การค้นหากฎความสัมพันธ์ด้วยการนับความถี่ของเส้นเชื่อมกราฟแบบเพิ่มเติมได้และสร้างกฎความสัมพันธ์แบบพลวัตสำหรับพาณิชย์อิเล็กทรอนิกส์

Incremental Association Rule Mining with Frequent Edge Graph and Dynamic Rule Generation for e-Commerce

ประมูล สุขสกาวผ่อง* และ พยุง มีสัจ

Pramool Suksakaophong* and Phayung Meesad

คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

Faculty of Information Technology, King Mongkut's University of Technology North Bangkok

*E-mail: pramool.s@email.kmutnb.ac.th

Received: Nov 21, 2018

Revised: Oct 24, 2019

Accepted: Nov 18, 2019

บทคัดย่อ

การหากฎความสัมพันธ์ เป็นกระบวนหนึ่งในเรื่องการทำเหมืองข้อมูลที่ค้นหาความสัมพันธ์ ระหว่างรายการสินค้าที่มักจะปรากฏด้วยกันบ่อยในรายการซื้อขายสินค้า สามารถนำไปเพิ่มยอดขาย ด้วยการทำโปรโมชั่นกับรายการสินค้าที่มักจะซื้อด้วยกันบ่อย แต่ร้านค้าปัจจุบันเป็นร้านค้าในรูปแบบพาณิชย์อิเล็กทรอนิกส์มากขึ้น สิ้นค้ามีจำนวนมาก การเพิ่มสินค้าใหม่และสินค้าหมดสต็อก เกิดขึ้นบ่อยมาก ทำให้การค้นหากฎความสัมพันธ์แบบเดิมไม่เหมาะกับข้อมูล ที่มีการเปลี่ยนแปลงตลอดเวลาแบบนี้ ผู้วิจัยจึงได้นำเสนอขั้นตอนวิธีใหม่ เรียกว่า (iARMFEG : incremental Association Rule Mining with Frequency Edge Graph) ในขั้นตอนการนับความถี่จะใช้กราฟถ่วงน้ำหนักในลักษณะที่นับเพิ่มขึ้นได้ การค้นหากฎความสัมพันธ์จากกราฟถ่วงน้ำหนักนี้ สามารถทำซ้ำได้โดยไม่ต้องเริ่มขั้นตอนการนับความถี่ใหม่ และยังค้าหาค้นหากฎความสัมพันธ์ได้จากทุกรายการสินค้าหรือระบุเจาะจงรายการสินค้าที่สนใจได้ สามารถแก้ปัญหาการค้นหาสินค้าที่มีรายการความถี่น้อยได้ การสร้างกราฟได้จากการอ่านข้อมูลจากฐานข้อมูลเพียงหนึ่งรอบเดียวในขณะที่ FP-Growth ต้องอ่านข้อมูลสองรอบ และพบว่าวิธีการที่นำเสนอใช้เวลาในการประมวลผลเร็วกว่าวิธี Apriori ถึง 95 เปอร์เซ็นต์

คำสำคัญ: กฎความสัมพันธ์ พาณิชย์อิเล็กทรอนิกส์ กราฟบริบูรณ์ เหมืองข้อมูล

Abstract

Association rule mining is one of the most common data mining tasks used to identify relations between items, which often appear in the same transaction data. This information can be used in such areas as boosting sales through promotional tactics that encourage purchases of items usually bought together. However, in today's shops, more are turning to the e-Commerce platform to sell their products. As in a real shop, product numbers increase with new items being added, while a few are out of stock. These kinds of situations occur around the clock and for this reason, the current association rule mining algorithms are no longer suited for such dynamic. Therefore, a new algorithm called iARMFEG (incremental Association Rule Mining with Frequency Edge Graph) is proposed. The iARMFEG brings in information from each transaction one at a time to build an incremental weight graph. The resulting rules are generated from the constructed weight graph. An advantage of the proposed method is the ability to retaliate rule generation multiple times without having to go through the Frequency Itemset Generation process again. Moreover, it can also search for rules from each list of items or only from a specified. By doing so,

the rare item problem is then resolved. The proposed method can successfully generate graphs by only reading data one time from a database, while FP-Growth must read two times. In addition, it was also found that the proposed method has a 95 percent faster processing time than Apriori methods because they also need to read data many times.

Keywords: Association rule mining, E-Commerce, Completed Graph, Data mining

1. บทนำ

การค้นหากฎความสัมพันธ์ (Association rule mining) มีจุดประสงค์เพื่อค้นหาความสัมพันธ์ที่น่าสนใจที่ซ่อนอยู่ในข้อมูลบันทึกรายการซื้อขายสินค้า (transaction data) จะได้ข้อมูลออกมาเป็นกฎความสัมพันธ์ (Association rule) ของการซื้อสินค้าของลูกค้าว่าจะซื้อสินค้าใดบ้างร่วมกันบ่อยจากบันทึกรายการซื้อขายสินค้า (Market basket transaction) เป็นพฤติกรรมการซื้อสินค้าของลูกค้า ยกตัวอย่างเช่น เมื่อลูกค้าซื้อนมแล้วจะซื้อขนมปังคู่กันด้วย เป็นต้น โดยกฎความสัมพันธ์สามารถสร้างขึ้นจากนับเซตรายการความถี่ถ้ามีค่ามากกว่าค่าสนับสนุนขั้นต่ำ (Minimum Support) เรียกย่อๆว่า *minsup* ที่กำหนดขึ้น ก็จะนำมาสร้างเป็นกฎความสัมพันธ์ โดยเปรียบเทียบค่าความเชื่อมั่นขั้นต่ำ (Minimum Confidence) เรียกย่อๆว่า *minconf* ถ้ามีค่ามากกว่าที่กำหนดก็นำมาสร้างเป็นกฎความสัมพันธ์ได้

แต่ปัญหาที่สำคัญในการค้นหากฎความสัมพันธ์จากข้อมูลบันทึกรายการซื้อขายสินค้าคือ ค่าที่เหมาะสมของค่าสนับสนุนขั้นต่ำ เพราะถ้ากำหนดค่าน้อยไป จำนวนกฎที่ได้ก็จะมากและถ้ากำหนดค่ามากไปจำนวนกฎที่ได้ก็จะน้อย นอกจากนี้ยังมีปัญหาในการค้นหากฎความสัมพันธ์คือ รายการสินค้าที่มีข้อมูลน้อย ซึ่งเป็นรายการสินค้าที่มีความถี่น้อยๆ แต่สินค้ามีมูลค่าสูง โดยขั้นตอนวิธีการค้นหากฎความสัมพันธ์ส่วนใหญ่จะตัดทิ้งเนื่องจากความถี่น้อยกว่าค่าสนับสนุนขั้นต่ำต่ำกว่าที่กำหนด

ขั้นตอนวิธีที่เป็นที่รู้จักในช่วงแรกคือ Apriori [1] โดยใช้ทฤษฎีเซต ในการจัดกลุ่มสร้างรายการเซ็ตรายการสินค้า 1ชิ้น เรียกว่า L1 เริ่มนับความถี่จากจากข้อมูลรายการซื้อขายสินค้าและเลือก ที่มีค่า มากกว่าค่า นำไปสร้างเซตรายการสินค้า 2ชิ้น เรียกว่า L2 ค้นหาความถี่จากข้อมูลรายการซื้อขายสินค้าอีกครั้ง ทำซ้ำจนถึง n รอบจนครบ การจำกัดการค้นหาข้อมูลด้วยการตัดรายการความถี่น้อยๆ ออกด้วยการกำหนดค่า และตามคุณสมบัติของ ก่อนจะนำมาสร้างกฎความสัมพันธ์ แต่ก็มีข้อเสียคือการอ่านข้อมูลหลายรอบ ต่อมาจึงผู้วิจัยสร้างขั้นตอนวิธี FP-Growth [2] โดยใช้โครงสร้างข้อมูลแบบต้นไม้ เรียกว่า FP-Tree ในการนับความถี่และสร้างกฎความสัมพันธ์ก็ทำงานได้รวดเร็วเพราะสแกนข้อมูลเพียง 2 รอบ

ปัญหาที่ยากในการหากฎความสัมพันธ์คือ การกำหนดค่าที่เหมาะสมของค่าสนับสนุนขั้นต่ำ เพราะถ้าตั้งค่าที่น้อยเกินไป ปริมาณกฎที่ได้จะเยอะหรือ ถ้าตั้งค่าสูงไปปริมาณกฎที่ได้จะน้อยหรือไม่สามารถหาได้เลย การกำหนดค่าการสนับสนุนขั้นต่ำโดยปกติจะทำโดยผู้ใช้ แต่อาจเป็นเรื่องยากที่จะหาค่าที่เหมาะสม เพราะข้อมูลแต่ละชุดจะมีลักษณะไม่เหมือนกัน การประมาณค่าที่เหมาะสมโดยผู้ใช้ (user) จะเป็นขั้นตอนที่ลองผิดลองถูกหลายครั้งกว่าจะหาค่าที่เหมาะสมได้ นอกจากนี้ การตัดความถี่น้อย ๆ ออกไปเพื่อลดเวลาที่ใช้ในการประมวลผล ทำให้ไม่พบข้อมูลที่ความถี่น้อยแต่อาจมีความสำคัญ เช่น ข้อมูลนั้นสามารถทำกำไรได้ดี แต่มีราคาแพง เช่นไข่ปลาคาร์เวียและ ไวน์ ผู้ค้าก็ต้องการทราบข้อมูลนั้นเพื่อต้องการทำโปรแกรมส่งเสริมการขาย ลักษณะนี้เป็นปัญหาค่าความถี่น้อยที่มักถูกตัดออก (Rare items problem) หรือต้องการระบุเจาะจงที่จะทำการส่งเสริมการขาย แบบเจาะจงก็จะทำไม่ได้ จึงมีผู้คิดค้น เพื่อกำหนดค่าถ่วงน้ำหนักให้แต่ละรายการมาแก้ปัญหาดังกล่าว

ดังนั้น ผู้วิจัยจึงคิดค้นขั้นตอนวิธีใหม่ โดยใช้โครงสร้างข้อมูลแบบกราฟจำลองข้อมูลโดยใช้ ขั้นตอนการนับความถี่แทนแต่ละรายการด้วยโหนดหรือจุดยอด และนับความถี่สะสมในเส้นเชื่อม ทำให้สามารถหาค่าถ่วงน้ำหนักได้ทุกๆ รายการสินค้า สามารถหากฎความสัมพันธ์จากทุกรายการสินค้า โดยจะใช้ค่าถ่วงน้ำหนัก ที่สร้างอันดับจากมากสุดแบบอัตโนมัติแทนการกำหนดค่าแบบเดิมซึ่งต้องกำหนดโดยผู้ใช้ และเมื่อมีข้อมูลใหม่มาสามารถนับความถี่เพิ่มจากเดิมได้ ไม่ต้องนับความถี่ใหม่ตั้งแต่ต้นทำให้ประหยัดเวลาในการประมวลผล

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

การค้นหากฎความสัมพันธ์ จากรายการซื้อขายสินค้ามีขั้นตอนหลัก ๆ 2 ขั้นตอนคือ ขั้นตอนแรก ขั้นตอนการนับความถี่ ที่มีความถี่สะสมมากกว่าค่าสนับสนุนขั้นต่ำ และขั้นตอนที่ 2 คือ การสร้างกฎความสัมพันธ์ จากรายการสินค้า

ทีมีค่าความเชื่อมั่นของกฎมากว่าค่าขั้นต่ำที่กำหนดจึงจะสร้างเป็นกฎความสัมพันธ์

2.1 นิยามศัพท์เฉพาะของกฎความสัมพันธ์

กำหนดให้ $I = \{i_1, i_2, \ldots i_d\}$ เป็นเซตข้อมูลรายการสินค้าที่ปรากฏในเซตบันทึกรายการซื้อขายสินค้า T โดยที่ $T = \{t_1, t_2, \ldots t_d\}$ แทน t_i เป็นเซตย่อยที่เป็นลำดับแต่ละรายการใบบันทึกรายการซื้อขายสินค้า ซึ่งจะมีรายการสินค้าอยู่ในเซต I

$$Support,\ s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N} \quad (1)$$

ค่าสนับสนุนขั้นต่ำ สามารถเขียนแทนด้วย *minsup*. การหาค่าสนับสนุน จากสมการที่ (1) มาจากการคำนวณความถี่ที่ข้อมูล X และ Y มักจะปรากฏด้วยกัน เปรียบเทียบกับจำนวนรายการซื้อขายสินค้าทั้งหมดคือ N

$$Confidence,\ c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \quad (2)$$

ค่าความเชื่อมั่นของกฎ สามารถเขียนแทนด้วย *minconf* ดังแสดงในสมการที่ (2) มาจากการคำนวณความถี่ที่ข้อมูล X และ Y ที่มักปรากฏด้วยกันบ่อย เปรียบเทียบกับจำนวนความถี่ของ X

จำนวนที่เป็นไปได้ของกฎ สามารถคำนวณในสมการ (3)

$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right] \quad (3)$$

$$= 3^d - 2^{d+1} + 1$$

ในสมการที่ (3) ตัวอักษร d คือจำนวนของ สินค้าตัวอย่างเช่น ถ้ามี 8 รายการ จำนวนกฎที่จะเป็นไปได้ คือ จำนวนกฎที่เป็นไปได้คือ 6,050 กฎ ถ้าจำนวน item การเพิ่มขึ้นจำนวนกฎจะมากขึ้นเป็นทวีคูณ

2.2 การหาค่าที่เหมาะสมของค่าสนับสนุนขั้นต่ำ

ปัญหาที่ยากของการค้นหากฎความสัมพันธ์แบบเดิมคือการที่ให้ผู้ใช้กำหนดค่าสนับสนุนขั้นต่ำและค่าความเชื่อมั่นของกฎเอง ซึ่งมีค่าระหว่าง 0.0-1.0 โดยมากแล้วผู้ใช้จะกำหนดแค่ทศนิยม 1-2 ตำแหน่ง เช่น 0.35 แต่ในข้อมูลขนาดใหญ่ ค่าสนับสนุนขั้นต่ำจะต่ำกว่า 0.01 และ มีค่าอยู่ในระหว่างทศนิยมห้าตำแหน่งขึ้นไปเช่น 0.01125-0.01535 ทำให้ผู้ใช้ ไม่สามารถกำหนดได้ถูกต้อง ซึ่งเป็นขั้นตอนที่ยุ่งยาก และต้องค้นหากันหลายรอบ ซึ่งก็มีงานวิจัยหลายเรื่องในการหาค่าที่เหมาะ เช่น การใช้ความฉลาดเชิงกลุ่ม Binary particle swarm optimization (BPSO) [3] และ Ant Colony optimization [4] นอกจากนี้ ยังมีวิธีการหาค่าความถี่สูงสุดที่อยู่ในข้อมูลรายการซื้อขายสินค้าแทนการใช้ค่าสนับสนุนขั้นต่ำเพื่อเป็นจุดเริ่มต้นในการค้นหากฎความสัมพันธ์ ตัวอย่างเช่น Top-K [5] และ MAFIA [6]

2.3 การแก้ปัญหาข้อมูลที่มีความถี่น้อย

การหากฎความสัมพันธ์ทั่วไปจะใช้วิธีการเล็ม (pruning) ข้อมูลที่มีความถี่น้อยๆออกไป ทำให้เกิดปัญหากับข้อมูลที่มีความถี่น้อย สินค้าที่มีความถี่น้อย แต่อาจมีสินค้าราคาแพงทำให้คนซื้อน้อย แต่ผู้ขายต้องการขายเพราะกำไรสูง จึงต้องการหากฎความสัมพันธ์ สำหรับจัดทำรายการส่งเสริมการขาย (promotion) จึงมีงานวิจัยเพื่อแก้ปัญหาหลากหลายรูปแบบ เช่น การกำหนดค่าสนับสนุนขั้นต่ำจำนวนหลายค่า (multi minimum support) [7] ทำให้สามารถค้นหาข้อมูลได้กว้างขึ้น การกำหนดค่าน้ำหนักให้แต่ละรายการสินค้า (Weighted Association Rule Mining (WARM) ผู้ใช้สามารถกำหนดค่าถ่วงน้ำหนัก (weight) ในรายการสินค้าที่ต้องการ ตัวอย่างงานวิจัยในกลุ่มนี้ เช่น WAR [8], WARM [9] และ WRARM [10] เป็นต้น

2.4 การแก้ปัญหาด้วยโครงสร้างข้อมูลแบบกราฟ

โครงสร้างข้อมูลแบบกราฟ G จะประกอบไป V แทนโหนด (Node) หรือ จุดยอด (Vertex) และ Eแทนการเชื่อมโยงระหว่างโหนดหรือเรียกว่า เส้นเชื่อม (Edge) ดังสมการที่ (4)

$$G = (V, E) \quad (4)$$

กราฟแบ่งได้เป็น 2 ประเภทหลักๆ คือ กราฟแบบมีทิศทางและกราฟแบบไม่มีทิศทาง โดยการงานวิจัยนี้เลือกใช้กราบริบูรณ์แบบไม่มีทิศทาง (undirected complete graph) ข้อดีของกราฟบริบูรณ์ หรือ กราฟแบบเชื่อมต่อถึงกันหมด คือ แข็งแรง (robust) ถ้าโหนดไหนไม่พร้อมใช้งาน เช่นสินค้าหมดก็สามารถยกเลิกใช้งานชั่วคราวได้ เมื่อสินค้านำเข้ามาใหม่ก็สามารถใช้งานต่อได้ เหมาะกับการใช้ในพาณิชย์อิเล็กทรอนิกส์ที่สินค้ามีการเปลี่ยนแปลงบ่อย มีสินค้าเข้ามาใหม่หรือหมดบ่อย ฉะนั้นระบบไม่ควรแนะนำให้ซื้อสินค้าที่หมดแล้วแต่ควรยกเลิกใช้งานชั่วคราวได้

กราฟ เป็นโครงสร้างข้อมูลที่แสดงความสัมพันธ์ของข้อมูลได้ดี กราฟถูกนำมาใช้ เช่น The Bipartite Graphm [11] แสดงความสัมพันธ์ของข้อมูลโดยแบ่งเป็น 2 ส่วน คือ ส่วนที่เป็นรายการสินค้า และ ส่วนที่เป็นลำดับรายการซื้อขายสินค้า FP-Growth-Graph [12] ใช้โครงสร้างข้อมูลแบบกราฟ แทน

โครงสร้างข้อมูลแบบต้นไม้ ในขั้นตอนการนับความถี่ GRAM [13] การใช้กราฟบริบูรณ์ (completed graph) เป็นการเปลี่ยนแต่ละรายการซื้อขายสินค้าไปเป็นกราฟบริบูรณ์หลายๆ กราฟมารวมกันแทนแต่กราฟด้วยสี ส่วนวิธีของ ARMFEG [14] ก็จะแทนแต่ละรายการสินค้า เป็นกราฟบริบูรณ์ และรวมกันเป็นกราฟแบบถ่วงน้ำหนัก

3. วิธีดำเนินงานวิจัย

ขั้นตอนวิธีในการหากฎความสัมพันธ์จะ ประกอบด้วย 3 ส่วนหลักๆ คือ ขั้นตอนการนับความถี่ (Frequent Itemset Generation) ขั้นตอนการหาค่าถ่วงน้ำหนักและการสร้างกฎความสัมพันธ์ (Rule Generation) โดยผลลัพธ์ที่ได้จากกระบวนการนับความถี่จะถูกเก็บใน เมทริกซ์แถวประชิด (Adjacency Matrix) ที่ใช้ชื่อว่า **FEG** และลำดับรายการซื้อขายสินค้าจะใช้ HashMap ที่ใช้ชื่อว่า **HMI** ส่วนขั้นตอนการหากฎความสัมพันธ์สามารถหาได้ 2 วิธี เลือกข้อมูลจากความถี่สูงสุด (*TopWeight*) หรือจากรายการสินค้าที่กำหนด

3.1 การนับความถี่แบบเพิ่มเติมได้

ในขั้นตอนการนับความถี่ จาก Figure 1 แสดงขั้นตอนวิธีของการนับความถี่แบบเพิ่มเติมได้ (Incremental Frequent Itemset Generation Algorithm) เริ่มจากการอ่านข้อมูลบันทึกรายการซื้อขายสินค้าเข้าสู่หน่วยความจำและเริ่มทำการวนรอบประมวลผลทีละรายการ เริ่มจากรายการแรก เมื่ออ่านข้อมูลนับจำนวนสินค้าที่มีอยู่ทั้งหมดที่ไม่ซ้ำกัน ทำการจำลองสร้างกราฟบริบูรณ์ตามจำนวนสินค้าในเมทริกซ์แถวประชิดชื่อว่า **FEG** และใส่ค่า 1 ทุกแถวยกเว้นเมื่อวนรอบแถวทแยงมุม จะทำการเก็บหมายเลขลำดับรายการซื้อขายสินค้าใน **HMI** (HashMap Items) ด้วยค่าหลักหรือกุญแจ (Key) คือ หมายเลขลำดับสินค้า (Transaction ID) และ ค่าของข้อมูล (Value) คือ หมายเลขลำดับรายการซื้อขายสินค้า รายการต่อมาก็จำลองเป็นกราฟบริบูรณ์ เท่ากับจำนวน ถ้ามีสินค้าใหม่ก็เพิ่มข้อมูลโหนด ใหม่ ในเมทริกซ์แถวประชิดและนับเพิ่มในแต่ละคู่ของโหนด ใน **FEG** นี้ ในรายการถัดมาก็ทำแบบเดียวกัน จนครบทุก รายการซื้อขายสินค้า ผลลัพธ์สุดท้ายจะได้กราฟถ่วงน้ำหนักที่แสดงภาพรวมของข้อมูลทั้งหมด

Table 1 An example of transaction database

TID	Items
01	Apple, Bacon, Cake
02	Apple, Cake, Donuts
03	Apple, Cake, Donuts, Eggs
04	Bacon, Cake, Eggs, Fish
05	Apple, Cake, Donuts, Eggs
06	Apple, Bacon, Cake, Eggs
07	Donuts, Fish
08	Bacon, Donuts, Eggs, Fish
09	Apple, Bacon, Cake, Eggs
10	Apple, Cake, Eggs

Table 1 แสดงบันทึกรายการซื้อขายสินค้า 6 รายการ แทน a=Apple b=Beer c=Cake d=Donuts e=Eggs and และ f=Fish

Algorithm1 Incremental Frequent Itemset Generation

```
for each transaction (t) in (T)
  NI = count Number of items in (t)
  while NI > 0
   if t = first transaction
      create adjacency matrix FEG
   else if node = new node
      extend new node in FEG
   Update +1 each pair of FEG(i,j)
   if FEG(i=j) HashMap HMI
      (Key=ItemsID,   Value=TID)
  end while
end for
```

Figure 1 pseudo code of Incremental Frequency Itemset Generation phase

ขั้นตอนการทำงานของการนับความถี่แบบเพิ่มเติมได้ เริ่มขั้นตอนแรก ระบบจะทำการอ่านรายการแรก TID 01 คือ จะประกอบไปด้วยโหนด {a, b, c} ทำการแปลงเป็นกราฟบริบูรณ์ที่มี 3 โหนด คือ a b และ c นำมาสร้างเป็นเมทริกซ์แถวประชิดขนาด 3 คูณ 3 และเก็บในตัวแปรชื่อ FEG ในการวนเก็บข้อมูลในเมทริกซ์จะวนเก็บข้อมูลตามลำดับ ในวงรอบที่ค่า

i=j จะเป็น self-loop หรือจะมีค่าเท่ากับเมทริกซ์ในแนวทแยงมุม จะเก็บ TID ในตัวแปร HMI ด้วยค่า key = ItemsID และ Value = TID ดังแสดงใน Figure 2(a)

ในลำดับต่อมาระบบจะอ่านรายการที่ 2 คือ TID02 คือ {a, c, d} มี 3 โหนด ทำการค้นหาในเมทริกซ์เดิมว่ามีโหนดเดิมอยู่หรือใน ในที่นี้ไม่พบโหนด d ก็จะทำการเพิ่มโหนด d เข้าไปในเมทริกซ์ FEG เดิม และทำการแปลงเป็นกราฟบริบูณ์ที่มี 3 โหนด และทำการ Update ข้อมูล +1 ในทุกคู่ของโหนด และในวงรอบที่ค่า i=j จะเป็น self-loop หรือจะมีค่าเท่ากับเมทริกซ์ในแนวทแยงมุมจะจัดเก็บ TID ใน HMI ด้วยค่า key = ItemsID และValue = TID แสดงใน Figure 2(b)

ในรายการที่ 3 คือ TID03 คือ {a, c, d,e } ทำการแปลงเป็นกราฟบริบูณ์ที่มี 4 โหนด และเพิ่มโหนดใหม่ คือ e ใน **FEG** และทำการ Update ข้อมูล +1 ในทุกคู่ของ node แสดงใน figure 2(c) และในวงรอบที่ค่า i=j จะเป็น self-loop หรือจะมีค่าเท่ากับเมทริกซ์ในแนวทแยงมุมจะจัดเก็บ TID ใน HMI ด้วยค่า key = ItemsID และValue = TID

ในรายการที่ 4 คือ TID04 คือ {b, c, e, f } ทำการแปลงเป็น กราฟบริบูณ์ ที่มี 4 โหนด และเพิ่มโหนดใหม่ คือ f ใน **FEG** และทำการ Update ข้อมูล +1 ในทุกคู่ของ node แสดงใน figure 2(d) และในวงรอบที่ค่า i=j จะเป็น self-loop หรือจะมีค่าเท่ากับเมทริกซ์ในแนวทแยงมุม จะจัดเก็บ TID ใน HMI ด้วยค่า key = ItemsID และValue = TID รายการที่เหลือก็ทำซ้ำจนครบทุกรายการตามข้อมูล Table 1 ผลลัพธ์สุดท้ายจะได้กราฟถ่วงน้ำหนักดัง Figure 3 แต่ละเส้นเชื่อม แสดงความถี่ของข้อมูล จะเก็บในเมทริกซ์แถวประชิด FEG และลำดับรายการซื้อขายเก็บใน HMI

3.2 การหาค่าถ่วงน้ำหนักสูงสุด

ผลลัพธ์ที่ได้จะเป็นกราฟถ่วงน้ำหนัก ที่แสดงค่าถ่วงน้ำหนักทุกคู่ของรายการสินค้า ในขั้นตอนนี้ต้องมีการกำหนดค่า k ไว้ เป็นค่าลำดับข้อมูลที่ต้องการค้นหา จากค่าสูงสุด TW (Top Weight) เช่น ถ้ากำหนดค่า k=2 ฉะนั้น TW[2] ก็นับเป็นลำดับที่3 จากค่าสูงสุดเพราะเริ่มต้นด้วย 0 จากข้อมูล FEG ทำการเรียง TW(7, 6, 5, 4, 4, 3, 3, 3, 3, 2) แสดงใน Table 2 เพราะฉะนั้น ค่า TW[2] = 5 ระบบก็จะใช้ค่า 5 นี้ในการเล็ม (pruning) กราฟ ในขั้นตอนต่อไป การหาค่า TW จากสมการที่ 5 ได้จากข้อมูลใน เมทริกซ์ FEG นำมาเรียงจากมากไปน้อย เมื่อลำดับ i มากกว่า j

$$TW = DESC\left[FEG_{i,j} | where i < j\right] \quad (5)$$

Table 2 ค่าถ่วงน้ำหนัก 10 อันดับแรก

k	*TopW*eight[*k*]	Frequent	*Support*	*Confidence*
0	{a, c }	7	0.7	1
1	{c, e}	6	0.6	0.75
2	{a, e}	5	0.5	0.71
3	{b, c}	4	0.4	0.8
4	{b, e}	4	0.4	0.8
5	{a, b}	3	0.3	0.42
6	{b, d}	3	0.3	0.6
7	{c, d}	3	0.3	0.37
8	{d, e}	3	0.3	0.6
9	{b, f}	2	0.2	0.4

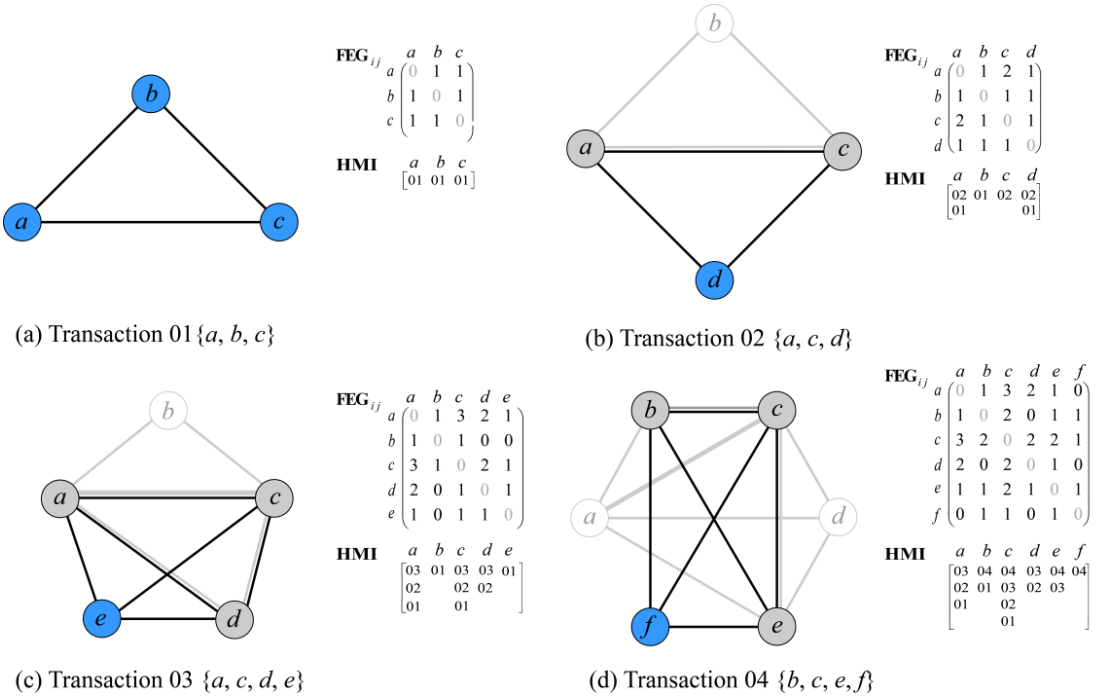
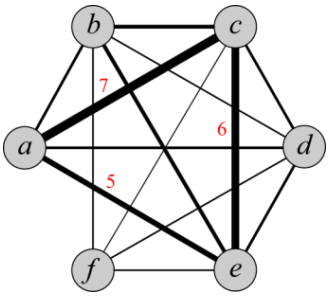


Figure 2 ขั้นตอนการนับความถี่แบบเพิ่มเติมได้



FEG *(Candidate 2-itemset)*

	a_{j0}	b_{j1}	c_{j2}	d_{j3}	e_{j4}	f_{j5}
a_{i0}	$0_{0,0}$	$3_{0,1}$	$7_{0,2}$	$3_{0,3}$	$5_{0,4}$	$0_{0,5}$
b_{i1}	$3_{1,0}$	$0_{1,1}$	$4_{1,2}$	$1_{1,3}$	$4_{1,4}$	$2_{1,5}$
c_{i2}	$7_{2,0}$	$4_{2,1}$	$0_{2,2}$	$3_{2,3}$	$6_{2,4}$	$1_{2,5}$
d_{i3}	$3_{3,0}$	$1_{3,1}$	$3_{3,2}$	$0_{3,3}$	$3_{3,4}$	$2_{3,5}$
e_{i4}	$5_{4,0}$	$4_{4,1}$	$6_{4,2}$	$3_{4,3}$	$0_{4,4}$	$2_{4,5}$
f_{i5}	$0_{5,0}$	$2_{5,1}$	$1_{5,2}$	$2_{5,3}$	$2_{5,4}$	$0_{5,5}$

HMI *(Candidate 1-itemset)*

a=7	*b*=5	*c*=8	*d*=5	*e*=7	*f*=3
10	09	10	08	10	08
09	08	09	07	09	07
06	06	06	05	08	04
05	04	05	03	06	
03	01	04	02	05	
02		03		04	
01		02		03	
		02			

Figure 3 ผลลัพธ์จากการนับความถี่แบบเพิ่มเติมได้

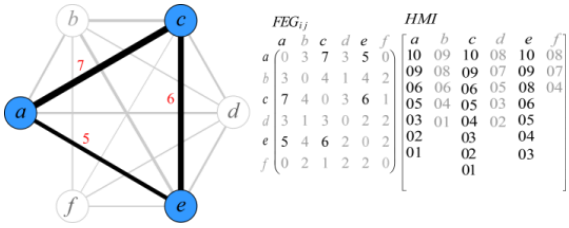
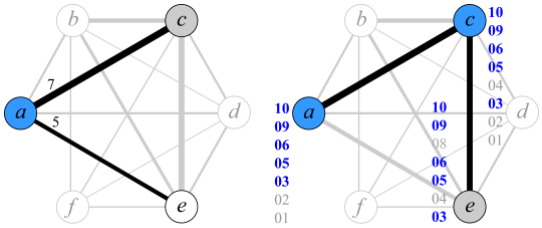


Figure 4 ผลจากการเล็มด้วยค่า TW[2]

จาก Figure 4 เมื่อได้ค่า TW[2]= 5 แล้ว สามารถนำมาเล็มกราฟได้ โดยตัดข้อมูลที่มีค่าน้อยกว่า 5 ออก โดยจะเหลือแค่เพียง 3 โหนด คือ (a, c และ e) ทำให้ลดปริมาณข้อมูลที่จะต้องใช้ในขั้นตอนต่อไปคือการสร้างกฎความสัมพันธ์

3.3 การหากฎความสัมพันธ์

ขั้นตอนการสร้างกฎความสัมพันธ์เริ่มจากการหาค่า TW[k] โดยค่า k ผู้ใช้เป็นคนกำหนดว่าต้องการหากฎที่มีค่าถ่วงน้ำหนักเท่าไหร่นับจากค่าสูงสุด (Top weight) จากนั้นนำมาเล็มกราฟจะได้ โหนดที่มีความถี่มากกว่าค่า TW[k] นำมาเก็บใน stack ที่ชื่อ NI เพื่อทำการค้นหาในแนวลึก Depth First Search (DFS) ในการค้นหากฎในกราฟที่มีค่าความถี่มากกว่า TW[k] การค้นหาจะใช้ทางเดินทางของกราฟในลักษณะวนลูป โดยจะใช้ Stack NI เป็นที่พักข้อมูลระหว่างกระบวนการ DFS และจะย้อนกลับเมื่อค่า TW[k] น้อยกว่าค่าที่กำหนด รหัสเทียมของการค้นหากฎความสัมพันธ์ แสดงใน Figure 6



(a) 2-Itemset $\{a\}\rightarrow\{c\}$ (b) 3-Itemset $\{ac\}\rightarrow\{e\}$

Figure 5 การหากฎความสัมพันธ์จากกราฟ

Algorithm2 Association rule Generation

```
k = defined level from TopWeight
TW = sort DESC uniqe weight from FEG
W = TW[k] (Weight pruning)
NI = Node list from HMI with weight >= W
while NI is not empty
NI.push(v)
  while NK is not empty
    DFS(FEG,v)   ( v is the vertex from NK )
    Stack S := {};   ( start an empty stack )
    for each vertex u, set visited[u] := false;
    push S, v;
     while (S is not empty) do
     u := pop S;
      if (not visited[u] && weight >= W)
         then
         visited[u] := true;
         for each unvisited neighbour
         w of u
         push S, w;
         RuleGen(R,S)
       end if
     end while
   end DFS()
  end while
Save R to file
```

Figure 6 Pseudo code of Rule Generation

จาก figure 5 แสดงตัวอย่างการสร้างกฎความสัมพันธ์ เมื่อผ่านการเล็ม ด้วยค่าถ่วงน้ำหนัก TW[2] = 5 แล้ว จะเหลือเพียง 3 โหนดที่นำมาใช้ในการหากฎ ขั้นตอนแรกก็นำโหนดทั้งหมดเก็บใน stack NI และนำมาหาเส้นทางของกราฟ ด้วยวิธีค้นหาแนวลึก DFS เริ่มจากโหนด a ใน Figure 5(a) แสดงโหนดที่เชื่อมต่อคือ c กับ e เมื่อคำนวณค่า $\{a\} \rightarrow \{c\}$ มีค่า 7 ซึ่งมากกว่า 5 ก็นำไปเก็บในตัวแปร R เพื่อแสดงผลกฎความสัมพันธ์ภายหลัง จากนั้น ก็ค้นหาต่อ $\{a, c\} \rightarrow \{e\}$ แสดง Figure 5(b) ซึ่งมีการเชื่อมโยงเกิน 2 โหนด จึงต้องนำค่าใน HMI ของแต่ละโหนด มา intersection กัน เพื่อหาความถี่ว่าเคยอยู่ในรายการซื้อขายเดียวกันกี่ครั้งจะเขียนในรูปเซตคือ $a \cap c \cap e = 5$ ก็ยังมีค่า ≥ 5 ฉะนั้น $\{a, c\} \rightarrow \{e\}$

จึงผ่านค่า TW[k] ขั้นต่ำ สามารถนำมาสร้างเป็นกฎได้ ก็นำไปเก็บในตัวแปร R กระบวนการ DFS เส้นทางของกราฟจะทำซ้ำจนครบทุกเส้นทาง ก็จะได้กฎความสัมพันธ์และทำการพิมพ์ออกมาแสดงในระบบ ตัวอย่างผลการค้นหาตามทางเดินเส้นกราฟ DFS จะได้กฎความสัมพันธ์ดังนี้

{apple} ==> {cake}
{apple, cake} ==> {eggs}
{apple} ==> {cake, eggs}
{apple} ==> {eggs}
{cake} ==> {eggs}
{cake} ==> { eggs, apple}
{cake, eggs} ==> {apple}
{cake} ==> {apple}
{eggs} ==> {apple}
{eggs, apple} ==> {apple}
{eggs} ==> {apple, cake}
{eggs} ==> {cake}

ผลจากกฎความสัมพันธ์ที่ได้ {apple} ==> {cake} แสดงว่า เมื่อมีการซื้อ apple ก็มักจะมีการซื้อ cake ด้วย ก็สามารถนำไปใช้ในการจัดตำแหน่งการวาง apple กับ cake ไว้ใกล้กัน หรือทำรายการส่งเสริมการขาย เช่น เมื่อซื้อ apple จะได้ส่วนลดในการซื้อ cake ด้วย ทำให้ยอดการซื้อ cake เพิ่มขึ้นได้ หรือ {apple, cake} ==> {eggs} แสดงว่าคนที่ซื้อ apple และ cake ด้วยกัน มักจะซื้อ eggs ด้วย ก็สามารถจัดทำรายการส่งเสริมการขาย เมื่อมากซื้อ apple และ cake สามารถซื้อ eggs ในราคาพิเศษ ในทางกลับกัน {eggs} ==> {apple, cake} ก็แสดงว่า คนที่ซื้อ eggs มีแนวโน้มจะซื้อ apple และ cake ด้วย

3.4 การค้นหากฎความสัมพันธ์แบบพลวัต

ข้อดีของการใช้โครงสร้างข้อมูลแบบกราฟคือมีการแยกขั้นตอนการเก็บความถี่และการค้นหากฎความสัมพันธ์ออกจากกันทำให้สามารถค้นหากฎความสัมพันธ์ซ้ำได้โดยไม่ต้องทำกระบวนการนับความถี่อีก ทำให้ประหยัดเวลาในการประมวลผลมากและการเก็บข้อมูลไว้ทุกรายการสินค้าโดยข้อมูลมีความอิสระต่อกันทำให้สามารถกฎความสัมพันธ์ได้ทุกรายการสินค้า เป็นการแก้ปัญหาข้อมูลที่มีความถี่น้อยที่มักจะโดนเล็มออกได้ จึงเหมาะกับการประยุกต์ใช้ในระบบตะกร้าสินค้าอิเล็กทรอนิกส์ (Shoping cart) เมื่อลูกค้า เลือกสินค้ามา 1 ชิ้น ระบบสามารถแนะนำสินค้าที่ซื้อด้วยกันบ่อยได้ทันที เพราะโครงสร้างข้อมูลแบบกราฟ จะทำงานได้รวดเร็วในในการประมวลผลข้อมูลเชื่อมโยงติดกันอยู่

การหากฎความสัมพันธ์แบบพลวัต (Dyanmic Rule Generation) โหนดเริ่มต้น สามารถเริ่มจากโหนดใดๆ ก็ได้ เมื่อเลือกโหนด เริ่มต้นแล้ว ระบบจะทำการหาค่าถ่วงน้ำหนักของโหนดนั้น ส่วนค่า k สามารถให้ค่าเดิมได้ โดยค่า k ผู้ใช้เป็นคนกำหนดว่าต้องการหากฎที่มีค่าถ่วงน้ำหนักเท่าไหร่นับจากค่าสูงสุด (Top weight) จากนั้นนำมาเล็มกราฟจะได้ โหนดที่มีความถี่มากกว่าค่า IW[k] (IW= Items Weight) นำมาเก็บในstack ที่ชื่อ NI เพื่อทำการค้นหาในแนวลึก ในการค้นหากฎในกราฟที่มีค่าความถี่มากกว่า IW[k]. การค้นหาจะใช้ทางเดินของกราฟในลักษณะวนลูบ โดยจะใช้ Stack NI เป็นที่พักข้อมูลระหว่างกระบวนการ DFS และจะย้อนกลับเมื่อค่า IW[k]น้อยกว่าค่าที่กำหนด pseudo code แสดงใน Figure 7

Algorithm3 Dynamic rule Generation

```
k = defined level from Maximum weight
IW = sort DESC uniqe weight from FEG
W = IW[k] (Weight pruning)
NK = Node list from HMI with weight >= W
while Nk is not empty
NK.push(v)
  while NN is not empty
    DFS(FEG,v)   ( v is the vertex from NK )
    Stack S := {};   ( start an empty stack )
    for each vertex u, set visited[u] := false;
    push S, v;
     while (S is not empty) do
     u := pop S;
      if (not visited[u] && weight >= W)
      then
      visited[u] := true;
      for each unvisited neighbour w of u
         push S, w;
         RuleGen(R,S)
      end if
     end while
    end DFS()
  end while
Save R to file
```

Figure 7 Pseudo code of Dynmic Rule Generation

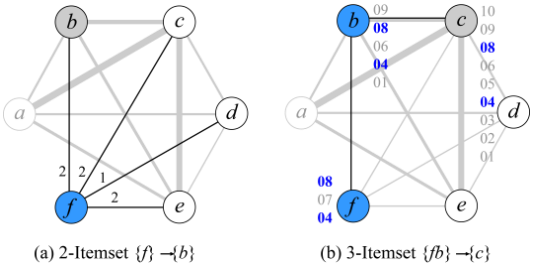


Figure 8 การหากฎความสัมพันธ์จากโหนด f

Table 3 ค่าถ่วงน้ำหนักของโหนด f

k	*ItemWeight[k]*	Frequent	*Support*	*Confidence*
0	{f, b}	2	0.7	1
1	{f, d}	2	0.6	0.75
2	**{f, e}**	**2**	**0.5**	**0.71**
3	{f, c}	1	0.4	0.8

จาก Figure 8(a) เริ่มจากเมื่อผู้ใช้เลือก node f เป็นโหนดเริ่มต้น ระบบจะนำค่า k ที่กำหนดไว้แล้วคือ 2 นำมาหาค่าถ่วงน้ำหนัก IW[2] = 2 จากค่า IW(2, 2, 2, 1) ดังแสดงใน Table 3 เป็นค่าถ่วงน้ำหนักของโหนด f เมื่อผ่านการเล็มด้วยค่าถ่วงน้ำหนัก IW[2] = 2 แล้ว จะเหลือเพียง 3 โหนดที่นำมาใช้ในการหากฎความสัมพันธ์คือ b d และ e จากนั้นก็นำโหนดทั้งหมดเก็บใน stack NI และนำมาหาเส้นทางของกราฟด้วยวิธีค้นหาแนวลึก DFS เริ่มจาก node f ใน Figure 8(a) แสดงโหนดที่เชื่อมต่อคือ f กับ b เมื่อคำนวณค่า $\{f\} \rightarrow \{b\}$ มีค่า 2 ซึ่งมากกว่าหรือเท่ากับ 2 ก็นำไปเก็บในตัวแปร R เพื่อแสดงผลกฎความสัมพันธ์ภายหลัง จากนั้น ก็ค้นหาต่อ $\{f, b\} \rightarrow \{c\}$ แสดงใน Figure 8(b) ซึ่งมีการเชื่อมโยงเกิน 2 โหนด จึงต้องนำ HMI ของแต่ละ node มา intersection กัน คือ $f \cap b \cap c = 2$ ก็ยังมีค่า ≥ 2 ฉะนั้น $\{f, b\} \rightarrow \{c\}$ จึงผ่านค่า IW[k] ขั้นต่ำ สามารถนำมาสร้างเป็นกฎได้ ก็นำไปเก็บในตัวแปร R และทำกระบวนการ DFS เส้นทางของกราฟ (path graph) จนครบและแสดงผลกฎความสัมพันธ์ออกมา การหาค่า IW จากสมการที่ 6 ได้จากการเรียงข้อมูลใน เมทริกซ์ FEG จากมากไปน้อย เมื่อลำดับ I คือ แถวลำดับของรายการสินค้าที่เลือก

$$IW = DESC\left[FEG_{i,j} \middle| where\, i = selected\ item\right] \qquad (6)$$

4. ผลการดำเนินงาน

เพื่อเปรียบเทียบประสิทธิภาพการทำงานของ Algorithm ใหม่ กับ Algorithm เดิมที่มีอยู่คือ Apriori กับ FP-Growth โดยใช้ชุดข้อมูลเดียวกัน ดัง Table 4 แสดงชุดข้อมูลที่ใช้ในการทดสอบประสิทธิภาพ มี 4 datasets ด้วยกัน คือ Retail T10I4D100K T40I10D100K และ OnlineRetail

Table 4 คุณลักษณะของชุดข้อมูลที่ใช้ทดสอบ

	Data	*Items*	*Avg*	*Transaction*
1	Retail	16,470	13	88,163
2	T10I4D100K	870	11	100,000
3	T40I10D100K	942	40	100,000
4	OnlineRetail	2,603	8	541,909

โดยชุดข้อมูล Retail จากการสร้างโดย Tom Brijs เป็นข้อมูลจากร้านค้าปลีก Belgian retail store มีจำนวนสินค้า 16,470 ชิ้น จำนวนการซื้อขายทั้งหมด 88,163 รายการ ชุดข้อมูล T10I4D100K และ T40I10D100K เป็นชุดข้อมูลจาก Frequent Itemset Mining Dataset Repository สร้างโดย IBM Almaden Quest research group แต่ละชุดข้อมูลมีจำนวนรายการซื้อขาย 100,000 รายการ T10I4D100K จะมีสินค้า 870 ชิ้น ค่าเฉลี่ยต่อ 1 รายการซื้อขายอยู่ที่ 11 ชิ้น ส่วน T40I10D100K จะมีสินค้า 942 ชิ้น ค่าเฉลี่ยต่อ 1 รายการซื้อขายอยู่ที่ 40 ชิ้น และ OnlineRetail นำมาจาก The UCI Machine Learning Repository ชุดข้อมูลประกอบไปด้วยรายการซื้อขาย 541,909 รายการ จำนวนสินค้า2603 ชิ้นเป็นข้อที่เก็บช่วง 01/12/2010 และ 09/12/2011 สำหรับ บริษัทค้าปลีกออนไลน์ที่ไม่ใช่ร้านค้าที่จดทะเบียนในประเทศอังกฤษและ บริษัท จำหน่ายของขวัญที่ ลูกค้าส่วนใหญ่ เป็น บริษัทเป็นผู้ค้าส่ง

Table 5 Top weight from retail dataset

	2-Itemsets	*Frequent*	*Support*	*Rules*
0	{39,48}	29,142	0.33054	2
1	{39,41}	11,414	0.12946	4
2	{38,39}	10,345	0.11733	6
3	**{41,48}**	**9,018**	**0.10228**	**8**
4	{32,39}	8,455	0.09590	10
5	{32,48}	8,034	0.09112	12
6	{38,48}	7,994	0.09067	12
7	{38,41}	3,897	0.04420	34
8	{32,41}	3,196	0.03625	36
9	{38,170}	3,031	0.03438	42

Table 6 Top weight from T10I4D100K

	2-Itemsets	*Frequent*	*Support*	*Rules*
0	{217,346}	1,336	0.01336	2
1	{368,829}	1,194	0.01194	6
2	{789,829}	1,194	0.01194	6
3	**{368,862}**	**1,193**	**0.01193**	**8**
4	{39,825}	1,187	0.01187	10
5	{39,704}	1,107	0.01107	12
6	{704,825}	1,102	0.01102	14
7	{227,390}	1,049	0.01049	16
8	{390,722}	1,042	0.01042	18
9	{227,722}	995	0.00995	26

ผลจากขั้นตอนการนับความถี่ (Frequency Itemset Generation) ดังแสดงใน Table 5 จะสามารถจำแนกข้อมูลมาแสดงในรูปแบบ TopWeight 10 อันดับแรก โดยสมมุติว่ามีการกำหนดค่า k = 3 TW[3] ของ Retail dataset จะเท่ากับ 0.10228 และจำนวนกฎที่จะเป็นไปได้ = 8

ใน Table 6 จะสามารถจำแนกข้อมูลมาแสดงในรูปแบบ TopWeight 10 อันดับแรก โดยสมมุติว่ามีการกำหนดค่า k = 3 TW[3] ของ T10I4D100K dataset จะเท่ากับ 0.01193 และจำนวนกฎ(number of rule) ที่จะเป็นไปได้ = 8

Table 7 Top weight from T40I10D100K

	2-Itemsets	*Frequent*	*Support*	*Rules*
0	{368,529}	7,500	0.07500	2
1	{368,829}	6,957	0.06957	2
2	{368,682}	6,130	0.06130	6
3	**{217,368}**	**6,125**	**0.06125**	**8**
4	{368,489}	6,120	0.06120	10
5	{368,692}	5,867	0.05867	12
6	{368,510}	5,716	0.05716	14
7	{529,829}	5,638	0.05638	16
8	{368,419}	5,569	0.05569	18
9	{368,914}	5,560	0.05560	20

ใน Table 7 จะสามารถจำแนกข้อมูลมาแสดงในรูปแบบ TopWeight 10 อันดับแรก โดยสมมุติว่ามีการกำหนดค่า k = 3 TW[3] ของ T40I10D100K dataset จะเท่ากับ 0.06125 และจำนวนกฎ(number of rule)ที่จะเป็นไปได้ = 8

Table 8 Top weight from OnlineRetail

	2-Itemsets	*Frequent*	*Support*	*Rules*
0	{1534,1943}	29,159	0.05381	2
1	{1816,1834}	20,265	0.03740	4
2	{225,1215}	18,970	0.03501	6
3	**{1384,1989}**	**15,134**	**0.02793**	**8**
4	{1534,1582}	14,136	0.02609	10
5	{225,1336}	13,793	0.02545	12
6	{1126,2183}	11,093	0.02047	14
7	{426,1534}	9,860	0.01819	16
8	{426,479}	9,752	0.01800	18
9	{90,1943}	9,590	0.01770	20

ใน Table 8 จะสามารถจำแนกข้อมูลมาแสดงในรูปแบบ TopWeight 10 อันดับแรก โดยสมมุติว่ามีการกำหนดค่า k = 3 TW[3] ของ OnlineRetail dataset จะเท่ากับ 0.02793 และจำนวนกฎ(number of rule)ที่จะเป็นไปได้ = 8

จาก Table 6 ถึง Table 8 จะเห็นได้ว่า การเรียงลำดับจากค่าถ่วงน้ำหนักสูงสุดแต่ละชุดข้อมูลไม่เท่ากัน ค่า Tw[0] คือข้อมูลที่มีค่าสูงสุดจะมีเพียงคู่เดียวและการจะเล็มด้วยค่าถ่วงน้ำหนักลำดับที่สาม TW[3] ผลที่ได้นำมาเขียนเป็นกราฟในแต่ละชุดข้อมูล สามารถแสดงได้ดัง Figure 9

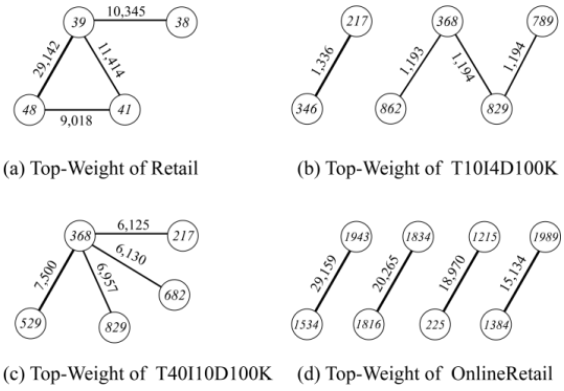


Figure 9 Node Top Weight

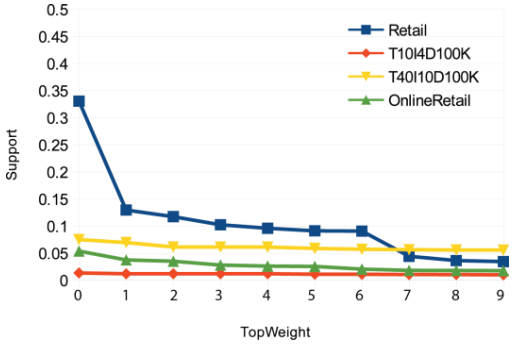


Figure 10 Support from Top Weight

จาก Table 6 ถึง Table 8 เมื่อมองในเรื่องค่าสนับสนุน (Support) ใน 10 อันดับแรก มาเขียนเป็นกราฟใน Figure 10 จะเห็นได้ว่ามีเพียง Retail dataset ที่มีค่าเกิน 0.1 โดยทั่วไปแล้วถ้าในร้านค้ามีสินค้ามาก ยอดขายของสินค้าแต่ละชิ้น จะไม่เกิน 10% ของปริมาณยอดขายทั้งหมด ทำให้การจะหาค่าสนับสนุนของสินค้าแต่ละชิ้นเป็นเรื่องยาก แต่การใช้เมทริกซ์แถวประชิดในการเก็บข้อมูล ทำให้สามารถหาค่าสนับสนุนได้ทุกรายการสินค้า จึงสามารถหากฎความสัมพันธ์ได้ทุกรายการสินค้า

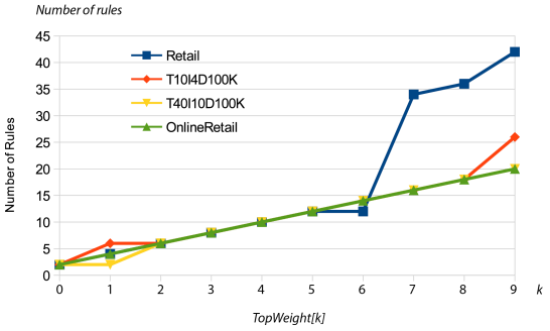


Figure 11 Number of Rule with Weight level

เมื่อนำค่าสนับสนุน จาก Table 6 ถึง Table 8 มาหากฎความสัมพันธ์จำนวนกฎที่ได้สามารถแสดงได้ดังกราฟใน Figure 11 แสดงความสัมพันธ์ของ Number of rules ต่อระดับของ TopWeight ที่กำหนดโดยค่า k ซึ่งการหากฎจาก TopWeight ทำให้สามารถคาดเดาจำนวนกฎที่จะเกิดขึ้นได้ หากเป็นข้อมูลที่เก็บจากการซื้อขายจริงเหมือนอย่าง OnlineRetail Dataset ข้อมูลแต่ละรายการจะไม่ค่อยมีซ้ำกัน ทำให้ผลของจำนวนของกฎจะเพิ่มขึ้นอย่างสม่ำเสมอ คือเท่ากับหรือมากกว่า ค่า k+1 คูณ 2 ดังสมการที่ 7 ซึ่งมีประโยชน์มากในการนำไปใช้กับการแนะนำสินค้าที่มักซื้อด้วยกันบ่อยในเวบไซต์ เพราะมีพื้นที่แสดงผลจำกัด ปัจจุบันนิยมเปิดด้วยโทรศัพท์มือถือหรือสมาร์ทโฟน หน้าจอมีขนาดเล็กและพื้นที่การแสดงผลน้อย จำนวนกฎความสัมพันธ์จึงไม่ควรเกิน 10 รายการ เพื่อให้สามารถแสดงผลได้อย่างเหมาะสมทั้งในจอคอมพิวเตอร์และบนสมาร์ทโฟน

$$NumberOfRule \geq 2(k+1) \qquad (7)$$

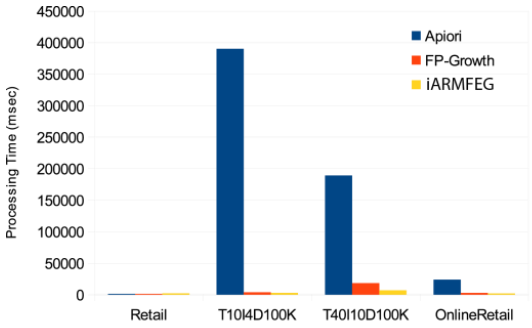


Figure 12 Processing time

จากผลการทดลองเรื่องเวลาที่ใช้ในการประมวลผลดังแสดงใน figure 12 เป็นการตั้งค่า minsup ไว้ที่ 95% จาก TopWeight ซึ่งจำนวนกฎที่ได้จะมีปริมาณน้อยอยู่ iARMFEG จะใช้เวลาในการประมวลผล (processing time) น้อยที่สุด รองลงมาคือ FP-Growth และ Apriori จะใช้เวลานานที่สุด เพราะมีการสแกนหลายรอบ ส่วน Apriori จะใช้เวลานานโดยเฉพาะ T40I10D100K ซึ่งจะมีค่าเฉลี่ยของ items ต่อ Transaction ID มาก ทำให้ยิ่งต้องใช้เวลาประมวลผลนาน

จากผลการทดลองเรื่องหน่วยความจำ (memory use) ที่ใช้ในการประมวลผลดังแสดงใน figure 13 เป็นการตั้งค่า minsup ไว้ที่ 95% จาก TopWeight ซึ่งจำนวนกฎที่ได้จะมีปริมาณน้อยอยู่ การใช้หน่วยความจำ จะไม่แตกต่างกันมากนัก โดย iARMFEG จะใช้หน่วยความจำมากใน Retail Dataset เพราะจำนวน items มีมากถึง 16,470 items ทำให้ต้องมีการสร้าง Adjacency Matrix ขนาดใหญ่ รองลงมาคือ FP-Growth และ Apriori จะใช้หน่วยความจำน้อยที่สุด แต่ iARMFEG จะมีข้อดีคือในการหากฎความสัมพันธ์ครั้งต่อๆ ไป ไม่ต้องทำขั้นตอนนับความถี่อีก ทำให้ประหยัดเวลาในส่วนนี้มาก ซึ่งโครงสร้างข้อมูลแบบกราฟที่ออกแบบไว้เมื่อนำไปใช้กับฐานข้อมูลแบบกราฟ จะลดเรื่องการใช้หน่วยความจำลงได้

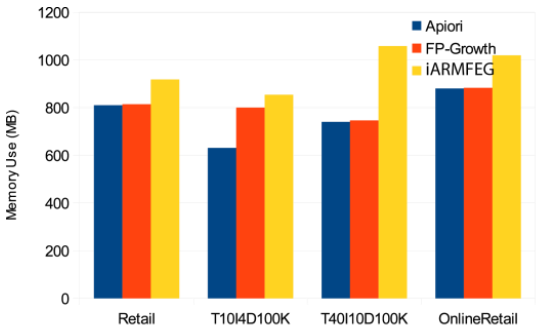


Figure 13 Memory used

5. สรุป

ผลการทดลอง iARMFEG ในชุดข้อมูลหลายตัวจะเห็นได้ว่า สามารถทำงานได้อย่างรวดเร็วในการหาค่าถ่วงน้ำหนักได้จากทุกชุดข้อมูล ลดขั้นตอนการลองผิดลองถูกเพื่อหาค่าที่เหมาะสมของค่าสนับสนุน ได้มาก ประสิทธิภาพการทำงานจะขึ้นอยู่กับจำนวน สินค้าทั้งหมดที่มีในชุดข้อมูล ที่นำมาสร้างเมทริกซ์แถวประชิด ซึ่งการค้นหาข้อมูลจะเท่ากับ $O(|V|^2)$ ข้อดีของการแยกส่วนของการนับความถี่และการค้นหากฎความสัมพันธ์ออกจากกัน ทำให้ในการค้นหากฎได้หลายๆ ครั้งโดยไม่ต้องทำขั้นตอนการนับความถี่อีก ถ้ามีรายการซื้อขายใหม่เข้ามาก็สามารถเพิ่มเติมจากรายการเดิมที่มีอยู่แล้วได้ ขณะที่ขั้นตอนวิธี Apriori และ FP-Growth ต้องอ่านข้อมูลใหม่ทุกครั้ง ทำให้ iARMFEG ใช้เวลาในการอ่านข้อมูลน้อยกว่า เพราะไม่ต้องมีการนับความถี่ใหม่ ทำให้ทำงานครั้งถัดมาได้รวดเร็ว สามารถนำไปประยุกต์ใช้กับระบบ shopping cart ในการแนะนำสินค้าที่มักซื้อด้วยกัน หรือสามารถใช้ ฐานข้อมูลแบบกราฟในการเก็บข้อมูลแทนได้

6. เอกสารอ้างอิง

[1] Agrawal R. and Srikant R. 1994. Fast Algorithms for Mining Association Rules in Large Databases. **Proceedings of the 20th International Conference on Very Large Data Bases**, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA: 487-499.

[2] Han J., Pei J. and Yin Y. 2000. Mining frequent patterns without candidate generation. **Proceeding SIGMOD '00 Proceedings of the 2000 ACM SIGMOD international conference on Management of data**. Dallas, Texas, USA: 1-12.

[3] Sarath K. N. V. D. and Ravi V. 2013. Association rule mining using binary particle swarm optimization. **Eng. Appl. Artif. Intell**. 26 : 1832-1840.

[4] Al-Dharhani G., Othman Z., and Bakar. A. 2014. A Graph-Based Ant Colony Optimization for Association Rule Mining. **Arabian Journal for Science and Engineering**. 39 : 4651-4665.

[5] Jiawei H., Jianyong W., Ying L. and Tzvetkov P. 2002. Mining top-k frequent closed patterns without minimum support. **2002 IEEE International Conference on Data Mining, 2002 Proceedings. Maebashi City, Japan.**: 211-218.

[6] Burdick D., Calimlim M., Flannick J., Gehrke J. and Yiu T. 2005. MAFIA: a maximal frequent itemset algorithm. **Knowledge and Data Engineering, IEEE Transactions**. 17 : 1490-1504.

[7] Uday Kiran R. and Krishna Re P. 2009. An improved multiple minimum support based approach to mine rare association rules. **IEEE Symposium on Computational Intelligence and Data Mining. Nashville, TN, USA.**: 340-347.

[8] Wang W., Yang J. and Yu P. 2004. WAR: Weighted Association Rules for Item Intensities. Knowledge and Information Systems, 6: 203-229.

[9] Pears R., Koh Y. S., Dobbie G. and Yeap W. 2013. Weighted association rule mining via a graph based connectivity model. **Information Sciences**. 218 : 61-84.

[10] Tao F., Murtagh F. and Farid M. 2003. Weighted Association Rule Mining using weighted support and significance framework. **Proceeding KDD '03 Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, Washington, D.C.:** 661-666.

[11] Duck J., Long J., Hwang B. and Keun-Ho R. 2007. Frequent Pattern Mining using Bipartite Graph. **18th International Workshop on Database and Expert Systems Applications (DEXA 2007). Regensburg, Germany**. : 182-186.

[12] Pears R., Pisalpanus S. and Koh Y. S. 2015. A graph based approach to inferring item weights for pattern mining. **Expert Systems with Applications**, 42 : 451-461.

[13] Dev P A., V S. N. and Joseph P. 2013. GARM: A Simple Graph Based Algorithm For Association Rule Mining. **International Journal of Computer Applications**. 76 : 1-4.

[14] Suksakaophong P., Meesad P. and Unger H. 2018. ARMFEG: Association Rule Mining by Frequency-Edge-Graph for Rare Items. **The Fourteenth International Conference on Computing and Information Technology. Chiang Mai, Thailand**,: 13-22.