การเปรียบเทียบประสิทธิภาพอัลกอริทึมสำหรับค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อย

A Comparison of Frequent Itemsets Mining Algorithms

ทวีศักดิ์ คงตุก[1*]

สาขาวิชาระบบสารสนเทศและคอมพิวเตอร์ธุรกิจ คณะบริหารธุรกิจและเทคโนโลยีสารสนเทศ

มหาวิทยาลัยเทคโนโลยีราชมงคลสุวรรณภูมิ ศูนย์สุพรรณบุรี[1]

บทคัดย่อ

การวิจัยครั้งนี้มีวัตถุประสงค์เพื่อ 1) ศึกษา และเปรียบเทียบอัลกอริทึมสำหรับค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อย (Frequent Item sets) ซึ่งเป็นขั้นตอนหนึ่งในกระบวนการทำเหมืองข้อมูลกฎความสัมพันธ์ (Association Rule Mining) 2) ศึกษาชุดข้อมูลที่ใช้สำหรับการทดสอบอัลกอริทึมสำหรับค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อย และ 3) สรุปผลได้ว่าอัลกอริทึมใด เหมาะกับชุดข้อมูลลักษณะใด และอัลกอริทึมใดทำงานได้เร็วที่สุด หรือใช้หน่วยความจำน้อยที่สุด

ผลการวิจัยพบว่า 1) อัลกอริทึมสำหรับค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อย มีข้อดีและข้อเสียแตกต่างกันไป ดังนั้นแต่ละอัลกอริทึมจะเหมาะสำหรับการใช้วิเคราะห์ชุดข้อมูลที่แตกต่างกัน 2) อัลกอริทึมที่ทำงานได้เร็วที่สุดสำหรับชุดข้อมูลขนาดใหญ่และมีความหนาแน่นมาก คืออัลกอริทึม FP-Growth, Apriori และ PrePost+ 3) อัลกอริทึมที่ทำงานได้เร็วที่สุดสำหรับชุดข้อมูลขนาดใหญ่และมีความหนาแน่นน้อย คืออัลกอริทึม LCMFreq 4) อัลกอริทึมที่ทำงานได้เร็วที่สุดสำหรับชุดข้อมูลขนาดเล็กและมีความหนาแน่นน้อย คืออัลกอริทึม LCMFreq และ 5) อัลกอริทึมที่ทำงานได้เร็วที่สุดสำหรับชุดข้อมูลขนาดเล็กและมีความหนาแน่นมาก คืออัลกอริทึม PrePost+, LCMFreq

คำสำคัญ: อัลกอริทึมอะไพรออริ, ไอเท็มเซตที่ปรากฏร่วมกันบ่อย, กฎความสัมพันธ์, เหมืองข้อมูล

ABSTRACT

This research aimed to 1) study and compare algorithm used for searching Frequent Itemsets, which is one of the process of association rule mining; 2) explore procedure of algorithm method and techniques used for searching of frequent Itemsets; and 3) summarize which algorithm is the best fit with each type of data and which one has the best performance of executing time or with the least memory.

According to the findings of the study, they can be concluded that (1) each algorithm used for searching Frequent Itemsets has different advantages and disadvantages. Thus, each algorithm will be used to analyze different type of dataset. (2) The fastest algorithm for large dataset with high density were FP-Growth, Apriori and PrePost+. (3) The algorithm that consumes the shortest time for large dataset with low density was LCMFreq. (4) The algorithm that consumes the shortest time for small dataset with low density was LCMFreq. And (5) the algorithm that consumes the shortest time for small dataset with high density were PrePost+ and LCMFreq.

Keyword: Apriori Algorithm, Frequent Itemsets, Association Rules Mining, Data Mining

บทนำ

การทำเหมืองข้อมูลกฎความสัมพันธ์เป็นการค้นหารูปแบบความสัมพันธ์ที่ซ่อนอยู่จากชุดข้อมูลขนาดใหญ่ เป็นเทคนิคหนึ่งที่ได้รับความนิยมเพื่อนำความรู้มาใช้วางแผนกลยุทธ์ต่างๆ การหากฎความสัมพันธ์ประกอบด้วย 2 ขั้นตอนคือ /ขั้นตอนการค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อย (Frequent Itemset Generation) และ การสร้างกฎความสัมพันธ์ (Rule Generation) ซึ่งระหว่างสองขั้นตอนนี้ ขั้นตอนแรกเป็นขั้นตอนที่ใช้เวลาและหน่วยความจำมากที่สุด จึงมีงานวิจัยและพัฒนาอัลกอริทึมเกิดขึ้นมากมายเพื่อแก้ไขปัญหาดังกล่าว อัลกอริทึมแรกที่ได้รับความนิยมคืออัลกอริทึม Apriori [1] ต่อมาได้มีการพัฒนาและนำเสนออัลกอริทึมมากมายได้แก่ ได้แก่ AprioriTid [1], Eclat [2] , FIN [3] , FP-Growth [4], HMine [5] , LCMFreq [6], PrePost+ [7] , Relim [8] และ dEclat [9]

คุณลักษณะของชุดข้อมูล (Data Characteristic) ที่นำมาเพื่อวิเคราะห์หาความสัมพันธ์นั้นมีผลโดยตรงต่อประสิทธิภาพของอัลกอริทึม ทั้งในด้านความเร็วและการใช้พื้นที่หน่วยความจำ ชุดข้อมูลที่มีขนาดเล็กและมีจำนวนไอเท็มเซตน้อยอาจทำงานได้ดีเมื่อวิเคราะห์ด้วยอัลกอริทึมอีกชนิดหนึ่ง แต่เมื่อชุดข้อมูลมีขนาดใหญ่และมีจำนวนไอเท็มเซตมากก็อาจจะเหมาะกับการวิเคราะห์ด้วยอัลกอริทึมอีกชนิดหนึ่ง เป็นต้น

ดังนั้น งานวิจัยนี้ต้องการเปรียบเทียบให้เห็นว่าอัลกอริทึมใด ควรเลือกใช้กับชุดข้อมูลที่มีคุณลักษณะแบบใดจึงจะเหมาะสม โดยพิจารณาในเรื่องของเวลาและหน่วยความจำที่ใช้ในการประมวลผล

1. วัตถุประสงค์การวิจัย

1. เพื่อศึกษา และสำรวจอัลกอริทึมการค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อยที่นิยมใช้งาน
2. เพื่อเปรียบเทียบความเร็วและการใช้หน่วยความจำของอัลกอริทึมการค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อย
3. เพื่อได้ข้อสรุปอัลกอริทึมใด เหมาะสมกับชุดข้อมูลที่มีคุณลักษณะอย่างไรสำหรับ การค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อย

2. เอกสารและงานวิจัยที่เกี่ยวข้อง

อัลกอริทึมแรกที่รู้จักกันแพร่หลายสำหรับใช้ค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อยคือ อัลกอริทึม Apriori [1] อัลกอริทึมนี้หาความสัมพันธ์โดยการสร้าง Candidate Itemsets ทีละระดับ (Level Wise) เพื่อค้นหา Frequent Itemsets ซึ่งจะมีการวนอ่าน Transaction ทั้งหมดเพื่อหาค่าสนับสนุน และเปรียบเทียบกับค่าสนับสนุนขั้นต่ำ (Minimum Support) ที่ผู้ใช้กำหนด Itemsets ที่มีค่าสนับสนุนเท่ากับหรือมากกว่าก็จะถือว่าเป็น Frequent Itemsets ส่วนไอเท็มเซ็ตตัวที่มีค่าสนับสนุนน้อยกว่าจะตัดทิ้งไป (Pruning) ในระดับถัดไปจะสร้าง Candidate Itemsets จาก Frequent Itemset รอบที่ผ่านมาเพื่อช่วยลดจำนวน Candidate อัลกอริทึม Apriori จะทำการสร้างและค้นหาว่าไอเท็มเซ็ตเป็น Frequent Itemset หรือไม่ โดยการวนนับ Transaction แบบนี้ไปเรื่อยๆ จนหมดข้อมูล และไม่สามารถสร้าง Candidate ที่ผ่านค่า Minimum Support ได้อีก การวนอ่าน Transaction เพื่อหาค่าสนับสนุนทำให้ใช้เวลามาก จึงมีวิธีการปรับปรุงการหาค่าความถี่ใหม่โดยการเก็บลำดับหมายเลขทรานแซคชั่นไว้แล้วจับคู่ลำดับรายการที่ตรงกันเพื่อหาจำนวนที่ปรากฏร่วมกัน ในอัลกอริทึม AprioriTid [1] และ Eclat [10] ซึ่งจะช่วยลดการสแกน Transaction ลงได้มาก ทำให้ I/O ระหว่างหน่วยความจำหลักและดิสค์ลดลงแต่มีข้อจำกัดในเรื่องของหน่วยความจำหลักต้องมีมากพอสำหรับชุดข้อมูล เพราะมิฉะนั้นอัลกอริทึมจะไม่สามารถทำงานได้

อัลกอริทึม FP-Growth [4] เป็นอัลกอริทึมที่หาความสัมพันธ์ของข้อมูลโดยไม่มีการสร้าง Candidate Itemset หลักการทำงานในระดับที่ 1 จะทำงานเช่นเดียวกับ Apriori ไอเท็มที่ไม่ผ่านค่าสนับสนุนขั้นต่ำ จะโดน

ตัดทิ้งไปในขั้นตอนแรก หลังจากนั้นจะนำข้อมูลทั้งหมดมาจัดเรียงใหม่จากค่าสนับสนุนมากไปหาน้อย โดยจัดเก็บด้วยโครงสร้างข้อมูลแบบ FP-tree ซึ่งเป็นข้อมูลในลักษณะที่บีบอัดให้เล็กลงเพื่อสามารถเก็บในหน่วยความจำได้เพียงพอ วิธีการนี้มีการวนอ่านทรานแซคชั่นเพียงแค่ 2 รอบเท่านั้น

วิธีดำเนินการวิจัย

1. ขั้นตอนการดำเนินการวิจัยประกอบด้วย 5 ขั้นตอน ดังนี้

1.1 ศึกษาความเป็นไปได้ และกำหนดปัญหาของระบบ

อัลกอริทึมสำหรับค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อยมีอยู่หลายอัลกอริทึม แต่ละอัลกอริทึมต่างใช้เทคนิคการค้นหาข้อมูลที่แตกต่างกันไป ปัญหาคืออัลกอริทึมใดทำงานได้เร็วที่สุดและ/หรือ ใช้หน่วยความจำน้อยที่สุดเมื่อใช้ค้นหากับชุดข้อมูลที่มีคุณลักษณะแบบใด เช่นข้อมูลขนาดใหญ่และมีจำนวนไอเท็มเซตมาก ควรเลือกใช้อัลกอริทึมใด ถ้าข้อมูลมีขนาดเล็กมาจำนวนไอเท็มเซตน้อย ควรเลือกใช้อัลกอริทึมใด เป็นต้น

1.2 วิเคราะห์ข้อมูลที่ได้จากการศึกษาในขั้นที่ 1 โดยวิเคราะห์ความเร็วและหน่วยความจำที่ใช้ในการค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อย ของแต่ละอัลกอริทึม จากชุดข้อมูลที่มีคุณลักษณะแตกต่างกันเพื่อจับคู่อัลกอริทึมที่เหมาะสมสำหรับชุดข้อมูลนั้นๆ

1.3 ออกแบบระบบโดยเลือกชุดข้อมูลทดลองจำนวน 10 ชุดที่นิยมใช้ในการทดสอบในงานวิจัยด้านนี้ เพื่อใช้ทดสอบอัลกอริทึมทั้งหมด 10 อัลกอริทึม

1.4 พัฒนาระบบโดยสามารถจับคู่ได้ว่าอัลกอริทึมใด ควรใช้ค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อยสำหรับชุดข้อมูลที่มีคุณลักษณะแบบใด

1.5 เก็บรวบรวมข้อมูล สรุป วิเคราะห์ และจัดทำคู่มือการใช้งานระบบ.

เริ่มจากการเลือกใช้ชุดข้อมูลมาตรฐานจาก UCI, NU-Mine Bench, KDD-Cup 2000 และ IBMที่นิยมใช้ในการทดสอบอัลกอริทึมการค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อย จำนวน 6 ชุดข้อมูล และเลือกอัลกอริทึมที่ใช้ค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อยจำนวน 10 อัลกอริทึมมาทดสอบ หลังจากนั้นจะเปรียบเทียบเวลาและหน่วยความจำที่ใช้กับอัลกอริทึมมาตรฐาน เพื่อสรุปผล

2. เครื่องมือการวิจัย

2.1 การทดลองนี้ทำบนเครื่องคอมพิวเตอร์ Lenovo Intel® Atom™ CPU N550 1.5 Ghz. RAM 2.00 Gb. 32 –bit Operating System Windows 7

2.2 ภาษาที่ใช้พัฒนาโปรแกรมคือภาษาจาวา

2.3 SPMF v. 2.11 โปรแกรมวิเคราะห์ข้อมูล

3. กลุ่มเป้าหมาย

การทดลองนี้คัดเลือกชุดข้อมูล 10 ชุด จาก UCI, NU-Mine Bench, KDD-Cup 2000 และ IBM ซึ่งมีรายละเอียด ดังนี้

ตารางที่ 1 รายละเอียดชุดข้อมูลที่ใช้ทดลอง (Data Characteristic)

ชื่อชุดข้อมูล	ขนาด (Size)	จำนวน Record	จำนวน Items	ขนาด (Size)	ความหนาแน่น
Chess	335kb	3,196	75	เล็ก	มาก
Mushrooms	590kb	8,124	119	เล็ก	มาก
Retail	4,070kb	88,162	16,469	ใหญ่	น้อย
Accidents	6,373kb	340,183	468	ใหญ่	มาก
BMSWebView1	935kb	59,602	497	เล็ก	น้อย
BMSWebView2	2,263kb	77,512	3,340	ใหญ่	น้อย
Pumsb	16,299kb	49,046	-	ใหญ่	มาก
t20i6d100k	7,650kb	99,922	893	ใหญ่	น้อย
t25i10d10k	949kb	9,976	929	เล็ก	น้อย
Chainstore	45,558kb	1,112,949	-	ใหญ่	น้อย

4. การทดลอง

การทดลองใช้อัลกอริทึมต่างๆ จำนวน 10 อัลกอริทึม ได้แก่ Apriori, AprioriTid, Eclat, FIN, FP-Growth, HMine, LCMFreq, PrePost+, Relim และ dEclat เพื่อเปรียบเทียบเวลาที่ใช้ในการประมวลผล (Time usage) และ หน่วยความจำที่ใช้ในการประมวลผล (Memory usage)

ผลการวิจัย

ผู้วิจัยเลือกกำหนดค่า Minimum Support (minSup) ให้กับชุดข้อมูลต่างๆ ดังนี้

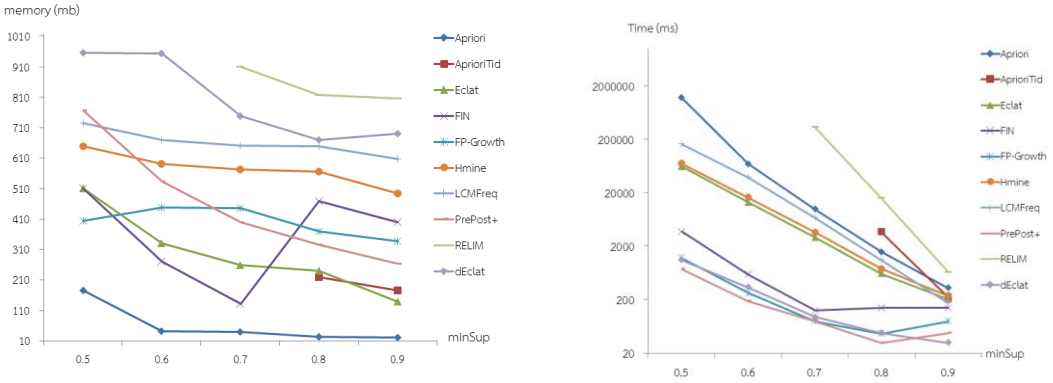
ตารางที่ 2 การกำหนดค่า Minimum Support สำหรับการทดลองแต่ละชุดข้อมูล

ชื่อชุดข้อมูล	ค่า Minimum Support (minSup)
Chess	0.5-0.9
Mushrooms	0.5-0.9
Retail	0.1-0.5
Accidents	0.5-0.9
BMSWebView1	0.01-0.05
BMSWebView2	0.009-0.04
Pumsb	0.95-0.99
t20i6d100k	0.06-0.10
t25i10d10k	0.06-0.10
Chainstore	0.01-0.05

กราฟเปรียบเทียบหน่วยความจำและเวลาที่ใช้ในการประมวลผลแต่ละอัลกอริทึมในชุดข้อมูลทดสอบทั้ง 10 ชุด

1. **ชุดข้อมูล** Chess

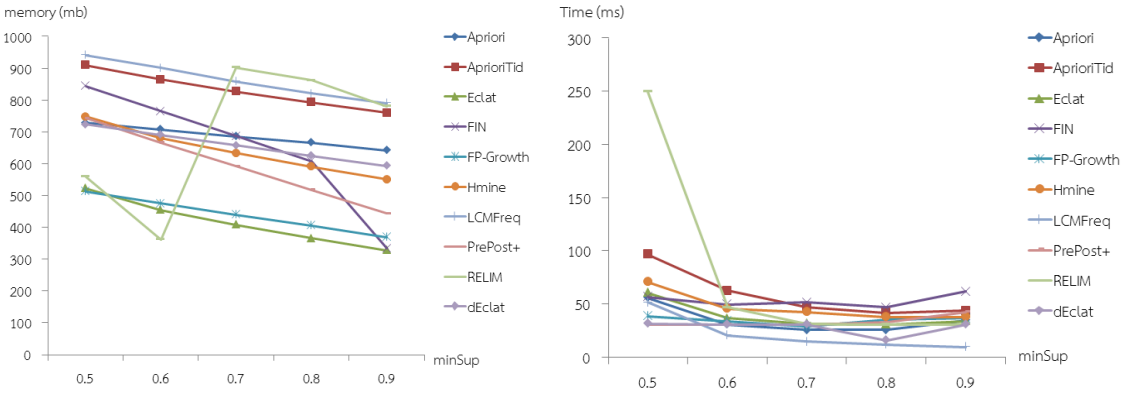
หน่วยความจำที่ใช้ประมวลผล (mb) เวลาที่ใช้ประมวลผล (ms)



ภาพที่ 1 กราฟเปรียบเทียบหน่วยความจำและเวลาที่ใช้ในการประมวลผลสำหรับชุดข้อมูล **Chess**

2. **ชุดข้อมูล** Mushrooms

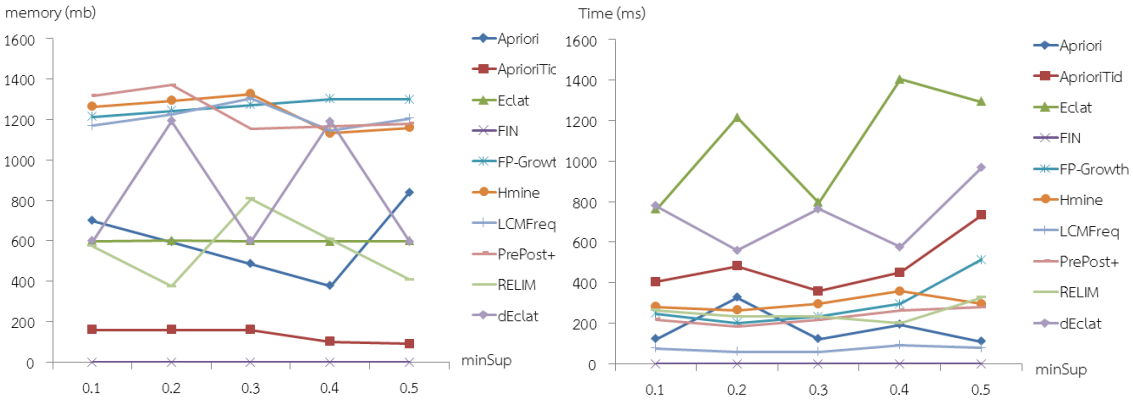
หน่วยความจำที่ใช้ประมวลผล (mb) เวลาที่ใช้ประมวลผล (ms)



ภาพที่ **2** กราฟเปรียบเทียบหน่วยความจำและเวลาที่ใช้ในการประมวลผลสำหรับชุดข้อมูล **Mushrooms**

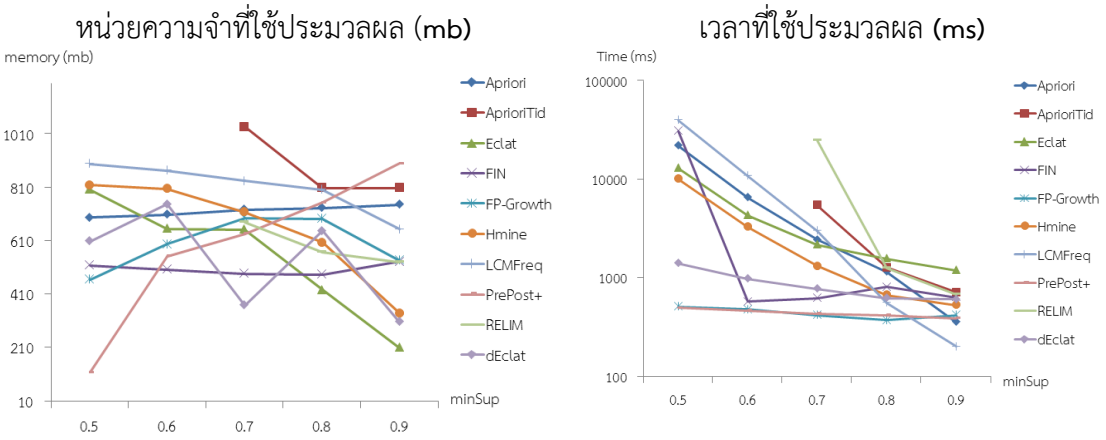
3. **ชุดข้อมูล** Retail

หน่วยความจำที่ใช้ประมวลผล (mb) เวลาที่ใช้ประมวลผล (ms)



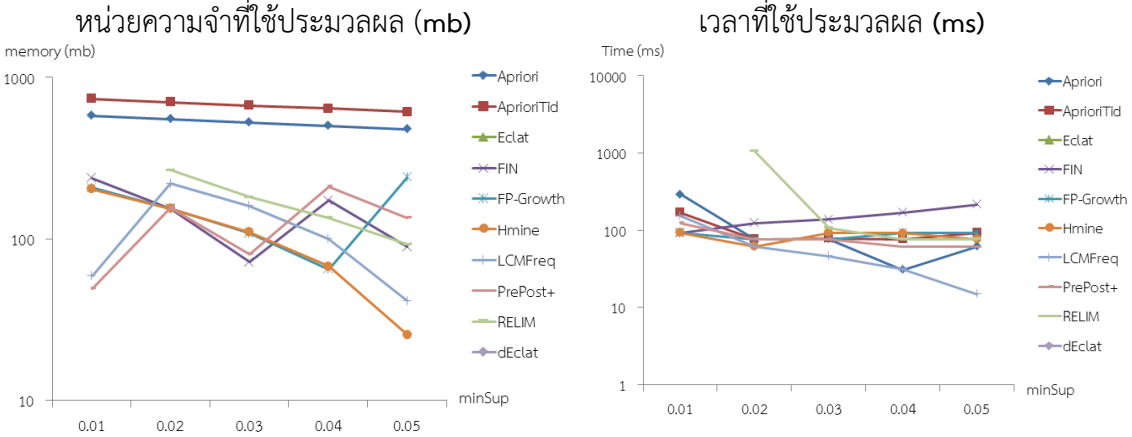
ภาพที่ **3** กราฟเปรียบเทียบหน่วยความจำและเวลาที่ใช้ในการประมวลผลสำหรับชุดข้อมูล **Retail**

4. **ชุดข้อมูล** Accidents



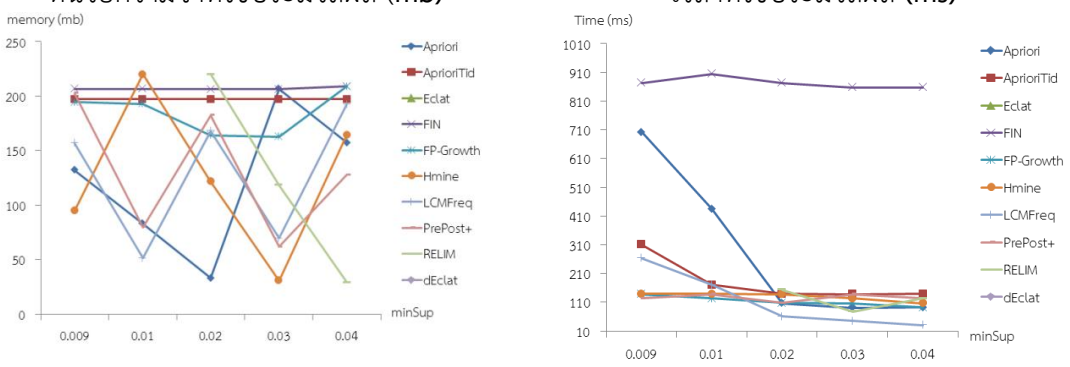
ภาพที่ 4 กราฟเปรียบเทียบหน่วยความจำและเวลาที่ใช้ในการประมวลผลสำหรับชุดข้อมูล **Accidents**

5. **ชุดข้อมูล** BMSWebView1



ภาพที่ 5 กราฟเปรียบเทียบหน่วยความจำและเวลาที่ใช้ในการประมวลผลสำหรับชุดข้อมูล **BMSWebView1**

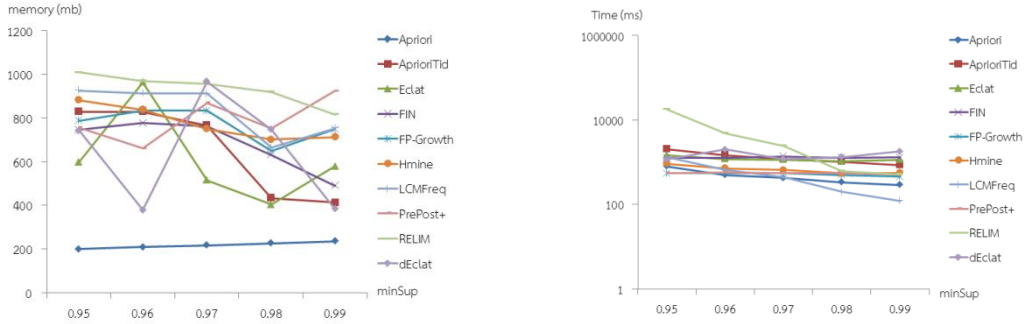
6. **ชุดข้อมูล** BMSWebView2



ภาพที่ 6 กราฟเปรียบเทียบหน่วยความจำและเวลาที่ใช้ในการประมวลผลสำหรับชุดข้อมูล **BMSWebView2**

7. **ชุดข้อมูล** Pumsb

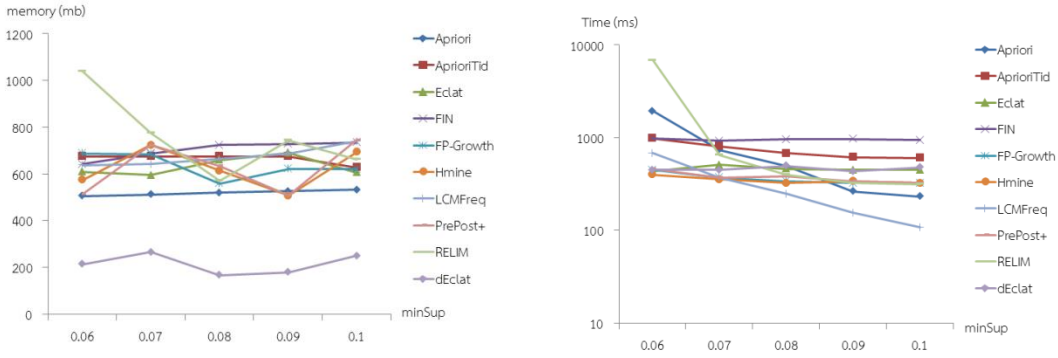
หน่วยความจำที่ใช้ประมวลผล (mb) เวลาที่ใช้ประมวลผล (ms)



ภาพที่ **7** กราฟเปรียบเทียบหน่วยความจำและเวลาที่ใช้ในการประมวลผลสำหรับชุดข้อมูล **Pumsb**

8 **ชุดข้อมูล** t20i6d100k

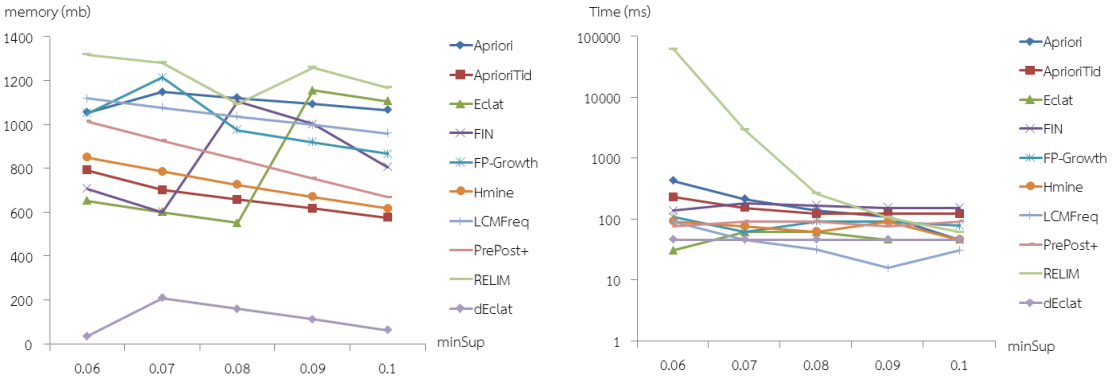
หน่วยความจำที่ใช้ประมวลผล (mb) เวลาที่ใช้ประมวลผล (ms)



ภาพที่ 8 กราฟเปรียบเทียบหน่วยความจำและเวลาที่ใช้ในการประมวลผลสำหรับชุดข้อมูล t20i6d100k

9**. ชุดข้อมูล** t25i10d10k

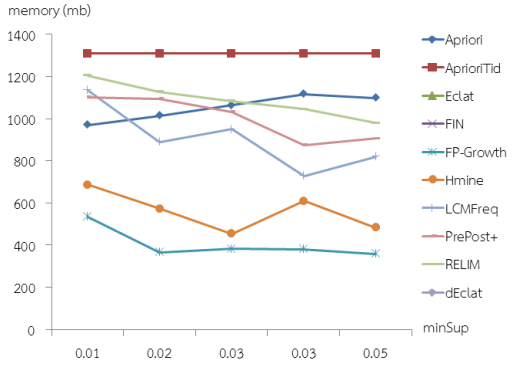
หน่วยความจำที่ใช้ประมวลผล (mb) เวลาที่ใช้ประมวลผล (ms)



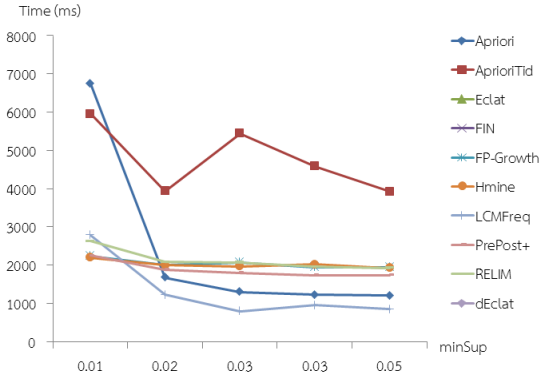
ภาพที่ **9** กราฟเปรียบเทียบหน่วยความจำและเวลาที่ใช้ในการประมวลผลสำหรับชุดข้อมูล t25i10d10k

10. **ชุดข้อมูล** Chainstore

หน่วยความจำที่ใช้ประมวลผล (mb)



เวลาที่ใช้ประมวลผล (ms)



ภาพที่ **10** กราฟเปรียบเทียบหน่วยความจำและเวลาที่ใช้ในการประมวลผลสำหรับชุดข้อมูล Chainstore

จากการทดลองพบว่า อัลกอริทึมที่ทำงานได้เร็วที่สุดสำหรับชุดข้อมูล Accidents และ Pumsb คือ อัลกอริทึม FP-Growth, Apriori และ PrePost+อัลกอริทึมที่ทำงานได้เร็วที่สุดสำหรับชุดข้อมูล Retail, BMSWebView2, t20i6d100k และ Chainstore คืออัลกอริทึม LCMFreq อัลกอริทึมที่ทำงานได้เร็วที่สุดสำหรับชุดข้อมูล BMSWebView1 และ t25i10d10k คืออัลกอริทึม LCMFreq และ อัลกอริทึมที่ทำงานได้เร็วที่สุดสำหรับชุดข้อมูล Chess และ Mushrooms คืออัลกอริทึม PrePost+, LCMFreq

สรุปผลการวิจัย

งานวิจัยนี้ชี้ให้เห็นว่า อัลกอริทึมที่ทำงานได้เร็วที่สุดสำหรับชุดข้อมูลขนาดใหญ่และมีความหนาแน่นมาก คืออัลกอริทึม FP-Growth, Apriori และ PrePost+ อัลกอริทึมที่ทำงานได้เร็วที่สุดสำหรับชุดข้อมูลขนาดใหญ่และมีความหนาแน่นน้อย คืออัลกอริทึม LCMFreq อัลกอริทึมที่ทำงานได้เร็วที่สุดสำหรับชุดข้อมูลขนาดเล็กและมีความหนาแน่นน้อย คืออัลกอริทึม LCMFreq และ อัลกอริทึมที่ทำงานได้เร็วที่สุดสำหรับชุดข้อมูลขนาดเล็กและมีความหนาแน่นมาก คืออัลกอริทึม PrePost+, LCMFreq

ข้อเสนอแนะ

สำหรับงานวิจัยต่อไปการศึกษาถึงกลไกการทำงานของอัลกอริทึมแต่ละอัลกอริทึมโดยเฉพาะอัลกอริทึมที่ทำงานได้เร็วและใช้หน่วยความจำน้อย เพื่อค้นหาแนวทางในการพัฒนาปรับปรุงประสิทธิภาพการทำงานของอัลกอริทึมนั้นๆ จะช่วยเพิ่มประสิทธิภาพในการค้นหาไอเท็มเซตที่ปรากฏร่วมกันบ่อยได้เป็นอย่างดี

เอกสารอ้างอิง

[1] R. Agrawal, and R. Srikant. (1994) . Fast algorithms for mining association rules**.** *In J.B. Bocca, Proceedings of the 20th International Confrerence on Very Large Data Bases (VLDB'94)*, Santiago dc Chile, Morgan Kaufmann.

[2] M.J.Zaki, and K.Gouda. (2000). Generating non-redundant association rules. *In 6th ACM SIGKDD Int'l Conf*. Knoledge Discovery and Data Mining, August 2000.

[3] Deng Z. H., Lv S. L. (2014). Fast mining frequent itemsets using Nodesets**.** *Expert systems with Applications,* No.41(2014), 4505-4512.

[4] J.Han, J. Pei, and Y. Yin (2000). Mining Frequent Patterns Without Candidate Generation**.** Proc. *ACM-SIGMOD Int'l Conf*, *Management of Data.* pp. 1-12, May 2000.

[5] J. Pei, et.al. (2007). *H-Mine: Fast and space-preserving frequent pattern mining in large databases***.** IIE transaction (2007) 39, 593-605.

[6] T. Uno, et.al. (2004). *LCM ver. 2: Efficient mining algorithms for Frequent/Closed/Maximal Itemsets***.** National Institute of informatics 2-1-2 Hototsubashi, Chiyoda-ku, Tokyo, Japan.

[7] Deng Z. H., and Lv S. L. (2015). PrePost+: An efficient N-lists based algorithm for mining frequent itemsets via Children-Parent Equivalence pruning**.** *Expert systems with Applications,* No.42(2015), 5424-5432.

[8] C. Borgelt. (2005). *Keeping Things Simple: Finding Frequent Item Sets by Recursive Elimination***.** University of Magdeburg, Germany.

[9] M.J.Zaki, and K.Gouda. (2001). *Fast Vertical Mining Using Diffsets***.** Rensselaer Polytechnic Institute, Troy, NY, USA. Kyushu University, Fukuoka Japan.

[10] M.J. Zaki, and W.M. JR. (2014). *Data Mining and Analysis Fundamental Concepts and Algorithms*. Cambridge University Press. New York, NY 10013-2473, USA.