filtering.

3. Time-space reduction by unbiased filtering for solving the 0/1-knapsack problem

Since 2013, we have intently researched to fulfill the exact solutions of the 0/1-KP (Knapsack problem), a popular NP-hard problem, for large n in polynomial time. This study proposes 1. the unbiased filtering for time-space reduction (in Section 3) to solve the 0/1 KP in $O(n + C')$ and 2. the parallel time-space reduction (in Section 4) for the optimized time complexity $O(n/p \log_2 p + C'/p)$ to be able to utilize in many real-time applications, especially the data-science computing on the big data, where a reduced capacity $C' \ll C$ and $C \leq n^d$ ($d \geq 1$). Clearly, our parallel reduction is necessary when $d \geq 2$. The key innovation and contribution of our new approach is assembled in Fig. 5, focusing on the unbiased filtering (for the time-space reduction) in polynomial time with the triple-right commitment (right man, right place, and right time) for the maximum total-profit. Our unbiased filtering is able to pre-select the outstanding (worth) objects to be in the knapsack, respected to the maximized total-profit $\sum_{i=0}^{n-1} p_i x_i$, based on the capacity C (or $\sum_{i=0}^{n-1} w_i x_i \leq C$), where $x_i \in \{0, 1\}$ for the objects $i = 0, 1, 2, 3, \dots, n-1$ (see Fig. 6).

Algorithm 3.1: Time-space reduction for the 0/1-KP in $O(n + C')$.

- Step 1: Apply Greedy Heuristic (GH) algorithm in $O(n)$ by feature (P/W , P , W) sorting (i.e., applying the existing Radix sort in $O(n)$) to find best initial solutions, see Section 3.1.
- Step 2: Object classification and unbiased filtering in $O(n)$ for time-space reduction, see Section 3.2.
- 2.1 Improve initial solutions by GH^+ algorithm.
 - 2.2 Compute dynamic weighting ($dw = wx_1 + wx_2 + wx_3$) by integration of weighted solutions of $GH1^+$ to $GH3^+$, where $GH1^+$ (P/W) $wx_1=5$, $GH2^+$ (P) $wx_2=3$, $GH3^+$ (W) $wx_3=1$ if $x_i=1$; otherwise ($x_i=0$) $wx_{GH}=0$.
 - 2.3 Unbiased filtering (Fig.6) for time-space reduction:
 - Filtering 1: Select worth objects ($x_i=1$) with $dw = 9$ (commonly selected objects) if $n' \leq 200$.
 - Filtering 2: When $n' > 200$, select other objects ($x_i=1$) with $dw = 8, 6, 5$, except the margin α and do not select objects ($x_i=0$) with $dw=0$, except the margin β .
- Step 3: Apply the modified DP (MDP) algorithm in $O(C')$ on remaining objects ($n' = \alpha + |g_3| + \beta \leq 200$), see Section 3.3.

Algorithm 3.1 (time-space reduction) is proposed to fulfill the optimal solutions of the 0/1-KP in polynomial time. Our approach consists of three main steps for 3-dimensional solving. First, the efficient Greedy Heuristic (GH) algorithm is applied by feature (P/W , P , W) sorting to construct the efficient initial solutions (in Section 3.1). In step 2, the time-space reduction is proposed (in

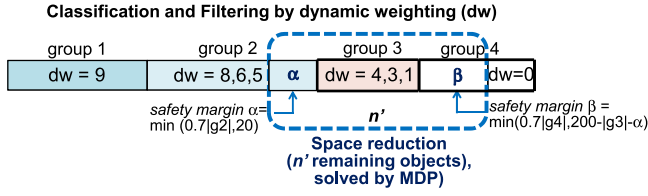


Fig. 6. Classification and filtering by the dynamic weighting (dw).

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
P	6	5	8	10	8	11	4	11	3	3	8	8	7	10	11	10	4	7	8	9
W	43	55	45	42	55	39	32	38	45	31	37	40	36	35	54	34	54	37	50	38
P/W	7.2	11	5.6	4.2	6.9	3.5	8	3.5	15	10	4.6	5	5.1	3.5	4.9	3.4	14	5.3	6.3	4.2
GH1	1	1	1	0	1	0	1	0	1	1	1	1	1	0	1	0	1	1	1	1
GH2	1	1	1	1	1	0	0	1	0	0	1	0	0	1	0	1	0	1	1	1
GH3	1	1	1	1	1	0	1	0	1	1	1	1	1	0	0	1	1	1	1	1
tw	99	652																		
tp	91	560																		
tp	98	640																		

Fig. 7. An example of initial solutions by GH (P/W, P, W) strategies.

Table 2
Multi-key processing for the efficient FS method.

GH1 (decreasing P/W)	GH2 (decreasing P)	GH3 (increasing W)
1st key is P/W and if $p_i/w_i = p_j/w_j$, 2nd key is P. else if $p_i = p_j$, 3rd key is W.	1st key is P and if $p_i = p_j$ then 2nd key is W.	1st key is W and if $w_i = w_j$ then 2nd key is P.

Section 3.2) to filtering-in ($x_i = 1$) the worth objects (i.e., pure diamonds) and filtering-out ($x_i = 0$) some worst objects (i.e., heavy rocks). Finally, in step 3 the modified DP is applied to complete the remaining solution of critical objects (small n') in Section 3.3.

3.1. Initial solutions by effective feature-sorting

For solving the 0/1-KP, there are three effective features (P/W, P, or W) in feature sorting (FS) with multi-key processing (see Table 2) to achieve best initial solutions. Among different benefits of these features (i.e., max P/W, max P, or min W), the P/W ranking yields the best results. However, for filtering the best P/W feature is not able to classify the objects ($x_i = 1$ or 0) correctly. In this study, we propose the unbiased filtering by integrating all features (P/W, P, and W) to classify the worth objects ($x_i = 1$) and the worst objects ($x_i = 0$). Thus, before filtering the efficient initial solutions will be initiated by the FS of GH strategies, including GH1 (P/W-ranking), GH2 (P-ranking), and GH3 (W-ranking). For example, n objects with limited capacity 100 ($C = 100$) and their features are demonstrated in Fig. 7 ($n = 20$) and three initial solutions ($\sum_{i=1}^{n-1} p_i x_i$) are GH1 (P/W) = 652, GH2 (P) = 560, and GH3 (W) = 640, whereas the optimal solution = 656.

3.2. Object classification and unbiased filtering

The significant key of our time-space reduction (for large n) is to filtering the certainly worth objects to reduce the searching space and time. Then, only the remaining (critical) objects (small n') will be solved by the modified DP. The main functions of our unbiased filtering are 1. improve solutions by the GH⁺ strategy (in Section 3.2.1); 2. compute dynamic weighting to classify objects (in Section 3.2.2); and 3. perform unbiased filtering (in Section 3.2.3).

3.2.1. Solution refinement by GH⁺ algorithms

Before filtering, the initial solutions must be improved by the GH⁺ (Greedy Heuristic⁺) algorithm (Algorithm 3.2) in $O(1)$ to increase the total profit, where the last selected object might be excluded ($x_{last} = 0$) and some of next 200 objects (i) in that rank may be included ($x_i = 1$) if they yield the better result. For

Table 3
Dynamic weighting (dw) and proper classification.

dw	Selected decision	Meaning [if ($x_i = 1$) $wx_1 = 5$, $wx_2 = 3$, $wx_3 = 1$; else $wx_i = 0$]
9	3 of 3 (P/W, P, W)	3-GHs ⁺ commonly select the object [$dw = 5 + 3 + 1$].
8	2 of 3 (P/W, P)	Top 2-GHs ⁺ select the object [$dw = 5 + 3$].
6	2 of 3 (P/W, W)	Two GHs ⁺ select the object [$dw = 5 + 1$].
5	1 of 3 (P/W)	Only top GH1 ⁺ (P/W) selects the object [$dw = 5$].
4	2 of 3 (P, W)	Last 2-GHs ⁺ select the object [$dw = 3 + 1$].
3	1 of 3 (P)	Only GH2 ⁺ (P) selects the object [$dw = 3$].
1	1 of 3 (W)	Only GH3 ⁺ (W) selects the object [$dw = 1$].
0	0 of 3 (–)	None of policies selects the object [$dw = 0$].

example, the GH1⁺ (P/W-ranking) can increase total profit (tp) from 652 to 656 (optimal); the GH2⁺ (P-ranking) can improve tp from 560 to 627; and lastly the GH3⁺ (W-ranking) can improve tp from 640 to 652.

Algorithm 3.2: Greedy Heuristic⁺ (GH⁺) algorithm in $O(1)$.

```

Algorithm Greedy Heuristic+ (GH+): [For GH1+, GH2+, or GH3+]
1. point ← the address of last selected object ( $x_{point} = 1$ )
2.  $tw' \leftarrow tw - w_{point}$ 
3.  $tp' \leftarrow tp - p_{point}$ 
4.  $i = point + 1$ 
5. while ( $i < \text{minimum}(\text{point} + 200, n)$ )
6.   while ( $p_i = p_{point}$  &  $w_i \geq w_{point}$ )  $i = i + 1$ 
7.   while ( $w_i = w_{point}$  &  $p_i \leq p_{point}$ )  $i = i + 1$ 
8.   if ( $tw' + w_i \leq C$ ) // start the improved process ( $tp', tw'$ )
9.      $solx' \leftarrow '1'$ 
10.     $tw' \leftarrow tw' + w_i$ ;  $tp' \leftarrow tp' + p_i$ 
11.   end if
12.    $i \leftarrow i + 1$ 
13. end while // concern 200 objects (from point) in GH+ ranking
14. if ( $tp' > tp$ ) // update improved results
15.    $solx_{point} \leftarrow '0'$  // unselect  $solx_{point}$ 
16.   update  $solx'$  // update new selected objects
17.    $tp \leftarrow tp'$ ;  $tw \leftarrow tw'$  // update new tp & new tw
18. end if

```

3.2.2. Assign dynamic weighting for proper classification

For the correct filtering, first we assign the proper weights to the results ($x_i = 1$) of three GH⁺ policies, according to the effects of their feature sorting, which are 1. let $wx_1 = 5$ for GH1⁺ (decreasing P/W), the most effective ordering; 2. let $wx_2 = 3$ for GH2⁺ (decreasing P), and 3. let $wx_3 = 1$ for GH3⁺ (increasing W), the least effective ranking (except on irregular datasets with biased W over P), which the solutions of GH3⁺ (W) may be better than that of GH2⁺ (P). The dynamic weighting (dw) is computed by the integration of the selected/unselected objects from all policies ($dw = wx_1 + wx_2 + wx_3$). Next, the dw-values are used to classify all objects into four groups, which are group 1 ($dw = 9$), group 2 ($dw = 8, 6, 5$), group 3 ($dw = 4, 3, 1$), and group 4 ($dw = 0$).

Table 3 shows the proper classification of all dw-values ($= 9, 8, 6, 5, 4, 3, 1, 0$), where the highest $dw = 9$ denotes the commonly selected object, which yields the benefit to all policies (high P/W, high P, small W), and the least $dw = 0$ (i.e., none of three policies selects the object). The idea of our classification is to finding the good (or worth) objects, accepted by all effective policies (i.e., max P/W, max P, and min W). Clearly, if an object (i) is commonly selected by all GHs⁺ ($dw = 9$), it will be the worth object and must be in the knapsack ($x_i = 1$). Otherwise, if an object (i) is not selected by all GHs⁺ ($dw = 0$), that object should be rejected ($x_i = 0$).

3.2.3. Unbiased filtering

In our study, we perform the unbiased filtering for the correct time-space reduction in order to solve the 0/1-KP correctly in polynomial time. According to the fact that for small n we can apply the DP to find the optimal solutions (in reasonable time). Thus, for large n if the filtering works correctly, the remaining objects (small n') can be solved by the modified DP in reasonable time.

Under the unbiased filtering, the *dw*-classification is the integration of three effective results, to achieve the good benefits for all different policies (max *P/W*, max *P*, and min *W*) since using only the best *P/W*-ranking is not able to define the proper classification, especially on irregular datasets (i.e., either biased *P* or biased *W*, etc.). Therefore, the correct filtering is our key innovation and contribution, which relies on the dynamic weighting (*dw*) of results from three effective policies to classify objects into four groups and within each group the objects are considered by the best *P/W*-ranking. Obviously, the worth objects are selected ($x_i = 1$) with $dw = 9$ since they are satisfied all (*P/W*, *P*, and *W*) policies, whereas some least ranking objects with $dw = 0$ are not selected ($x_i = 0$). Moreover, for handling irregular datasets, we introduce the double filtering technique (i.e., *Filtering 1* for small *n* and *Filtering 2* for large *n*). In *Filtering 1*, all objects in group 1 ($dw = 9$) are certainly selected into the knapsack and the remaining objects $n' = |g2| + |g3| + |g4| (\leq 200 \text{ objects})$ are solved by the modified DP. For $n' > 200$, *Filtering 2* will be required to select more objects from group 2 ($dw = 8, 6, 5$) and reduce remaining objects (n').

Filtering 1 (for small *n*)

Fig. 8(a) displays the *dw*-classification, which are group 1 ($dw = 9$), group 2 ($dw = 8, 6, 5$), group 3 ($dw = 4, 3, 1$), and group 4 ($dw = 0$), respectively. In this filtering, we select objects that have the highest $dw = 9$ (selected by all of GH^+ strategies). For example (in Fig. 7 ($n = 20$)), after the *dw*-classification (Fig. 8(b)), we will select all objects with $dw = 9$ (or worth objects, $x_i = 1$). In this case, there are 12 objects ($dw = 9$) with partial $solx = \{0, 1, 2, 4, 8, 9, 10, 11, 14, 16, 17, 18\}$, partial $soltw = 79$, and partial $soltp = 546$. Then, there exist 8 remaining objects ($n' = 8$) $x' = \{3, 5, 6, 7, 12, 13, 15, 19\}$ and remaining capacity $C' = 21$.

Filtering 2 (for large *n*)

After *Filtering 1*, if uncertain objects $n' > 200$ then *Filtering 2* will be applied to select some of worth objects from group 2 and reduce uncertain objects (in groups 3 and 4 for $dw = 4, 3, 1, 0$). In this case, some objects in group 2 ($dw = 8, 6, 5$) are selected, except uncertain α objects (with least *P/W*-ranking), and some β objects on top *P/W*-ranking in group 4 ($dw = 0$) are also uncertain objects. The parameters α , β , and n' (remaining objects) are defined in Eqs. (3.1)–(3.3).

$$\alpha = \min(70\%|g2|, 20) \quad \text{if } n \leq 30000$$

$$= \min(100\%|g2|, 30) \quad \text{otherwise} \quad (3.1)$$

$$\beta = \min(70\%|g4|, 200 - |g3| - \alpha) \quad \text{if } n \leq 30000$$

$$= \min(100\%|g4|, 300 - |g3| - \alpha) \quad \text{otherwise} \quad (3.2)$$

$$n' = \alpha + |g3| + \beta \leq 200 \quad \text{if } n \leq 30000$$

$$\leq 300 \quad \text{otherwise} \quad (3.3)$$

Fig. 9 illustrates a simplified example of *Filtering 2* (on $n' = 8$), the result from *Filtering 1* in Fig. 8(b). In *Filtering 2*, the uncertain objects are objects in group 3 ($|g3| = 2$) plus the (left) margin $\alpha = \min(70\%|g2|, 20) = \min(2, 20) = 2$ and the (right) margin $\beta = \min(70\%|g4|, 200 - |g3| - \alpha) = \min(2, 196) = 2$ from group 4 ($dw = 0$). Thus, *Filtering 2* selects one object ($x_6 = 1$) on top *P/W*-ranking in group 2, rejects one object ($x_{15} = 0$) on last *P/W*-ranking in group 4, and the remaining objects $n' = \alpha + |g3| + \beta = 2 + 2 + 2 = 6$. After *Filtering 2*, there are 13 selected objects ($x_i = 1$) with partial $solx = \{0, 1, 2, 4, 6, 8, 9, 10, 11, 14, 16, 17, 18\}$, partial $soltw = 83$, and partial $soltp = 578$, where remaining objects $n' = 6$ ($i = 3, 5, 7, 12, 13, 19$) with $C' = 17$. Finally, these uncertain n' objects are determined by the modified DP in Section 3.3 to select more objects in order to complete the solution. Note that in this research, we have studied a variety of datasets and found that a number of uncertain objects can be $n' \leq 200$ for $n \leq 30000$ to confirm the optimal solutions; otherwise ($n > 30000$) we set $n' \leq 300$.

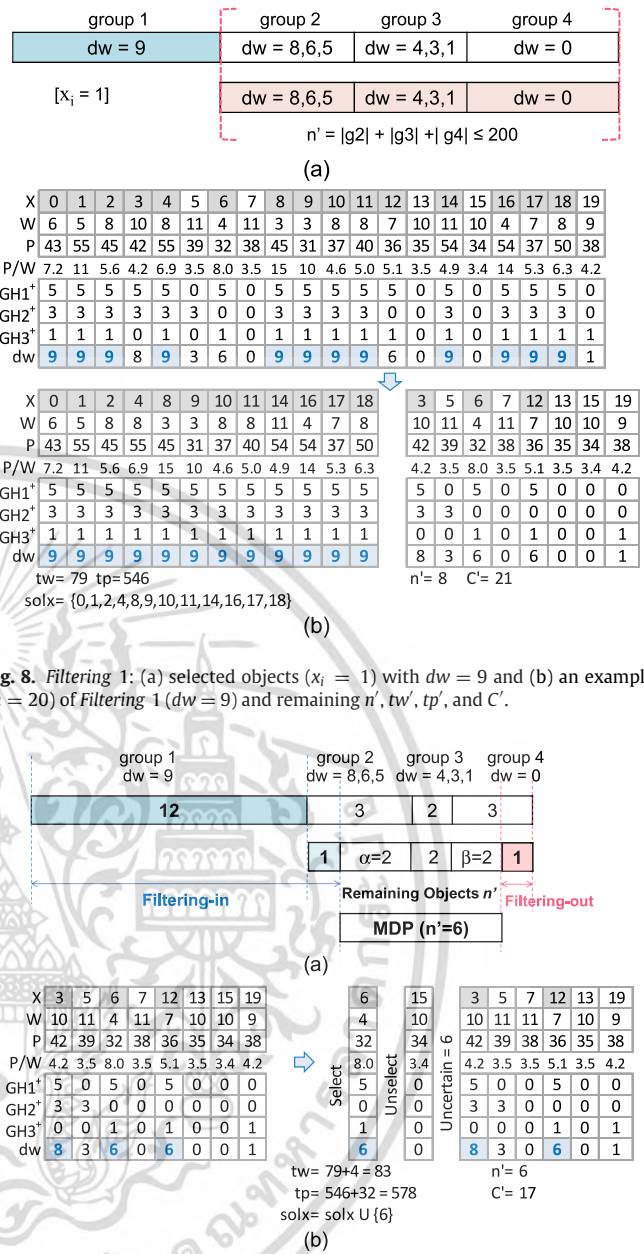


Fig. 9. Filtering 1–2: (a) a number of selected objects ($x_i = 1$) with $dw = 9, 8, 6, 5$ and uncertain objects (n'); (b) an example of *Filtering 2*.

3.3. Select remaining objects by the modified DP

After filtering, the uncertain objects in groups 3 ($dw = 4, 3, 1$) plus two uncertain margins α (from group 2) and β (from group 4) must be verified carefully by the modified DP (from Algorithm 2.2) to select some (k') of remaining objects (n') to complete the solution. In our reduction approach, if the filtering is defined correctly, then all of the worth objects will be accepted ($x_i = 1$), some of the worst objects will be rejected ($x_i = 0$) in $O(n)$ and only uncertain objects (small n') will be searched by the modified DP in $O(C')$ since $n' \leq 300$ and remaining limited $C' \ll C$. In addition, our unbiased filtering can move all uncertain objects from group 2 ($dw = 8, 6, 5$) and group 4 ($dw = 0$) to a new group $n' = \alpha + |g3| + \beta$, where $\alpha = \min(70\%|g2|, 20)$ and $\beta = \min(70\%|g4|, 200 - |g3| - \alpha)$.

For example, Fig. 10 illustrates the MDP process on the remaining objects $n' = 6$ (objects $j = 3, 5, 7, 12, 13, 19$). By applying

d	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$Y=0\ tp$	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42
r	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$j=3, w_i=10, p_i=42$																		
$Y=1\ tp$	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42
r	00	00	00	00	00	00	00	00	00	00	10	10	10	10	10	10	10	10
$j=5, w_i=11, p_i=39$																		
$Y=2\ tp$	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42
r	000	000	000	000	000	000	000	000	000	000	100	100	100	100	100	100	100	100
$j=7, w_i=11, p_i=38$																		
$Y=3\ tp$	0	0	0	0	0	0	0	0	0	36	36	36	42	42	42	42	42	78
r	0000	0000	0000	0000	0000	0000	0000	0001	0001	0001	1000	1000	1000	1000	1000	1000	1000	1001
$j=12, w_i=7, p_i=36$																		
$Y=4\ tp$	0	0	0	0	0	0	0	0	0	36	36	36	42	42	42	42	42	78
r	00000	00000	00000	00000	00000	00000	00000	00000	00000	00001	00001	00010	10000	10000	10000	10000	10000	10010
$j=13, w_i=10, p_i=35$																		
$Y=5\ tp$	0	0	0	0	0	0	0	0	0	36	36	38	42	42	42	42	42	74
r	000000	000000	000000	000000	000000	000000	000000	000000	000000	000001	000001	000010	100000	100000	100000	100000	100000	100101
$j=19, w_i=9, p_i=38$																		

Fig. 10. An example of the modified DP ($tp[d] = tp[d - w_j] + p_j$ if $tp[d - w_j] + p_j > tp[d]$, $d = w_j$ to C') on the remaining objects $n' = 6$ and $C' = 17$.

the MDP, the first step is to find an array $y = \{0, 1, 2, 3, 4, 5\}$ or $x' = \{3, 5, 7, 12, 13, 19\}$, the remaining objects ($n' = 6$) with $P' = \{42, 39, 38, 36, 35, 38\}$, $W' = \{10, 11, 11, 7, 10, 9\}$, and $C' = 17$. Then, the next step is to apply the MDP algorithm (see results $tp[C']$ and binary r in Fig. 10) for the remaining solution ($solx', soltp', soltw'$), which are $solx' = \{3, 12\}$, $soltp' = 78$, and $soltw' = 17$. Therefore, $soltp = soltp + soltp' = 578 + 78 = 656$, $soltw = soltw + soltw' = 83 + 17 = 100$, and the final solution $solx = solx \cup solx' = \{0, 1, 2, 4, 6, 8, 9, 10, 11, 14, 16, 17, 18\} \cup \{3, 12\} = \{0, 1, 2, 3, 4, 6, 8, 9, 10, 11, 12, 14, 16, 17, 18\}$.

Clearly, time complexity of our time-space reduction is $O(n + C')$, which is polynomial time on a constant of remaining objects $n' \leq 300$ and a limited capacity (after filtering) $C' \ll C$, where normally $C' \sim C/n$ and $n < C \leq n^2$. For large n and assume $C \leq n^2$, the existing DP cannot find the optimal solutions in reasonable time since $O(knC) = O(n^4)$, while our reduction approach can find the good solutions in $O(k'n'C') = O(n)$, according to a constant $k'n'$ (where $k' < n' \leq 200$). However, on big-data n , we set more $n' \leq 300$ (for higher precision), and hence consume more time for $k'n'C'$ in $O(C')$. Thus, the parallel 0/1-KP reduction is proposed (in Section 4) to find the good solutions.

4. Optimized parallel time-space reduction

This section proposes the parallel time-space reduction for the efficient 0/1-KP solving by p PEs (processing elements) on a theoretical PRAM model as well as practical machines with efficient interconnection networks (i.e., static hypercube networks and dynamic MINs⁺ (multistage interconnection networks⁺) [26], see Fig. 11(a) for big data n and massive C , especially $C \geq n^2$. In our parallel study, we focus on utilizing all PEs with efficient communication for solving the 0/1-KP in reasonable time. In this study, we achieve the optimized time complexity (see Table 4) as well as most of our solutions are optimal. In parallel (Algorithm 4.1), 3-dimensional steps of the parallel 0/1-KP solving are designed by focusing on efficient communication after workload partitioning ($w = n/p$) and local processing on p PEs. Our parallel 0/1-KP reduction is proposed for not only the fine-grain processing ($p = n$) but also the medium- and coarse-grain computing ($p < n$).

Algorithm 4.1: Parallel time-space reduction for solving the 0/1-KP.

- Step 1: Parallel GH (pGH) algorithm for initial solutions by optimized Bitonic ($P/W, P, W$) sorting and parallel prefix-weight computing for selected/unselected objects, see Section 4.1.
- Step 2: Parallel classification and filtering for time-space reduction, see Section 4.2.
- Step 3: Parallel MDP algorithm on remaining n' , see Section 4.3.

Table 4 presents optimized time complexity of 0/1-KP solving by our parallel (time-space) reduction algorithm. In step 1 (pGH),

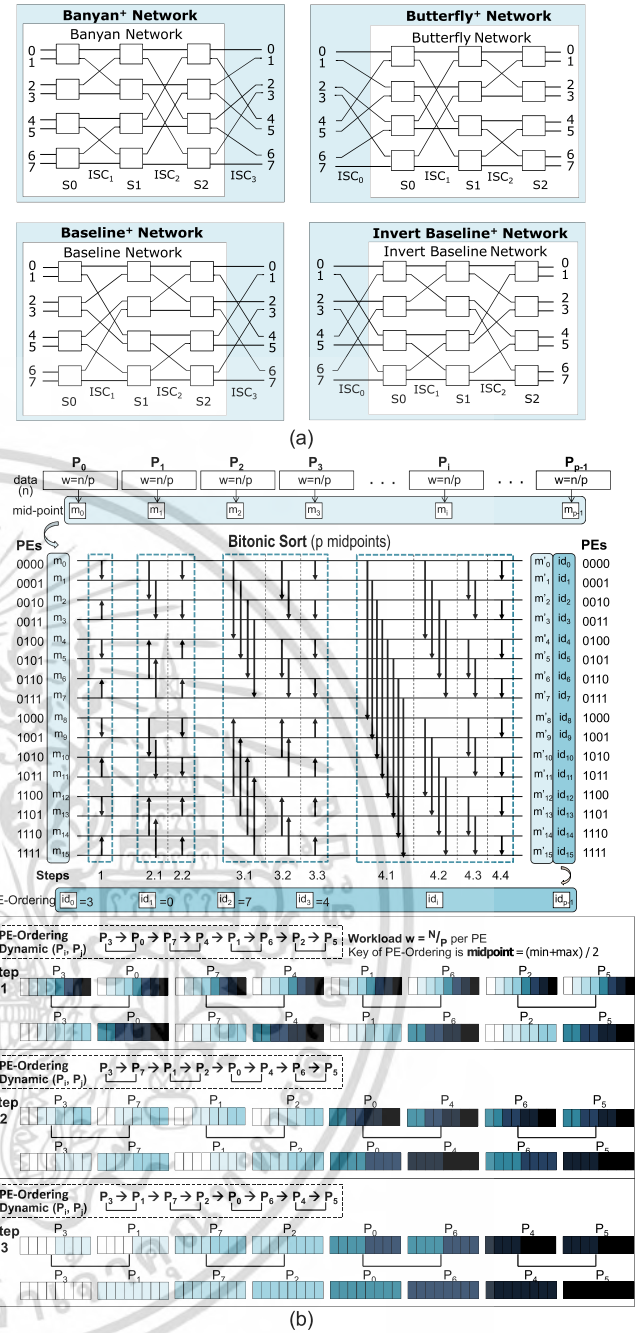


Fig. 11. (a) An example of MIN⁺ computing machines ($n = 8$) and (b) midpoint-based dynamic communication of the optimized Bitonic sort.

we apply the optimized Bitonic sort [33], see Fig. 11(b), for feature ($P/W, P, W$) sorting and introduce our parallel prefix-weight computing by all P_i ($i = 0$ to $n - 1$) that assign $solx(x_i = 1)$ if the prefix $\sum_0^i x_i w_i \leq C$ in $O(\log_2 n)$ for $p = n$, and $O(n/p \log_2 p)$ for $p < n$. In step 2 (parallel filtering), the improved pGH⁺ and filtering can be processed in $O(1)$ for $p = n$ and $O(n/p)$ for $p < n$. Lastly, the parallel MDP is $O(1)$ on $p = n$ ($C' \leq n$) or $O(C'/n)$ if $n < C'$ and $O(C'/p)$ for $p < n$.

4.1. Parallel GH algorithm for initial solutions

In our parallel GH (pGH) algorithm (Algorithm 4.2), we apply the optimized parallel sorting ($p \leq n$) [33] with local radix sort [15]

Table 4

Time complexity of our efficient parallel 0/1-KP algorithm.

Functions	$p = 1$	$p = n$	$p < n$
Step 1: feature sorting	$O(n)$	$O(\log_2 n)^2$	$O(n/p \log_2 p)$
select objects	$O(n)$	$O(\log_2 n)$	$O(n/p \log_2 p)$
Step 2: filtering	$O(n)$	$O(1)$	$O(n/p)$
Step 3: modified DP	$O(C')$	$O(C'/n)$ or $O(1)$ if $C' \leq n$	$O(C'/p)$
Total time complexity	$O(n+C')$	$O(\log_2 n)^2 + C'/n$	$O(n/p \log_2 p + C'/p)$

plus multi-key processing (Table 2) for efficient parallel FS before parallel prefix-computing of weight (W) to select objects in $O(\log_2 n)$ for $p = n$ and $O(n/p \log_2 p)$ for $p < n$.

Let A_i be a prefix-sum of weight ($A_i = w_0 + w_1 + \dots + w_i$). In parallel prefix-sum with (P_j, P_i) communication, in case of $p = n$ all P_j ($j = i - 2^k; k = 0, 1, \dots, \log_2 p - 1$) send A_j to P_i , and hence all P_i ($i = 2^k$ to $n-1$) compute $A_i = A_i + A_j$ in $O(\log_2 p)$.

Algorithm 4.2: Parallel GH (pGH) algorithm.

```

Parallel Greedy Heuristic (pGH) Algorithm:  $O(n/p \log_2 p)$ 
1. Parallel Sort  $P/W$  (or  $P$ , or  $W$ ) by optimized Bitonic sorting
2. forall  $P_i$  ( $i=0, 1, 2, \dots, p-1$ ) pardo // workload  $w=n/p$  per PE
3.   for  $j=0$  to  $n/p-1$  do // initial local prefix-sum
4.      $l \leftarrow i(n/p)+j$ 
5.     if ( $j=0$ )  $W[l] \leftarrow w[l]$  else  $W[l] \leftarrow w[l]; W[l] \leftarrow W[l]+W[l-1]$ 
6.   end for
7. end forall
8. for  $k=0$  to  $\log_2 p-1$  do // communication ( $\log_2 p$  steps)
9.   forall  $P_i$  ( $i=0, 1, 2, \dots, p-1$ ) pardo
10.    for  $j=0$  to  $n/p-1$  do // local workload  $n/p$  per PE
11.       $l \leftarrow i(n/p)+j$ 
12.      if ( $l \geq (n/p)2^k$ )
13.         $W[l] \leftarrow W[l]+W[(i-2^k+1)n/p-1]$  // 1-data exchange
14.      end if
15.    end for
16.  end forall
17. end for k

18. Parallel SelectObjects:  $O(n/p)$ 
19. forall  $P_i$  ( $i=0, 1, 2, \dots, p-1$ ) pardo
20.   for  $j=0$  to  $n/p-1$  do
21.      $l \leftarrow i(n/p)+j$ 
22.     if ( $W[l] \leq C$ )  $solx[l] \leftarrow '1'$  else  $solx[l] \leftarrow '0'$ 
23.   end for
24. end forall

25. ComputeTotalProfit:  $O(\log_2 p)$ 
26. forall  $P_i$  ( $i=0, 1, 2, \dots, p-1$ ) pardo
27.   for  $j=0$  to  $n/p-1$  do // local  $w=n/p$  per PE
28.      $l \leftarrow i(n/p)+j$ 
29.      $tp[l] \leftarrow tp[l]+(p[l]*solx[l])$ 
30.      $tw[l] \leftarrow tw[l]+(w[l]*solx[l])$ 
31.     if ( $solx[l]='1'$ )  $id[l] \leftarrow l$ 
32.   end for
33.   for  $k=0$  to  $\log_2 p-1$  do // communication ( $\log_2 p$ )
34.     if ( $i2^k=0$  and  $i+2^k < p$ )
35.        $tp[i] \leftarrow tp[i] + tp[i+2^k]$  //  $tp[0]$ : total profit
36.        $tw[i] \leftarrow tw[i] + tw[i+2^k]$  //  $tw[0]$ : total weight
37.        $id[i] \leftarrow \max(id[i], id[i+2^k])$  //  $id[0]$ : max id (or point)
38.     end for k
39.   end forall

```

In case of $p < n$, the efficient communication of our parallel prefix-weight computing (i.e., workload $w = n/p$ but only 1-data exchange) is illustrated in Fig. 12(a), simplified by $p = 8$. After each P_i ($i = 0, 1, 2, \dots, p-1$) computes a prefix $\sum_{i=0}^i x_i w_i$; where $i = 0, 1, 2, \dots, n-1$, it can select object i ($solx_i = 1$) if the prefix $\sum_{i=0}^i x_i w_i \leq C$ (see Fig. 12(b)). Finally, parallel $\sum_{i=0}^{n-1} x_i p_i$ is computed in $\log_2 p$ steps by p PEs. Note that for particular (equal) workload $w = n/p$ (where $n\%p = 0$), the index l of workload for each P_i is $l = i(n/p)+j$ by all P_i ($i = 0, 1, \dots, p-1$), corresponding to local iterations $j = 0, 1, 2, \dots, w-1$.

In addition, for arbitrary workload $n\%p \neq 0$, we introduce the flexible workload w' , defined in Eqs. (4.1)–(4.2).

$$l = i(n/p + 1) + j \quad \text{and } w' = n/p + 1 \quad \text{if } (i < n\%p) \quad (4.1)$$

$$l = i(n/p) + j \quad \text{and } w' = n/p \quad \text{if } (i \geq n\%p) \quad (4.2)$$

Table 5Efficient utilization of p PEs corresponding to C and C' .

C	$C' \sim C/n$	p PEs and efficient time complexity
$C \leq n^2$	$C' \leq n$	$O(1)$ with $p = n$; $O(C'/p)$ with $p < n$
$C \leq n^3$	$C' \leq n^2$	$O(n)$ with $p = n$; $O(1)$ with $p = n^2$; $O(C'/p)$ with $p < n^2$, etc.

4.2. Parallel classification and filtering

In our parallel classification and filtering (Algorithm 4.3), the improved pGH^+ of each feature ($P/W, P, W$) must be computed first on some of unselected objects ($x_i = 0$) by 3 PEs (P_0, P_1, P_2) by applying Algorithm 3.2 in $O(1)$. Then, parallel dynamic weighting (dw_i) and classification for filtering are processed by all P_i in $O(1)$ for $p = n$ and $O(n/p)$ for $p < n$ without any communication. Fig. 13 displays an example of the overall process on $n = 20, p = 4$, and $w = n/p = 5$, similar to Figs. 7–9 ($p = 1$). Then, the remaining n' objects are processed by the parallel MDP (using p PEs) on remaining capacity C' for k' selected objects to fulfill the solution (in Section 4.3).

Algorithm 4.3: Improved pGH^+ and Parallel Filtering.

```

Improved Parallel Greedy Heuristic ( $pGH^+$ ) and Filtering
1. apply Algorithm 3.2 to improve GH1–GH3 by  $P_0 - P_2$  in  $O(1)$ 
2. call ParallelAssignWeight ( $c=5, wx[n]$ ) //  $pGH1^+(P/W)$   $O(n/p)$ 
3. call ParallelAssignWeight ( $c=3, wx[n]$ ) //  $pGH2^+(P)$   $O(n/p)$ 
4. call ParallelAssignWeight ( $c=1, wx[n]$ ) //  $pGH3^+(W)$   $O(n/p)$ 
5. forall  $P_i$  ( $i=0, 1, 2, \dots, p-1$ ) pardo
6.   if ( $i < n\%p$ )  $w' \leftarrow n/p+1$ ; else  $w' \leftarrow n/p$ 
7.   for  $j=0$  to  $w'-1$  do // workload  $w'$  per PE
8.     if ( $i < n\%p$ )  $l \leftarrow i(w'+1)+j$ ; else  $l \leftarrow iw' + (n\%p)+j$ ;
9.      $dw[l] \leftarrow wx_1[l] + wx_2[l] + wx_3[l]$ 
10.    assign group {1,2,3, or 4} to object  $l$ , according to  $dw[l]$ 
11.   end for
12. end forall

13. ParallelAssignWeight( $c, wx[n]$ ) //  $wx_1=5; wx_2=3; wx_3=1$  of  $GH^+1-3$ 
14. forall  $P_i$  ( $i=0, 1, 2, \dots, p-1$ ) pardo
15.   if ( $i < n\%p$ )  $w' \leftarrow n/p+1$ ; else  $w' \leftarrow n/p$ 
16.   for  $j=0$  to  $w'-1$  do // workload  $w'$  per PE
17.     if ( $i < n\%p$ )  $l \leftarrow i(w'+1)+j$ ; else  $l \leftarrow iw' + (n\%p)+j$ ;
18.     if ( $solx[l] \leftarrow '1'$ )  $wx_{GH}[l] \leftarrow c$ ; else  $wx_{GH}[l] \leftarrow 0$  //  $c = 5, 3$ , or  $1$ 
19.   end for
20. end forall

```

4.3. Parallel MDP for remaining solution

Algorithm 4.4 demonstrates the parallel MDP (pMDP) on remaining objects (n'), based on the sequential Algorithm 2.2 (the complete DP). For example, Fig. 14 illustrates the process of pMDP on the remaining objects $n' = 6$; $C' = 17$; $x' = \{3, 5, 7, 12, 13, 19\}$; $P' = \{42, 39, 38, 36, 35, 38\}$, $W' = \{10, 11, 11, 7, 10, 9\}$ by $p = 4$ PEs with workload $w = C'/p$. In parallel, p PEs can process in $O(1)$ for $p = C' \leq n$ (or $O(C'/n)$ if $n < C'$) and $O(C'/p)$ for $p < C'$ by the efficient parallel point-to-point communication (with only 1-data exchange). Time complexity of Algorithm 4.4 is $O(C'/p)$ since remaining $n' \leq 300$ (see Eq. (3.3)). For $p = C'$, the efficient (P_{d-wj}, P_d) communication requires either PRAM or MIN⁺ (i.e., for object j all P_d ($d = w_j$ to C') compute $tp[d-w_j]+p_j$ and update in $tp[d]$ if it is better). For $p < C'$, all P_i check the condition to compute the index $D = iw' + d + 1$ (if $l < C'\%p$); otherwise $D = iw' + C'\%p + d + 1$, similar to index l in Eqs. (4.1)–(4.2) for arbitrary workload ($C'\%p \neq 0$), before updating total profit ($tp[D]$ by P_i) and selected objects that can put into the Knapsack with capacity C' . In general, Table 5 presents efficient utilizing p PEs, according to capacity C and remaining C' in parallel MDP for regular $n < C \leq n^2$ and infrequent cases ($C > n^2$), according to weights of objects i (i.e., $1 \leq w_i \leq n$) for an average weight $\sim n/2$. Thus, the capacity is approximately $C \leq k(n/2)$ or C

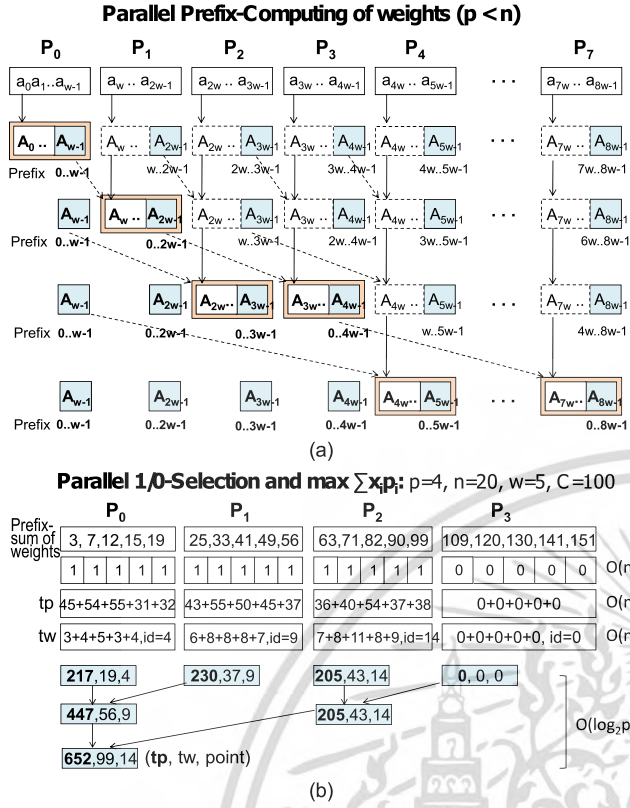


Fig. 12. (a) An efficient communication of parallel prefix-computing of weight (W) for $p < n$ and b) parallel 0/1-selection with prefix $\sum_{i=0}^n x_i w_i \leq C$ and maximized $\sum_{i=0}^n x_i p_i$.

$< n^2$ (for k selected objects) but in case $n/2 \leq w_i \leq n$, the capacity $C < n^3$.

Algorithm 4.4: The parallel MDP.

```

Parallel MDP Algorithm: return ( $C'$ , soltp', solx'):  $O(C'/p)$ 
1. do // repeat for  $k' = |\text{selected objects}|$ 
2.    $w \leftarrow C'/p$  //  $w$ : workload (per PE)
3.   forall  $P_i (i=0 \text{ to } p-1)$  pardo
4.     If  $(i < C'/p)$   $w' \leftarrow w+1$  else  $w' \leftarrow w$ 
5.     for  $d=0$  to  $w'-1$  do
6.       If  $(i < C'/p)$   $D \leftarrow iw' + d+1$  else  $D \leftarrow iw' + C'/p + d+1$ 
7.        $tp[D] \leftarrow 0$ ;  $x[D] \leftarrow -1$ 
8.     end for  $d$ 
9.   end forall
10.  for  $j=0$  to  $n'-1$  do
11.    forall  $P_i (i=0 \text{ to } p-1)$  pardo
12.      If  $(i < C'/p)$   $w' \leftarrow w+1$  else  $w' \leftarrow w$ 
13.      for  $d=w'-1$  down to  $0$  do
14.        if  $(i < C'/p)$   $D \leftarrow iw' + d+1$  else  $D \leftarrow iw' + C'/p + d+1$ 
15.        if  $(D \geq w_j)$  &  $(tp[D-w_j] + p_j) > tp[D]$  then  $x[D] \leftarrow j$ ;  $tp[D] \leftarrow tp[D-w_j] + p_j$ 
16.      end for  $d$ 
17.    end forall
18.  end for  $j$ 
19.   $r \leftarrow x[C']$ ;  $n' \leftarrow r$ ;  $C' \leftarrow C' - w$ ; solx[ $k$ ]  $\leftarrow r$ ;  $k \leftarrow k+1$ 
20.  while  $C' > 0$ 
21.    soltp  $\leftarrow tp[C']$ 

```

5. Algorithm analysis

In Section 3, we design the time-space reduction algorithm to solve the 0/1-KP in efficient time and most of our solutions are optimal. Thus, in Section 5.1 we perform the correctness analysis, based on the nature of data and the simulation study on a variety of datasets. Practically, we expect to find the optimal solutions (100% on the regular datasets and 99% on the irregular datasets i.e., in cases that solutions of the GH3 (W) are better than those of the GH2

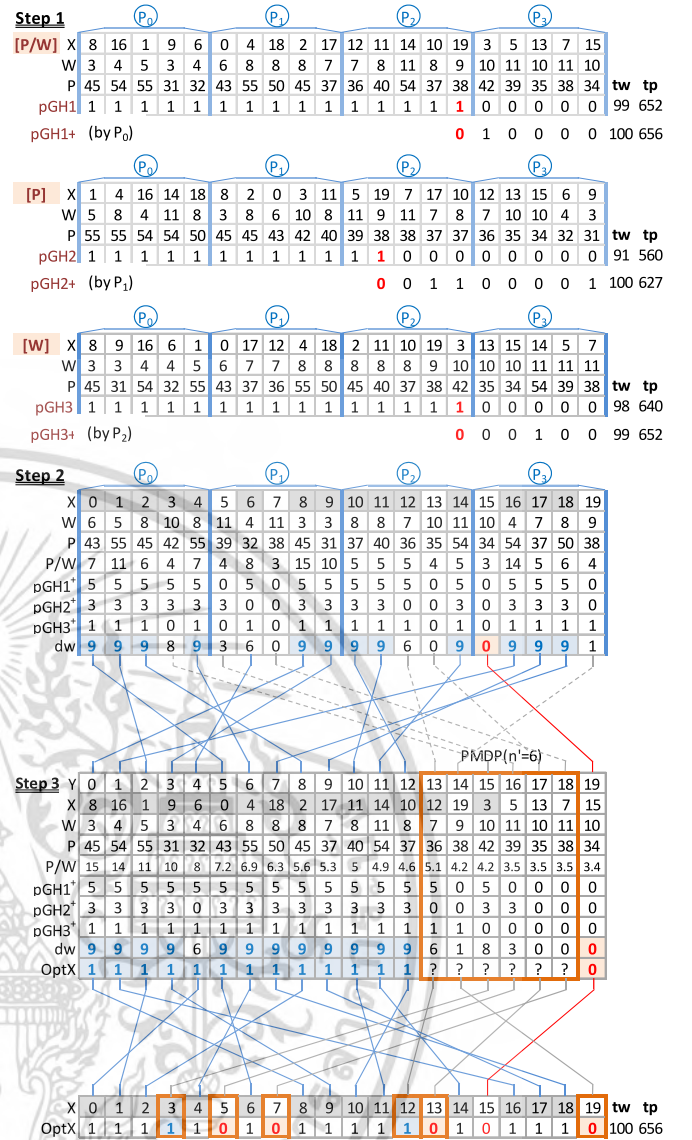


Fig. 13. An example of the overall process of solving the 0/1-KP by our parallel time-space reduction, focusing on unbiased filtering.

(P), etc.). Section 5.2 presents time complexity of our time-space reduction ($TS_{\text{Reduction}}$), the quick GH, and the optimal DP.

5.1. Correctness of $TS_{\text{Reduction}}$ algorithm

The objective of the 0/1-KP is to select the proper (or worth) objects, respected to the maximized total-profit ($\sum_{i=0}^{n-1} p_i x_i$) within a knapsack capacity C ($\sum_{i=0}^{n-1} w_i x_i \leq C$). Obviously, the optimal solutions by the DP algorithm are better than the approximate solutions by the quick GH1 (P/W) algorithm. However, for large n the optimal solutions by the DP cannot be searched in reasonable time, while the quick GH can find the approximate solutions in polynomial time but the solutions may not be optimal, especially in cases of irregular datasets (see Figs. 15–16).

Fig. 15 shows the results of 0/1-KP solving on a benchmark dataset ($n = 5$, $C = 26$) by the quick GH1 (P/W) with total profit = 47 (not optimal). In this case, the DP and our time-space reduction ($TS_{\text{Reduction}}$) can find the optimal solution (see shaded $x_i \{1, 2, 3\}$ with max total profit = 51). Since an object (No. 0) with the profit (P) = 24 (max) and weight (W) = 12 (max) is ranked in

		P_0					P_1					P_2					P_3					
$w_i=10$	($x=3$)	d	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	soltp	
$p_i=42$	$Y=0$	tp	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42	
$w_i=11$	($x=5$)												42	42	42	42	42	42	42	42	42	
$p_i=39$	$Y=1$	tp	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42	
$w_i=11$	($x=7$)												42	42	42	42	42	42	42	42	42	
$p_i=38$	$Y=2$	tp	0	0	0	0	0	0	0	0	0	0	42	42	42	42	42	42	42	42	42	
$w_i=7$	($x=12$)												42	42	42	42	42	42	42	42	42	
$p_i=36$	$Y=3$	tp	0	0	0	0	0	0	0	36	36	36	42	42	42	42	42	42	42	42	78	
$w_i=10$	($x=13$)									36	36	36	42	42	42	42	42	42	42	42	78	
$p_i=35$	$Y=4$	tp	0	0	0	0	0	0	0	36	36	36	42	42	42	42	42	42	42	42	78	
$w_i=9$	($x=19$)									36	36	36	42	42	42	42	42	42	42	42	78	
$p_i=38$	$Y=5$	tp	0	0	0	0	0	0	0	36	36	38	42	42	42	42	42	42	42	74	78	

Fig. 14. An example of $tp[C']$ results in n' iterations by our parallel MDP for the remaining solution ($n' = 6, p = 4$).

X	0	1	2	3	4
W	12	7	11	8	9
P	24	13	23	15	16
P/W	2	1.9	2.1	1.9	1.8

P/W ranking

Y	0	1	2	3	4
X	2	0	3	1	4
W	11	12	8	7	9
P	23	24	15	13	16
P/W	2.1	2	1.9	1.9	1.8

Fig. 15. A benchmark dataset ($n = 5$) with an uncertain object (No. 0) in the beginning of P/W ranking but is not included in the knapsack.

the second best position and it is selected by the GH1 (P/W) but it is not in the optimal solution. In fact, such an object is selected by the GH1 (P/W) and the GH2 (P), except the GH3 (W).

However, this is not the problem in our $TS_{Reduction}$ approach since we use the dw -integration of GH1⁺ – GH3⁺ results ($dw = 0$ to 9) to classify all objects before filtering. Clearly, the worth objects ($dw = 9$) will be selected ($x_i = 1$) and some of least ranking objects ($dw = 0$) will be rejected ($x_i = 0$), while uncertain objects (such as the object No.0) will be verified ($x_i = 0/1$) by the modified DP.

In our correctness analysis, we study the nature of data and benchmark (special) datasets (i.e., Figs. 15–16). Under these special datasets, some data may be “outliers”, as biased P or biased W, that have high ranking but are not selected in the knapsack. For instance, such outliers may be the objects with medium P/W but high profit (P) and high weight (W), or the objects with medium P/W but small weight and small profit, etc. After unbiased filtering, they must be verified carefully by the MDP, see the confirmed results in Tables 9–10.

Note: for simplifying the analysis, let's assume the profit of object (or human) be the merit of human and weight represent the ego of human. Usually, if a human has a maximum profit (or merit) then he/she should have a small weight (or ego) and will be the worth object ($x_i = 1: dw = 9$). In contrast, if a human has the maximum weight (or ego) then he/she should have a small profit (or merit) and should not be in the knapsack ($x_i = 0: dw = 0$).

Fig. 16 illustrates “outlier-spreading” under GH (P/W, P, W) policies, leading to non-optimal solutions. Under our time-space reduction, Filtering 1 ($dw = 9$) for small n (Fig. 17(a)), can remove the outliers (i.e., objects 0, 7, 9, 12, 22, 30, 32, etc. in Fig. 16) to the uncertain group ($n' \leq 200$ objects) to be solved by the MDP in reasonable time. Only if $n' > 200$, we will apply Filtering 2 (Fig. 17(b)), which can manage the outliers by defining the safety margins (α and β) and solved by the MDP. Note: With outliers, the best P/W-ranking of the GH3 cannot find the optimal solution, especially if outliers are close to the top (or best) positions of

Y	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
X	2	4	6	15	1	23	10	3	0	14	31	34	18	32	9	12	7	30	29	22	13	26	20	24	17	21	19	5	8	11	33	25	16	27	28
W	74	70	75	77	71	72	74	75	71	75	74	86	84	75	74	75	84	96	98	83	100	95	93	101	103	93	92	98	91	102	98	101			
P	506	471	502	504	462	463	471	476	450	470	454	496	484	429	420	425	463	411	504	450	505	505	427	503	471	460	499	491	443	438	463	416	434	414	426
P/W	6.8	6.7	6.7	6.5	6.5	6.4	6.4	6.3	6.3	6.3	6.1	5.8	5.8	5.7	5.7	5.7	5.5	5.5	5.4	5.4	5.3	5.2	5.1	5	5	4.9	4.9	4.8	4.8	4.8	4.7	4.6	4.3	4.2	4.2

Fig. 16. A dataset ($n = 35$) with uncertain objects (data No. 0, 7, 9, 12, 22, 30, 32) in the middle of P/W ranking but not in the knapsack.

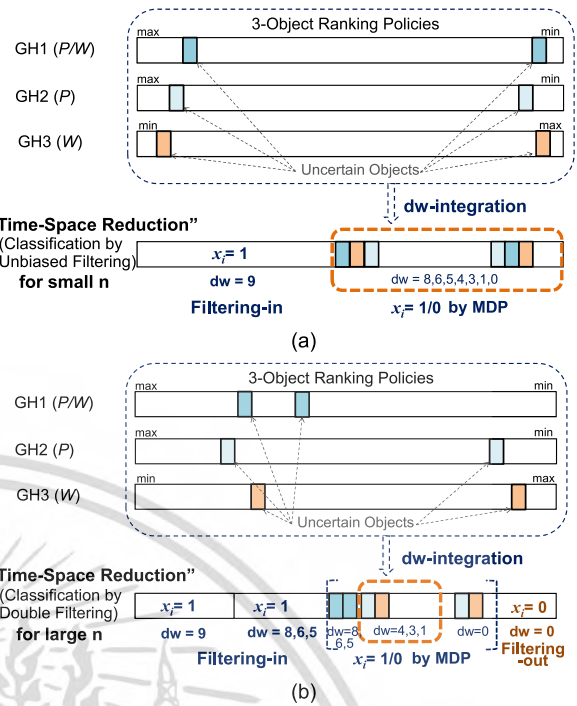


Fig. 17. Diagrams show some outliers occurred in GHs but can be solved by our new $TS_{Reduction}$ approach with proper filtering: (a) Filtering 1 (for small n) and (b) Double filtering (Filtering 1 + Filtering 2 for large n).

P/W-ranking. In our new $TS_{Reduction}$ approach, we introduce the dw -integration and $[\alpha, \beta]$ -safety margins to solve this problem. For example, some objects in group 2 ($dw = 8, 6, 5$) may be outliers, which should be in the α -margin (i.e., the object with $dw = 8$ of biased P (high P-ranking of large P (merit) but also large W (ego)), the object with $dw = 6$ of biased W (high W-ranking of small W (ego) but also small P (merit)), etc.). These outliers must be determined as uncertain objects plus objects in groups 3 ($dw = 4, 3, 1$) and some objects on top of groups 4 ($dw = 0$) in the β -margin. Finally, all of them are solved by the MDP.

In fact, the 0/1-KP decisions (by the existing GH methods and other heuristic algorithms) may cause two types of errors (see Fig. 18).

- (Decision error I) If some objects are selected with $dw = 4, 3, 1$ (at least one GH select $x_i = 1$) but in fact they are not selected ($x_i = 0$) in the optimal solution, or
- (Decision error II) If some objects are not selected ($x_i = 0$) with $dw = 0$ but in fact they must be selected ($x_i = 1$) in the optimal solution.

In our $TS_{Reduction}$ approach, we solve such errors by the dw (dynamic weighting) and the integration of three effective GH⁺ policies (P/W, P, W) plus the $[\alpha, \beta]$ -safety margin in order to remove the outliers into the uncertain group, solved by the MDP. Clearly, under our unbiased filtering, we can move most of uncertain objects and irregular data from group 2 ($dw = 8, 6, 5$) and group 4 ($dw = 0$) into a new group of uncertain objects (n') with the margins α and β , as defined in Eqs. (3.1)–(3.3). Thus, the correctness of our $TS_{Reduction}$ relies on the proper filtering.

Table 6
Completeness and time complexity of 0/1-KP algorithms.

Algorithms	Completeness of solutions			Performance	
	Total weight (<i>soltw</i>)	Total profit (<i>soltp</i>)	Sol. X(<i>solx</i>)	Solution	Time complexity
GH1(<i>P/W</i>)	✓	✓	✓	n/a Opt.	$O(n)$
GH2(<i>P</i>)	✓	✓	✓	n/a Opt.	$O(n)$
GH3(<i>W</i>)	✓	✓	✓	n/a Opt.	$O(n)$
Fast DP	✓	✓	×	Optimal	$O(nC)$
Complete DP	✓	✓	✓	Optimal	$O(knC)$
Proposed $TS_{Reduction}$	✓	✓	✓	Most sols. are Opt.	$O(n+C')$

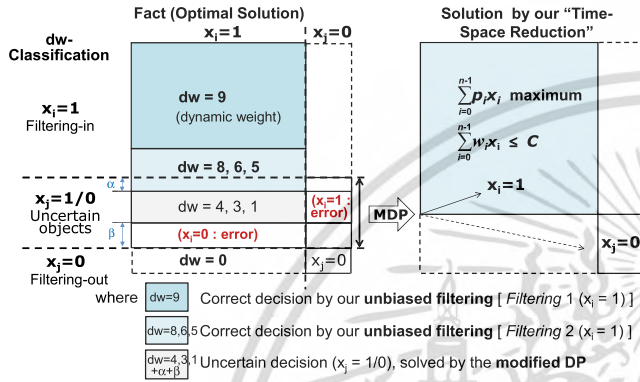


Fig. 18. The 0/1-KP solving by our $TS_{Reduction}$ with unbiased filtering and outlier handling.

Table 7
Time complexity of parallel 0/1-KP algorithms.

Algorithms	Optimal solutions	Time complexity	
		$p = n$	$p < n$
Parallel GH	n/a Opt.	$O((\log_2 n)^2)$	$O(n/p \log_2 p)$
Parallel DP	Optimal	$O(kC)$	$O(knC/p)$
Parallel TS_R	Mots sols. are Opt.	$O((\log_2 n)^2 + C'/n)$	$O(n/p \log_2 p + C'/p)$

By studying the nature of data in most of datasets, there exist three main classes of objects. Let assume objects be a variety of stones for *dw*-classification and filtering.

1. Objects with high profit and light weight or $dw = 9$ (such as pure diamonds) should be selected ($x_i = 1$).
2. Objects with large weight and little profit or $dw = 0$ (i.e., heavy boulders (or rocks)) might not be selected ($x_i = 0$).
3. Objects with medium profit (*P*) and weight (*W*).
 - 3.1 If both *P* and *W* are small, *W*-biased may cause $dw = 6$.
 - 3.2 If both *P* and *W* are large, *P*-biased may cause $dw = 8$.

Otherwise, objects will have $dw = 8, 6, 5, 4, 3$, or 1.

5.2. Time complexity of $TS_{Reduction}$ algorithm

In this section, we present time complexity of our $TS_{Reduction}$ algorithm, compared to those of the quick GH and the optimal DP, illustrated in Tables 6, 8 (sequential algorithms) and 7 (parallel algorithms). The fast GH1-GH3 can find the total profit, the total weight, and the solution set *X* in $O(n)$ but results may not be optimal. In optimal DP, there are two versions: 1. the basic DP and 2. the efficient DP, including the fast DP (with *soltw*, *soltw*) and the complete DP with the full solution (*soltw*, *soltw*, *solx*).

Time complexity of the fast DP with a single array (for *soltw* and *soltw*) is $O(nC)$, used as a standard DP to compare with our optimal

Table 8
Strength and weakness of three 0/1-KP approaches.

Algorithms	Methodology	Strength	Weakness
GH1(<i>P/W</i>)	Best ranking by combining <i>P/W</i>	– Fast finding initial solutions in $O(n)$.	– Not guarantee the optimal solutions.
GH2(<i>P</i>)	Ranking by <i>P</i> (profit)	– Fast finding initial solutions in $O(n)$.	– Not guarantee the optimal solutions.
GH3(<i>W</i>)	Raking by <i>W</i> (weight)	– Fast finding initial solutions in $O(n)$.	– Not guarantee the optimal solutions.
Basic DP	(Simple) Dynamic Programming (using 2-D array).	– Found the optimal solutions in $O(nC)$. – Get the full solution (<i>solx</i> , <i>soltw</i> , <i>soltw</i>).	– Use 2-D array ($n \times C$) that cannot be simply implemented for large <i>n</i> and massive $C \leq n^2$.
Fast DP	(Improved) Dynamic Programming (on single array).	– Found the optimal solutions in $O(nC)$. – Save space with a single 1-D array.	– Compute only total profit (<i>soltw</i>) and total weight (<i>soltw</i>).
Complete DP	(Completed) Dynamic Programming of the fast DP.	– Found the optimal solutions. – Save space with a single 1-D array. – Get the full solution (<i>solx</i> , <i>soltw</i> , <i>soltw</i>).	– After compute total profit (<i>soltw</i>), repeat process for the full solution (<i>solx</i> , <i>soltw</i>) in $O(knC)$, where $k = \text{selected objects} < n$.
Our Time-space reduction ($TS_{Reduction}$)	Integration of 1. the (fast) GH (<i>P/W</i> , <i>P</i> , <i>W</i>), 2. our unbiased filtering (to reduce <i>n</i>), and 3. the (optimal) DP on small <i>n'</i> .	– Found good solutions in $O(n+C')$ and most of solutions are optimal. – Save space with a single array. – Get the full solution (<i>solx</i> , <i>soltw</i> , <i>soltw</i>). – Combine the fast GH on large <i>n</i> and optimal DP on small <i>n'</i> .	–

solutions, while the complete DP yields the full solution (including *solx*, *soltw*, *soltw*) with time complexity $O(knC)$. However, all of DPs are non-polynomial time for large *n*, *k* ($< n$), and massive *C* ($n < C < n^d$). In this study, we present the 0/1-KP solving in $O(n + C')$ by our time-space reduction ($TS_{Reduction}$), including 1. the GH1-GH3 for initial solutions in $O(n)$, 2. the filtering for the (time-space) reduction in $O(n)$, and 3. the MDP for the remaining objects (*n'*) in $O(C')$, $C' \ll C$. For our parallel 0/1-KP, all functions of the $TS_{Reduction}$ algorithm is designed with efficient communication ($\log_2 p$), especially the parallel 0/1-object-selection (in Step 1) by the parallel prefix-weight computing $\sum_{i=0}^n x_i w_i$ in $O(\log_2 n)$ for $p = n$ and $O(n/p \log_2 p)$ when $p < n$. In Step 2, our parallel filtering is $O(n/p)$ without any communication and Step 3 (parallel MDP) is $O(C'/p)$ with efficient parallel point-to-point communication.

Table 9
Experimental results (total profits) of benchmark datasets.

Datasets: n	0/1-KP algorithms				
	GH1(P/W)	GH2(P)	GH3(W)	Fast DP	TS _{Reduction}
Dataset #1:5	47	47	44	51	51
Dataset #2:10	266	179	309	309	309
Dataset #3:15	1302	1315	1249	1458	1458
Dataset #14:100	10592	9788	8166	10785	10785
Dataset #18:500	81254	75326	72106	81522	81522
Dataset #22:900	135320	124014	127496	135632	135632
Dataset #23:1000	82996	82512	82996	83022	83022
Dataset #26:4000	75544	45586	59874	75606	75606
Dataset #31:9000	84723	79944	60504	84897	84897

6. Experimental results

Experiments for the 0/1-KP solving in this study were processed on a number of datasets. The first and second classes of datasets were standard (or benchmark) datasets from available resources [10,11] and the third class of datasets were randomly generated by simulation study.

(1) Standard Datasets (Class 1)

The standard dataset (class 1) was downloaded from [11], including three datasets namely Dataset No. 1 ($n = 5$), Dataset No. 2 ($n = 10$) and Dataset No. 3 ($n = 15$).

Dataset No. 1 ($n = 5$ objects): $C = 26$

$W = \{12, 7, 11, 8, 9\}$

$P = \{24, 13, 23, 15, 16\}$

Dataset No. 2 ($n = 10$ objects): $C = 165$

$W = \{23, 31, 29, 44, 53, 38, 63, 85, 89, 82\}$

$P = \{92, 57, 49, 68, 60, 43, 67, 84, 87, 72\}$

Dataset No. 3 ($n = 15$ objects): $C = 750$

$W = \{70, 73, 77, 80, 82, 87, 90, 94, 98, 106, 110, 113, 115, 118, 120\}$

$P = \{135, 139, 149, 150, 156, 163, 173, 184, 192, 201, 210, 214, 221, 229, 240\}$

(2) Standard Datasets (Class 2)

The standard dataset (class 2), other benchmark datasets, were downloaded from [10], see observed results in Table 9.

(3) Generated Datasets (Class 3)

Random datasets were generated using our generator for a variety of problem sizes (n), such as $5 \leq n \leq 30000$ (for $p = 1$), see results in Tables 10–12 and Fig. 19. For parallel performance ($n > 30000$, p PEs), Table 13 shows the speedup on a big data $n = 5000000$ by $p = 2, 4, 6, \dots, 4096$ PEs and Fig. 20 displays the corresponding efficiency ($2 \leq p \leq 8192$).

In performance evaluation, we evaluate not only the correctness of solutions but also the runtime performance (i.e., response time ($p = 1$) and speedup ($p > 1$)). The results of our TS_{Reduction} are compared with the optimal solutions (*soltp* (max total profit)) of the fast DP. A number of experiments have been investigated (by simulation study) on a variety of problem sizes ($n = 5, 6, 7, 8, 9, 10, \dots, 29999, 30000$) to verify the correctness. Moreover, for each special (or irregular) dataset, five different samples have been repeated to confirm the correctness. In regular datasets, the GH1(P/W) yielded the best solutions among three policies. Otherwise, either the GH3(W) or the GH2(P) yielded better results, and hence we call irregular datasets in this study.

In the correctness evaluation, Tables 10 and 11 display some of our experimental results ($5 \leq n \leq 30000$), including irregular

Table 10
Experimental results (total profits) of generated datasets.

Datasets	0/1-KP algorithms				
	GH1(P/W)	GH2(P)	GH3(W)	Fast DP	TS _{Reduction}
$n = 20$	614	560	590	621	621
$n = 35$	6638	5975	6410	6768	6768
$n = 50$	7907	7734	6833	7907	7907
$n = 200$	26086	24899	21674	26091	26091
$n = 400$	41220	41266	32500	41404	41404
$n = 600$	71886	68929	69375	71968	71968
$n = 1000$	82996	42512	82996	83022	83022
$n = 2000$	376785	281510	343765	376926	376926
$n = 3000$	84462	64261	75635	84493	84493
$n = 4000$	75544	45586	59782	75606	75606
$n = 5000$	440412	344114	417261	440653	440653
$n = 6000$	412266	360074	366835	412465	412465
$n = 7000$	212589	154512	175217	212816	212816
$n = 8000$	288179	144014	272014	288228	288228
$n = 9000$	84723	79768	60176	84897	84897
$n = 10000$	211869	157430	196542	211929	211929
$n = 20000$	480954	306778	433961	480974	480974
$n = 30000$	1553299	1208633	1510874	1553414	1553414

Table 11
Experimental results (total weights) of generated datasets.

Datasets	C	0/1-KP Algorithms				
		GH1(P/W)	GH2(P)	GH3(W)	Fast DP	TS _{Reduction}
$n = 20$	200	191	195	186	200	200
$n = 35$	1100	1053	1062	1030	1096	1096
$n = 50$	1500	1496	1475	1442	1496	1496
$n = 200$	6582	6577	6539	6522	6582	6582
$n = 400$	15800	15709	15747	15712	15800	15800
$n = 600$	25520	25478	25477	25461	25520	25520
$n = 1000$	12822	12796	12812	12796	12822	12822
$n = 2000$	25200	25182	25176	25164	25200	25200
$n = 3000$	10750	10745	10732	10739	10750	10750
$n = 4000$	11350	11338	11337	11345	11350	11350
$n = 5000$	65000	64953	64975	64981	65000	65000
$n = 6000$	56500	56464	56491	56500	56500	56500
$n = 7000$	27060	27024	27060	27057	27060	27060
$n = 8000$	21840	21835	21769	21829	21840	21840
$n = 9000$	42200	42108	42128	42198	42200	42200
$n = 10000$	35975	35962	35970	35938	35975	35975
$n = 20000$	75200	75196	75197	75151	75200	75200
$n = 30000$	102500	102490	102498	102477	102500	102500

Table 12
Response time (seconds) of three 0/1-KP approaches.

n	GH (P/W)	Fast DP (<i>soltp</i> , <i>soltw</i>)	Complete DP (<i>soltp</i> , <i>soltw</i> , <i>solx</i>)	Proposed TS _{Reduction}
500	0.002	0.051	5.091	0.015
1000	0.007	0.089	33.16	0.031
2000	0.013	0.374	10.76	0.015
3000	0.018	0.283	43.33	0.031
4000	0.040	0.267	30.36	0.053
5000	0.043	2.111	222.45	0.084
6000	0.077	2.077	102.04	0.115
7000	0.100	1.118	101.99	0.153
8000	0.115	1.036	314.03	0.200
9000	0.146	2.180	196.38	0.246
10000	0.181	2.440	2261.65	0.284
20000	0.697	9.028	3522.29	1.130
30000	1.551	11.952	58675.84	2.409

datasets. In this experiment, all of our solutions (*soltp*, *soltw*, and *solx*) are optimal by verified with the optimal solutions (*soltp* and *soltw*) of the fast DP.

Table 12 shows the response time of three approaches, where the quick GH yielded the minimum response time but most of solutions are not optimal, while our TS_{Reduction} yielded the reasonable runtime and most of our solutions are optimal. For the optimal study of the DP processing, the fast DP consumed more runtime, according to $O(nC)$ complexity and the complete DP was very time

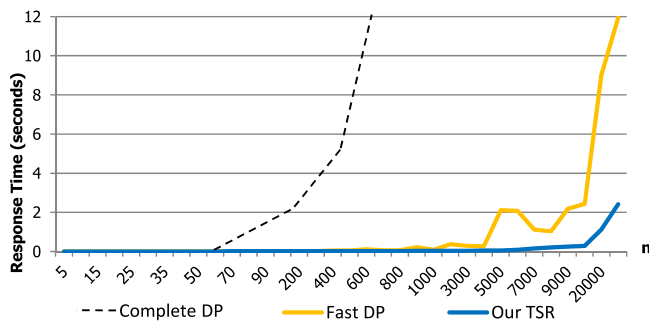


Fig. 19. The response time of our efficient $TS_{\text{Reduction}}$ and the fast DP on $n = 5$ to $n = 30000$.

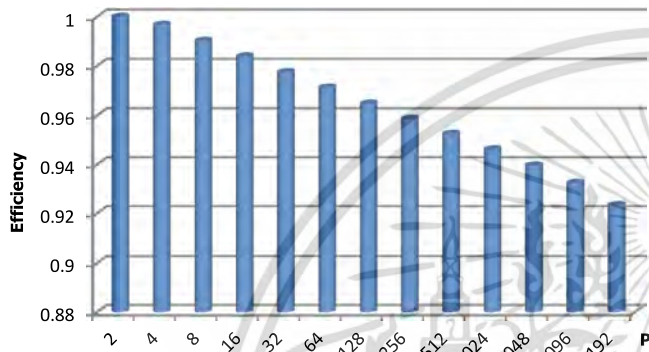


Fig. 20. The significant efficiency of our parallel $TS_{\text{Reduction}}$ ($E_p > 0.9$) on large $n = 5000000$ by $p = 2, 4, 8, 16, \dots, 8192$ PEs.

consuming because of its $O(knC)$ complexity. Fig. 19 displays the efficient response time of our $TS_{\text{Reduction}}$, which outperformed not only the complete DP but also the fast DP. Clearly, the complete DP is appropriate for small n , since it guarantee the optimal solutions (i.e., < 30 s for $n < 4000$), see Table 12. But for large n (especially $n > 10000$), the response time of our $TS_{\text{Reduction}}$ is much better (i.e., 2.4 s for $n = 30000$) and in this experiment most of our solutions are optimal.

Practically, for big data n (i.e., $n > 30000$) the parallel $TS_{\text{Reduction}}$ was evaluated in terms of speedup and efficiency. The speedup is $S_p = T_1/T_p$ and the efficiency is $E_p = S_p/p$, where T_1 is runtime of sequential $TS_{\text{Reduction}}$ and T_p is runtime of executing $pTS_{\text{Reduction}}$ (p PEs). Table 13 displays increasing speedup and significant efficiency of the parallel 0/1-KP reduction ($pTS_{\text{Reduction}}$) by utilizing $p = 2, 4, 8, 16, \dots, 4096$ PEs on a large problem size $n = 5000000$, where in cases of small p (i.e., $p = 2, 4, 8, 16$) the results are close to ideal cases ($S_p = p$ and $E_p = 1$). In case of $16 < p \leq 512$, the efficiency of our $pTS_{\text{Reduction}}$ is $0.95 < E_p \leq 0.98$ and for $512 < p \leq 4096$, $0.93 < E_p \leq 0.95$, according to our efficient communication.

7. Conclusion

This paper presents the unbiased filtering for time-space reduction and the efficient parallel time-space reduction for solving the 0/1-KP, an NP-hard problem, in polynomial time. In our innovation and contribution, we integrated the feature-sorting (FS) of heuristic strategy (to find the best initial solutions), the unbiased filtering (for the time-space reduction), and the modified DP (dynamic programming) for solving the remaining (critical) objects (n') and capacity C' . Our key contribution relies on the unbiased filtering for the triple-right commitment (right man, right place, and right time), which aims to process as fast as the FS and to find the optimal solutions as the DP. In parallel time-space reduction, we achieved

Table 13

Speedup and efficiency on $n = 5000000$ by $p \leq 4096$.

Coarse-grained processing			Medium-grained processing		
p	S_p	E_p	p	S_p	E_p
2	2.00	1.0000	128	123.51	0.9649
4	3.99	0.9967	256	245.44	0.9587
8	7.92	0.9902	512	487.71	0.9526
16	15.74	0.9838	1024	968.98	0.9463
32	31.28	0.9774	2048	1924.46	0.9397
64	62.15	0.9711	4096	3818.93	0.9324

the optimized time complexity $O(n/p \log_2 p + C'/p)$, according to efficient communication. The performance was evaluated by simulation study, compared with the optimal solutions (max total profits) of the fast DP. In this experiment, most of our solutions ($n \leq 30000$) were optimal and the benchmark datasets were investigated to confirm the correctness. In parallel (for big data n), our parallel $TS_{\text{Reduction}}$ yielded the remarkable speedup as well as efficiency on $p = 2, 4, 8, 16, 32, 64, 128, \dots, 1024, 2048, 4096$ PEs, where $0.93p < S_p \leq p$ and $0.93 < E_p \leq 1$.

In our future study, we will apply the idea of our unbiased filtering for solving other NP-hard problems with expecting high precision in achieving the optimal solutions in polynomial time.

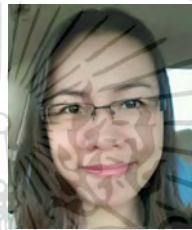
Acknowledgments

The authors want to be deeply thankful to the faculty of Science, King Mongkut's Institute of Technology Ladkrabang (KMUTL), Bangkok, Thailand for the research grant to Miss Patcharin Buayen, a doctoral student, on her tuition fee during 2013–2015, and gratefully thank the “Put-ta-tum” Buddhism-handbook by Phra-phrom Kunaporn (P.A. Pa-yut-tao) for fulfilling our best attitude in positive thinking and decision-making, and lastly glorious honor of our beloved King Bhumibol Adulyadej for the great inspiration in this optimistic study, especially being able to apply his philosophy of sufficient economy for our better life.

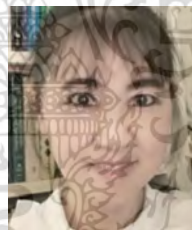
References

- [1] S.J. Ahmad, V.S.K. Reddy, A. Damodaram, P.R. Krishna, Delay optimization using knapsack algorithm for multimedia traffic over MANET, *Eur. J. Exp. Syst. Appl.* 42 (2015) 6819–6827.
- [2] F. Aisopos, K. Tserpes, T. Varvarigou, Resource management in software as a service using the knapsack problem model, *Int. J. Prod. Econ.* 141 (2013) 465–477.
- [3] J.C. Bansal, K. Deep, A modified binary particle swarm optimization for knapsack problems, *Appl. Math. Comput.* 218 (2012) 11042–11061.
- [4] E. Bas, A capital budgeting problem for preventing workplace mobbing by using analysis hierarchy process and fuzzy 0-1 bidimensional knapsack model, *Exp. Syst. Appl.* 38 (2011) 12415–12422.
- [5] K.K. Bhattacharjee, S.P. Sarmah, Shuffled frog leaping algorithm and its application to 0/1-KP, *Appl. Soft Comput.* 19 (2014) 252–263.
- [6] J.M. Calvin, J.Y.-T. Leung, Average-case analysis of a greedy algorithm for the 0/1-KP, *Oper. Res. Lett.* 31 (2003) 202–210.
- [7] V.C.B. Camargo, L. Mattioli, F.M.B. Toledo, A knapsack problem as a tool to solve the production planning problem in small foundries, *J. Comput. Oper. Res.* 39 (2012) 86–92.
- [8] C. Changdar, G.S. Mahapatra, R.K. Pal, An ant colony optimization approach for binary knapsack problem under fuzziness, *Appl. Math. Comput.* 223 (2013) 243–253.
- [9] A.S. da Cunha, L. Bahiense, A. Lucena, C.C. de Souza, A new lagrangian based branch and bound algorithm for the 0-1 knapsack problem, *Elect. Note. Discret. Math.* 36 (2010) 623–630.
- [10] [dataset] CERMSEM. <http://cermsem.univ-paris1.fr/pub/CERMSEM/hihi/KSP>.
- [11] [dataset] Free Software Foundation, Inc. Data for the 01 Knapsack Problem. http://people.sc.fsu.edu/~jburkardt/datasets/knapsack_01/knapsack_01.html.
- [12] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco, 1979.
- [13] M. Gupta, A fast and efficient genetic algorithm to solve 0-1 knapsack problem, *Int. J. Digit. Appl. Contemp. Res.* 6 (2013).

- [14] C.V. Henkel, T. Bäck, J.N. Kok, DNA computing of solution to knapsack problems, *Biosystems* 88 (2007) 156–162.
- [15] D. Jiménez-González, J. Navarro, J.-L. Larriba-Pey, The Effect of local sort on parallel sorting algorithms, in: *proceedings of Euromicro Workshop on Parallel, Distributed and Network-based Processing*, Canary Islands, Spain, 2002, pp. 360–367.
- [16] F. Jolai, M.J. Rezaee, M. Rabbani, J. Razmi, P. Fattahi, Exact algorithm for bi-objective 0-1 knapsack problem, *Appl. Math. Comput.* 194 (2007) 544–551.
- [17] H. Kellerer, U. Pferschy, D. Pisinger, *Knapsack Problem*, Springer-Verlag Berlin, New York, 2004.
- [18] X. Kong, L. Gao, H. Ouyang, S. Li, A simplified binary harmony search algorithm for large scale 0-1 knapsack problem, *Exp. Syst. Appl.* 42 (2015) 5337–5355.
- [19] T.Y. Lim, M.A. Al-Betar, A.T. Khader, Taming the wild with 0/1-KP monogamous pairs genetic algorithm, *Exp. Syst. Appl.* 54 (2016) 241–250.
- [20] J. Lv, X. Wang, M. Huang, H. Cheng, F. Li, Solving 0-1 knapsack problem by greedy degree and expectation efficiency, *Appl. Soft Comput.* 41 (2016) 94–103.
- [21] S. Martello, P. Toth, *Knapsack Problems: Algorithms and Computer Implementation*, J. Wiley & Sons, Chichester, 1990.
- [22] R.C. Merkle, M.E. Hellman, Hiding information and signatures in trapdoor knapsacks, *IEEE Trans. Inform. Theory* IT-24 (1978) 525–530.
- [23] M. Monaci, U. Pferschy, P. Serafini, Exact solution of the robust knapsack problem, *Comput. Oper. Res.* 40 (2013) 2625–2631.
- [24] N. Moosavian, Soccer league competition algorithm for solving knapsack problem, *Swarm Evol. Comput.* 20 (2015) 14–22.
- [25] R.S. Pavithr, Gursaran Quantum inspired social evolution (QSE) algorithm for 0-1 knapsack problem, *Swarm Evol. Comput.* 29 (2016) 33–46.
- [26] R. Petagon, J. Werapun, Embedding the optimal all-to-all personalized exchange on multistage interconnection networks, *J. Parallel Distrib. Comput.* 88 (2016) 15–30.
- [27] X. Qiu, W. Kern, Improved approximation algorithm for a bilevel knapsack problem, *Theoret. Comput. Sci.* 595 (2015) 120–129.
- [28] A. Rong, J.R. Figueira, K. Klamroth, Dynamic programming based algorithms for the discounted {0-1} knapsack problem, *Appl. Math. Comput.* 218 (2012) 6921–6933.
- [29] A. Rong, J.R. Figueira, M.V. Pato, A two state reduction based dynamic programming algorithm for the bi-objective 0-1 knapsack problem, *Comput. Math. Appl.* 62 (2011) 2913–2930.
- [30] A. Rong, J.R. Figueira, M.V. Pato, Dynamic programming algorithm for the bi-objective integer knapsack problem, *European J. Oper. Res.* 236 (2014) 85–99.
- [31] R.P. Roederkerk, H.J. van Herte, Robust optimization of the 0-1 knapsack problem: balancing risk and return in assortment optimization, *European J. Oper. Res.* 250 (2016) 842–854.
- [32] C. Sachdeva, S. Goel, An improved approach for solving 0-1 knapsack problem in polynomial time using genetic algorithms, in: *IEEE International Conference on Recent Advances and Innovations in Engineering*, 2014.
- [33] T. Thanakulwarapas, J. Werapun, An optimized Bitonic sorting strategy with midpoint-based dynamic communication, *J. Parallel Distrib. Comput.* 84 (2015) 37–50.
- [34] T.K. Truong, K. Li, Y. Xu, Chemical reaction optimization with greedy strategy for the 0-1 knapsack problem, *Appl. Soft Comput.* 13 (2013) 1774–1780.
- [35] D.C. Vanderster, N.J. Dimopoulos, R. Parra-Hernandez, R.J. Sobie, Resource allocation on computations grids using a utility model and the knapsack problem, *Futur. Gen. Comput. Syst.* 25 (2009) 35–50.
- [36] E. Wang, Y. Yang, J. Wu, A Knapsack-based buffer management strategy for delay-tolerant networks, *J. Parallel Distrib. Comput.* 86 (2015) 1–15.
- [37] W. Wedashwara, S. Mabu, M. Obayashi, T. Kuremoto, Combination of genetic network programming and knapsack problem to support record clustering on distributed databases, *Exp. Syst. Appl.* 46 (2016) 15–23.
- [38] J. Zhang, Comparative study of several intelligent algorithms for knapsack problem, *Proced. Environ. Sci.* 11 (2011) 163–168.
- [39] X. Zhang, S. Huang, Y. Hu, Y. Zhang, S. Mahadevan, Y. Deng, Solving 0-1 knapsack problem based on amoeboid organism algorithm, *Appl. Math. Comput.* 219 (2013) 9959–9970.
- [40] J. Zhao, F. Pang, T. Huang, Y. Liu, Genetic algorithm based on greedy strategy in the 0-1 knapsack problem, in: *IEEE 3rd International Conference on Genetic and Evolutionary Computing*, 2009.
- [41] Y. Zhou, X. Chen, Guo Zhou, An improved monkey algorithm for 0-1 knapsack problem, *Appl. Soft Comput.* 38 (2016) 817–830.
- [42] D. Zou, L. Gao, S. Li, J. Wu, Solving 0-1 knapsack problem by novel global harmony search algorithm, *Appl. Soft Comput.* 11 (2011) 1556–1564.



Patcharin Buayen received the B.S. degree in Computer Science in 1999 from Lampang Rajabhat University, Lampang, Thailand. She received the M.S. degree in Computer Science in 2008 from King Mongkut's Institute of Technology Ladkrabang (KMUTL), Bangkok, Thailand and received the excellent science award from the foundation of Professor Dr. Thae Nilanithi, faculty of Science, Chulalongkorn University, Bangkok, Thailand. Since 2011, she has been a lecturer of Business Computer at Rambhi Barni Rajabhat University, Chantaburi, Thailand. Miss Buayen is a Ph.D. candidate (in Computer Science), at KMUTL, Bangkok, Thailand. Her research interests include data structures and computer algorithms, NP-hard problems, and parallel algorithms.



Jeeraporn Werapun received the B.S. degree from Kasetsart University and the M.S. degree from Chulalongkorn University, Bangkok, Thailand. She received the M.S. degree in Computer Science in 1993 and the D.Sc. degree in Computer Engineering (Parallel and Distributed Systems) in 1999 from the George Washington University, Washington, D.C., U.S.A. Dr. Werapun is an Associate Professor of Computer Science at King Mongkut's Institute of Technology, Ladkrabang (KMUTL), Bangkok, Thailand. Her research interests include parallel algorithms, parallel architectures, parallel and distributed computing, and multistage interconnection networks (MINs).

ภาคผนวก ข

ผลงานวิจัยตีพิมพ์ที่ Algorithms

Journal homepage : <https://www.mdpi.com/journal/algorithms>

P. Buayen, J. Werapun, “Efficient 0/1-Multiple-Knapsack Problem Solving by Hybrid DP Transformation and Robust Unbiased Filtering.” *Algorithms*. 15 (2022), 366, 1-34.
<https://doi.org/10.3390/a15100366>



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Article

Efficient 0/1-Multiple-Knapsack Problem Solving by Hybrid DP Transformation and Robust Unbiased Filtering

Patcharin Buayen *  and Jeeraporn Werapun

Department of Computer Science, Faculty of Science, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand

* Correspondence: 56605013@kmitl.ac.th

Abstract: The multiple knapsack problem (0/1-mKP) is a valuable NP-hard problem involved in many science-and-engineering applications. In current research, there exist two main approaches: 1. the exact algorithms for the optimal solutions (i.e., branch-and-bound, dynamic programming (DP), etc.) and 2. the approximate algorithms in polynomial time (i.e., Genetic algorithm, swarm optimization, etc.). In the past, the exact-DP could find the optimal solutions of the 0/1-KP (one knapsack, n objects) in $O(nC)$. For large n and massive C , the unbiased filtering was incorporated with the exact-DP to solve the 0/1-KP in $O(n + C')$ with 95% optimal solutions. For the complex 0/1-mKP (m knapsacks) in this study, we propose a novel research track with hybrid integration of DP-transformation (DPT), exact-fit (best) knapsack order ($m!$ -to- m^2 reduction), and robust unbiased filtering. First, the efficient DPT algorithm is proposed to find the optimal solutions for each knapsack in $O([n^2, nC])$. Next, all knapsacks are fulfilled by the exact-fit (best) knapsack order in $O(m^2[n^2, nC])$ over $O(m![n^2, nC])$ while retaining at least 99% optimal solutions as $m!$ orders. Finally, robust unbiased filtering is incorporated to solve the 0/1-mKP in $O(m^2n)$. In experiments, our efficient 0/1-mKP reduction confirmed 99% optimal solutions on random and benchmark datasets ($n \leq 10,000$, $m \leq 100$).

Keywords: multiple 0/1-knapsack problem (0/1-mKP); efficient NP-hard problem solving; exact-DP transformation; exact-fit (best) knapsack order; robust unbiased filtering



Citation: Buayen, P.; Werapun, J. Efficient 0/1-Multiple-Knapsack Problem Solving by Hybrid DP Transformation and Robust Unbiased Filtering. *Algorithms* **2022**, *15*, 366. <https://doi.org/10.3390/a15100366>

Academic Editor: Jesper Jansson

Received: 16 August 2022

Accepted: 27 September 2022

Published: 30 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Presently, a variety of NP-hard problems are involved in many real-world applications and AI computing. Solving specific NP-hard problems (with high performance in efficient time) for those applications is challenging. Some of the interesting NP-hard problems are the 0/1-KP (knapsack problem), the 0/1-mKP (multiple m knapsacks), etc.

Formally, the 0/1-KP is defined as follows: Consider a set of n objects and a knapsack capacity C , where each object j ($=0, 1, \dots, n-1$) has profit p_j and weight w_j .

The objective of the 0/1-KP is to select some objects for the maximum total profit kept in the knapsack that cannot exceed the knapsack capacity (C), defined in Equations (1)–(3). Recently (2018), unbiased filtering [1] was proposed (for the 0/1-KP) to select outstanding objects (from n objects) before applying the exact DP (dynamic programming) algorithm on small remaining n' (≤ 200) in efficient time for most optimal solutions (at least 95%) of regular and irregular datasets.

$$\text{Maximize } \sum_{j=0}^{n-1} p_j x_j \quad (1)$$

$$\text{Subject to } \sum_{j=0}^{n-1} w_j x_j \leq C \quad (2)$$

$$\text{and } x_j \in \{0, 1\}; j = 0, 1, 2, \dots, n-1 \quad (3)$$

For the complex 0/1-mKP (m knapsacks), the objective is to select some objects for multiple knapsacks (each selected object j ($x_{ij} = 1$) in a proper knapsack i) that cannot exceed

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

For the complex 0/1-mKP (m knapsacks), the objective is to select some objects for multiple knapsacks (each selected object j ($x_{ij} = 1$) in a proper knapsack i) that cannot exceed each of the knapsack capacities ($C_i; i = 1, 2, 3, \dots, m$), see Figure 1 (one and m knapsacks), for the maximized total profit, defined in Equations (4)–(6).

Maximize $\sum_{i=1}^m \sum_{j=0}^{n-1} p_j x_{ij}$ (4) (4)

Subject to $\sum_{j=0}^{n-1} w_j x_{ij} \leq C_i; i = 1, 2, \dots, m; j = 0, 1, 2, \dots, n-1$ (5) (5)

and $\sum_{i=1}^m x_{ij} \leq 1; x_{ij} \in \{0, 1\}; \forall i, j$ (6) (6)

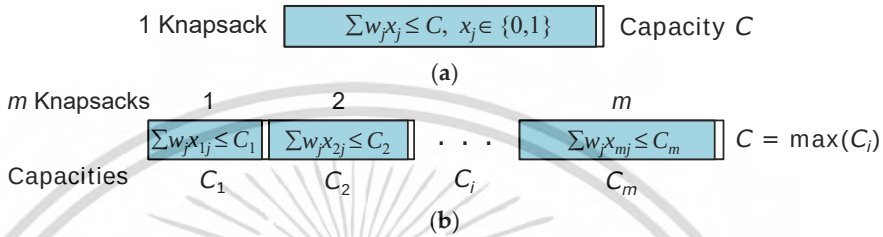


Figure 1. Figure 1 illustrates the difference between a simple 0/1-KP (one knapsack with capacity C) and (b) the complex 0/1-mKP (m knapsacks with capacity $C_i; i = 1, 2, \dots, m$).

The 0/1-KP and 0/1-mKP are constructive for science and engineering applications, such as resource allocation [2,3], capital budgeting [4], production planning [5], multicriteria scheduling [6], risk balancing and asset optimization [7]. Other applications in network systems [8–11], etc. However, finding the optimal solution of the 0/1-mKP is much harder than that of the 0/1-KP since each selected object must specify a proper knapsack capacity ($C_i; i = 1, \dots, m$) from the available knapsacks.

In popular 0/1-KP research, two approaches are extensively studied: the exact approach for optimal solutions (but in exponential time) and the fast approximate approach (but the optimal solution may not be found). In the optimal solution of the 0/1-KP can be computed by DP (dynamic programming) algorithms in $O(nC)$ [11–15], or BnB (branch and bound) [16] and backtracking [17] algorithms in exponential time ($O(2^n)$). In practice, approximate methods (i.e., greedy methods [18,19], kernel search [20,21], genetic algorithms [22,23], swarm optimization [24–26], hybrid methods [29–32], hyper-heuristic method [33], etc.) can find the good solutions in polynomial time. Recently, the time-space reduction algorithm [1] was proposed to solve the 0/1-KP in $O(n \log n)$ for large n by unbiased filtering to preselect the outstanding objects (from n objects) before applying the exact DP algorithm on remaining n' and C' ($n' \leq 200$, $C' \leq 200$, and massive C may not be a polynomial bound of n), which could find most optimal solutions (at least 95%) in experiments.

In current KP-researches, a variety of 0/1-KPs have been studied, including the multiple KP (0/1-mKP) [34–36], no shared x_{ij} in m knapsacks ($\sum_{j=0}^{n-1} w_j x_{ij} \leq C_i$), the multidimensional KP (0/1-MKP) [37–41], $\sum_{j=0}^{n-1} w_j x_{ij} \leq C_i$ shared x_{ij} in m knapsacks, and the multidimensional multiple-choice KP (MCMKP) [42]. However, the exact solutions of those complex KP could be easily found only for small n . Recently, the mathematical HyMKP [34] was proposed in $O(mnC)$ for the 0/1-mKP with most optimal solutions (in s) on $n \leq 500$. For large n and massive C , the existing metaheuristic algorithms for the 0/1-KP can be applied to solve the 0/1-mKP in polynomial time (but requiring the proper knapsack orders). For high performance, the exact DP could find the optimal solution of the 0/1-KP in $O(nC)$ and $O(mnC)$ for the 0/1-mKP (with m orders, $C = \max(C_i)$) for at least 99% optimal solutions (but for small m, n , and C). In this study, we are interested to solve the 0/1-mKP for large m, n, C with the proper orders in efficient time. Our hypothesis is “For each of m knapsacks, apply unbiased filtering before using the exact DP on remaining n' and C' can find most optimal solutions in efficient time”.

เอกสารนี้เป็นเอกสารต้นฉบับที่ยังไม่ผ่านการตรวจสอบและแก้ไขครั้งสุดท้าย. โปรดอย่าใช้เอกสารนี้เพื่อเผยแพร่หรือใช้เพื่อวัตถุประสงค์อื่นใดโดยไม่ได้รับอนุญาตจากสำนักพิมพ์. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or by any information storage and retrieval system, without permission in writing from the publisher. For more information, contact the publisher.

In this research, we introduce a novel research track (a hybrid approach of time-space reduction) for solving the 0/1-mKP in efficient time with expected 99% optimal solutions. In our hybrid approach, we propose the integration of DP transformation (reducing C to C'), exact-fit (best) knapsack-order (reducing $m!$ to m^2), and 3. robust unbiased filtering (for polynomial time). First, we propose the DP transformation (DPT) algorithm to find the optimal solutions of the 0/1-KP (for each of m knapsacks) in $O([n^2 nC])$, or $O(n^2)$ in the best case and $O(nC)$ in the worst case, before being applied for m knapsacks. Second, for the 0/1-mKP (m knapsacks), we propose the exact-fit (best) knapsack-order (in our multi-DPT) in $O(m^2 nC)$ for achieving the good solutions as m orders (at least 99% optimal solutions). Third, robust unbiased filtering is incorporated to solve the 0/1-mKP in polynomial time ($O(n^2 nC)$) while retaining 99% optimal solutions in the best case and $O(nC)$ in the worst case, before being applied for m knapsacks. Second, the correctness and complexity of the DPT and multi-DPT algorithms are analyzed. In experiments, the original and multi-DPT algorithms are evaluated on random and benchmark datasets ($n \leq 10,000$, $m \leq 100$). The main parts of this paper are organized as follows: Section 2 reviews the related work. Section 3 presents the DPT algorithm to find the optimal solutions of the 0/1-KP in $O([n^2 nC])$ and the multi-DPT + robust unbiased filtering are evaluated on random and benchmark datasets ($n \leq 10,000$, $m \leq 100$). The main parts of this paper are organized as follows: Section 2 reviews the related work. Section 3 presents the DPT algorithm and the optimal solutions of the 0/1-KP in $O([n^2 nC])$ and the multi-DPT + robust unbiased filtering are evaluated on random and benchmark datasets ($n \leq 10,000$, $m \leq 100$). Section 4 proposes the multi-DPT algorithm with the exact-fit (best) knapsack order to solve the 0/1-mKP in $O(n^2 n^2 nC)$ and reduced to $O(n^2 n)$ by our robust unbiased filtering. Section 5 provides the algorithm analysis (correctness and complexity). Section 6 performs the experiments to evaluate the performance of our multi-DPT algorithm and robust unbiased filtering. Section 7 concludes this study.

2. Related Work

For 0/1-mKP research, finding most optimal solutions ($\geq 99\%$ optimal performance) in Related Work is challenging. First, Section 2.1 reviews the exact DP algorithms to find the optimal solutions of the 0/1-KP. Section 2.2 explores the time-space reduction algorithm to solve the 0/1-mKP in poly. serial time. Section 2.3 revisits DP algorithm (quantum-inspired differential evolution for the 0/1-KP) with 2 adaptive grey wolf optimization) for the 0/1-KP. Section 2.4 explores the polynomial time. Section 2.5 reviews the recent QDGWC (quantum-inspired differential evolution with adaptive grey wolf optimizer) for the 0/1-mKP. Section 2.6 explores the efficient mathematical HyMKP model for the 0/1-mKP.

For 0/1-KP solving, let $tp[C]$ be an array of total profits, $solt_p$ be a maximum total-profit, $solt_w$ be a total-weight ($\leq C$), and $solt_x[n]$ be an array of solution X ($x_i = 0/1$). Algorithm 1 presents the basic DP [11] with two functions (preprocessing and X-tracking) on a 2D array $dp[n][C]$ (with two functions (preprocessing and X-tracking) on a 2D array). For example, consider the optimal solution $(42, 39, 28, 37, 35, 38)$ and $W = 160$ (example 8, 10, 9). Figure 2 displays the result of preprocessing ($solt_p[79] = 10$) and X-tracking ($solt_x[4] = 1$) from displaying the result of preprocessing ($solt_p[79] = 79$) and X-tracking ($solt_x[4] = 1$) from space, which is not practical if $C \gg n$. Thus, the fast DP (Algorithm 2) [11] uses a 1D array (of C elements) to find $solt_p$ in $O(nC)$ but without $solt_x[n]$, since the 1D array of the last tp -result cannot support the X-tracking (for all selected objects). Finally, the complete DP (Algorithm 3) [11] (p. 24) uses a 1D array for the full optimal solution ($solt_p, solt_w, solt_x[n]$) by repeating the fast DP for selected objects (for k selected objects) in $O(knC)$.

[illegible]

Figure 2. An example of D/HKBSolving ($n=6$, $(8)=18$) of the basic DPr (Algorithm 1) its results and $\text{soltp} = 79$ (optimal in an $\text{O}(n^3)$ matrix and X-tracking for $\text{sol}/4 \notin \{1, 4\}$).

```
soltp = 79 (optimal) in an nx6 matrix and X-tracking for sol = sp/4 ≠ {1, 4}.
```

Algorithm 1: Basic DP with 2D array (for $soltp$, $soltw$, $solx[n]$) in $O(nC)$.

```

1.   for ( $d = 0$  to  $C$ )  $tp[0,d] = 0$ ;
2.   for ( $j = 1$  to  $n$ ) do           // preprocessing for  $soltp$ 
3.       for ( $d = 0$  to  $w_j - 1$ )  $tp[j,d] = tp[j - 1,d]$ ;
4.       for ( $d = w_j$  to  $C$ ) do
5.           if ( $tp[j - 1,d - w_j] + p_j > tp[j - 1,d]$ ) then  $tp[j,d] = tp[j - 1,d - w_j] + p_j$ ;
6.           else  $tp[j,d] = tp[j - 1,d]$ ;
7.       end for  $d$ ;
8.   end for  $j$ ;
9.    $soltp = tp[n,C]$ ;
10.   $d = C$ ;  $j = n$ ;  $solx = \emptyset$ ;           // X-tracking for  $solx$ 
11.  do // X-tracking for  $solx$ 
12.      while ( $tp[j,d] = tp[j - 1,d]$ )  $j = j - 1$ ;           // move up
13.       $solx = solx \cup \{j\}$ ;  $pp = tp[j,d] - p_j$ ;  $j = j - 1$ ;
14.      while ( $tp[j,d - 1] \geq pp$  and  $pp > 0$ )  $d = d - 1$ ;       // move left
15.  while ( $pp > 0$  and  $j \geq 1$ ).

```

Algorithm 2: Fast DP with 1D array (for $soltp$, $soltw$) in $O(nC)$.

```

1.   for ( $d = 0$  to  $C$ )  $tp[d] = 0$ ;
2.   for ( $j = 1$  to  $n$ ) do           // preprocessing for  $soltp$ 
3.       for ( $d = C$  down to  $w_j$ ) do
4.           if ( $tp[d - w_j] + p_j > tp[d]$ ) then  $tp[d] = tp[d - w_j] + p_j$ ;
5.       end for  $d$ ;
6.   end for  $j$ ;  $soltp = tp[C]$ .

```

Algorithm 3: Full DP with 1D array (for $soltp$, $soltw$, $solx[n]$) in $O(knC)$.

```

1.    $solx = \emptyset$ ;  $C' = C$ ;  $n' = n$ ;
2.   do
3.       for ( $d = 0$  to  $C'$ ) do  $tp[d] = 0$ ;
4.       for ( $j = 1$  to  $n'$ ) do           // preprocessing for  $soltp$ 
5.           for ( $d = C'$  down to  $w_j$ ) do
6.               if ( $tp[d - w_j] + p_j > tp[d]$ ) then
7.                    $x[d] = j$ ;  $tp[d] = tp[d - w_j] + p_j$ ;
8.               end for  $d$ ;
9.           end for  $j$ ;
10.       $r = x[C']$ ;           // find  $solx$  (a selected object)
11.       $solx = solx \cup \{r\}$ ;  $k = k + 1$ ;
12.       $n' = r - 1$ ;  $C' = C' - w_r$ ;
13.  while ( $C' > 0$ );           // repeat for  $k$  selected objects
14.   $soltp = tp[C]$ .

```

2.2. 0/1-KP Solving by Time-Space Reduction Algorithm

In 2018, time-space reduction ($TS_{\text{Reduction}}$) algorithm [1] (p. 198) was proposed to solve the 0/1-KP in $O(n + C')$ for large n by focusing on unbiased filtering. That reduction method (Algorithm 4) consists of three main steps: 1. find the best three initial solutions; 2. perform object classification and unbiased filtering; and 3. apply the full DP on the remaining objects ($n' \leq 200$); see Equations (7)–(9). The advantages of the $TS_{\text{Reduction}}$ algorithm are the efficient time complexity $O(n + C')$ and the good performance (95% optimal solutions). See an example ($n = 20$, $C = 100$) of the 0/1-KP solved by the $TS_{\text{Reduction}}$ algorithm in Figure 3 ($soltp = 656$ (optimal)).

Step 1 Initial solutions (by GH) and improved solutions (by GH⁺)

[P/W]	X	8	16	1	9	6	0	4	18	2	17	12	11	14	10	19	3	5	13	7	15		
	W	3	4	5	3	4	6	8	8	8	7	7	8	11	8	9	10	11	10	11	10		
	P	45	54	55	31	32	43	55	50	45	37	36	40	54	37	38	42	39	35	38	34	tw	tp
	GH1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	99	652
	GH1+															0	1	0	0	0	0	100	656

[P]	X	1	4	16	14	18	8	2	0	3	11	5	19	7	17	10	12	13	15	6	9		
	W	5	8	4	11	8	3	8	6	10	8	11	9	11	7	8	7	10	10	4	3		
	P	55	55	54	54	50	45	45	43	42	40	39	38	38	37	37	36	35	34	32	31	tw	tp
	GH2	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	91	560
	GH2+													0	0	1	1	0	0	0	1	100	627

[W]	X	8	9	16	6	1	0	17	12	4	18	2	11	10	19	3	13	15	14	5	7		
	W	3	3	4	4	5	6	7	7	8	8	8	8	8	9	10	10	10	11	11	11		
	P	45	31	54	32	55	43	37	36	55	50	45	40	37	38	42	35	34	54	39	38	tw	tp
	GH3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	98	640
	GH3+															0	0	0	1	0	0	99	652

Step 2 Object classification and unbiased filtering

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
GH1 ⁺	5	5	5	5	5	0	5	0	5	5	5	5	5	0	5	0	5	5	5	0
GH2 ⁺	3	3	3	3	3	0	0	0	3	3	3	3	0	0	3	0	3	3	3	0
GH3 ⁺	1	1	1	0	1	0	1	0	1	1	1	1	1	0	1	0	1	1	1	1
dw	9	9	9	8	9	3	6	0	9	9	9	9	6	0	9	0	9	9	9	1

Filtering-in (13)={0,1,2,4,6,8,9,10,11,14,16,17,18}, Filtering-out(1)={15}

Step 3 DP on n'(6)={3,5,7,12,13,19}, C'=17

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	tw	tp
solx	1	1	1	1	0	1	0	1	1	1	1	1	1	0	1	0	1	1	1	0	100	656

Figure 3: An example of 0/1-KP solving ($n=20$, $C=100$) by TS Reduction (Algorithm 4), $solp=656$ (optimal).

Algorithm 4. Time-space reduction for 0/1-KP in $O(n + C')$.

- Step 1:** Apply the GH (greedy heuristic) algorithm by sorting 3 features ($P/W, P, W$) for top 3 initial solutions in $O(n)$. Note: Sorting (in each GH) relies on the major-minor keys.
- For P/W -decreasing sort, a major key is P/W and 2 minor keys are P & W .
 - For P -decreasing sort, a major key is P and a minor key is W .
 - For W -increasing sort, a major key is W and a minor key is P .
- Step 2:** Object classification and unbiased filtering in $O(n)$.
- 2.1 Improve 3 initial solutions by the GH⁺ algorithm.
 - 2.2 Compute dynamic weight (dw) by integrating 3 solutions of GH⁺ to GH⁺ (to support unbiased selection) where $dw = wt_1 \times x_1 + wt_2 \times x_2 + wt_3 \times x_3$, $wt_1 = 5, wt_2 = 3$, and $wt_3 = 1$ iff ($x_i = 1$); otherwise $dw = 0$ (when $x_i = 0$).
 - 2.3 Classify objects and perform unbiased filtering (Group 1 ($dw=9$), Group 2 ($dw=8, 6, 5$), Group 3 ($dw=3, 4$), and Group 4 ($dw \neq 0$)).
 - Filtering in/out (to reduce $n' \leq 200$): Select worth objects ($x_i = 1$) with $dw = 9$ (selected by all 3-GH policies).
 - Select other objects ($x_i = 1$) with $dw = 8, 6, 5$, except uncertain α (in Group 2) and do not select worst objects ($x_i = 0$) with $dw = 0$ except β (in Group 4).
- Step 3:** Apply the DP in $O(C')$ on remaining $n' = \alpha + |\text{Group3}| + \beta$.

$$n' = \alpha + |\text{Group3}| + \beta \leq 200 \quad (7)$$

$$\alpha = \min(0.7 \times |\text{Group2}|, 20) \quad (8)$$

$$\alpha = \min(0.7 \times |\text{Group2}|, 20) \quad (8)$$

$$\beta = \min(0.7 \times |\text{Group4}|, 200 - |\text{Group3}| - \alpha) \quad (9)$$

$$\beta = \min(0.7 \times |\text{Group4}|, 200 - |\text{Group3}| - \alpha) \quad (9)$$

2.3. 0/1-KP Solving by Quantum-Inspired Differential Evolution Algorithm

Recently, the QDGWO (Quantum-inspired differential evolution with adaptive grey wolf optimizer) algorithm (Figure 4) [43] was proposed in 2021 for solving the 0/1-KP by adopting the quantum computing principles plus mutation and crossover operations. In that study, the adaptive mutation operations, the crossover operations, and the quantum

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์และเผยแพร่โดยไม่หวังผลกำไร
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Recently, the QDGWO (Quantum-inspired differential evolution with adaptive grey wolf optimizer) algorithm (Figure 4) [43] was proposed in 2021 for solving the 0/1-KP by adopting the quantum computing principles plus mutation and crossover operations. In that study, the adaptive mutation operations, the crossover operations, and the quantum observation are combined to generate new solutions as trial individuals in the solution space. In experiments, the fast convergent of QDGWO (on $n = 50$ to $n = 3000$ objects) was compared with the existing quantum-based methods (QEA, AQOE, and QSE) using maximum 1000 iterations. The QDGWO results outperformed those of existing Q-based algorithms. However, the QDGWO algorithm could not guarantee the optimal solutions, especially on the irregular datasets.

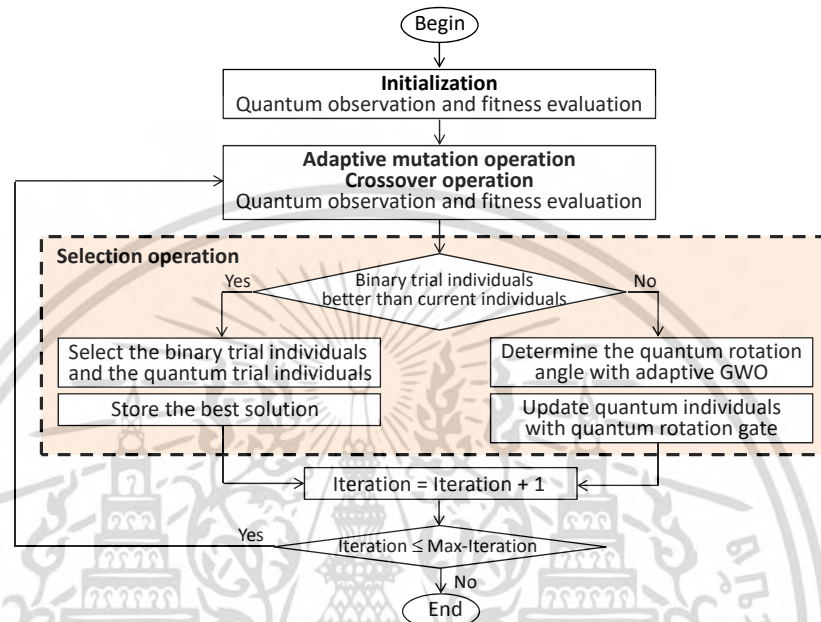


Figure 4. Framework of QDGWO (Quantum-inspired Differential Evolution with adaptive Grey Wolf Optimizer (GWO)).

2.4. 0/1-mKP Solving by Mathematical HyMKP Algorithm

For solving the 0/1-mKP (m knapsacks, n objects), the mathematical HyMKP model (Algorithm 5) [34] (p. 893) was proposed. That hybrid model includes the MULKNAP (well-known partial BnB) program and the create-reflect-multigraph-MKP (Algorithm 6) [34] (p. 891) and two decomposition methods. Algorithm 6 was modified from the arc-flow model and the reflect model in $O(mnC)$, $C = \max(C_i, i = 1, 2, \dots, m)$, by starting with decreasing weights (of n items/objects) for the good initial-solution (in m knapsacks). Then, that solution was improved by the knapsack-based decomposition with v iterations (in τ secs.). In the past, such mathematical models were used to solve the classical stock problem (CSP: 1999–2017) and the cutting and packing problems (1970–1977). In experiments (on OR-benchmark datasets), the mathematical HyMKP algorithm (with time complexity $O(mnC)$) yielded 99.9% optimal solutions (in τ secs.) for small $n \leq 500$.

Algorithm 5: Mathematical HyMKP model.

- Step 1:** perform the existing preprocessing: *Instance reduction*, *Capacity lifting*, and *Item dominance*.
- Step 2:** call MULKNAP *branch-and-bound* (BnB) for τ secs.
if (the solution is optimal) then return.
- Step 3:** call *Create-reflect-multigraph-MKP* (Algorithm 6).
- Step 4:** for ($i = 1$ to v) do
execute the *knapsack-based decomposition*.
if (an optimal solution has been obtained) then return.
else add the resulting no-good-cut.
end for i
- Step 5:** if (the instance is not solved) then
execute the *reflect-based decomposition* and return.

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์โดยมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Algorithm 6: Create-reflect-multigraph-MKP in $O(mnC)$.

```

1.    $N = \emptyset; A_s = \emptyset; A_r = \emptyset; A_c = \emptyset; A_l = \emptyset; A = \emptyset; s = 0; C = \max\{C_i\}_{i=1,2,\dots,m}$ 
2.   for  $(l = 1$  to  $C/2)$  do  $T[l] = 0;$ 
3.    $T[s] = 1; N = N \cup \{s\};$ 
4.   for  $(j = 1$  to  $n)$  do
5.       for  $(l = C/2 - 1$  down to  $0)$  do
6.           if  $(T[l] = 1)$  then
7.               if  $(l + w_j \leq C/2)$  then
8.                    $A_s = A_s \cup \{(l, l + w_j, j, 0)\}; T[l + w_j] = 1; N = N \cup \{(l + w_j)\};$ 
9.               for  $(i = 1$  to  $m)$  do
10.                  if  $(l + w_j > C/2$  and  $l \leq C_i - (l + w_j))$  then
11.                       $A_r = A_r \cup \{(l, C_i - (l + w_j), j, i)\}; N = N \cup \{(C_i - (l + w_j))\};$ 
12.                  end for  $i;$ 
13.              end if;
14.          end for  $l;$ 
15.      end for  $j;$ 
16.      for  $(i = 1$  to  $m)$  do  $N = N \cup \{C_i/2\};$ 
17.      for  $(i = 1$  to  $m)$  do  $A_c = A_c \cup \{(C_i/2, C_i/2, 0, i)\};$ 
18.      for  $(l \in N)$  do  $A_l = A_l \cup \{(l, l', 0, 0); l' = \min\{e \in N: e > l\}\};$ 
19.       $A = A_s \cup A_r \cup A_c \cup A_l;$ 
20.      return  $N, A;$ 

```

In particular, the arc-flow model was modified to fill a knapsack as a path in a graph, where arcs were items/objects. The looping conditions of Algorithm 6 (Lines 4–15) are similar to the basic DP (Algorithm 1) for each of m knapsacks with time complexity in $O(mnC)$. Let (d, e, j, i) denote an arc in a set A from nodes d to e (Lines 8, 11 and 17–19).

$A_s = \{(d, d + w_j, j, 0), 1 \leq j \leq n\}$ is the set of standard item arcs.

$A_r = \{(d, C_i - (d + w_j), j, i), 1 \leq i \leq m, 1 \leq j \leq n\}$ is the set of reflected item arcs (satisfying $d + w_j > C_i/2$ and $d \leq C_i - w_j$).

$A_c = \{(C_i/2, C_i/2, 0, i)\}$ is the set of reflected connection arcs.

$A_l = \{(d, e, 0, 0), d < e\}$ is the set of loss arcs.

For solving the 0/1-mKP (in theory), the existing researches were focused on the exact algorithms, such as the branch-and-bound algorithm [35] and the mathematical HyMKP [34], where their performance results were compared to the (known) optimal solutions but those algorithms could execute in reasonable time on small $n \leq 500$ only.

For large n (in practice), each of the efficient meta-heuristic algorithms [26,30,32,43] proposed for the 0/1-KP (i.e., GA, swarm optimization, quantum computing, hybrid method, etc.) can be applied to solve the 0/1-mKP (for each of m knapsacks) with proper orders. For example, we can apply the recent QDGWO [43]) for the 0/1-KP (one knapsack) to the 0/1-mKP (m knapsacks). However, that meta-heuristic algorithm cannot guarantee the optimal solution for each knapsack, especially on the irregular datasets, and hence the total profit from many knapsacks ($m > 10$) may be near-optimal only.

Therefore, to solve the 0/1-mKP ($m > 10$) for large n , we are interested to find the high optimal performance in efficient time by our hybrid approach. In this study, our proposed algorithms (in the hybrid approach) are

1. Exact DP transformation (DPT) algorithm (in Section 3) to find the optimal solutions of the 0/1-KP (in each knapsack).
2. $m!$ -to- m^2 knapsack-order reduction (in Section 4.1.2) to define the exact-fit (best) knapsack order for the 0/1-mKP (m knapsacks).
3. Robust unbiased filtering (in Section 4.2) to improve/reduce time complexity to polynomial time while retaining the high optimal performance.

In this 0/1-mKP research, we propose “a novel research track (a hybrid approach of the exact DPT + robust unbiased filtering) to solve the 0/1-mKP in efficient time with expected at least 99% optimal solutions”. We start with our exact-DP transformation for 0/1-KP in $O([n^2, nC])$ over $O(nC)$ for each knapsack (in Section 3) before being applied to 0/1-mKP (m knapsacks) with the exact-fit (best) knapsack order in $O(m^2[n^2, nC])$ over $O(m![n^2, nC])$ in Section 4 and reduce it to $O(m^2n)$ by our efficient unbiased filtering.

3. 0/1-KP Solving by DP Transformation to List-Based Time-Space Reduction

First, in our new research track (for solving the 0/1-mKP), we propose the DP transformation to list-based time-space reduction (DPT-List_{TSR}) algorithm to find the optimal solutions of the 0/1-KP in $O(n^2)$ in the best case and $O(nC)$ in the worst case. Our DPT-List_{TSR} algorithm was renovated from the basic DP (Algorithm 1). That original DP can find the optimal solution of the 0/1-KP in $O(nC)$ by using the 2D-array ($n \times C$) in all (best, average, and worst) cases. In this study, the DPT-List_{TSR} algorithm can find the optimal solution of the 0/1-KP by introducing the lists of effective nodes (e-nodes) in efficient $O([n^2, nC])$. The contribution of our DPT-List_{TSR} is the forward reduction (F-reduction) in the preprocessing (for the (original) e-nodes) and the backward reduction (B-reduction) in the X-tracking (for the tight bound of the original e-nodes).

Next, to simplify our DPT-List_{TSR} process (the preprocessing in Section 3.1 and the X-tracking in Section 3.2), the following data structures and proper functions are predefined. See a corresponding example of our DP transformation in Figure 5 ($n = 5, C = 18$).

The e-node is an effective node with improved tp (total profit) by object j at capacity d that is more than tp at previous capacity $d - 1$.

The original e-node is the original improved e-node by object j at capacity d (not by object $j + 1$ at the same d).

The F-list (forward list) is a list of e-nodes (used in object $j - 1$ and object j).

The B-list (backward list) is a list of original e-nodes (for X-tracking).

F-reduction is a function used to reduce e-nodes to the original e-nodes.

B-reduction is another function applied after finding $x_j = 1$ in X-tracking (to simplify the remaining X-tracking process).

3.1. Preprocessing of the DPT-List_{TSR} Algorithm

In our preprocessing (Algorithm 7), two (temporary) F-lists of e-nodes (the previous F-list $j - 1$ and the current F-list j) must be built first; see Figure 5 ($j = 0$ to $n - 1$). Since the previous F-list $j - 1$ can inherit all e-nodes of object 0 to object $j - 1$, the current F-list j can be constructed from F-list $j - 1$ (to inherit the previous e-nodes and fulfill the current F-list j with new e-nodes before keeping only the original e-nodes in another B-list L_j). Our preprocessing can reduce the computing time and the using space on a variety of datasets, such as $O(n^2)$ in the best case and $O(nC)$ in the worst case, as proven in Section 5.2. The current F-list j of object j (w_j, p_j) can be created in two main steps. In Step 1 (Algorithm 7: Line 3), some e-nodes ($cn = (tp, d)$) from the head of F-list $j - 1$ are copied to F-list j (while $cn.tp < p_j$ and $cn.d < w_j$). In Step 2 (Algorithm 7: Lines 4–10 (to fulfill F-list j with new e-nodes)), from each e-node (en) of F-list $j - 1$ (head to tail) Step 2.0 checks to inherit each remaining e-node (from step 1) to F-list j (in a proper location) if it is worth (see detail in Section 5.1), Step 2.1 computes ($d = en.d + w_j, tp = en.tp + p_j$), and Step 2.2 adds new e-node to tails of F-list j and B-list j if ($d \leq C$ and $tp > TP$). In this step, the desired TP (at $d = en.d + w_j$) can be decoded (from d and n_1 in F-list $j - 1$).

object $j + 1$ at the same d .

The F-list (forward list) is a list of e-nodes (used in object $j - 1$ and object j).

The B-list (backward list) is a list of original e-nodes (for X-tracking).

F-reduction is a function used to reduce e-nodes to the original e-nodes.

B-reduction is another function applied after finding $x_j = 1$ in X-tracking (to simplify the remaining X-tracking process).

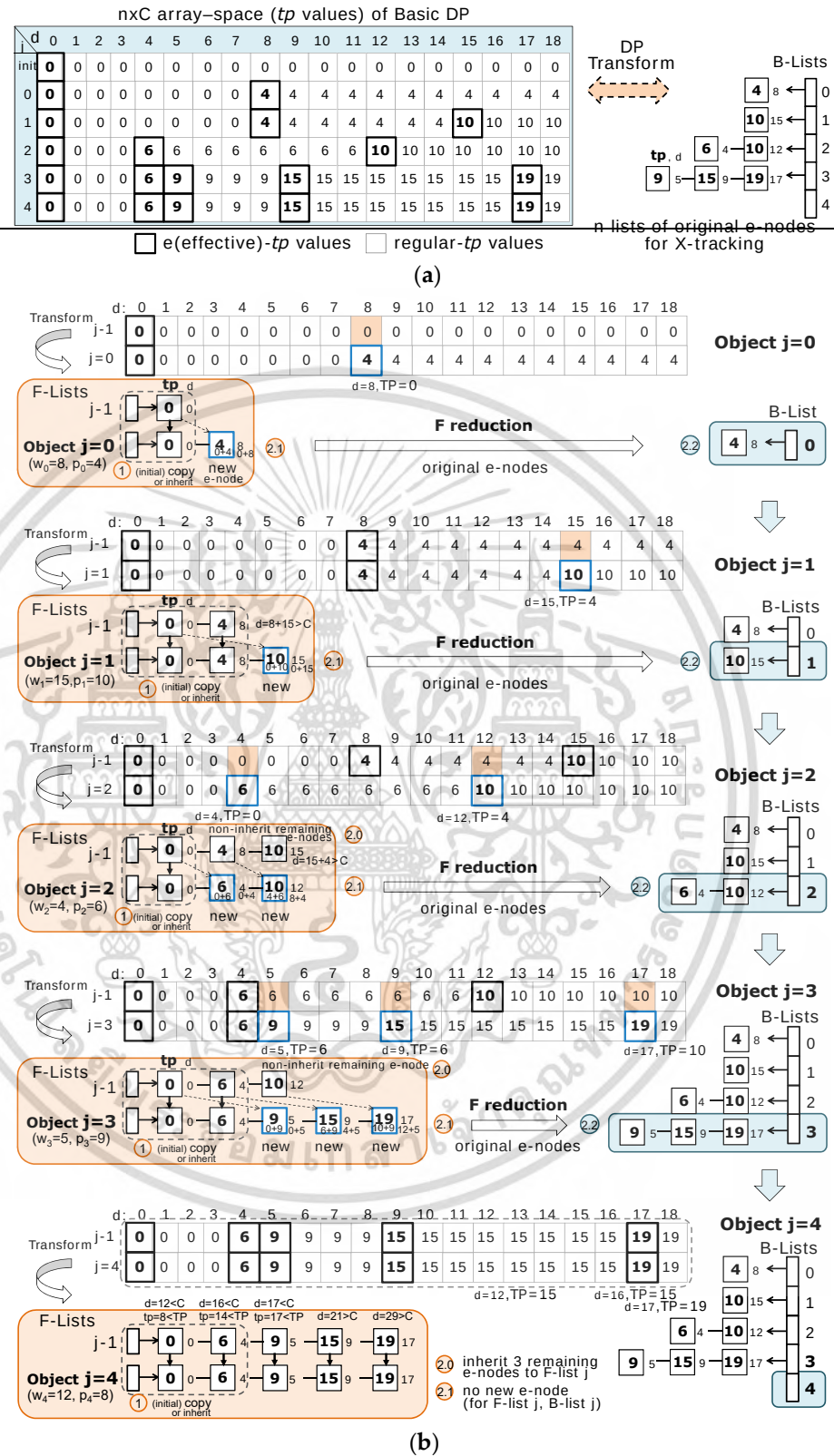


Figure 5. Figure 5a: Example of preprocessing process using $C = 18$ (a) nxC array (tp) by the basic DP and (b) two F-lists and B-lists by DPT-List algorithm for objects $j = 0-4$.

3.1. Preprocessing of the DPT-List_{TSR} Algorithm

In our preprocessing (Algorithm 7), two (temporary) F-lists of e-nodes (the previous F-list $j - 1$ and the current F-list j) must be built first; see Figure 5 ($j = 0$ to $n - 1$). Since the previous F-list $j - 1$ can inherit all e-nodes of object 0 to object $j - 1$, the current F-list j can be constructed from F-list $j - 1$ (to inherit the previous e-nodes and fulfill the current F-list j with new e-nodes before keeping only the original e-nodes in another B-list L_j). Our preprocessing can reduce the computing time and the using space on a variety of da-

Algorithm 7: Preprocessing of the DPT-List_{TSR} algorithm.

```

1.   create initial F-list  $j - 1$  (for  $j = 0$ ) with one e-node ( $tp = 0, d = 0$ );
2.   for ( $j = 0$  to  $n - 1$ ) do // to fulfill F-list  $j$  of object  $j$ 
3.       initial copy e-nodes ( $cn$ ) of F-list  $j - 1$  to F-list  $j$  (while ( $cn.tp < p_j$  &  $cn.d < w_j$ )); // Step 1
4.       e-node  $en = \text{head}(\text{F-list } j - 1)$ ;
5.       for (each e-node ( $en$ ) in F-list  $j - 1$ ) do (from head to tail) // Step 2
6.           inherit remaining e-node (if it is worth) to F-list  $j$  (in a proper location); // Step 2.0
7.           compute  $d = en.d + w_j$ ;  $tp = en.tp + p_j$ ;  $n_1 = en$ ; // Step 2.1
8.            $n_1 = \text{decode}(n_1, d)$ ;  $n_2 = n_1.\text{next}$ ;  $TP = n_1.tp$  (if  $n_1.d \leq d < n_2.d$ );
9.           if ( $d \leq C$  &  $tp > TP$ ) add new e-node to tails of F-list  $j$  and B-list  $j$ ; // Step 2.2
10.      end for (F-list  $j - 1$ );
11.  end for  $j$ ;
12.   $soltp = \max(ori-en.tp)$  of original e-node in B-list  $j$ ;

```

For example ($n = 5, C = 18$), $P = \{4, 10, 6, 9, 8\}$ and $W = \{8, 15, 4, 5, 12\}$, Figure 5a shows the B-list reduction (right) of seven original e-nodes (see detail in Figure 5b), compared to total profits $= n \times C = 90$ elements (left) of the basic DP. Figure 5b displays the preprocessing of DPT-List_{TSR} from $j = 0$ to 4.

- For object $j = 0$ ($w_0 = 8, p_0 = 4$), Step 1 copies the first e-node $cn = (tp, d) = (0, 0)$ of F-list $j - 1$ to F-list j (since $cn.tp < p_0$ and $cn.d < w_0$). In Step 2 (from head of F-list $j - 1$), at e-node $en = (0, 0)$, Step 2.1 computes $d = en.d + w_0 = 8$, $tp = en.tp + p_0 = 4$, and decode $TP = 0$ [$(n_1.d = 0, n_1.tp = 0)$, $(n_2.d = C = 18, n_2.tp = 0)$]. Since $d < C$ and $tp = 4 > TP = 0$, Step 2.2 adds new e-node $= (4, 8)$ in F-list $j = 0$ and B-list L_0 .
- For object $j = 1$ ($w_1 = 15, p_1 = 10$), Step 1 copies $(0, 0)$, $(4, 8)$ of F-list $j - 1$ to F-list j (while $cn.tp < p_1$ and $cn.d < w_1$). In Step 2, at e-node $en = (0, 0)$ of F-list $j - 1$, compute $d = 15$, $tp = 10$, $TP = 4$ [$(n_1.d = 8, n_1.tp = 4)$, $(n_2.d = C = 18, n_2.tp = 4)$]. Since $tp = 10 > TP = 4$, add new e-node $(10, 15)$ in F-list $j = 1$ and B-list L_1 . At $en = (4, 8)$, $d = 8 + 15 = 23 > C$ (no new e-node added).
- For object $j = 2$ ($w_2 = 4, p_2 = 6$), Step 1 copies $(0, 0)$ of F-list $j - 1$ to F-list j . In Step 2, at $en = (0, 0)$ of F-list $j - 1$, $d = 4$, $tp = 6 > TP = 0$, add new e-node $(6, 4)$ in F-list $j = 2$ and B-list L_2 . At $en = (4, 8)$ of F-list $j - 1$, $d = 8 + 4 = 12$, $tp = 4 + 6 = 10 > TP = 4$, add new e-node $(10, 12)$. At $en = (10, 15)$, $d = 15 + 4 = 19 > C$ (no new e-node added).
- For object $j = 3$ ($w_3 = 5, p_3 = 9$), Step 1 copies $(0, 0)$, $(6, 4)$ of F-list $j - 1$ to F-list j . In Step 2, at $en = (0, 0)$ of F-list $j - 1$, $d = 5$, $tp = 9 > TP = 6$ (add new e-node $(9, 5)$). At $en = (6, 4)$ of F-list $j - 1$, $d = 4 + 5 = 9$, $tp = 6 + 9 = 15 > TP = 6$ (add new e-node $(15, 9)$). At $en = (10, 12)$, $d = 17$, $tp = 19 > TP = 10$ (add new e-node $(19, 17)$).
- For object $j = 4$ ($w_4 = 12, p_4 = 8$), Step 1 copies $(0, 0)$, $(6, 4)$ of F-list $j - 1$ to F-list j . In Step 2, at $en = (0, 0)$ of F-list $j - 1$, compute $d = 12$, $tp = 8 < TP = 15$ (no new e-node added). At $en = (6, 4)$, compute $d = 4 + 12 = 16$, $tp = 6 + 8 = 14 < TP = 15$ (no new e-node added). At $en = (9, 5)$, add this remaining (worth) e-node to F-list j and compute $d = 5 + 12 = 17$, $tp = 9 + 8 = 17 < TP = 19$ (no new e-node added). At $en = (15, 9)$, add this remaining e-node to F-list j and compute $d = 9 + 12 = 21 > C$ (no new e-node added). At $en = (19, 17)$, add this remaining e-node to F-list j and compute $d = 17 + 12 = 29 > C$ (no new e-node added).

Figure 6 shows another example ($n = 15, C = 40$), $P = \{17, 14, 14, 15, 12, 16, 13, 15, 16, 18, 22, 24, 21, 13, 11\}$ and $W = \{11, 14, 7, 5, 10, 12, 5, 8, 5, 11, 9, 10, 8, 5, 6\}$. The 223 e-nodes (black + gray) and 103 original e-nodes (black) are reduced over $nC = 15 \times 40 = 600$ cells.



4.0/1-mKP Solving by Multi-DP-T-List-Plus Unbiased Filtering in Efficient Time

To solve the 0/1 mknP, we propose an efficient novel research track (Figure 8a) by

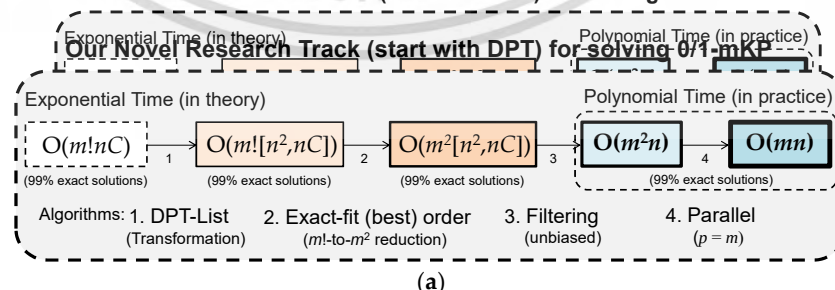
[illegible]

Figure 8. Cont.

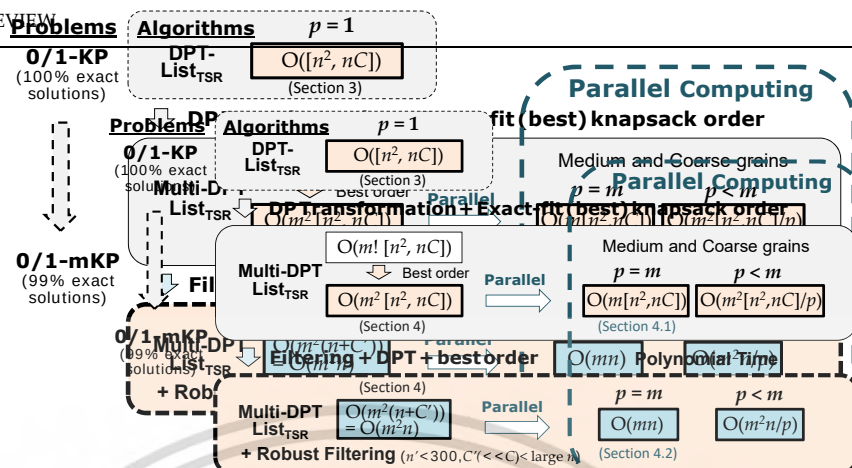


Figure 8. Figure 8 (a) Our novel research track for solving 0/1-mKP in polynomial time with 99% optimal solutions and (b) DPT-List_{TSR} and Multi-DPT-List_{TSR} algorithm and parallel model.

Figure 8. (a) Our novel research track for solving 0/1-mKP in polynomial time with 99% optimal solutions and (b) DPT-List_{TSR} and Multi-DPT-List_{TSR} algorithm and parallel model.

4.1. Efficient Multi-DPT-List_{TSR} Algorithm for Solving 0/1-mKP

The 0/1-mKP is one of the hardest KPs since it is difficult to find the optimal solutions in $O(m^2, nC)$ by the BB algorithm, except on small n . For large n , we study the DPT-List_{TSR} algorithm (in Section 3) first for the 0/1-KP since its optimal solution can be computed in $O(n^2, nC)$ in each knapsack (or the internal effect for the 0/1-mKP (m knapsacks)). Next, we can use m orders (m knapsacks) directly (for the external effect) in our multi-DPT-List_{TSR} algorithm for at least 99% optimal solutions in $O(m^2, nC)$ (since the more orders there are, the higher the optimal precision. However, the exponential complexity grows as the number of orders increases). However, that exponential complexity cannot support large m , n , and C . Thus, we propose two efficient order reductions: 1. the top nine (knapsack) orders in $O(m^2, nC)$ for the regular datasets and 2. the exact fit (best) knapsack order in $O(m^2, nC)$ for the irregular datasets.

4.1.1. Top Nine Knapsack Orders for Regular Datasets

Initially, the top nine (knapsack) orders are introduced in our multi-DPT-List_{TSR} algorithm, which are good enough to solve the 0/1-mKP with 99% optimal solutions for the regular datasets. Each of the top nine orders is obtained by sorting m capacities ($C_i, i = 1, 2, 3, \dots, m$). For example ($m = 5$), the forward order (F) of capacities $C = (66, 26, 80, 96, 70)$ is (1, 2, 3, 4, 5) and the top three orders are increasing (inc) = (2, 1, 5, 3, 4), decreasing (dec) = (4, 3, 5, 1, 2), and combined inc-dec = (2, 4, 1, 3, 5). In this study, the top nine effective orders include increasing (inc), decreasing (dec), combining inc-dec, combining dec-inc, orders in forward (f) backward (b), odd-even (of) and even-odd (eo), and odd-even (of) dec, see a corresponding example in Figure 9. Moreover, the top nine orders can be improved by the Latin square (LS) permutation to achieve at least 99% optimal solutions for the regular datasets. In practice, the parallel LS permutation of the top nine orders are used to preserve the complexity in $O(m^2, nC)$ for $m > 9$.

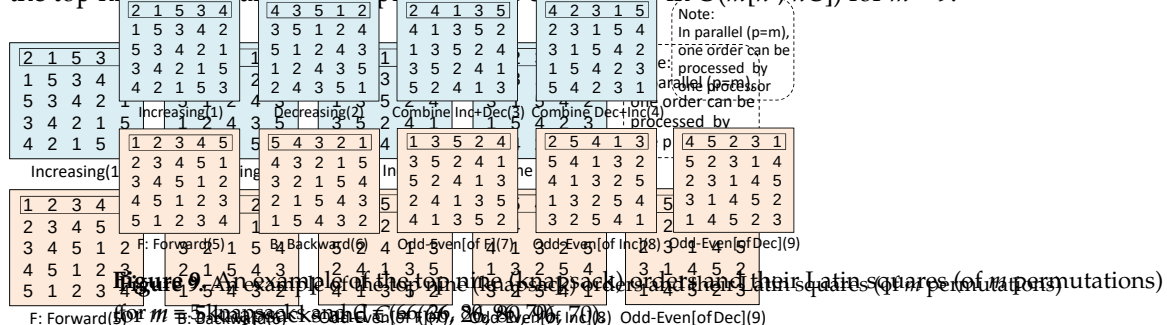


Figure 9. An example of the top nine knapsack orders (of nine orders) in $O(m^2, nC)$. Moreover, in some cases, the top nine orders are used to preserve the complexity in $O(m^2, nC)$ for $m > 9$.

Figure 9. An example of the top nine knapsack orders (of nine orders) in $O(m^2, nC)$. Moreover, in some cases, the top nine orders are used to preserve the complexity in $O(m^2, nC)$ for $m > 9$.

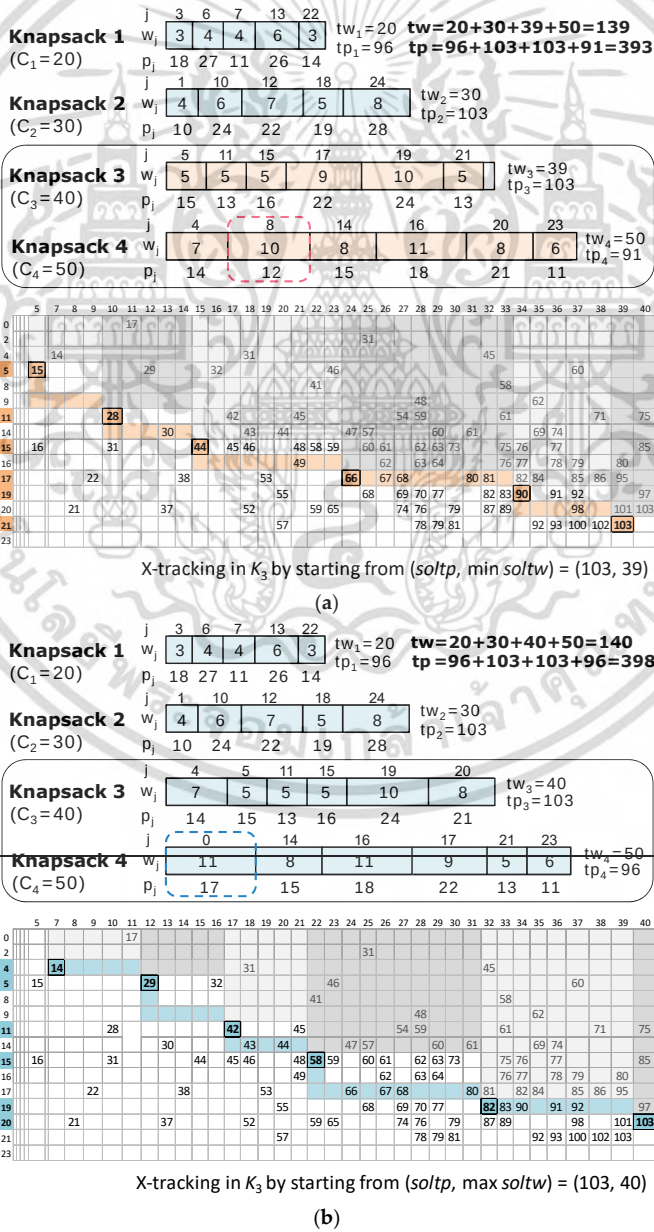
[illegible]

Figure 10. An example of start tracking from $(20, 25, 40, 50)$ of two tracking with nonunique solution Xs in K3: (a) start tracking from $(\text{soltp}, \text{min soltp})$ and (b) start tracking from $(\text{soltp}, \text{max soltp})$.

Algorithm 9: Multi-DPT-List_{TSR} for one proper order: $O(m[n^2, nC])$.

1. $n^* = n$;
2. for $(i = 1 \text{ to } m)$ do
3. apply DPT-List_{TSR} (n^* objects) on knapsack i (K_i);
4. call Algorithm 7: // preprocessing (of DPT-List_{TSR})

Algorithm 9: Multi-DPT-List_{TSR} for one proper order: $O(m[n^2, nC])$.

```

1.    $n^* = n$ ;
2.   for ( $i = 1$  to  $m$ ) do
3.       apply DPT-ListTSR ( $n^*$  objects) on knapsack  $i$  ( $K_i$ );
4.       call Algorithm 7; // preprocessing (of DPT-ListTSR)
5.       if ( $i < m$ ) start X-tracking at ( $\max tp$ ,  $\max tw$ );
6.       else ( $i = m$ ) start X-tracking at ( $\max tp$ ,  $\min tw$ );
7.       call Algorithm 8; // X-tracking (of DPT-ListTSR) for  $\max tp$ 
8.       Total profit = Total profit +  $\max tp$ ;
9.       update  $n^*$  (exclude  $k_i$  selected objects of knapsack  $i$ );
10.  end for  $i$ ;
11.  return Total profit.

```

4.1.2. The Exact-Fit (Best) Knapsack Order for Regular and Irregular Datasets

For the irregular datasets, we may use all possible $m!$ orders to find at least 99% optimal solutions in $O(m! [n, nC])$ but $m!$ orders work on small m only. Thus, for large m , to achieve the optimal precision as $m!$ orders, we propose the exact-fit (best) knapsack order (Algorithm 10) in $O(m^2[n^2, nC])$, where both internal and external effects must be solved by the exact DPT-List_{TSR} algorithm. For the external effect (among m knapsacks), the DPT-List_{TSR} algorithm is used for computing the exact (TP_i, TW_i) in each of available knapsacks before selecting K_i with the best exact-fit $_i = \min(dFit_i)$, where different Fit_i ($dFit_i$) = $C_i - TW_i$, $\forall i \leq m$. For instance, Figure 11 shows the exact-fit (best) order for $m = 5$, $C = (66, 26, 80, 96, 70)$, $n = 33$, $P = \{18, 44, 7, 21, 22, 29, 42, 24, 36, 17, 13, 23, 12, 25, 15, 41, 15, 19, 33, 5, 8, 18, 28, 25, 12, 30, 19, 14, 48, 25, 16, 23, 25\}$, and $W = \{6, 12, 16, 12, 14, 14, 5, 12, 12, 15, 10, 17, 14, 9, 19, 5, 7, 12, 8, 14, 14, 15, 14, 12, 7, 6, 13, 15, 10, 14, 8, 12, 10\}$. In Figure 11b, the best order (2, 4, 1, 5, 3) is computed in $m(m+1)/2 = 15$ steps by our exact DPT-List_{TSR} algorithm to achieve the optimal result (726).

- In the first K_i selection, there are m $dFit_i$ -results (in $m = 5$ steps) with two $\min(dFit_i) = 0$ (in K_2, K_3) and K_2 ($\min C_2$) is selected (see conditions in Step 2 of Algorithm 10).
- In the second K_i selection, there are 4 $dFit_i$ -results and K_4 ($\min(dFit_4) = 0$) is selected.
- In the third K_i selection, there are 3 $dFit_i$ -results and K_1 ($\min(dFit_1) = 0$) is selected.
- In the fourth K_i selection, there are 2 $dFit_i$ -results and K_5 ($\min(dFit_5) = 0$) is selected.
- In the fifth K_i selection, the last K_3 ($\min(dFit_3) = 0$) in the last step is selected.

Moreover, for critical decisions in some datasets, there are equal $\min(dFit_i)$ s in $K_{i'} - K_{i''}$. Then, three extra policies (Step 2 in Algorithm 10) are introduced to find the best of the three best results (for the good results as $m!$ orders as much as possible).

Algorithm 10: multi-DPT-List_{TSR} (the exact-fit (best) knapsack order).

Step 1: apply DPT-List_{TSR} for (TP_i, TW_i) on each of m knapsacks in $O(m[n^2, nC])$ and $O([n^2, nC])$ in parallel ($p = m$).

Step 2: select best K_i with $\min(dFit_i)$; $dFit_i = C_i - TW_i$ ($i = 1, 2, \dots, m$).

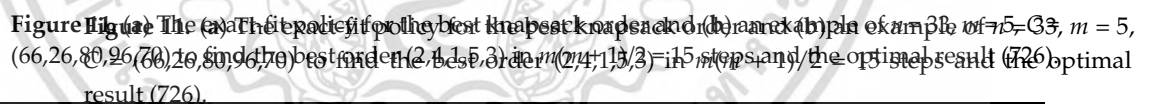
In Step 2, for critical $\min(dFit_i)$, each of the three policies is applied.

Policy 1: if (there are equal $\min(dFit_i)$ s), select best K_i with $\min(C_i)$; if (there are equal $\min(C_i)$ s), select best K_i with $\max(TP_i)$;
Policy 2: if (there are equal $\min(dFit_i)$ s), select best K_i with $\max(TP_i/TW_i)$; if (there are equal $\max(TP_i/TW_i)$ s), select best K_i with $\min(C_i)$; if (there are equal $\min(C_i)$ s), select best K_i with $\max(TP_i)$;
Policy 3: if (there are equal $\min(dFit_i)$ s), select best K_i with $\max(TP_i)$; if (there are equal $\max(TP_i)$ s), select best K_i with $\min(C_i)$;

Step 3: update unselected $n^* = n - k$ and $m' = m - 1$.

Step 4: repeat Step 1–3 on n^* and m' until $m' = 1$.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Step 1: apply $\text{DPT-List}_{\text{SR}}$ for (I_P, I_W) on each of m knapsacks, in $\mathcal{O}(m[n^2, nC])$ and $\mathcal{O}(n^2, nC)$ by $p = m$. However, $\mathcal{O}(mnC)$ in the worst case is not efficient for large m, n , and C .

Step 2: select best K with $\min(d, it_i)$; $d, it_i = C - I W_i (i = 1, 2, \dots, m)$.
 $O(m^2 n)$ by $p = 1$ and $O(mn)$ by $p = m$ while retaining 99% exact precision.
 In Step 2, for critical $\min(d, it_i)$, each of the three policies is applied.

Policy 1: if (there are equal $\min(dFit)$ s), select best K_i with $\min(C_i)$;
Policy 2: Efficient Robust Unbiased Filtering for Polynomial Time Reduction
 if (there are equal $\min(C_i)$ s), select best K_i with $\max(TP_i)$;
 In our novel research track (Figure 8a), the contribution in polynomial time is achieved by robust unbiased filtering in $O(m/(n+1)C_i)$ or $O(n/n)$ on $C_i \ll C$, large n while retaining 99% optimal solutions. Our (fast and efficient) unbiased filtering can select the outstanding objects (from n objects), and only uncertain objects ($n' < 300$) are considered by the DPT-List_{TSR} algorithm (in each knapsack). For the 0/1-mKP, the parameter (γ, α, β) -setting (in Equations (10)-(14)) was our key contribution to retain 99% optimal precision, as in our previous work (Algorithm 4) [1]. Usually, the critical and uncertain objects ($\gamma, \alpha,$ and

In parallel, the multi-DPI-List_{FSR} (the exact-fit (best) order) can be processed in $O(m|n^2, nC|)$ by $p = m$. However, $O(mn)$ in the worst case is not efficient for large m , n , and C . Thus, in Section 4.2, robust unbiased filtering (our key contribution) is presented in efficient $O(m \cdot n)$ by $p = 1$ and $O(mn)$ by $p = m$ while retaining 99% exact precision.

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$\gamma' = \min(10, 0.15 \times |\text{Group1}|), \max \gamma' = 10 \quad (11)$$

$$\beta = \min(50, 0.70 \times |\text{Group}_4|); \max \beta = 50 \quad (13)$$

$$U = \min(\min(200, \text{GroupSize}), \max(U, 200)) \quad (14)$$

From the four groups of object classification (in Figure 12), the dynamic critical ratios were studied to find the critical number of objects (300) and after filtering retaining 99 in group 1, 101 in group 2, 103 in group 3, and 105 in group 4. The large number of variable n is $n = 100, 254, 500, 1205 \pm 285, 200 \pm 285$ since for the large n a large number of filtering of objects (objects) and for small n a small number of filtering of objects (objects). Efficient filtering (filtering) algorithm (Algorithm 3) is required (required) before applying DPT-List-DR algorithm and C++ code C1. Note that (1) remaining objects after filtering (remaining objects) of objects knapsack is different. For example, Example 3 Figure 13 shows the object classification for filtering (for filtering) $n = 2530, 402, P = \{30, 100\}$ and $1, 4, 8, 17, 115, 127, 181, 142, 113, 27, 13, 22, 26, 23, 16, 18, 22, 15, 24, 28, 22, 14, 24, 28\}$ and $4/11, 28\}$ and $1/8 = \{5, 4, 4, 3, 12, 35, 45, 47, 16, 8, 25, 61, 5, 7, 5, 80, 58, 13, 9, 6, 80\}$ with four groups of classification (classification) of objects (objects) and Figure 14 demonstrates the results of filtering (filtering) of objects (objects) of objects (objects) and (3) comparing filtering (filtering) objects (objects) of objects (objects) of objects (objects) and (4) and (5) comparing filtering (filtering) objects (objects) of objects (objects) of objects (objects) and (6) and (7) and (8) and (9) and (10) and (11) and (12) and (13) and (14) and (15) and (16) and (17) and (18) and (19) and (20) and (21) and (22) and (23) and (24) and (25) and (26) and (27) and (28) and (29) and (30) and (31) and (32) and (33) and (34) and (35) and (36) and (37) and (38) and (39) and (40) and (41) and (42) and (43) and (44) and (45) and (46) and (47) and (48) and (49) and (50) and (51) and (52) and (53) and (54) and (55) and (56) and (57) and (58) and (59) and (60) and (61) and (62) and (63) and (64) and (65) and (66) and (67) and (68) and (69) and (70) and (71) and (72) and (73) and (74) and (75) and (76) and (77) and (78) and (79) and (80) and (81) and (82) and (83) and (84) and (85) and (86) and (87) and (88) and (89) and (90) and (91) and (92) and (93) and (94) and (95) and (96) and (97) and (98) and (99) and (100) and (101) and (102) and (103) and (104) and (105) and (106) and (107) and (108) and (109) and (110) and (111) and (112) and (113) and (114) and (115) and (116) and (117) and (118) and (119) and (120) and (121) and (122) and (123) and (124) and (125) and (126) and (127) and (128) and (129) and (130) and (131) and (132) and (133) and (134) and (135) and (136) and (137) and (138) and (139) and (140) and (141) and (142) and (143) and (144) and (145) and (146) and (147) and (148) and (149) and (150) and (151) and (152) and (153) and (154) and (155) and (156) and (157) and (158) and (159) and (160) and (161) and (162) and (163) and (164) and (165) and (166) and (167) and (168) and (169) and (170) and (171) and (172) and (173) and (174) and (175) and (176) and (177) and (178) and (179) and (180) and (181) and (182) and (183) and (184) and (185) and (186) and (187) and (188) and (189) and (190) and (191) and (192) and (193) and (194) and (195) and (196) and (197) and (198) and (199) and (200) and (201) and (202) and (203) and (204) and (205) and (206) and (207) and (208) and (209) and (210) and (211) and (212) and (213) and (214) and (215) and (216) and (217) and (218) and (219) and (220) and (221) and (222) and (223) and (224) and (225) and (226) and (227) and (228) and (229) and (230) and (231) and (232) and (233) and (234) and (235) and (236) and (237) and (238) and (239) and (240) and (241) and (242) and (243) and (244) and (245) and (246) and (247) and (248) and (249) and (250) and (251) and (252) and (253) and (254) and (255) and (256) and (257) and (258) and (259) and (260) and (261) and (262) and (263) and (264) and (265) and (266) and (267) and (268) and (269) and (270) and (271) and (272) and (273) and (274) and (275) and (276) and (277) and (278) and (279) and (280) and (281) and (282) and (283) and (284) and (285) and (286) and (287) and (288) and (289) and (290) and (291) and (292) and (293) and (294) and (295) and (296) and (297) and (298) and (299) and (300) and (301) and (302) and (303) and (304) and (305) and (306) and (307) and (308) and (309) and (310) and (311) and (312) and (313) and (314) and (315) and (316) and (317) and (318) and (319) and (320) and (321) and (322) and (323) and (324) and (325) and (326) and (327) and (328) and (329) and (330) and (331) and (332) and (333) and (334) and (335) and (336) and (337) and (338) and (339) and (340) and (341) and (342) and (343) and (344) and (345) and (346) and (347) and (348) and (349) and (350) and (351) and (352) and (353) and (354) and (355) and (356) and (357) and (358) and (359) and (360) and (361) and (362) and (363) and (364) and (365) and (366) and (367) and (368) and (369) and (370) and (371) and (372) and (373) and (374) and (375) and (376) and (377) and (378) and (379) and (380) and (381) and (382) and (383) and (384) and (385) and (386) and (387) and (388) and (389) and (390) and (391) and (392) and (393) and (394) and (395) and (396) and (397) and (398) and (399) and (400) and (401) and (402) and (403) and (404) and (405) and (406) and (407) and (408) and (409) and (410) and (411) and (412) and (413) and (414) and (415) and (416) and (417) and (418) and (419) and (420) and (421) and (422) and (423) and (424) and (425) and (426) and (427) and (428) and (429) and (430) and (431) and (432) and (433) and (434) and (435) and (436) and (437) and (438) and (439) and (440) and (441) and (442) and (443) and (444) and (445) and (446) and (447) and (448) and (449) and (450) and (451) and (452) and (453) and (454) and (455) and (456) and (457) and (458) and (459) and (460) and (461) and (462) and (463) and (464) and (465) and (466) and (467) and (468) and (469) and (470) and (471) and (472) and (473) and (474) and (475) and (476) and (477) and (478) and (479) and (480) and (481) and (482) and (483) and (484) and (485) and (486) and (487) and (488) and (489) and (490) and (491) and (492) and (493) and (494) and (495) and (496) and (497) and (498) and (499) and (500) and (501) and (502) and (503) and (504) and (505) and (506) and (507) and (508) and (509) and (510) and (511) and (512) and (513) and (514) and (515) and (516) and (517) and (518) and (519) and (520) and (521) and (522) and (523) and (524) and (525) and (526) and (527) and (528) and (529) and (530) and (531) and (532) and (533) and (534) and (535) and (536) and (537) and (538) and (539) and (540) and (541) and (542) and (543) and (544) and (545) and (546) and (547) and (548) and (549) and (550) and (551) and (552) and (553) and (554) and (555) and (556) and (557) and (558) and (559) and (560) and (561) and (562) and (563) and (564) and (565) and (566) and (567) and (568) and (569) and (570) and (571) and (572) and (573) and (574) and (575) and (576) and (577) and (578) and (579) and (580) and (581) and (582) and (583) and (584) and (585) and (586) and (587) and (588) and (589) and (590) and (591) and (592) and (593) and (594) and (595) and (596) and (597) and (598) and (599) and (600) and (601) and (602) and (603) and (604) and (605) and (606) and (607) and (608) and (609) and (

Group 1	Group2	Group3	Group4
dw = 9	γ dw = 8,6,5	α dw = 4,3,1	β dw = 0
$(x_j = 1)$	$(x_j = 1)$	$(x_j = 0)$	$(x_j = 0)$

Figure 12. Four groups of object classification and efficient filtering for remaining $n' < 300$.

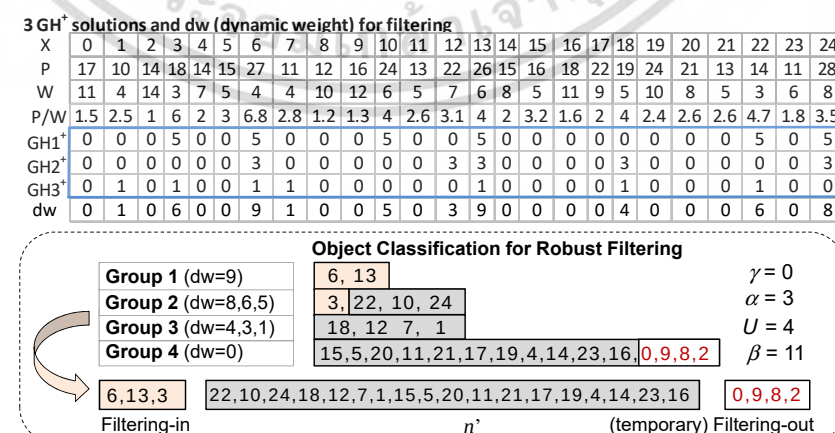
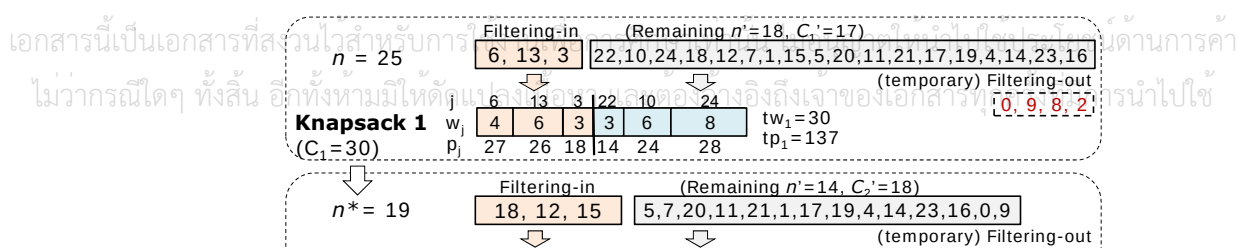


Figure 13. An example ($n=25, m=2, C=(30, 40)$) and object classification for knapsack₁($\mathbb{K}$).



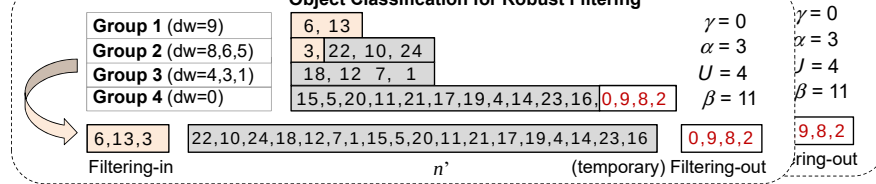


Figure 13. An example of robust unbiased filtering before applying DPT-ListTSR in K_1 and K_2 .

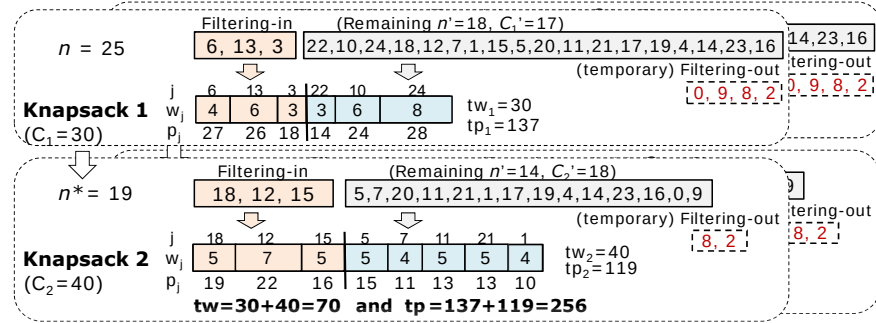


Figure 14. An example of robust unbiased filtering before applying DPT-ListTSR in K_1 and K_2 .

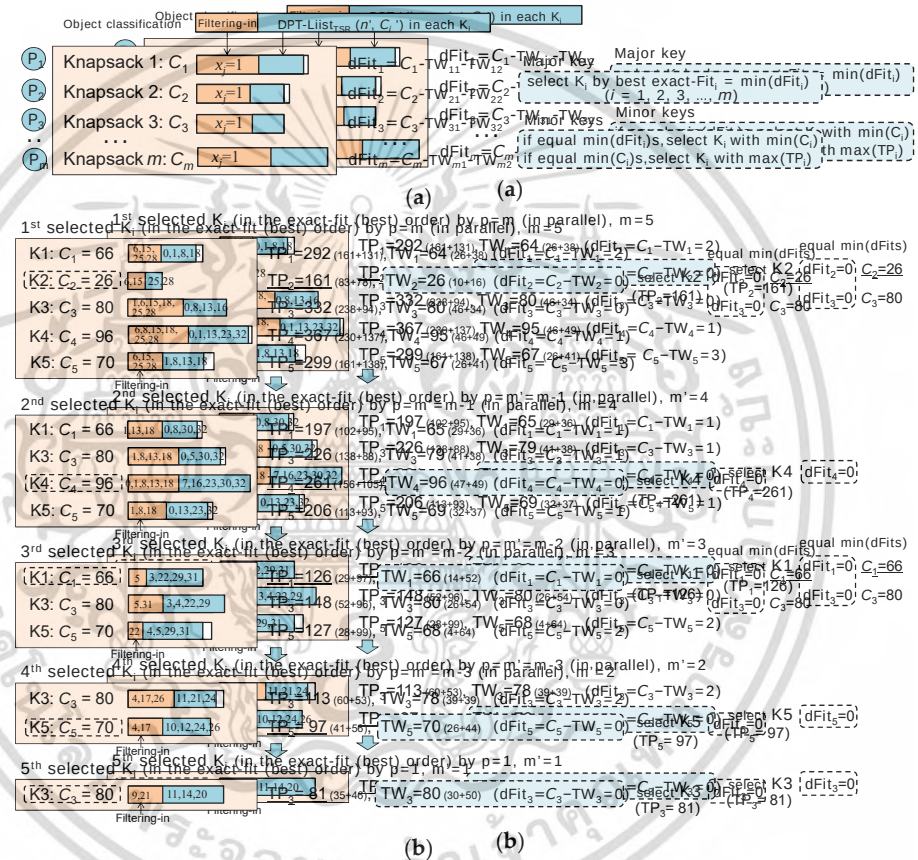


Figure 15. (a) The exact-fit policy plus efficient filtering for the best knapsack order and (b) an example of $n=33$, $m=5$, $C=(66, 26, 80, 96, 70)$ to find the best order (2, 4, 1, 5, 3) in $m(m+1)/2=15$ steps and the optimal result (726) by the multi-DPT-ListTSR + robust unbiased filtering.

Algorithm 11: Multi-DPT-ListTSR + robust unbiased filtering.

1. $n^* = n$;
2. for $(i = 1 \text{ to } m)$ do
3. do object classification and unbiased filtering (for Filter- tp) on n^* ;
4. apply DPT-ListTSR ($n' < 300$) on knapsack i (C_i');
5. call Algorithm 7 (preprocessing on remaining n' , C_i');
6. if $(i < m)$ start = (max tp , max tw) else start = (max tp , min tw);
7. call Algorithm 8 (X-tracking on n' for $solx$ from max tp);
8. Total profit = Total profit + Filter- tp + max tp ;
9. update n^* (exclude k_i selected objects of knapsack i);
10. end for i ;
11. return Total profit.

5. Analysis of Proposed Algorithms

The correctness of the DPT-List_{TSR} algorithm for solving the 0/1-KP was proven in Section 5.1 and its complexity was analyzed in Section 5.2. For solving the 0/1-mKP, the high (optimal) precision (as $m!$ orders) of the exact-fit (best) knapsack order was presented in Section 5.3. Finally, the 99% optimal precision of the robust unbiased filtering for solving the 0/1-KP and the 0/1-mKP were analyzed in Section 5.4.

5.1. Correctness of the DPT-List_{TSR} Algorithm

The DPT-List_{TSR} algorithm (in Section 3) was designed to solve the 0/1-KP in $O([n^2, nC])$ on the efficient lists, which can find the optimal solutions as the best DP (Algorithm 1: $O(nC)$ on a 2D-array ($n \times C$)) before being applied in each of m knapsacks.

Our DPT-List_{TSR} algorithm can reduce not only the redundant computing time but also the space consumption (of the basic DP: Algorithm 1) while retaining the correctness. Our focus is the DP transformation of the 2D array ($n \times C$) to the efficient lists of e-nodes. Our preprocessing (Algorithm 7) employs two (temporary) F-lists (of objects $j-1$ and j) to inherit all worth e-nodes (of objects 0 to $j-1$) and compute new e-nodes (improved tp values by the current object j) before saving only the original e-nodes in B-list j .

To clarify our correct transformation, Figure 16 shows the construction of e-nodes (of F-lists j) in Figure 5 ($n = 5$). For object $j = 1$ ($p_1 = 10, w_1 = 15$), Figure 16a displays the F-list j construction. After the initial copy of two e-nodes ($cn = (tp, d) = (0, 0)$ and $(4, 8)$) from F-list $j-1$ (while $cn.tp < p_1$ and $cn.d < w_1$) to F-list j , the rest of F-list j is fulfilled. For the first e-node $en = (0, 0)$ of F-list $j-1$, a new e-node $(10, 15)$ with $d = 15 < C$ and $tp = 10 > TP = 4$ is added to the end of F-list j . For the next $en = (4, 8)$, compute $d = 8 + 15 = 23 > C$ (no new e-node is added). For object $j = 3$ ($p_3 = 9, w_3 = 5$), Figure 16b shows the F-list j construction in three steps. After the initial copy of two e-nodes ($cn = (0, 0)$ and $(6, 4)$) from F-list $j-1$ (while $cn.tp < p_3$ and $cn.d < w_3$) to F-list j , the rest of F-list j is fulfilled. For $en = (0, 0)$, a new e-node $(9, 5)$ is added to F-list j . Second, for $en = (6, 4)$, a new e-node $(6 + 9, 4 + 5) = (15, 9)$ is added to F-list j . Third, for $en = (10, 12)$, this remaining e-node is not inherited, whereas a new e-node $(10 + 9, 12 + 5) = (19, 17)$ is added to F-list j . Note: Function “inherit remaining e-node” (in Algorithm 7: Line 6) is presented in Figure 16b; see the complex inherited results in Figure 6 ($n = 15$). Finally, our X-tracking (Algorithm 8) can find $solx[n]$ from the original e-nodes (on the B-lists in Figure 7), similar to the basic DP (on the 2D-array).

Algorithms 2022, 15, 366 FOR PEER REVIEW 20 of 35

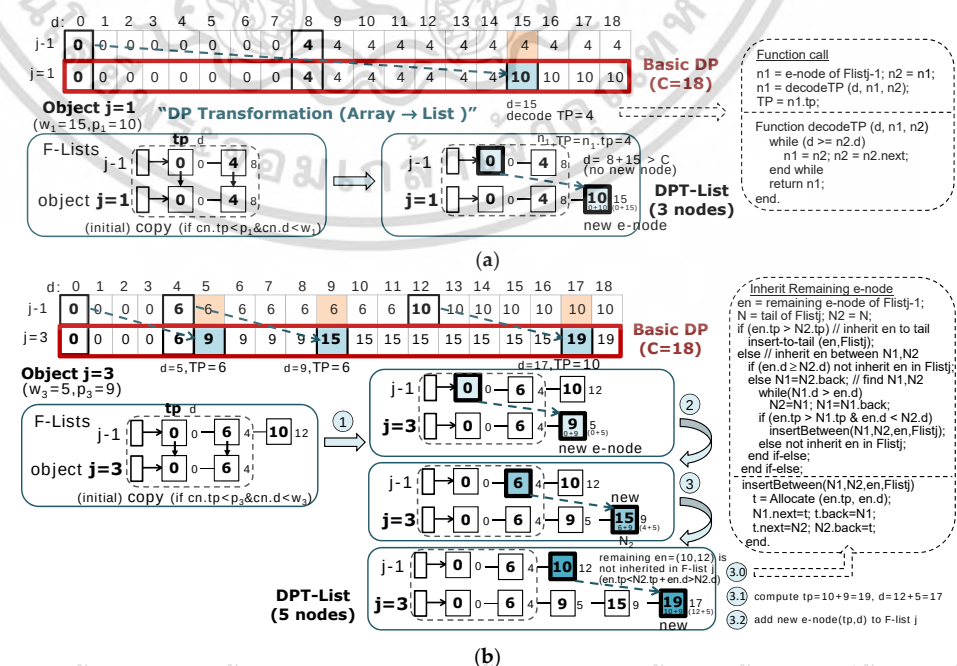


Figure 16. An example of the correct F-list j construction ($n = 5$ in Figure 5): (a) F-list $j = 1$ (add a new e-node to tail of F-list j) and (b) F-list $j = 3$ (add each of three new e-nodes to tail of F-list j).

5.2. Complexity Analysis of the DPT-List_{TSR} Algorithm

The time complexity of our DPT-List_{TSR} algorithm for solving the 0/1-KP is $O([n^2, nC])$, according to the efficient reduction of computing time and using space; see Figure 17 (our efficient time-space reduction). Our time complexity depends on the number of e-nodes of the (temporary) F-list j for all $j = 0, 1, 2, 3, \dots, n-1$, where $|F\text{-list } j| \leq 2|F\text{-list } j-$

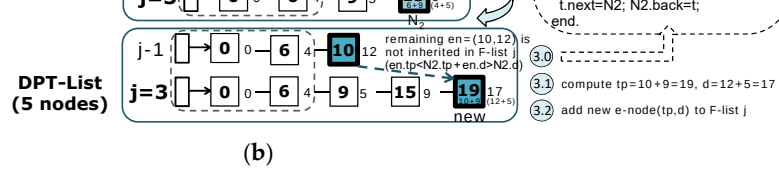


Figure 16. An example of the correct F-list j construction ($n = 5$ in Figure 5): (a) F-list $j = 1$ (add a new e-node to tail of F-list j) and (b) F-list $j = 3$ (add each of three new e-nodes to tail of F-list j).

5.2: Complexity Analysis of the DPT-List_{TSR} Algorithm

The time complexity of our DPT-List_{TSR} algorithm for solving the 0/1-KP is $O(n^2)$, according to the efficient reduction of computing time and using space; see Figure 17 (our efficient time-space reduction). Our time complexity depends on the number of e-nodes of the (temporary) F-list j for all $j = 0, 1, 2, 3, \dots$, where $1 \leq |F\text{-list } j| \leq 2|F\text{-list } j-1|$; see a simple example ($n = 5$) in Figure 5. The (initial) F-list j contains one e-node ((tp, d)). For object $p = 0$, there are at most two e-nodes (≤ 2 nodes). For object j (object nodes) and for any j (nodes), the time complexity of our DPT-List_{TSR} algorithm can be the best, average, or worst cases, depending on the datasets. Figure 5 displays one of the best cases ($n = 5, C = 18$, nodes = 19) and original e-nodes = 7, reduced from 15 elements) analysis; in this analysis, the best and worst cases can be derived as follows:

- **Best case:** Total steps (objects ($j = 0$ to $n - 1$)) are approximately $1 + 2 + 4 + \dots + 2^{(j-2)} + 1 + 2n \approx n(n+1)/2 = O(n^2)$.
- **Worst case (rarely occurs):** Total steps are approximately $1 + 2 + (4+4) + (8+8) + \dots + (2^{(j-2)} + 2^{(j-2)}) \times 2 \approx n \times C/2 = O(nC) = O(nC)$.

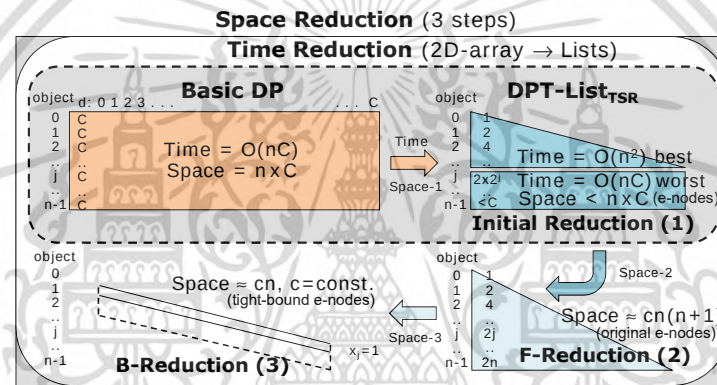


Figure 17. Time and space reduction of DPT-List_{TSR} for the 0/1-KP.

The worst case (the e-nodes of $F\text{-list}_{n-1} = |F\text{-list}_{n-1}| \approx C$) hardly occurs due to some remaining e-nodes of $F\text{-list } j - 1$ are not inherited to $F\text{-list } j$ (see a clarified example in Figure 16b) and no additional new e-nodes in $F\text{-list } j$ when considering some (worst) objects j (such as large w_j or tiny profit p_j) by two conditions: 1. $node.tp + p_j < tp[d]$ (no improved tp) and 2. $node.d + w_j > C$ (at $d + w_j$, object j cannot be packed in the knapsack), such as no additional new e-node for object $j = 4$ (in Figure 5). Figure 6 shows an example of a regular/average case ($n = 15, C = 40, nC = 600$, e-nodes = 223, and original e-nodes = 103). The time complexity of the average case arises in most datasets ($\approx (\text{best} + \text{worst})/2 < nC/2$). Since a weight w_j of object j can be $1 \leq w_j < C$, the average w_j is approximately C/n . For $w_j \geq C/n$, usually no e-node is added (because of the condition $node.d + w_j > C$).

For m knapsacks, the time complexity of our multi-DPT-list_{TSR} algorithm for solving the 0/1-mKP is $O(m[n^2, nC])$ with any effective knapsack order (including top nine orders) and $O(m^2[n^2, nC])$ for the exact-fit (best) order in $m(m+1)/2$ steps; see Section 5.3.

For space complexity, Figure 17 illustrates our three steps of space reduction: 1. e-nodes ($< nC$), 2. original e-nodes, and 3. tight bound of e-nodes; see Figure 5 ($n = 5, C = 18$) and Figure 7b ($n = 15, C = 40$). In our experiment, Section 6.1, displays the observed results ($n \leq 3000$), where after F-reduction the original e-nodes are a function of cn^2 ($< n^3$, $c = \text{a constant}$) and hence after B-reduction the original e-nodes are less than n^2 .

5.3. High (Optimal) Precision (as $m!$ Orders) of the Exact-Fit (Best) Knapsack Order

In our novel research track (Figure 8), we study by starting with the exact DP for the optimal solution in one knapsack to m knapsacks. In Section 3, we propose the DPT-List_{TSR} algorithm to find the optimal solution in $O(n^2, nC)$ for each knapsack. In Section 4, we propose the efficient order reduction for m knapsacks (over $m!$ orders), which are 1. the

$= a \text{ constant})$ and hence after B-reduction the original e-nodes are less than n^2 .

5.3. High (Optimal) Precision (as $m!$ Orders) of the Exact-Fit (Best) Knapsack Order

In our novel research track (Figure 8), we study by starting with the exact DP for the optimal solution in one knapsack to m knapsacks. In Section 3, we propose^{21 of 34} the DPT-List_{TSR} algorithm to find the optimal solution in $O([n^2, nC])$ for each knapsack. In Section 4, we propose the efficient order reduction for m knapsacks (over $m!$ orders), which are effective to obtain the optimal solution in $O(m! \log(m! \log C))$ for the regular datasets and $O(m! \log(m! \log C))$ for the irregular datasets (but 2, three exact-fit (best) DPT-List_{TSR} algorithm DPT-List_{TSR} for the regular datasets where regular DPT-List_{TSR} algorithm DPT-List_{TSR} applied in the internal and external effects in each knapsack and among k knapsacks and among m knapsacks).

For the regular datasets, an effective order is increasing C_i ; see Figure 18 (selecting k candidates (objects), $0 \leq i \leq m-1$, for for positions (knapsacks) with capacity C_i , $i = 1, 2, \dots, m$). In a company/organization, the profit (knowledge) and the weight (negative attitude/greedy weight). For large, the first top nine effective orders (in Section 4.1.1) are presented and later improved by the LS of m permutations, where the partial LS (first nine permutations of each order) is concerned in $O(m! \log(m! \log C))$ see Table 1 (the proposed reduction orders and all possible $m!$ orders (for $5 \leq m \leq 100$)).

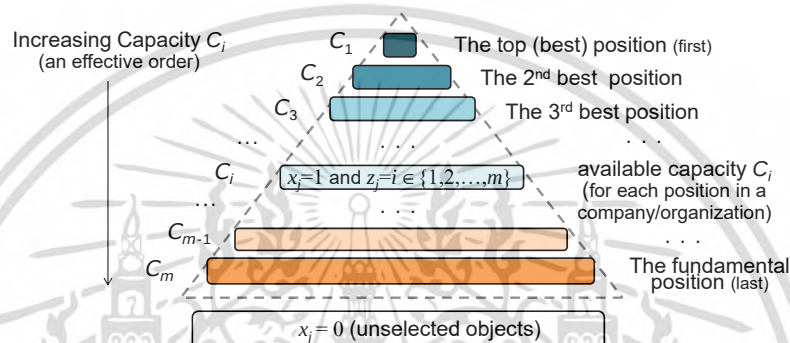


Figure 18. An example of increasing C_i (an effective knapsack order) for the 0/1-mKP.

Table 1. All possible ($m!$) knapsack orders and the proposed effective orders for the 0/1-mKP.

m Knapsacks	All Orders ($m!$)	Exact-Fit/Best ($m(m+1)/2$)	Top (9) Orders	Partial LS Min (9 m , 9×9)	Full LS (9 m)
5	120	15	9	45	45
6	720	21	9	54	54
7	5040	28	9	63	63
8	40,320	36	9	72	72
9	326,880	45	9	81	81
10	3,268,800	55	9	81	90
20	20!	210	9	81	180
50	50!	1275	9	81	450
100	100!	5050	9	81	900

For the irregular datasets, we can use $m!$ orders to achieve at least 99% optimal solutions but in exponential time $O(m! [n^2, nC])$. To reduce the time complexity and retain 99% optimal precision, the exact DPT-List_{TSR} algorithm is applied for not only the internal effect (for the optimal solution in each knapsack) but also the external effect (for the best order among m knapsacks). For the external effect, the DPT-List_{TSR} algorithm is used to find all exact-fit knapsacks before selecting the best knapsack and repeating the same process for the remaining objects and knapsacks. The exact-fit (best) order (Algorithm 10) is determined in $m(m+1)/2$ steps in $O(m^2 [n^2, nC])$. In particular, the best knapsack K_i is selected by the best exact-fit _{i} = $\min(dFit_i)$; $dFit_i = C_i - TW_i$, where the exact TP_i and TW_i are computed by the DPT-List_{TSR} algorithm in each of the available knapsacks; see Figure 11 ($n = 33$, $m = 5$). Moreover, if there are equal $\min(dFit_i)$ s in more than one K_i (in the critical decision) in step 2 of Algorithm 10, then three proper policies are used to find the best of three best solutions. See the confirmed results (99% optimal solutions) in Section 6.3.

5.4. High (Optimal) Precision of the Robust Unbiased Filtering

The exact DP + unbiased filtering [1] can solve the 0/1-KP in $O(n + C')$, $C' \ll C$ with at least 95% optimal precision. Thus, we can adopt the process of unbiased filtering for the

0/1-mKP. Initially, all objects are classified (into four groups) by dynamic weighting (dw), integrated from three effective ranks ($P/W, P, W$); see Figure 19a, where two parameters (α, β) were defined to handle special objects (called outliers) in unbiased filtering [1] for the 0/1-KP with 95% optimal solutions. In this study, to achieve 99% optimal solutions for the 0/1-mKP, the (γ, α, β) parameters are introduced by studying all datasets (i.e., most datasets are regular ($\approx 90\%$) and irregular datasets are $\approx 10\%$). In our robust unbiased filtering, the (γ, α, β, U) parameters are determined in Equations (10)–(14), where the critical objects are in low rank in Group 1 (γ) and Group 2 (α) and in top rank in Group 4 (β), and most uncertain objects (U) are in Group 3. Figure 19a shows that some objects (around the critical points in the three ranks ($P/W, P$, or W)) are the critical objects (γ, α, β). All uncertain/critical objects ($n' < 300$) can be solved by the DPT-List_{TSR} algorithm in $O(n + C')$. Figure 19b shows the idea of 99% optimal precision (due to our robust unbiased filtering) in each knapsack of the 0/1-mKP; see in the observed results (in Section 6).

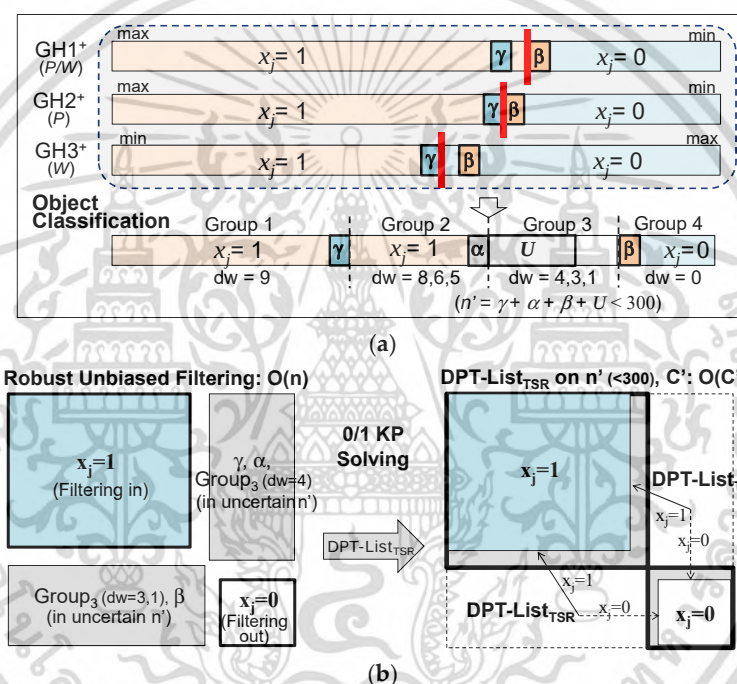


Figure 19. High (optimal) precision of robust unbiased filtering for each knapsack of 0/1-mKP: (a) object classification and (b) robust unbiased filtering and DPT-List_{TSR} for remaining $n' < 300$.

6. Experimental Results

Experiments were conducted to evaluate the DPT-List_{TSR} algorithm and robust unbiased filtering for the 0/1-KP (Section 6.1) to ensure at least 99% optimal precision (in each knapsack) before applying to the 0/1-mKP (Sections 6.2 and 6.3) by the multi-DPT-List_{TSR} algorithm (the best knapsack order for m knapsacks) and robust unbiased filtering.

6.1. Results of the DPT-List_{TSR} (One Knapsack) and Robust Unbiased Filtering

For solving the 0/1-KP, we generated a variety of random datasets (dynamic seeds) with the 0/1-KP, of uniform distributions for both the profits and weights of n objects with a number of $n \leq 10,000$. The experiments to obtain the profits and weights performance of robust unbiased filtering. In the experimental results, performance of robust unbiased filtering could find the exact solutions in DPT of the datasets (in 10,000) while there were 223 (of 10,000) datasets for which the recent TSF $\leq 10,000$ unbiased filtering [1] could not find the optimal solutions. Table 2 shows the empirical results of the first 23 of 223 special datasets (or irregular datasets), $n = 12, 14, 21, 26, 39, \dots, 385$ (observed on $n = 5, 6, 7, \dots, 9999, 10,000$).

Table 2. Experimental results of irregular datasets (23 of 223), observed from $n = 5, 6, 7, \dots, 10,000$.

Total Weight (solwt)	Total Profit (soltp)
TSR +	TSR +

Table 2. Experimental results of irregular datasets (23 of 223), observed from $n = 5, 6, 7, \dots, 10,000$.

n	C	Total Weight (<i>soltw</i>)			Total Profit (<i>soltp</i>)		
		DPT-List (Opt.)	DPT + rFilter	TSR + uFilter [1]	DPT-List (Opt.)	DPT + rFilter	TSR + uFilter [1]
12	96	96	96	92	282	282	280
14	112	111	111	109	365	365	362
21	168	168	168	167	500	500	498
26	208	208	208	203	637	637	636
39	312	312	312	312	868	868	866
45	360	360	360	360	1177	1177	1172
73	510	510	510	509	1640	1640	1639
80	559	559	559	558	1712	1712	1711
81	566	566	566	566	1888	1888	1886
143	1000	1000	1000	1000	3084	3084	3082
147	1028	1028	1028	1028	3250	3250	3239
155	1084	1084	1084	1083	3440	3440	3437
166	1161	1161	1161	1161	3617	3617	3616
182	1273	1273	1273	1273	3967	3967	3966
197	1378	1378	1378	1378	4534	4534	4533
199	1392	1392	1392	1391	4561	4561	4560
247	1481	1481	1481	1480	4822	4822	4821
276	1655	1655	1655	1655	5446	5446	5445
286	1715	1715	1715	1715	5889	5889	5888
316	1895	1895	1895	1895	6266	6266	6257
329	1973	1973	1973	1973	6484	6484	6469
360	2159	2159	2159	2159	6985	6985	6984
385	2309	2309	2309	2309	7710	7710	7709

Table 3 presents the optimal performance of our DPT-List_{TSR} + robust unbiased filtering for at least 99% optimal solutions (on $n \leq 10,000$) compared to our previous work [1]. In this experiment, there exist irregular datasets $\approx 10\%$ (from all 10,000 random datasets), where unbiased filtering (in TS_{Reduction}) [1] could handle $\approx 5\%$ and our robust unbiased filtering (in DPT-List_{TSR}) could handle $\approx 9.9\%$. Table 4 displays our space reduction, observed on $n \leq 3000$ (with runtime < 1 min per n). Our F-reduction can save space 69–92%, and our B-reduction can save space 84–93%.

Table 3. Optimal precision of the DPT-List_{TSR} + robust filtering ($n \leq 10,000$).

n : Datasets	DPT-List + Robust Filtering			TSR + Unbiased Filtering [1]		
	not Opt.	Optimal	Precision	not Opt.	Optimal	Precision
$5 \leq n \leq 100$	0	95	99.9%	9	86	90.0%
$5 \leq n \leq 200$	0	195	99.9%	16	179	92.0%
$5 \leq n \leq 500$	0	495	99.9%	23	472	95.4%
$5 \leq n \leq 2000$	0	1995	99.9%	28	1967	98.6%
$5 \leq n \leq 5000$	0	4995	99.9%	109	4886	97.8%
$5 \leq n \leq 10,000$	0	9995	99.9%	223	9772	97.8%

Note: 99% optimal solutions refer to “For 100 observed datasets, we could find 99 optimal solutions”.

Table 4. Performance (percentage) of space reduction by the DPT-List_{TSR} ($n \leq 3000$).

n	$n \times C$ (Full Space)	e-Nodes (1. Initial Reduction)		Original e-Nodes (2. F-reduction)		Tight-Bound e-Nodes (3. B-Reduction)	
5	90	13	86%	7	92%	6	93%
15	600	223	63%	103	83%	50	92%
50	17,450	6559	62%	2465	86%	1917	89%
100	69,900	35,852	49%	15,263	78%	10,849	84%
200	239,800	150,518	37%	66,883	72%	35,932	85%

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับงานวิจัยทางวิชาการเท่านั้น ไม่ควรนำข้อมูลไปใช้ในการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Table 4. Cont.

n	$n \times C$ (Full Space)	e-Nodes (1. Initial Reduction)		Original e-Nodes (2. F-reduction)		Tight-Bound e-Nodes (3. B-Reduction)	
500	1,250,000	915,303	27%	374,729	70%	173,786	86%
1000	5,000,000	3,832,827	23%	1,566,414	69%	692,691	86%
1500	11,250,000	8,669,932	23%	3,364,525	70%	1,581,133	86%
2000	11,998,000	10,322,643	14%	3,293,641	73%	1,115,040	91%
2500	18,747,500	16,071,768	14%	5,338,435	72%	1,752,052	91%
3000	26,997,000	23,225,460	14%	7,497,627	72%	2,620,552	90%

6.2. Results of the Multi-DPT-List_{TSR} (m Knapsacks) and Robust Unbiased Filtering

For solving the 0/1-mKP, the optimal performance of our multi-DPT-List_{TSR} algorithm with the proper (knapsack) orders (i.e., top nine orders, Latin squares, the exact-fit (best) order) was evaluated by comparison to the optimal solutions. In practice, the fast response time of our multi-DPT-List_{TSR} with robust unbiased filtering was observed, while retaining the high performance. In this experiment, a number of random datasets were generated for n ($\leq 10,000$) objects and m (≤ 100) knapsacks with a variation of capacities (i.e., $C_i \pm 10$, $C_i \pm 15$, $C_i \pm 20$, etc.). In addition, the benchmark datasets [34] were observed and the empirical results were compared to the optimal solutions.

In performance (total profit) evaluation, we focus on the investigation of 1. the exact-fit best (knapsack) order in our multi-DPT-List algorithm (for 99% optimal solutions in theory) and 2. the robust unbiased filtering (in polynomial time) to confirm 99% optimal solutions. We implemented our multi-DPT-List (the exact-fit best order) and the fast multi-DPT-List + robust filtering compared to the optimal solutions. For the practical polynomial-time evaluation, the fast response time of our multi-DPT-List + robust filtering was compared to the quick multi-GH⁺ (a well-known heuristic algorithm).

For the performance comparison, the (known) optimal solutions of the 0/1-mKP (in Column 2 of Tables 5–10) can be computed by using a large knapsack ($C^s = \sum_{i=1}^m C_i$) by the exact DP or our DPT-List in the (regular and irregular) datasets.

The implemented programs of three main approaches (in this experiment) are

Exact	1.1 Multi-DPT-List (exact-fit best order)	$O(m^2[n^2, nC])$
Exact + Filtering	2.1 Fast multi-DPT-List + filtering (exact-fit best order)	$O(m^2n)$
	2.2 Fast multi-DPT-List + filtering (top 9 orders + partial LSs)	$O(mn)$
	2.3 Fast multi-DPT-List + filtering (top 9 orders)	$O(mn)$
Heuristic	3.1 Quick multi-GH (P/W rank) (top 9 orders)	$O(mn)$
	3.2 Improved multi-GH ⁺ (P/W rank) (top 9 orders)	$O(mn)$
	3.3 Improved multi-GH ⁺ (P/W rank) (top 9 + full LS orders)	$O(m^2n)$

Table 5. Results (total profits) of datasets with capacities $C \pm 10$ ($m = 2$).

$m = 2$	Optimal	mDPT-L $m! O(m[n^2, nC])$	mDPT-L + Filter $m! O(mn)$	mGH $m! O(mn)$	mGH ⁺ $m! O(mn)$
n		$m! = 2$	$m! = 2$	$m! = 2$	$m! = 2$
15	315	315	315	308	310
20	420	420	420	393	420
30	800	800	800	790	796
40	1050	1050	1050	1022	1047
50	1019	1019	1019	1006	1013
100	2359	2359	2359	2313	2357
200	3878	3878	3878	3860	3870
300	6202	6202	6202	6171	6196
400	7686	7686	7686	7654	7683
500	9074	9074	9074	9045	9072
1000	18,038	18,038	18,038	18,002	18,031

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการใช้งานเท่านั้น ไม่อนุญาตให้นำไปใช้เพื่อการค้าโดยไม่ได้รับอนุญาต
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Table 6. Results (total profits) of datasets with capacities $C \pm 15$ ($m = 3$).

$m = 3$	Optimal	mDPT-L $m! O(m[n^2, nC])$	mDPT-L + Filter $m! O(mn)$	mGH $m! O(mn)$	mGH+ $m! O(mn)$
n		$m! = 6$	$m! = 6$	$m! = 6$	$m! = 6$
15	327	327	327	320	322
20	466	466	466	454	466
30	840	840	840	829	839
40	1103	1103	1103	1090	1099
50	1067	1067	1067	1031	1062
100	2427	2427	2427	2390	2426
200	3949	3949	3949	3908	3943
500	9150	9150	9150	9133	9148
1000	18,112	18,112	18,112	18,088	18,110
2000	25,547	25,547	25,547	25,524	25,544

Table 7. Results (total profits) of datasets with capacities $C \pm 20$ ($m = 4$).

$m = 4$	Optimal	mDPT-L $m! O(m[n^2, nC])$	mDPT-L + Filter $m! O(mn)$	mGH $m! O(mn)$	mGH+ $m! O(mn)$
n		$m! = 24$	$m! = 24$	$m! = 24$	$m! = 24$
20	495	495	495	490	490
30	884	884	884	884	884
40	1200	1200	1200	1187	1192
50	1137	1137	1136	1121	1136
60	1504	1504	1504	1472	1499
100	2548	2548	2548	2534	2546
200	4098	4098	4098	4071	4088
500	9318	9318	9318	9297	9314
1000	18,284	18,284	18,284	18,249	18,280
2000	25,760	25,760	25,760	25,735	25,752
3000	38,941	38,941	38,941	38,899	38,930

Table 8. Results (total profits) of datasets ($C \pm 20$), $n = 1000$ – 5000 ($m = 6$ – 50).

m	Optimal	mDPT-L $O(m^2[n^2, nC])$	mDPT-L + LS Filter $O(m^2n)$	mDPT-L + Filter $O(mn)$	mGH+ $O(mn)$	mGH+ + LS $O(m^2n)$
$n = 1000$		Best	$9 \times m$	9	9	$9 \times m$
6	18,541	18,541	18,541	18,541	18,535	18,535
7	18,703	18,703	18,703	18,703	18,684	18,693
8	18,889	18,889	18,889	18,889	18,879	18,884
9	19,079	19,079	19,079	19,079	19,068	19,071
$n = 2000$		Best	9×9	9	9	$9 \times m$
12	27,769	27,769	27,769	27,769	27,742	27,753
13	28,154	28,154	28,154	28,154	28,127	28,127
14	28,547	28,547	28,547	28,547	28,498	28,517
15	28,948	28,948	28,948	28,948	28,900	28,935
$n = 5000$		Best	9×9	9	9	$9 \times m$
20	54,736	54,736	54,736	54,736	54,680	54,695
30	62,496	62,496	62,496	62,496	62,410	62,450
40	72,417	72,417	72,417	72,417	72,323	72,331
50	84,051	84,051	84,051	84,051	83,943	83,963

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Table 9. Results (total profits) of datasets ($C \pm 10, 20$), $n = 9000$ ($m = 40$ – 90).

m	Optimal	mDPT-L $O(m^2[n^2, nC])$	mDPT-L + LS Filter $O(m^2n)$	mDPT-L + Filter $O(mn)$	mGH ⁺ $O(mn)$	mGH ⁺ + LS $O(m^2n)$
$C_i \pm 10$		Best	9×9	9	9	9 m
40	53,669	53,669	53,669	53,669	53,494	53,520
50	54,803	54,803	54,803	54,803	54,538	54,655
60	58,614	58,614	58,614	58,614	58,425	58,450
70	64,018	64,018	64,018	64,018	63,689	63,784
$C_i \pm 20$		Best	9×9	9	9	9 m
40	85,380	85,380	85,380	85,380	85,203	85,286
50	100,859	100,859	100,859	100,859	100,683	100,724
60	118,729	118,729	118,729	118,729	118,525	118,616
70	138,195	138,195	138,195	138,195	137,963	138,038
80	158,983	158,983	158,983	158,983	158,722	158,857
90	180,054	180,054	180,054	180,054	179,873	179,981

Table 10. Results (total profits) of irregular datasets ($m = 3$ – 7 , $n \leq 10,000$).

$n \leq 10,000$	Optimal	mDPT-L $O(m^2[n^2, nC])$	mDPT-L + LS Filter $O(m^2n)$	mDPT-L + Filter $O(mn)$	mGH ⁺ $O(mn)$	mGH ⁺ + LS $O(m^2n)$
$m:n$		$m!$	$m!$	$m!$	$m!$	$m!$
3:51	1318	1318	1317	1317	1315	1315
3:73	1727	1727	1725	1725	1725	1725
4:33	752	752	747	747	747	747
4:49	1264	1264	1263	1263	1254	1254
4:50	1137	1137	1136	1136	1136	1136
4:65	1565	1565	1563	1563	1563	1563
$m:n$		Best	9 m	9	9	9 m
5:89	2366	2366	2365	2365	2359	2359
7:77	1834	1834	1833	1833	1829	1830
7:138	3263	3263	3262	3262	3256	3256
7:148	3780	3780	3799	3799	3773	3777

First, we evaluated the performance of our mDPT-List and fast mDPT-List + filtering with $m!$ orders (for small $m = 2, 3, 4$), compared to the optimal solutions. For $m \leq 4$, our approach can find the optimal solutions in most datasets; see Columns 3 and 4 in Tables 5–7. For $m > 4$, we investigated the effect of robust filtering plus the top nine effective orders and partial Latin squares (Columns 4 and 5 in Tables 8 and 9). For the regular datasets ($n \leq 10,000$, $m \leq 100$), our mDPT-List + filtering (top nine orders) yielded 99% optimal solutions.

Second, we aimed to compare among the fast polynomial-time algorithms ($O(mn) - O(m^2n)$) by observing the effect of the top nine effective orders; see Columns 4–7 in Tables 8 and 9. For $m \leq 100$ and $n \leq 10,000$, the results (total profits) of our mDPT-List + filtering (in Column 5) were compared to those of the quick mGH⁺ (P/W rank) in Columns 6–7 (response time < 1 s). For the regular datasets, our fast mDPT-List + filtering (in Column 5) yielded most optimal solutions, while the results of the quick mGH⁺ (in Column 6) and its improvement with LS of 9 m orders (in Column 7) were far from the optimal solutions, especially when using many knapsacks ($m > 10$). Note: GH (P/W rank) is frequently used in many meta-heuristic algorithms (i.e., GA, swarm, etc.) for the good initial solutions to solve the 0/1-KP and GH⁺ is used in unbiased filtering [1] (p. 199) and in robust unbiased filtering (in this study). In the comparison, we use the improved mGH⁺ with the Latin squares of top nine orders (for 9 m orders/iterations to emulate the evolution process of GA/swarm optimization). For most datasets, the mGH⁺ (9 m orders) could not find the optimal solutions in each knapsack since it included uncertain object(s) in the solution.

However, it is not the problem in our robust unbiased filtering since all uncertain objects ($n' < 300$) were considered by the exact DPT-List with 99% optimal precision.

In our initial observation and analysis, for $m = 2$ ($n \leq 10,000$), the mDPT-List + filtering ($m!$ orders) in Table 5 yielded 100% optimal solutions. For $m = 3, 4$ ($n \leq 10,000$), our approach ($m!$ orders) in Tables 6 and 7 yielded 99.9% optimal solutions. Next, we found that (for the irregular datasets) the top nine orders were not sufficient to find the optimal solutions $\geq 99\%$, especially $m \geq 5$. Then, we investigated the effect of the LS of the top nine orders (see Column 4 in Tables 8–10). Moreover, Tables 10 and 11 report the irregular datasets found during the execution of each dataset ($n \leq 10,000$), where any dataset is called “irregular” when the top nine orders could not find the optimal solution. For $m \geq 5$, we performed an intensive study and experiment to observe each of $n \leq 10,000$ ($m \leq 100$) and found that a number of irregular datasets increased when m increased (see Column 6 in Table 11). Hence, the exact-fit (best) knapsack order is applied to solve this problem.

Table 11. Observed frequency of nonoptimal solutions (in $n \leq 10,000$ per m), $m = 5, 6, 7, \dots, 53, 54$.

$n \leq 10,000$	mDPT-L: $O(m^2[n^2, nC])$		mDPT-L + Filtering: $O(m^2n)$		
m	Best	9 m (LS)	Best	9 m (LS)	Top 9
5	0	0	0	1	2
6	0	0	0	1	3
7	0	1	0	3	6
8–14	0	0	0	2	1.6 (ave. per m)
15–19	0	0	0	0	1.6 (ave. per m)
20–24	0	0	0	0	2.6 (ave. per m)
25–29	0	0	0	0	4.8 (ave. per m)
30–34	0	0	0	0	5.4 (ave. per m)
35–39	0	0	0	0	13.8 (ave. per m)
40–44	0	0	0	0	31.4 (ave. per m)
45–49	0	0	0	0	34.8 (ave. per m)
50–54	0	0	0	0	69.2 (ave. per m)

Note: When observing the irregular datasets, using top 9 orders (Column 6) in our mDPT-List + filtering could not find the optimal solutions in approximate 69 datasets (in average) of $n \leq 10,000$, $m = 54$ in the (random) regular and irregular datasets, while using the best order (Column 4) could find all optimal solutions.

After performing the intensive study and comparison (on large $n \leq 10,000$), we found that for the irregular datasets, our mDPT-List + robust unbiased filtering (the exact-fit (best) order in $O(m^2n)$) could find at least 99% optimal solutions as those of the original mDPT-List ($O(m^2[n^2, nC])$); see a report of observed frequency of nonoptimal solutions (0%) of our approach in Table 11 (Column 4).

Finally, we performed an extra experiment to evaluate the performance of our mDPT-List on the benchmark datasets [34] available at <http://or.dei.unibo.it/library> (accessed on 13 June 2020). Tables 12 and 13 show the empirical results of our mDPT-List (the best knapsack order in $O(m^2[n^2, nC])$) and our fast mDPT-List + robust filtering (the best knapsack order in $O(m^2n)$), compared to the regular mDP ($O(m^2nC)$) and the optimal solutions.

For ($n:m = 100:10$) 10 datasets [34], the results (in Table 12) showed that our mDPT-List (without/with filtering (the best order, LS orders, top nine orders)) could find the optimal solutions (Columns 4–7), while the results (Column 8) of the quick mGH⁺ (9 m orders in 9 m iterations) were not optimal.

Table 12. Results (total profits) of 10 benchmark datasets ($n = 100, m = 10$).

Research Approach		Exact		Exact + Filtering		Heuristic
		mDP	mDPT-L	mDPT-L + Filter: $O(m^2n)$		mGH ⁺
$n:m$	Optimal	Best	Best	Best	9 m	9 m
100:10-1	26,797	26,797	26,797	26,797	26,797	26,763
100:10-2	24,116	24,116	24,116	24,116	24,116	24,093
100:10-3	25,828	25,828	25,828	25,828	25,828	25,812
100:10-4	24,004	24,004	24,004	24,004	24,004	23,977
100:10-5	23,958	23,958	23,958	23,958	23,958	23,933
100:10-6	24,650	24,650	24,650	24,650	24,650	24,614
100:10-7	23,911	23,911	23,911	23,911	23,911	23,886
100:10-8	26,612	26,612	26,612	26,612	26,612	26,579
100:10-9	24,588	24,588	24,588	24,588	24,588	24,565
100:10-10	24,617	24,617	24,617	24,617	24,617	24,591

Table 13. Results (total profits) of 20 benchmark datasets ($200 \leq n \leq 500, 20 \leq m \leq 50$).

Research Approach		Exact		Exact + Filtering		Heuristic
Over Packing *		mDPT-L $O(m^2[n^2, nC])$		mDPT-L + Filter $O(m^2n)$		mGH ⁺ $O(m^2n)$
$n:m$	Optimal	Best of this study	Best +extra	Best	Best	9 m
200:20-1	80,260 *	80,205	80,205	80,163	80,196	80,121
200:20-2	80,171 *	80,122	80,122	80,122	80,121	80,069
200:20-3	79,101 *	79,083	79,083	79,061	79,083	79,041
200:20-4	76,264 *	76,208	76,208	76,174	76,174	76,149
200:20-5	79,619	79,619	79,619	79,581	79,581	79,515
200:20-6	76,749 *	76,711	76,711	76,711	76,711	76,612
200:20-7	76,543 *	76,474	76,474	76,429	76,474	76,402
300:30-1	121,806 *	121,756	121,742	121,742	121,756	121,654
300:30-2	119,877 *	119,828	119,828	119,795	119,828	119,743
300:30-3	119,806 *	119,762	119,762	119,756	119,749	119,684
300:30-4	115,567 *	115,556	115,529	115,516	115,556	115,434
300:30-5	117,204 *	117,175	117,175	117,160	117,168	117,065
300:30-6	118,516 *	118,493	118,493	118,493	118,450	118,386
300:30-7	115,793 *	115,752	115,752	115,706	115,693	115,641
300:30-8	123,664 *	123,624	123,624	123,620	123,620	123,552
500:50-1	205,672 *	205,645	205,645	205,645	205,645	205,488
500:50-2	199,868 *	199,781	199,775	199,775	199,781	199,681
500:50-3	202,321 *	202,286	202,286	202,277	202,277	202,164
500:50-4	136,669 *	136,657	136,657	136,653	136,652	136,595
500:50-5	135,806 *	135,796	135,795	135,795	135,796	135,736

Note: The symbol * (in Column 2) means that the (extra) solution may be overpacking.

For ($n:m = 200:20, 300:30, 500:50$) 20 datasets [34], most optimal solutions of these critical datasets were unknown (see Table 13) since the DP-packing in one large knapsack ($C^s = \sum_{i=1}^m C_i$) may be overpacking. Figure 20a shows an example of overpacking, when some objects in the critical datasets (such as some valuable objects j (high p_j/w_j) but large w_j) cannot be packed in any knapsack i with capacity C_i , except in the extra space of one knapsack of large capacity C^s . In these critical datasets, our mDPT-List with the best order (in Columns 5 and 6) yielded good results, which were close to or equal to the optimal solutions and outperformed those of LSs (9m) of top nine orders (in Columns 7–8).

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Figure 20. (a) A policy (in the space of parameters) with large impact in critical datasets and (b) an example of eight effective policies to handle the critical decisions).

In our contribution, we focus on the large The Fast mDPT-List (the best order) filtering (top order) in $\mathcal{O}(mn)$ is good for the regular datasets with 99% optimal solutions (Tables 5-8 and 12). For the irregular datasets (Table 11), our fast mDPT-List (the best order) filtering in $\mathcal{O}(m^2n)$ can find 99% optimal solutions similar to our original mDPT-List (the best order). Thus, for the regular and irregular datasets, our fast mDPT-List (the best order) is sufficient to achieve 99% optimal solutions in polynomial time $\mathcal{O}(m^2n)$. Moreover, for the critical/special benchmark datasets, we have intensively studied by the experiment (in Section 6.3) to improve the solutions (Column 4 in Table 13).

6.3. Extra Experiment and Additional Improvement on Critical Datasets

To improve the results of the critical datasets (benchmark datasets [34]), we have to find all possible critical decisions, such as 1. nonunique X_s (in X -tracking (see an example in Figure 10) in Section 4.1.1) and 2. equal $\min(dFit_i)$ in more than one knapsacks (in the exact-fit (best) order (see an example in Figure 11) in Section 4.1.2) and provide the right policies to handle them. Clearly, if there is only one $\min(dFit_i)$, $dFit_i = C_i - TW_i$, we can select the best knapsack K_i directly for the best order. By the DP packing, there may be many equal $\min(dFit_i)$ s in $K_i - K_j$, but only one K_i is selected (at a time), and this decision may cause the local optimal problem. To handle this problem, the top three effective policies are introduced in Algorithm 10, and the best of three best results is our final solution. However, to achieve the better results of these critical/special datasets, we add the other effective policies 4-8 in Algorithm 10 (step 2) to cover the other critical decisions; see an example in Figure 20b, i.e., select K_i with $\min(dFit_i)$ at the first, last, second, before last, and mid policies (in policies 4-8). Figure 20b shows the detail of selecting the best K_i ($m = 10$ knapsacks (in policies 4-8). Figure 20c shows the detail of selecting the best K_i ($m = 10$ knapsacks are six $\min(dFit_i)$ s 1, 8, 10, 12, 25, 27) with eight critical decisions (i.e., assume there are six $\min(dFit_i)$ s 6, 7, 9, 10, 11, 12) for selecting the best K_i (in the same order) with one policy for one result (1, 3, 7, 9, 10) for selecting the best K_i (in the best order) with one policy for one result (1, 3, 7, 9, 10). In this experiment, the improved results (max best order) with one policy for one result (1, 3, 7, 9, 10) were stable under these eight policies. In each dataset, the results (Column 14) were improved due to the exact DP list packing plus the proper critical handling by our eight policies for the best knapsack order.

In this regard, on 20 critical datasets (in Table 10), our unbiased filtering (using knapsack three policies) yielded (4 of 20) best results (Column 3), which outperformed the top three policies yielded (9 of 20) best results (Column 6), which outperformed the results (Column 7) of using LSs of top nine orders (9 m). In the superior improvement of our eight effective policies (in the extra experiment), the extra mdPT-List yielded (16 of 20) best results (Column 4), while the other 4 of 20 best results (Column 3) were fulfilled by robust filtering due to the (unbiased) preselecting and the less problem of nonunique

(Column 7) of using LSs of top nine orders (9 m). In the superior improvement of our eight effective policies (in the extra experiment), the extra mDPT-List yielded (16 of 20) best results (Column 4), while the other 4 of 20 best results (Column 3) were fulfilled by robust filtering due to the (unbiased) preselecting and the less problem of nonunique Xs in the X-tracking by DPT-List (on small $n' < 300$) in each knapsack. Obviously, for the unique X, our mDPT-List (with/without filtering) yields the same result.

In addition, the response times of three mDP algorithms (basic mDP, mDPT-List, and mDPT-List + filtering) were compared in this experiment (under the same 99% optimal precision). In theory, three different time complexities of these mDP algorithms are 1. $O(m^2n^2C)$ in the basic mDP (orders), 2. $O(m^2n^2C)$ in the mDPT-List (the best order), and 3. $O(m^2n)$ in the fast mDPT-List + filtering (the best order). Due to our efficient unbiased filtering, the response time of our fast mDPT-List + filtering for $n = 20,000$ and $m = 20$ was less than one second, that of the best mDPT-List for $n = 20,000$ was 10 min, that of the basic mDP for $n = 20,000$ was more than one hour, and so on for other large n .

Next to simplify the comparison and discussion for the 0/1-mKP solving with the critical datasets, we employ the triple-right rule (right man, right place, and right time). Figure 21 displays the improvement of our multi-DPT-List algorithm (our research track in Figure 8) from 199% optimal solutions in efficient time.

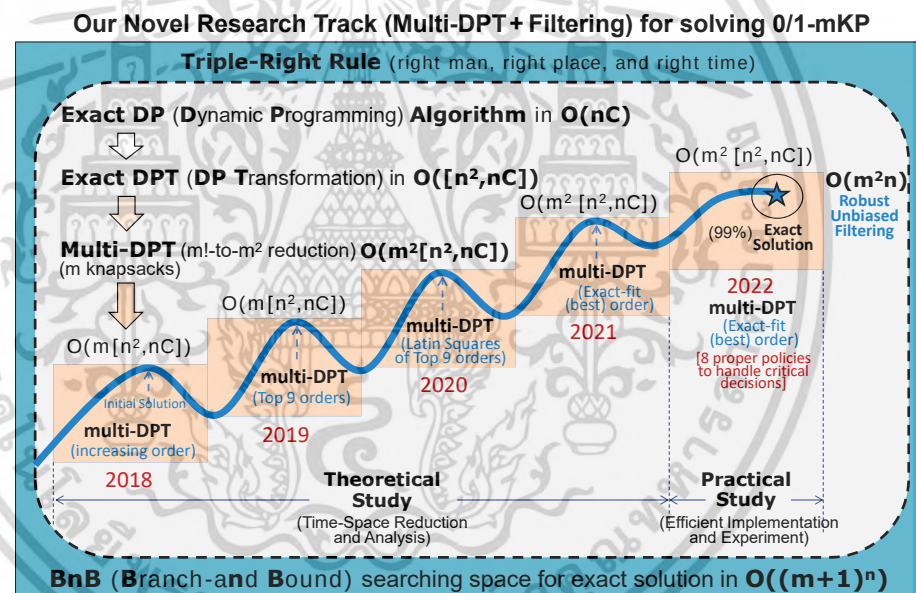


Figure 21. The improvement of our multi-DPT-List algorithm for solving the 0/1-mKP (m knapsacks) when dealing with the critical datasets.

In theory, the exact BnB algorithm can find the exact solution of the 0/1-mKP but in $O((m+1)^n)$ with the right man and right place but not the right time. In practice, the multi-GH* (with 9 m LS-orders) can find the good solutions in polynomial time but those solutions may not be optimal because it only confirms the right time rule. In this study (Figure 21), we study and apply the exact DPT (the man) and the efficient unbiased filtering (in practice) to achieve triple-right packing (packing a right man, right place, right time, and right time) $O(m^2n)$ in the multi-DPT reduction, see the highlight space of right space or not in Figure 22. This right bound reduction starts with the exact DPT for one knapsack (selecting the right object, uses the exact-fit (best) order for n knapsacks (putting the right object in the right knapsack), and ends with robust unbiased filtering (putting the right object in the right knapsack at the right time).

The comparison of our multi-DPT-List + robust filtering and the recent HyMKP [34] is demonstrated in Table 14. In practice (with large n ($< 10,000$ in this experiment)), our multi-DPT-List + robust filtering yielded 99% optimal results in $O(m^2n)$; see results in Tables 5–12, while the results of quick multi-GH* in $O(m^2n)$ were not optimal. For the HyMKP study [34], there is no available result for $n > 500$ since for large n the partial BnB (MULKNAP program) in the HyMKP (Algorithm 5) may not find the optimal solution in τ secs. Then, the reflect multi-graph MKP (with increasing n -weights (w)) in Algorithm 6

while the results of quick multi-GH⁺ in $O(m^2n)$ were not optimal. For the HyMKP study [34], there is no available result for $n > 500$ since for large n the partial BnB (MULKNAP program) in the HyMKP (Algorithm 5) may not find the optimal solution in τ secs. Then, the reflect multi-graph MKP (with increasing n -weights (w_j) in Algorithm 6 ($O(mnC)$) can provide the comparable results to our multi-DPT-List (with top nine orders) in the regular datasets, according to the looping on weights and C , similar to the DP (Algorithm 1) for each of m knapsacks. For the irregular datasets, v -rounds of decompositions of HyMKP are used to improve the initial solution. However, (for large n) the process of Algorithm 6 may take a long time to reach the 99% optimal solutions due to its complexity $O(mnC)$, while our exact-fit best (knapsack) order of multi-DPT-List + efficient filtering in $O(m^2n)$ can find 99% optimal solutions as $m!$ orders.

Table 14. Comparison of our multi-DPT-List + robust filtering and the mathematical HyMKP.

For Regular and Irregular Datasets ($n \leq 10,000, m \leq 100$)		
Exact + Filtering	Our multi-DPT-List + robust filtering (the exact-fit best order) could find most optimal solutions ($\geq 99\%$) in efficient response time (< 1 s per n); see confirmed results in Tables 5–12.	$O(m^2n)$
Exact	Mathematical HyMKP [34] can execute in τ secs. with Algorithm 6 (reflect multi-graph MKP with decreasing n weights (w_j)) like the basic DP for each of m knapsacks. That initial solution can be improved by the knapsack decomposition in v iterations to find the optimal solution ($n \leq 500$) in τ secs. However, no available results for $n > 500$ in that study.	$O(mnC)$
Heuristic	Multi-GH⁺ (Latin squares of top nine orders) could find good solutions in efficient time (< 1 s) but they are not optimal (see the last column results in Tables 5–10 and Table 12). Note: LSs of top 9 orders could emulate $9m$ iterations/evolutions in the GA/swarm optimization with good results (near optimal in each knapsack for small m).	$O(m^2n)$
For critical and special benchmark datasets ($n \leq 500$) [34]		
Exact	Partial BnB (in HyMKP) [34]: The existing BnB (MULKNAP program) could find most optimal solutions ($\geq 99.9\%$) in τ secs for $n \leq 500$.	$O((m+1)^n)$
Exact + Filtering	Our multi-DPT-List + robust filtering (the best order): For critical datasets in 0/1-mKP applications, we can adopt the MULKNAP program [34] for $n \leq 500$ in our approach to achieve 99.9% optimal solutions. For $n > 500$ we can apply our efficient multi-DPT-List + filtering in efficient time.	$O(m^2n)$

For $n \leq 500$ (in the critical datasets), the HyMKP model yielded 99.9% optimal solutions by the partial BnB (MULKNAP) program in τ secs. Thus, for $n \leq 500$ we can adopt that MULKNAP program in our approach for achieving 99.9% optimal solutions.

Finally, after achieving the good performance (99% optimal solutions) of our multi-DPT-List_{TSR} + robust filtering in the efficient time $O(m^2n)$, we can improve the time complexity to $O(mn)$ in parallel (by using $p = m$ processors).

Moreover, to handle the critical datasets in parallel, we can achieve the global best result in parallel (by $p = m$ processors), such as Column 3 (in Table 13), by combining the local best result of the parallel multi-DPT-List_{TSR} in $O(m[n^2, nC])$ in Column 4 and the local best result of the parallel multi-DPT-List_{TSR} + robust filtering in $O(mn)$ in Column 6 for the best of the best results (in Column 3) in $O(m[n^2, nC])$, which is efficient, especially in average $\approx O(mn^3)$ if $C = \max(C_i) \leq n^2$.

In practical 0/1-mKP applications (for large n), if the fast computing time is the most important factor (in the regular and irregular datasets), our multi-DPT-List + robust filtering in $O(m^2n)$ or $O(mn)$ in parallel ($p = m$) with 99% optimal solutions is good enough. However,

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้สำหรับใช้ในวงจำกัดเท่านั้น ไม่สามารถเผยแพร่หรือใช้เพื่อการค้าโดยไม่ได้รับอนุญาต
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

if the high optimal performance is the most important factor (in the critical datasets and in the critical 0/1-mKP applications), the integration (of the original multi-DPT-List_{TSR} and the fast multi-DPT-List_{TSR} + robust filtering) provides higher precision (i.e., 99.9% optimal solutions) in efficient $O(m[n^2, nC])$ in parallel ($p = m$) or $O(mn^2)$ in the best case and $O(mn^3)$ in average if $C \leq n^2$.

7. Conclusions

In this study, to solve the complex 0/1-mKP (m knapsacks) in polynomial time we introduced a novel research track with hybrid integration of DP transformation (for the optimal solution in each knapsack) and robust unbiased filtering (for polynomial time). First, the efficient DPT-List_{TSR} algorithm was proposed to find the optimal solutions of the 0/1-KP in $O([n^2, nC])$ over $O(nC)$ before being applied in the 0/1-mKP. Second, for solving the 0/1-mKP we proposed the multi-DPT-List_{TSR} with the exact-fit (best) knapsack order ($m!$ -to- m^2 reduction) with 99% optimal solutions in $O(m^2[n^2, nC])$ over $O(m![n^2, nC])$. Third (for large n , massive C), robust unbiased filtering was incorporated into our multi-DPT-List_{TSR} to solve the 0/1-mKP in efficient $O(m^2n)$ over $O(mnC)$ of the recent HyMKP, while retaining 99% optimal solutions. The experiment was conducted to evaluate the performance of our multi-DPT-List + robust unbiased filtering (with 99% optimal solutions) on random and benchmark datasets ($n \leq 10,000$, $m \leq 100$). Practically (for large m , n , and C), our multi-DPT-List_{TSR} + robust unbiased filtering ($O(m^2n)$) could find 99% optimal solutions (as the original multi-DPT-List_{TSR} ($O(m^2[n^2, nC])$)) in polynomial time.

In our current research, we apply our multi-DPT-List_{TSR} + robust unbiased filtering to solve the multi-container packing. In the future study, we will modify our unbiased filtering idea to solve another popular NP-hard problem (i.e., traveling salesman and logistic transportation, etc.) in efficient time with expected high optimal performance.

Author Contributions: Conceptualization, P.B. and J.W.; methodology, P.B.; software, P.B.; validation, P.B. and J.W.; formal analysis, J.W.; investigation, P.B.; writing—original draft preparation, P.B.; writing—review and editing, J.W.; visualization, P.B.; supervision, J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the School of Science, King Mongkut's Institute of Technology Ladkrabang (KMUTL), Bangkok, Thailand, grant number [CW-011-2/2562].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to deeply thank the School of Science, King Mongkut's Institute of Technology Ladkrabang (KMUTL), Bangkok, Thailand for the Ph.D. research grant (CW-011-2/2562) to Patcharin Buayen. In addition, they would like to gratefully thank Phra-phrom Kunaporn (P.A. Pa-yut-tao) with his "Put-ta-tum" Buddhism-handbook for fulfilling their exact understanding and good decision-making. Last, they would like to gloriously honor their beloved King Bhumibol Adulyadej for the great inspiration in the optimistic study through positive thinking plus critical rethinking.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Buayen, P.; Werapun, J. Parallel time-space reduction by unbiased filtering for solving the 0/1-knapsack problem. *J. Parallel Distrib. Comput.* **2018**, *122*, 195–208. [\[CrossRef\]](#)
2. Aisopos, F.; Tserpes, K.; Varvarigou, T. Resource management in software as a service using the knapsack problem model. *Int. J. Prod. Econ.* **2013**, *141*, 465–477. [\[CrossRef\]](#)
3. Vanderster, D.C.; Dimopoulos, N.J.; Parra-Hernandez, R.; Sobie, R.J. Resource allocation on computational grids using a utility model and the knapsack problem. *Future Gener. Comput. Syst.* **2009**, *25*, 35–50. [\[CrossRef\]](#)

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งยังมีให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. Bas, E. A capital budgeting problem for preventing workplace mobbing by using analytic hierarchy process and fuzzy 0-1 bidimensional knapsack model. *Expert Syst. Appl.* **2011**, *38*, 12415–12422. [\[CrossRef\]](#)
5. Camargo, V.C.B.; Mattioli, L.; Toledo, F.M.B. A knapsack problem as a tool to solve the production planning problem in small foundries. *Comput. Oper. Res.* **2012**, *39*, 86–92. [\[CrossRef\]](#)
6. Fukunaga, A.S.; Korf, R.E. Bin completion algorithms for multicontainer packing, knapsack, and covering problems. *J. Artif. Intell. Res.* **2007**, *28*, 393–429. [\[CrossRef\]](#)
7. Rooderkerk, R.P.; van Herrde, H.J. Robust optimization of the 0-1 knapsack problem: Balancing risk and return in assortment optimization. *Eur. J. Oper. Res.* **2016**, *250*, 842–854. [\[CrossRef\]](#)
8. Ahmad, S.J.; Reddy, V.S.K.; Damodaram, A.; Krishna, P.R. Delay optimization using Knapsack algorithm for multimedia traffic over MANETs. *Expert Syst. Appl.* **2015**, *42*, 6819–6827. [\[CrossRef\]](#)
9. Wang, E.; Yang, Y.; Wu, J. A Knapsack-based buffer management strategy for delay-tolerant networks. *J. Parallel Distrib. Comput.* **2015**, *86*, 1–15. [\[CrossRef\]](#)
10. Wedashwara, W.; Mabu, S.; Obayashi, M.; Kuremoto, T. Combination of genetic network programming and knapsack problem to support record clustering on distributed databases. *Expert Syst. Appl.* **2016**, *46*, 15–23. [\[CrossRef\]](#)
11. Kellerer, H.; Pferschy, U.; Pisinger, D. Dynamic Programming. In *Knapsack Problem*; Springer: Berlin, Germany; New York, NY, USA, 2004; pp. 20–26.
12. Monaci, M.; Pferschy, U.; Serafini, P. Exact solution of the robust knapsack problem. *Comput. Oper. Res.* **2013**, *40*, 2625–2631. [\[CrossRef\]](#) [\[PubMed\]](#)
13. Rong, A.; Figueira, J.R.; Klamroth, K. Dynamic programming based algorithms for the discounted {0-1} knapsack problem. *Appl. Math. Comput.* **2012**, *218*, 6921–6933. [\[CrossRef\]](#)
14. Rong, A.; Figueira, J.R.; Pato, M.V. A two state reduction based dynamic programming algorithm for the bi-objective 0-1 knapsack problem. *Comput. Math. Appl.* **2011**, *62*, 2913–2930. [\[CrossRef\]](#)
15. Rong, A.; Figueira, J.R. Dynamic programming algorithms for the bi-objective integer knapsack problem. *Eur. J. Oper. Res.* **2014**, *236*, 85–99. [\[CrossRef\]](#)
16. Cunha, A.S.; Bahiense, L.; Lucena, A.; Souza, C.C. A new lagrangian based branch and bound algorithm for the 0-1 knapsack problem. *Electron. Notes Discret. Math.* **2010**, *36*, 623–630. [\[CrossRef\]](#)
17. Li, X.; Liu, T. On exponential time lower bound of knapsack under backtracking. *Theor. Comput. Sci.* **2010**, *411*, 1883–1888. [\[CrossRef\]](#)
18. Calvin, J.M.; Leung, J.Y.-T. Average-case analysis of a greedy algorithm for the 0/1 knapsack problem. *Oper. Res. Lett.* **2003**, *31*, 202–210. [\[CrossRef\]](#)
19. Truong, T.K.; Li, K.; Xu, Y. Chemical reaction optimization with greedy strategy for the 0-1 knapsack problem. *Appl. Soft Comput.* **2013**, *13*, 1774–1780. [\[CrossRef\]](#)
20. Balas, E.; Zemel, E. An algorithm for large zero-one knapsack problems. *Oper. Res.* **1980**, *28*, 1130–1154. [\[CrossRef\]](#)
21. Guastaroba, G.; Savelsbergh, M.; Speranza, M.G. Adaptive kernel search: A heuristic for solving mixed integer linear programs. *Eur. J. Oper. Res.* **2017**, *263*, 789–804. [\[CrossRef\]](#)
22. Lim, T.Y.; Al-Betar, M.A.; Khader, A.T. Taming the 0/1 knapsack problem with monogamous pairs genetic algorithm. *Expert Syst. Appl.* **2016**, *54*, 241–250. [\[CrossRef\]](#)
23. Sachdeva, C.; Goel, S. An improved approach for solving 0/1 knapsack problem in polynomial time using genetic algorithms. In Proceedings of the IEEE International Conference on Recent Advances and Innovations in Engineering, Jaipur, India, 9–11 May 2014.
24. Bansal, J.C.; Deep, K. A modified binary particle swarm optimization for knapsack problems. *Appl. Math. Comput.* **2012**, *218*, 11042–11061. [\[CrossRef\]](#)
25. Bhattacharjee, K.K.; Sarmah, S.P. Shuffled frog leaping algorithm and its application to 0/1 knapsack problem. *Appl. Soft Comput.* **2014**, *19*, 252–263. [\[CrossRef\]](#)
26. Moosavian, N. Soccer league competition algorithm for solving knapsack problem. *Swarm Evol. Comput.* **2015**, *20*, 14–22. [\[CrossRef\]](#)
27. Zhang, J. Comparative study of several intelligent algorithms for knapsack problem. *Procedia Environ. Sci.* **2011**, *11*, 163–168. [\[CrossRef\]](#)
28. Zou, D.; Gao, L.; Li, S.; Wu, J. Solving 0-1 knapsack problem by novel global harmony search algorithm. *Appl. Soft Comput.* **2011**, *11*, 1556–1564. [\[CrossRef\]](#)
29. Lv, J.; Wang, X.; Huang, M.; Cheng, H.; Li, F. Solving 0-1 knapsack problem by greedy degree and expectation efficiency. *Appl. Soft Comput.* **2016**, *41*, 94–103. [\[CrossRef\]](#)
30. Pavithr, R.S. Gursaran, Quantum inspired social evolution (QSE) algorithm for 0-1 knapsack problem. *Swarm Evol. Comput.* **2016**, *29*, 33–46. [\[CrossRef\]](#)
31. Zhao, J.; Huang, T.; Pang, F.; Liu, Y. Genetic algorithm based on greedy strategy in the 0-1 knapsack problem. In Proceedings of the 2009 Third International Conference on Genetic and Evolutionary Computing, Guilin, China, 14–17 October 2009.
32. Zhou, Y.; Chen, X.; Zhou, G. An improved monkey algorithm for 0-1 knapsack problem. *Appl. Soft Comput.* **2016**, *38*, 817–830. [\[CrossRef\]](#)

요약: 이 논문은 0-1 배낭 문제를 해결하기 위한 다양한 알고리즘을 비교하고, 그 성능을 평가한다. 특히, 유전 알고리즘, 개미 군집 알고리즘, 양자 영감 사회 진화 알고리즘 등 다양한 메타휴리스틱 알고리즘을 소개하고, 그 성능을 비교한다.

키워드: 0-1 배낭 문제, 유전 알고리즘, 개미 군집 알고리즘, 양자 영감 사회 진화 알고리즘, 메타휴리스틱 알고리즘.

33. Sanchez-Diaz, X.; Ortiz-Bayliss, J.C.; Amaya, I.; Cruz-Duarte, J.M.; Conant-Pablos, S.E.; Terashima-Marin, H. A feature-independent hyper-heuristic approach for solving the knapsack problem. *Appl. Sci.* **2021**, *11*, 10209. [\[CrossRef\]](#)
34. Dell'Amico, M.; Delorme, M.; Iori, M.; Martello, S. Mathematical models and decomposition methods for the multiple knapsack problem. *Eur. J. Oper. Res.* **2019**, *274*, 886–899. [\[CrossRef\]](#)
35. Fukunaga, A.S. A branch-and-bound algorithm for hard multiple knapsack problems. *Ann. Oper. Res.* **2011**, *184*, 97–119. [\[CrossRef\]](#)
36. Martello, S.; Monaci, M. Algorithmic approaches to the multiple knapsack assignment problem. *Omega* **2020**, *90*, 102004. [\[CrossRef\]](#)
37. Angelelli, E.; Mansini, R.; Speranza, M.G. Kernall search: A general heuristic for the multi-dimensional knapsack problem. *Comput. Oper. Res.* **2010**, *37*, 2017–2026. [\[CrossRef\]](#)
38. Haddar, B.; Khemakhem, M.; Hanafi, S.; Wilbaut, C. A hybrid quantum particle swarm optimization for the multidimensional knapsack problem. *Eng. Appl. Artif. Intell.* **2016**, *55*, 1–13. [\[CrossRef\]](#)
39. Meng, T.; Pan, Q.-K. An improved fruit fly optimization algorithm for solving the multidimensional knapsack problem. *Appl. Soft Comput.* **2017**, *50*, 79–93. [\[CrossRef\]](#)
40. Wang, L.; Yang, R.; Ni, H.; Ye, W.; Fei, M.; Pardalos, P.M. A human learning optimization algorithm and its application to multi-dimensional knapsack problems. *Appl. Soft Comput.* **2015**, *34*, 736–743. [\[CrossRef\]](#)
41. Zhang, X.; Wu, C.; Li, J.; Wang, X.; Yang, Z.; Lee, J.-M.; Jung, K.-H. Binary artificial algae algorithm for multidimensional knapsack problems. *Appl. Soft Comput.* **2016**, *43*, 583–595. [\[CrossRef\]](#)
42. Gao, C.; Lu, G.; Yao, X.; Li, J. An iterative pseudo-gap enumeration approach for the multidimensional multiple-choice knapsack problem. *Eur. J. Oper. Res.* **2017**, *260*, 1–11. [\[CrossRef\]](#)
43. Wang, Y.; Wang, W. Quantum-inspired differential evolution with gray wolf optimizer for 0-1 knapsack problem. *Mathematics* **2021**, *9*, 1233. [\[CrossRef\]](#)

ประวัติผู้เขียน

ชื่อ	นางสาวพัชรินทร์ บัวเย็น
วัน เดือน ปีเกิด	19 พฤศจิกายน 2520
ที่อยู่ปัจจุบัน	19/80 หมู่ 6 ตำบลพลอยแหวน อำเภอท่าใหม่ จังหวัดจันทบุรี 22120
ประวัติการศึกษา	2551 วิทยาศาสตรมหาบัณฑิต สาขาวิทยาการคอมพิวเตอร์ เกรดเฉลี่ย 3.84 สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (รางวัลผลการศึกษายอดเยี่ยมชั้นวิทยาศาสตรมหาบัณฑิต มูลนิธิศาสตราจารย์ ดร.แถบ นีละนิธิ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย) 2542 วิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ เกรดเฉลี่ย 3.17 สถาบันราชภัฏรำไพพรรณี
ทุนการศึกษาที่ได้รับ	ทุนสนับสนุนการศึกษาระดับบัณฑิตศึกษา คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (หมายเลขทุน CW-011-2/2562)
ผลงานทางวิชาการ	1. P. Buayen and J. Werapun, Parallel time-space reduction by unbiased filtering for solving the 0/1-Knapsack problem, J. Parallel Distrib. Comput. 122 (2018), 195-208. 2. P. Buayen, J. Werapun, "Efficient 0/1-Multiple-Knapsack Problem Solving by Hybrid DP Transformation and Robust Unbiased Filtering. 12 (2022), 366, 1-34.