การค้นหารูปแบบปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแสด้วยเทคนิคการเลื่อนหน้าต่าง

ธาชินี มีเสมา

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรมหาบัณฑิต
สาขาวิชาเทคโนโลยีสารสนเทศ
คณะวิทยาการสารสนเทศ มหาวิทยาลัยบูรพา
สิงหาคม 2561

MINING TOP-K FREQUENT-REGULAR ITEMSETS FROM DATA STREAMS BASED ON SLIDING WINDOW TECHNIQUE

TASHINEE MESAMA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE MASTER DEGREE OF SCIENCE IN INFORMATION TECHNOLOGY
FACULTY OF INFORMATICS BURAPHA UNIVERSITY
AUGUST 2018

คณะกรรมการควบคุมวิทยานิพนธ์และคณะกรรมการสอบงานนิพนธ์ ได้พิจารณาวิทยานิพนธ์ของ ธาชินี มีเสมา ฉบับนี้แล้ว เห็นสมควรรับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ ของมหาวิทยาลัยบูรพาได้

คณะกรรมการควบคุมงานนิพนธ์

.....อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร.โกเมศ อัมพวัน)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธาน
(ผู้ช่วยศาสตราจารย์ ดร.อนุชิต จิตพัฒนกุล)

.....กรรมการ
(ผู้ช่วยศาสตรจารย์ ดร.โกเมศ อัมพวัน)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.อุรีรัฐ สุขสวัสดิ์ชน)

คณะวิทยาการสารสนเทศ อนุมัติให้รับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ ของมหาวิทยาลัยบูรพา

.....คณบดีคณะวิทยาการสารสนเทศ
(ผู้ช่วยศาสตราจารย์ ดร.กฤษณะ ชินสาร)
วันที่.....๒๖..... เดือน สิงหาคม พ.ศ. 2561

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดีด้วยความกรุณาจากผู้ช่วยศาสตราจารย์ ดร. โกเมศ อัมพวัน อาจารย์ที่ปรึกษาวิทยานิพนธ์ที่เสียสละเวลาให้ความรู้ ให้คำปรึกษา แนะแนวทางในการทำวิทยานิพนธ์ตลอดจนตรวจสอบแก้ไขข้อบกพร่องต่างๆ ตลอดช่วงเวลาของการทำวิทยานิพนธ์ในครั้งนี้ รวมถึงให้กำลังใจผู้วิจัยเสมอมา ผู้วิจัยรู้สึกซาบซึ้งในความปรารถนาดีที่มอบให้ จึงขอกราบขอบคุณมา ณ โอกาสนี้

ขอขอบคุณคณาจารย์คณะวิทยาการสารสนเทศที่คอยอบรมสั่งสอนและให้ความรู้ ประสบการณ์ตลอดช่วงเวลาที่ทำการศึกษา ทำให้เกิดองค์ความรู้อันเป็นประโยชน์ในการทำวิทยานิพนธ์และการดำรงค์ชีวิตต่อไป

ขอขอบคุณผู้ช่วยศาสตราจารย์ ดร.อนุชิต จิตพัฒนกุล ที่กรุณาให้เกียรติเป็นประธาน โดยมีผู้ช่วยศาสตราจารย์ ดร.อุรีรัฐ สุขสวัสดิ์ชน เป็นกรรมการในการสอบวิทยานิพนธ์

สุดท้ายนี้ ขอขอบคุณครอบครัว ที่พยายามเข้าใจในขั้นตอนของการทำวิทยานิพนธ์ คอยเป็นกำลังใจและอยู่เบื้องหลังทุกความสำเร็จของผู้วิจัยตลอดมา รวมถึงเพื่อนสาขาเทคโนโลยีสารสนเทศ รุ่นที่ 10, เพื่อนห้องปฏิบัติการ CIL, เพื่อนที่ทำงาน และบุคคลทุกท่านที่ผ่านเข้ามาในชีวิตที่คอยส่งเสริม ช่วยเหลือ เป็นกำลังใจให้กันมาโดยตลอดโดยเฉพาะคุณบุญพร้อม ปัญญาใส และคุณพจน์สพร แซ่ลิ้ม ที่คอยชี้แนะและคอยช่วยเหลือผู้วิจัยด้วยดีเสมอมา

ธาชินี มีเสมา

57920140: สาขาวิชา: เทคโนโลยีสารสนเทศ; วท.ม. (เทคโนโลยีสารสนเทศ)
คำสำคัญ: การทำเหมืองข้อมูล, รูปแบบปรากฏบ่อยและปรากฏอย่างสม่ำเสมอ, รูปแบบปรากฏบ่อยสุดเคอันดับแรกและปรากฏอย่างสม่ำเสมอ, บิตเวกเตอร์, ข้อมูลกระแส, เทคนิคการเลื่อนหน้าต่าง

ธาชินี มีเสมา: การค้นหารูปแบบปรากฏบ่อยสุดเคอันดับและปรากฏอย่างสม่ำเสมอจากฐานข้อมูลกระแสด้วยเทคนิคการเลื่อนหน้าต่าง (MINING TOP-K FREQUENT-REGULAR ITEMSETS FROM DATA STREAMS BASED ON SLIDING WINDOW TECHNIQUE)
คณะกรรมการควบคุมงานวิทยานิพนธ์: ผู้ช่วยศาสตราจารย์ ดร. โกเมศ อัมพวัน, Ph.D., 55 หน้า. ปี พ.ศ. 2561.

การค้นหารูปแบบที่ปรากฏบ่อยและปรากฏอย่างสม่ำเสมอได้รับความสนใจอย่างมากและถูกนำมาประยุกต์ใช้ในงานด้านต่างๆ อาทิเช่น งานด้านวิทยาศาสตร์ ด้านการแพทย์ ด้านการทำธุรกรรมต่างๆ เป็นต้น การค้นหารูปแบบที่ปรากฏบ่อยและปรากฏอย่างสม่ำเสมอภายใต้กรอบการพิจารณาความสำคัญของความถี่/ความบ่อย และความสม่ำเสมอของการปรากฏซ้ำ ซึ่งการกำหนดค่าขีดแบ่งสนับสนุนที่เหมาะสมและความสม่ำเสมอ เพื่อวัดความน่าสนใจของเซตรายการนั้นสามารถทำได้ยาก ซึ่งอาจส่งผลให้ไม่มีผลลัพธ์ที่เพียงพอต่อความต้องการซึ่งทำให้ผลลัพธ์ที่ได้ไม่มีประโยชน์ต่อการนำข้อมูลไปใช้การวิเคราะห์ นอกจากนี้ยังมีการค้นหารูปแบบที่น่าสนใจจากฐานข้อมูลกระแสที่มีการไหลเวียนของข้อมูลอย่างต่อเนื่อง เพื่อประยุกต์ใช้กับธุรกิจที่มีการเพิ่มขึ้นของทรานแซกชันเป็นจำนวนมาก อาทิเช่น ข้อมูลการใช้บริการอินเทอร์เน็ต ข้อมูลหุ้น บริการที่ลูกค้าใช้บริการบ่อยในช่วงเวลาหนึ่งๆ จึงได้มีการคิดค้นการค้นหารูปแบบที่ปรากฏบ่อยสุดเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากฐานข้อมูลกระแส โดยการใช้เทคนิคการเลื่อนหน้าต่างในการพิจารณาเหตุการณ์ที่เกิดขึ้นล่าสุด ซึ่งถือได้ว่ามีความสำคัญมากกว่าเหตุการณ์ที่เกิดก่อนหน้านี้ ขั้นตอนวิธีที่นำเสนอเรียกว่า TFRIM-DS (Top-k frequent-regular itemsets from data streams) ซึ่งจะได้ผลลัพธ์ของเซตรายการที่ปรากฏสม่ำเสมอและมีค่าสนับสนุนสูงสุดเคอันดับแรกในหน้าต่างการพิจารณาในปัจจุบัน นอกจากนี้ยังมีการใช้บิตเวกเตอร์ที่ออกแบบมาเพื่อใช้ในการจัดเก็บเซตรายการ และนำบิตเวกเตอร์มาใช้ซ้ำเมื่อมีการเปลี่ยนแปลงของหน้าต่างการพิจารณา ในการทดลองจะแสดงให้เห็นถึงประสิทธิภาพของขั้นตอนวิธี TFRIM-DS เพื่อใช้สำหรับค้นหารูปแบบที่ปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากฐานข้อมูลกระแสด้วยเทคนิคการเลื่อนหน้าต่าง

57920140: MAJOR: INFORMATION TECHNOLOGY; M.Sc. (INFORMATION TECHNOLOGY)
KEYWORDS: DATA MINING, FREQUENT-REGULAR PATTERN, TOP-K FREQUENT-REGULAR PATTERN, BIT-VECTOR, DATA STREAM, SLIDING WINDOW

TASHINEE MESAMA: MINING TOP-K FREQUENT-REGULAR ITEMSETS FROM DATA STREAM BASED ON SLIDING WINDOW TECHNIQUE. ADVISORY COMMITTEE: KOMATE AMPHAWAN, Ph.D., 55 P. 2018.

Frequent-regular itemset mining has achieved a great attention and applied in several applications. In this framework, an itemset that frequently and regularly occurs in a database is identified as interesting. However, without prior knowledge, the setting appropriate support and regularity thresholds to measure interestingness of itemsets is quite difficult. This may lead to none, only few or overwhelm of generated results causing users cannot further take advantages from these itemsets. In addition, mining interesting itemsets over data streams is a challenging task on various domains. Therefore, to cope with these issues, we here propose an approach to mine top-k frequent-regular itemsets over data streams. To mine such itemsets, the concept of sliding window is applied in which recent occurrences are considered to be more important than the former occurrences. An efficient single-pass algorithm, called "TFRIM-DS" is also introduced to mine a set of k itemsets that regularly occur and have highest support in the current considered window. In addition, a bit-vector with a reuse technique is applied and designed to efficiently maintain occurrence information of each itemset. Experiments were conducted and showed efficiency of our proposed TFRIM-DS to mine top-k frequent-regular itemsets over sliding window of data streams.

สารบัญ

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในยุคปัจจุบันเทคโนโลยีสารสนเทศมีความสำคัญอย่างมากในการวิเคราะห์ข้อมูลในด้านต่างๆ ซึ่งปัจจุบันบริษัทหรือองค์กรต่างๆ มีการเก็บข้อมูลเป็นปริมาณมากจากระบบการดำเนินการ (Operational system) ที่ใช้ในการดำเนินธุรกิจ เช่น งานด้านวิทยาศาสตร์ ด้านการแพทย์ ด้านพันธุวิศวกรรมศาสตร์ ด้านกฎหมาย การทำธุรกรรมต่างๆ การติดตามการผันผวนของหุ้น ธุรกิจอสังหาริมทรัพย์ ธุรกิจการค้าหรือธุรกิจค้าปลีก นอกจากนั้นในยุคปัจจุบันเป็นยุคที่ธุรกิจการค้ามีการแข่งขันสูง ธุรกิจแบบผูกขาดมีจำนวนลดลง ซึ่งเป็นปัจจัยที่ทำให้เกิดการแข่งขันทางการตลาดมากขึ้น ดังนั้นถ้าบริษัทหรือองค์กรใดมีข้อมูลเชิงกลยุทธ์ที่ดีจะช่วยให้มีความสามารถในการแข่งขันทางธุรกิจได้ ด้วยเหตุนี้องค์กรต่างๆ จึงมีความสนใจที่จะได้รับข้อมูลข่าวสาร ข้อมูลเชิงกลยุทธ์ เพื่อช่วยในการประกอบการตัดสินใจในด้านธุรกิจ การลงทุน ความคุ้มค่าการลงทุน รวมถึงการวางแผนทางด้านธุรกิจในอนาคต

จากความต้องการเกี่ยวกับข้อมูลเชิงกลยุทธ์ รวมถึงองค์ความรู้จากข้อมูลเพื่อใช้ประกอบการตัดสินใจที่เพิ่มขึ้น การทำเหมืองข้อมูล (Data Mining) ที่ซึ่งเป็นกระบวนการหนึ่งในการค้นหาหรือประมวลผลข้อมูลเชิงกลยุทธ์ด้วยการค้นหาองค์ความรู้ที่ถูกซ่อนอยู่ในข้อมูลปริมาณมากจึงถูกคิดค้นขึ้นและพัฒนาในวงกว้าง โดยเทคนิคการทำเหมืองข้อมูลจะประกอบไปด้วยการหากฎความสัมพันธ์ของข้อมูล (Association Rules) การจำแนกประเภทของข้อมูล (Classification) การจัดกลุ่มข้อมูล (Clustering) การค้นหาความผิดปกติของข้อมูล (Outlier analysis) และอื่นๆ โดยในปัจจุบันได้มีการนำเทคนิค Data Mining ไปประยุกต์ใช้ในงานด้านต่าง ๆ มากขึ้น ทั้งในด้านการส่งเสริมการขายสินค้า การจัดวางสินค้าในห้างสรรพสินค้า ด้านการวิเคราะห์เครดิตลูกค้าในธนาคาร ด้านวิทยาศาสตร์ ด้านการสนับสนุนการรักษาพยาบาล ด้านอสังหาริมทรัพย์ และในด้านอื่นๆ อีกมาก

จากข้างต้น การหากฎความสัมพันธ์ของข้อมูลเป็นกระบวนการหนึ่งในการทำเหมืองข้อมูลที่ได้รับความนิยมมากในการหาความสัมพันธ์ระหว่างข้อมูลสองรายการใดๆ หรือข้อมูลที่มีมากกว่าสองรายการขึ้นไปจากฐานข้อมูลขนาดใหญ่ โดยในการวัดความน่าสนใจของความสัมพันธ์จะวัดจากความถี่ (หรือความบ่อย) ในการปรากฏร่วมกันของรายการนั้นๆ การหากฎความสัมพันธ์ของข้อมูลจะประกอบด้วย 2 ขั้นตอนหลักคือ 1) การค้นหาเซตรายการที่ปรากฏร่วมกันบ่อยๆ หรือถี่ๆ (โดยเรียกเซตรายการนี้ว่า Frequent Itemset หรือ Frequent Pattern) และ 2) การสร้างกฎความสัมพันธ์จากเซตรายการที่ปรากฏร่วมกันบ่อยๆ ที่ซึ่งค้นพบจากขั้นตอนแรก ตามลำดับ โดยผลลัพธ์ที่ได้

สามารถเขียนได้ในรูปเซตของรายการที่เป็นเหตุ ไปสู่เซตของรายการที่เป็นผลที่ซึ่งสามารถทำให้ทราบถึงพฤติกรรมของการปรากฏขึ้นของเซตรายการนั้นๆ โดยในเบื้องต้นการหากฎความสัมพันธ์ของข้อมูลได้ถูกประยุกต์ใช้ในการวิเคราะห์พฤติกรรมการซื้อสินค้าของลูกค้าจากการจัดเก็บข้อมูลการซื้อสินค้า ณ จุดรับชำระเงิน (Point of Sell, PoS) เพื่อใช้หากลุ่มสินค้าที่ถูกซื้อร่วมกันบ่อยๆ ที่ซึ่งจะสามารถช่วยในเรื่องของการจัดทำโปรโมชัน การจัดชั้นวางสินค้าให้สินค้าที่ถูกซื้อร่วมกันบ่อยๆ อยู่ใกล้ๆ กัน (เพื่อเพิ่มความสะดวกให้แก่ลูกค้า) การจัดทำแค็ตตาล็อกสินค้าเพื่อส่งเสริมการขาย เป็นต้น จาก 2 ขั้นตอนหลักของการหากฎความสัมพันธ์ของข้อมูล เราจะสังเกตุได้ว่าขั้นตอนการค้นหาเซตรายการที่ปรากฏบ่อยจะเป็นขั้นตอนที่ต้องการการประมวลผลที่ค่อนข้างมาก (สืบเนื่องมาจากปริภูมิสถานะที่ค่อนข้างใหญ่เมื่อมีรายการบรรจุในฐานข้อมูลมาก) นี้จึงเป็นเหตุให้มีการพัฒนาขั้นตอนวิธีเพื่อเพิ่มประสิทธิภาพในการค้นหาเซตรายการที่ปรากฏบ่อย รวมถึงการพัฒนาการค้นหารูปแบบหรือเซตรายการที่น่าสนใจภายใต้การวัดความน่าสนใจของรูปแบบนั้นๆ ในรูปแบบต่างๆ

การค้นหารูปแบบปรากฏบ่อยและสม่ำเสมอเป็นการค้นหารูปแบบแบบหนึ่งที่ถูกพัฒนาจากการค้นหารูปแบบที่ปรากฏบ่อย โดยการค้นหารูปแบบนี้จะทำการวัดความน่าสนใจของรูปแบบใน 2 ลักษณะคือ 1) ความบ่อย (ความถี่) และ 2) ความสม่ำเสมอ ของการปรากฏขึ้นของรูปแบบนั้นๆ โดยรูปแบบที่มีความน่าสนใจจะต้องปรากฏบ่อยเกินกว่าค่าขีดแบ่งสนันสนุน (Support threshold) และจะต้องปรากฏสม่ำเสมอกว่าค่าขีดแบ่งความสม่ำเสมอ (Regularity threshold) กล่าวคือ รูปแบบที่น่าสนใจจะต้องมีค่าความถี่มากกว่าหรือเท่ากับค่าขีดแบ่งสนับสนุน และ จะต้องมีค่าความสม่ำเสมอน้อยกว่าหรือเท่ากับค่าขีดแบ่งความสม่ำเสมอ โดยผู้ที่ต้องการค้นหารูปแบบจะเป็นผู้กำหนดค่าขีดแบ่งทั้งสอง แต่อย่างไรก็ตาม การกำหนดค่าขีดแบ่งสนับสนุนโดยที่ไม่ทราบถึงคุณลักษณะของข้อมูลจะเป็นเรื่องยาก หากผู้ใช้กำหนดค่าขีดแบ่งสนับสนุนมากเกินไปอาจทำให้มีรายการที่ปรากฏบ่อยปริมาณน้อย แต่ในทางตรงกันข้ามหากกำหนดค่าขีดแบ่งสนับสนุนน้อยอาจทำให้มีรายการที่ปรากฏบ่อยมีปริมาณมาก ซึ่งอาจมากกว่าที่จะสามารถนำรูปแบบดังกล่าวไปประกอบการตัดสินใจได้ ด้วยเหตุนี้จึงได้เกิดกระบวนการการค้นหารูปแบบปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอ โดยที่ผู้ใช้ไม่จำเป็นต้องกำหนดค่าขีดแบ่งสนับสนุน เพื่อลดความยุ่งยากของการกำหนดค่าขีดแบ่งสนับสนุน วิธีการนี้ผู้ใช้จำเป็นต้องกำหนดจำนวนผลลัพธ์ที่ต้องการเคอันดับแทนการกำหนดค่าขีดแบ่งสนับสนุน และกำหนดค่าขีดแบ่งความสม่ำเสมอ ซึ่งจะทำให้สามารถหารูปแบบเครูปแบบที่มีค่าสนับสนุนสูงสุดและปรากฏอย่างสม่ำเสมอ

จากที่ได้กล่าวมาข้างต้น การค้นหารูปแบบปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจะดำเนินการกับฐานข้อมูลรายการที่มีทรานแซกชันที่แน่นอน แต่ไม่สามารถค้นหารูปแบบปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากฐานข้อมูลที่มีความเปลี่ยนแปลงของทรานแซกชันได้ (กล่าวคือ ฐานข้อมูลที่มีการเพิ่มหรือลบของทรานแซกชัน) ดังนั้น ในงานวิทยานิพนธ์นี้จะ

มุ่งเน้นที่การค้นหารูปแบบปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากฐานข้อมูลกระแส (Data Stream, DS) ภายใต้การกำหนดกรอบของหน้าต่างการพิจารณา (Sliding Window) ซึ่งการค้นหารูปแบบดังกล่าวสามารถประยุกต์ใช้กับธุรกิจที่มีการเพิ่มขึ้นของทรานแซกชันเป็นจำนวนมากจากฐานข้อมูลกระแสที่มีการไหลเวียนข้อมูลอย่างต่อเนื่อง อาทิเช่น ห้างสรรพสินค้าที่มีหลายสาขาและได้รับความนิยมเป็นอย่างมาก ข้อมูลหุ้น ข้อมูลจากบริษัทผู้ให้บริการโทรศัพท์/อินเทอร์เน็ต และอื่นๆ ซึ่งจากการค้นพบรูปแบบข้างต้นจะสามารถช่วยให้ทราบถึงสินค้าและบริการที่ถูกลูกค้าซื้อ/ใช้บริการบ่อยและสม่ำเสมออันนำมาซึ่งการปรับปรุงคุณภาพของสินค้าและบริการที่ซึ่งจะช่วยให้สามารถรักษาฐานลูกค้าและช่วงชิงส่วนแบ่งทางการตลาดได้

1.2 วัตถุประสงค์ของการวิจัย

1. คิดค้นและออกแบบขั้นตอนวิธีสำหรับค้นหารูปแบบปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแสด้วยเทคนิคการเลื่อนหน้าต่าง

2. สร้างต้นแบบสำหรับการค้นหารูปแบบในลักษณะข้างต้นที่จะสามารถประยุกต์ใช้ในการดำเนินธุรกิจหรืองานทางด้านวิทยาศาสตร์ต่างๆ

3. ผู้ที่สนใจสามารถนำแนวคิดที่ได้รับจากงานวิจัยไปศึกษา เพื่อพัฒนาหรือประยุกต์ใช้ในงานวิจัยหรือประยุกต์ใช้ในการตัดสินใจพัฒนาหรือปรับปรุงกระบวนการทางธุรกิจ

1.3 ประโยชน์ที่คาดว่าจะได้รับจากการวิจัย

1. ได้ต้นแบบสำหรับการค้นหารูปแบบปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแสด้วยเทคนิคการเลื่อนหน้าต่าง เพื่อนำข้อมูลมาพัฒนา ปรับปรุง ในงานวิจัยหรือการตัดสินใจในการพัฒนาธุรกิจ เช่น การจัดทำโปรโมชัน การจัดชั้นวางสินค้า การนำเสนอสินค้าใหม่ และอื่นๆ โดยการวิเคราะห์ข้อมูลสามารถปรับใช้กับธุรกิจที่มีความหลากหลายทั้งทางด้านวิทยาศาสตร์ ด้านการแพทย์ เป็นต้น

2. สามารถนำขั้นตอนวิธีที่นำเสนอไปใช้ในการพัฒนาโปรแกรมให้สามารถใช้งานได้จริง เพื่อช่วยในการสนับสนุนการวิเคราะห์ข้อมูลต่างๆ ทางด้านธุรกิจต่างๆ ด้านวิทยาศาสตร์ ด้านการแพทย์ เป็นต้น

3. ขั้นตอนวิธีที่นำเสนอสามารถถูกใช้เป็นแนวทางในการศึกษาพัฒนาต่อยอดในงานวิจัยขั้นสูง เพื่อให้ได้ขั้นตอนวิธีที่มีประสิทธิภาพมากยิ่งขึ้น

1.4 ขอบเขตของการวิจัย

1. งานวิจัยนี้จะทำการพิจารณาความสำคัญหรือความน่าสนใจของรูปแบบในเชิงความถี่และความสม่ำเสมอของการปรากฏขึ้นของรูปแบบนั้นๆ

2. ผู้ใช้ต้องทำการกำหนดพารามิเตอร์ที่จำเป็นต่อการหาผลลัพธ์ที่จะประกอบไปด้วย 1) จำนวนรูปแบบที่ต้องการค้นหา k 2) ค่าขีดแบ่งความสม่ำเสมอ (Regularity threshold, σ_r) 3) กำหนดขนาดของหน้าต่างการพิจารณา (w) เพื่อบ่งบอกถึงขอบเขตของการพิจารณา

3. ข้อมูลที่ใช้ในการทดสอบจะต้องเป็นข้อมูลแบบฐานข้อมูลกระแส (Data Streams)

4. การทดสอบประสิทธิภาพของขั้นตอนวิธีที่คิดค้นและออกแบบจะดำเนินการทดสอบในแง่มุมของความถูกต้องของผลลัพธ์และประสิทธิภาพในการค้นหาผลลัพธ์ในด้านเวลาในการประมวลผลและพื้นที่หน่วยความจำที่ใช้จัดเก็บข้อมูล

1.5 แผนการดำเนินงาน

ในการดำเนินการวิจัยนี้ดำเนินการภายใต้ระยะที่ระบุไว้ในตารางที่ 1.1

ตารางที่ 1.1 ระยะเวลาในการดำเนินการวิจัย

ปี	แผนดำเนินงานวิจัย	เดือน											
		1	2	3	4	5	6	7	8	9	10	11	12
2558	1. ศึกษางานวิจัยที่เกี่ยวข้อง และจัดเตรียมข้อมูลในการทำงานวิจัย			←	—	—	—	—	—	→			
2558	2. กำหนดขอบเขตของงานวิจัย									←	—	—	→
2559	3. ศึกษาทฤษฎีที่เกี่ยวข้อง และกำหนดนิยามพื้นฐานที่เกี่ยวข้อง	←	—	—	—	—	—	—	→				
2559	4. ศึกษาขั้นตอนวิธีที่เกี่ยวข้อง								←	—	—	—	→
2560	5. ออกแบบขั้นตอนวิธีและพัฒนาโปรแกรมสำหรับการทดสอบ	←	—	—	—	—	—	—	—	—	—	—	→

ตารางที่ 1.1 (ต่อ)

ปี	แผนดำเนินงานวิจัย	เดือน											
		1	2	3	4	5	6	7	8	9	10	11	12
2561	6. จัดทำเอกสารและสอบโครงร่างวิทยานิพนธ์	←				→							
2561	7. ตรวจสอบข้อบกพร่องและแก้ไข ขั้นตอนวิธีให้มีประสิทธิภาพมากยิ่งขึ้น	←			→								
2561	8. พัฒนาโปรแกรมเพื่อเปรียบเทียบประสิทธิภาพ และจัดทำบทความตีพิมพ์ในงานประชุมวิชาการ				←			→					
2561	9. จัดทำเอกสารฉบับสมบูรณ์ และสอบวิทยานิพนธ์				←				→				

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงทฤษฎีพื้นฐาน งานวิจัยที่เกี่ยวข้อง และคุณลักษณะของฐานข้อมูลรายการที่ใช้ในการทดลอง รวมไปถึงนิยามต่างๆ ที่เกี่ยวข้องกับการค้นหารูปแบบที่ปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอ ดังนี้

2.1 ทฤษฎีพื้นฐาน

2.1.1. การค้นหารูปแบบปรากฏบ่อย (Mining frequent itemsets)

การค้นหารูปแบบปรากฏบ่อยเป็นการค้นหารูปแบบรายการที่ปรากฏร่วมกันบ่อยๆ ภายใต้การพิจารณาจำนวนครั้ง/ความถี่/ความบ่อยในการปรากฏขึ้นของชุดรายการในฐานข้อมูลที่มีขนาดใหญ่หรือข้อมูลกระแส โดยการค้นหารูปแบบที่ปรากฏบ่อยจะมุ่งเน้นในการค้นหาเซตรายการที่ปรากฏร่วมกันบ่อย เช่น สินค้าในห้างสรรพสินค้าที่ถูกซื้อร่วมกันบ่อยๆ, ยารักษาโรคที่แพทย์มักจะสั่งร่วมกัน, ลักษณะของโรคที่มักจะเกิดกับการดำรงค์ชีวิต เพื่อให้ผู้ประกอบการสามารถนำข้อมูลที่ได้ไปวิเคราะห์แนวโน้มทางการตลาดของสินค้า การจัดโปรโมชัน การจัดการคลังสินค้า การป้องกันโรค และอื่นๆ โดยสามารถนิยามได้ดังนี้

นิยามที่ 2.1 *$I = \{i_1, i_2, i_3, \dots, i_n\}$ เป็นรายการ (items) ที่อาจหมายถึงสิ่งของหรือเหตุการณ์ที่ต้องการหาความสัมพันธ์ โดยเรียก $i_j \in I$ ว่า "รายการ"*

นิยามที่ 2.2 *เซต $X = \{i_1, i_2, i_3, \dots, i_n\} \subseteq I$ เรียกว่า เซตรายการ (Set of items, an itemset, a pattern) เมื่อ เซต X ประกอบไปด้วยรายการทั้งสิ้น k รายการ จะเรียกว่า k-itemset, k-pattern*

นิยามที่ 2.3 *$TDB = \{t_1, t_2, t_3, \dots, t_n\}$ คือ ฐานข้อมูลรายการหรือฐานข้อมูลแบบทรานแซกชัน (Transaction Database) ซึ่งแต่ละทรานแซกชัน $TDB = \{t_1, t_2, t_3, \dots, t_n\}$ จะประกอบด้วย 1) หมายเลขกำกับทรานแซกชัน (Unique Transaction Identifier, tid) $tid = j$ และ 2) ชุดของรายการ Y ถูกบรรจุอยู่ในทรานแซกชันนั้นๆ ดังแสดงในภาพที่ 2.1*

itd	item
1	a, b, c, d
2	a, b, d
3	a, c, d
4	a, b, c, d, e
5	b, d, e
6	a, e
7	a, b, c
8	a, b, d, e
9	a, b, c, d, e
10	c

ภาพที่ 2.1 ตัวอย่างฐานข้อมูลรายการที่ประกอบไปด้วยหมายเลขทรานแซกชัน (tid) และชุดรายการที่ปรากฏในทรานแซกชัน

ถ้าชุดรายการ $X \subseteq Y$ จะสรุปได้ว่าชุดรายการ X ปรากฏขึ้นใน X ทรานแซกชัน t^j หรือ ทรานแซกชัน t^j มี X บรรจุอยู่ สามารถแสดงในรูปแบบของสัญลักษณ์ได้เป็น t_j^x ดังนั้นเมื่อทำการตรวจสอบชุดรายการ X ว่าปรากฏขึ้นในทรานแซกชันใดบ้างในฐานข้อมูล TDB จะทำให้ทราบถึง T^x

นิยามที่ 2.4 *$T^x = \{t_j^x, t_{j+1}^x, \dots, t_k^x\}$ เมื่อ $1 \le j < k \le |TDB|$ คือ ชุดของหมายเลขทรานแซกชัน (tid) ที่ถูกเรียงลำดับที่ซึ่งมี X อยู่ในทรานแซกชัน สามารถอธิบายโดยย่อได้เป็น (tidset) ดังนั้นเราสามารถทราบถึงจำนวนครั้งในการปรากฏขึ้นของชุดรายการ X ในฐานข้อมูล (ค่าความถี่หรือค่าสนับสนุน)*

นิยามที่ 2.5 *s^x คือจำนวนของการปรากฏของชุดรายการ X หรือเรียกว่าค่าสนับสนุนของรายการ X โดยสามารถคำนวนได้เป็น $s^x = |T^x|$ ชุดรายการ X จะเป็นรายการที่ปรากฏบ่อยก็ต่อเมื่อ s^x มีค่ามากกว่าหรือค่าเท่ากับค่าขีดแบ่งสนับสนุน (Minimum support threshold, min_sup (σ_s)) ที่ผู้ใช้กำหนด*

2.1.2. การค้นหารูปแบบปราฏบ่อยและปรากฏสม่ำเสมอ (Mining frequent-regular itemsets)

การค้นหารูปแบบปรากฏบ่อยและปรากฏสม่ำเสมอเป็นการค้นหารายการภายใต้การพิจารณาจำนวนครั้ง/ความถี่/ความบ่อย ร่วมกับค่าความสม่ำเสมอที่ปรากฏขึ้นในฐานข้อมูลที่มีขนาดใหญ่หรือข้อมูลกระแส เนื่องจากความถี่ของการเกิดข้อมูลในรูปแบบต่างๆ อาจจะไม่เพียงพอที่จะเป็นเงื่อนไขของการหารูปแบบที่มีความหมาย ดังนั้นลักษณะของการเกิดรูปแบบที่มีความสัมพันธ์กันจึง

เป็นปัจจัยที่สำคัญของการวิเคราะห์ข้อมูลกับงานหลายๆ ด้าน รูปแบบของการเกิดความสัมพันธ์ของข้อมูลที่เกิดขึ้นอย่างสม่ำเสมอคือ ข้อมูลจะปรากฏขึ้นโดยมีระยะห่างในช่วงที่ผู้ใช้ให้ความสำคัญ ซึ่งในการหาข้อมูลที่เกิดขึ้นในลักษณะดังกล่าว ได้มีการนำเสนอเกณฑ์การวัดความสำคัญของเซตรายการกับช่วงเวลาการปรากฏซ้ำของเซตรายการที่มุ่งเน้นในเรื่องความสม่ำเสมอของการปรากฏ ดังนั้นการค้นหารูปแบบที่ปรากฏบ่อยและปรากฏอย่างสม่ำเสมอสามารถนิยามได้ดังนี้

นิยามที่ 2.6 *ค่าความสม่ำเสมอของเซตรายการ* X (r^x) *คือ ระยะห่างมากที่สุดในระหว่างสองทรานแซกชันที่ต่อเนื่องกันในฐานข้อมูลทรานแซกชันที่มีชุดราย* X *ปรากฏอยู่ ดังนั้นค่าความสม่ำเสมอของของรายการ* X *ภายใต้การปรากฏในครั้งหนึ่งๆ ของ* X *ในทรานแซกชัน* $t_j^x \in T^x$ *กล่าวคือ* $r_{t_j}^x$ *จะสามารถคำนวณได้จาก 3 กรณีดังนี้*

1. ถ้าทรานแซกชัน t_j^x *เป็นทรานแซกชันที่มีชุดรายการ* X *ปรากฏขึ้นครั้งแรก ค่าความสม่ำเสมอ* $r_{t_j}^x$ *จะมีค่าเท่ากับ* j *ค่าของ* $r_{t_j}^x$ *จะแสดงถึงช่วงของทรานแซกชันตั้งแต่* t_1*จนกระทั่งมีการปรากฏขึ้นครั้งแรกของชุดรายการ* X *ในทรานแซกชัน* t_j^x

2. ถ้า t_k^x *เป็นลำดับของทรานแซกชันหลังจาก* t_j^x *ที่มีชุดรายการ* X *ปรากฏ และ* t_j^x *เป็นทรานแซกชันระหว่าง* t_j^x *และ* t_k^x *ที่ไม่มีชุดรายการ* X *ปรากฏขึ้น สามารถคำนวณได้จาก* $r_{t_j}^x = k - j$ *จะแสดงถึงระยะห่างของการปรากฏขึ้นของชุดรายการ* X *ระหว่าง* t_j^x *และ* t_k^x

3. ถ้าทรานแซกชัน t_k^x *เป็นทรานแซกชันที่มี* X *ปรากฏขึ้นครั้งสุดท้าย ค่าความสม่ำเสมอ* $r_{t_k}^x$ จะมีค่าเท่ากับ $|TDB| - t_k^x$ *ค่า* $r_{t_k}^x$ *จะแสดงถึงช่วงของทรานแซกชันเริ่มตั้งแต่* $|TDB|$ *จนกระทั่งมีการปรากฏขึ้นครั้งสุดท้ายของ* X *ในทรานแซกชัน* t_k^x *เมื่อพิจารณาการปรากฏขึ้นของชุดรายการที่ได้จากการหาค่าความสม่ำเสมอ จะทำให้ได้ค่าความสม่ำเสมอของชุดรายการที่มีการปรากฏซ้ำ โดยสามารถหาได้จากสมการ* $r^x = max(r_{t_j}^x, r_{t_{j+1}}^x, \dots, r_{t_k}^x)$

รายการ X *จะเป็นรูปแบบที่ปรากฏบ่อยและปรากฏสม่ำเสมอได้จะต้องมีค่าสนับสนุนมากกว่าค่าขีดแบ่งสนับสนุน (minimum support threshold,* σ_s*) และมีค่าความสม่ำเสมอน้อยกว่าหรือเท่ากับค่าขีดแบ่งความสม่ำเสมอ (maximum regularity threshold,* σ_r*) ซึ่งค่าขีดแบ่งทั้งสองถูกกำหนดโดยผู้ใช้*

ปัญหาของการกำหนดเกณฑ์ค่าขีดแบ่งสนับสนุนที่เหมาะสมกับข้อมูลรายการนั้นยังทำได้ยาก เนื่องจากหากกำหนดค่าขีดแบ่งสนับสนุนน้อยเกินไปจำนวนผลลัพธ์ที่ได้จะมีปริมาณมาก ผลลัพธ์อาจไม่เป็นที่น่าสนใจ ซึ่งใช้เวลาและพื้นที่ในการจัดเก็บข้อมูลจำนวนมาก และในทางกลับกันหากกำหนดค่าขีดแบ่งสนับสนุนมากเกินไปจำนวนผลลัพธ์ที่ได้จะมีปริมาณน้อย หรืออาจไม่พบผลลัพธ์ที่สอดคล้องกับค่าขีดแบ่งสนันสนุนนี้เลย ซึ่งทำให้เกิดการวิจัยในการทำเหมืองข้อมูลเพื่อค้นหารูปแบบที่

ปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอ โดยการใช้การกำหนดจำนวนผลลัพธ์ที่มีค่าสนับสนุนสูงสุดในเคอันดับแรก แทนการกำหนดค่าขีดแบ่งสนับสนุน

ตัวอย่างที่ 2.1 *จากภาพที่ 2.1 สามารถระบุถึงเซตของทรานแซกชันที่มีชุดรายการ* $\{a,b\}$ *ปรากฏอยู่ได้ดังนี้* $T^{ab} = \{t_1^{ab}, t_2^{ab}, t_4^{ab}, t_7^{ab}, t_8^{ab}, t_9^{ab}\}$ *ค่าสนับสนุนของ* $s^{ab} = 6$ *และสามารถคำนวณหาค่าความสม่ำเสมอที่ปรากฏในแต่ละทรานแซกชันของชุดรายการ* $\{a,b\}$ *โดย* $r^{ab} = max(1, 2-1, 4-2, 7-4, 8-7, 9-8, 10-9)$ *จะได้* r^{ab} *เท่ากับ* 3 *ถ้ากำหนดค่าขีดแบ่งสนับสนุน เท่ากับ* 5 *ค่าขีดแบ่งความสม่ำเสมอ เท่ากับ* 3 *ดังนั้นจึงสรุปได้ว่า ชุดรายการ* $\{a,b\}$ *เป็นชุดรายการที่ปรากฏบ่อยและปรากฏสม่ำเสมอ*

2.1.3. การค้นหารูปแบบปรากฏบ่อยเคอันดับแรกและปรากฏสม่ำเสมอ (Mining top-k frequent–regular itemsets)

การค้นหารูปแบบปรากฏบ่อยและปรากฏอย่างสม่ำเสมอ จะมีปัญหาในการที่ผู้ใช้จะต้องกำหนดค่าขีดแบ่งค่าสนับสนุน ซึ่งเป็นเรื่องยากสำหรับผู้ใช้งาน เพราะส่วนใหญ่ไม่มีประสบการณ์ในการพิจารณาค่าขีดแบ่งความสม่ำเสมอที่เหมาะสมกับข้อมูล อาจทำให้เกิดปัญหาดังนี้

1. หากกำหนดค่าขีดแบ่งสนับสนุนน้อยเกินไปจะทำให้เกิดรูปแบบปรากฏบ่อยจำนวนมาก รูปแบบที่ได้รับอาจไม่น่าสนใจ ทำให้สิ้นเปลืองพื้นที่จัดเก็บและใช้เวลาในการค้นหามาก

2. หากกำหนดค่าขีดแบ่งสนับสนุนน้อยเกินไป จะทำให้รูปแบบที่ปรากฏบ่อยมีจำนวนน้อย ผลลัพธ์ที่ได้อาจไม่เป็นที่ต้องการ

ด้วยเหตุนี้จึงเกิดการกำหนดจำนวนผลลัพทธ์ที่ต้องการแทนการกำหนดค่าขีดแบ่งสนับสนุนที่เรียกว่า “top-k” ซึ่งจะกำหนดจำนวนผลลัพธ์ที่มีค่าความถี่ในการปรากฏสูงสุดเคอันดับแรก ผู้ใช้จึงไม่ต้องกำหนดค่าขีดแบ่งสนับสนุน นอกจากนั้นยังมีการเพิ่มเกณฑ์ค่าขีดแบ่งความสม่ำเสมอในการค้นหาการปรากฏซ้ำของรูปแบบที่ปรากฏบ่อย เพื่อเพิ่มประสิทธิภาพในการวัดความสำคัญของการค้นหารูปแบบการปรากฏบ่อยและปรากฏอย่างสม่ำเสมอ สามารถนิยามได้ดังนี้

นิยามที่ 2.7 *เซตรายการ X จะเป็นรูปแบบที่ปรากฏบ่อยสุดเคอันดับแรกและปรากฏอย่างสม่ำเสมอได้จะต้องมีคุณสมบัติดังนี้*

1. เซตรายการที่ได้มีค่าขีดแบ่งความสม่ำเสมอน้อยกว่าหรือเท่ากับค่าขีดแบ่งความสม่ำเสมอ

2. ค่าสนับสนุนของเซตรายการอยู่ในลำดับที่ไม่เกินลำดับที่เค

การค้นหารูปแบบที่ปรากฏบ่อยสุดเคอันดับแรกและปรากฏอย่างสม่ำเสมอที่ผู้ใช้ไม่ต้องกำหนดค่าขีดแบ่งสนับสนุน ทำให้ประหยัดเวลาและพื้นที่ในการจัดเก็บข้อมูลถึงแม้ว่าจะกำหนดจำนวนของผลลัพธ์มากหรือน้อย ข้อมูลมีความกระจุกหรือกระจายตัว และสามารถทำได้โดยการอ่านข้อมูลจากฐานข้อมูลเพียงครั้งเดียว

2.2 งานวิจัยที่เกี่ยวข้อง

การค้นหารูปแบบที่ปรากฏบ่อยเริ่มจาก (Agrawal, R., และ Srikant, R., 1994) ได้นำเสนอขั้นตอนวิธีสำหรับการค้นหารูปแบบที่ปรากฏบ่อย (Frequent itemset mining, FIM) ที่พิจารณาความน่าสนใจของเซตรายการในรูปแบบของความถี่ด้วยขั้นตอนวิธีอะพริออริ (Apriori) ที่ง่ายและไม่ซับซ้อน จึงได้รับการยอมรับและเป็นที่นิยมมาก อะพริออริเป็นขั้นตอนวิธีพื้นฐานในการหาความสัมพันธ์ของข้อมูลโดยใช้หลักการค้นหาแบบบวงกว้าง ซึ่งจะทำการสร้างและอ่านข้อมูลจากฐานข้อมูลทีละทรานแซกชัน แต่เนื่องจากขั้นตอนวิธีนี้ไม่เหมาะกับการทำงานกับข้อมูลจำนวนมากหรือฐานข้อมูลขนาดใหญ่ เพราะจะต้องสร้างเซตรายการที่มีน่าจะเป็นกฎความสัมพันธ์จำนวนมากและมีการอ่านข้อมูลจากฐานข้อมูลหลายครั้ง (Han, J., และคณะ, 2000) จึงได้นำเสนอขั้นตอนวิธีเอฟพี-โกรท (Frequent pattern growth: FP-growth) ในการค้นหารูปแบบที่ปรากฏบ่อยโดยไม่ต้องสร้างเซตรายการที่น่าจะเป็นกฎความสัมพันธ์และทำการอ่านข้อมูลจากฐานข้อมูลเพียงสองครั้ง เพื่อลดระยะเวลาในการประมวลผลให้สามารถทำงานได้เร็วขึ้น มีประสิทธิภาพมากขึ้นและมีความยืดหยุ่นสูง ต่อมา (Zaki, M.J., 2000) ได้พัฒนาขั้นตอนวิธีเพื่อลดระยะเวลาการประมวณผลให้สามารถทำงานได้ดีขึ้น โดยทำการอ่านฐานข้อมูลเพียงครั้งเดียวและไม่มีการสร้างเซตรายการที่น่าจะเป็นกฎความสัมพันธ์ ในการค้นหารูปแบบความถี่ที่ปรากฏบ่อยจากเซตรายการยังมีอีกหลายงานวิจัย อาทิเช่น การค้นหาเซตรายการที่ปรากฏบ่อยในข้อมูลกระแสโดยใช้ประสิทธิภาพของเทคนิคการเลื่อนหน้าต่าง (Li, H., และ Lee, S., 2009) การค้นหาเซตรายการที่ปรากฏบ่อยเคอันดับแรกโดยหลีกเลี่ยงการกำหนดค่าขีดแบ่งสนับสนุน (Amphawan, K., และคณะ, 2009) การค้นหาเซตรายการที่ปรากฏบ่อยจากข้อมูลกระแส (Calders, T., และคณะ, 2014) ซึ่งงานวิจัยดังกล่าวจะพิจารณาในแง่มุมความถี่ของการปรากฏเท่านั้น

ต่อมาได้มีการพิจารณารูปแบบที่ปรากฏบ่อยและปรากฏอย่างสม่ำเสมอ (Tanbeer, S. K., และคณะ, 2009) ได้เสนอขั้นตอนวิธีการค้นหารูปแบบที่ปรากฏบ่อยและปรากฏอย่างสม่ำเสมอ โดยนำการปรากฏอย่างสม่ำเสมอและค่าขีดแบ่งสนับสนุนมาเป็นเกณฑ์วัดความสัมพันธ์ของเซตรายการในฐานข้อมูล ทำให้ทราบถึงเซตรายการที่ปรากฏบ่อยและปรากฏอย่างสม่ำเสมอ แนวความคิดนี้สามารถนำไปประยุกต์ใช้ได้งานหลายๆ ด้าน อาทิเช่น พฤติกรรมการซื้อสินค้าในห้างสรรพสินค้าในรายการสินค้าที่ถูกซื้อบ่อยๆ และถูกซื้ออย่างสม่ำเสมอ เพื่อใช้ในการพิจารณาเรื่องการจัดวางสินค้า การจัดวางสินค้าร่วมกับสินค้าอื่น การจัดโปรโมชันของสินค้านั้นๆ ในส่วนของอสังหาริมทรัพย์สามารถพิจารณาถึงช่วงอายุ อาชีพ การใช้ชีวิต เพื่อประกอบการตัดสินใจเลือกทำเล ออกแบบตกแต่ง กำหนดราคา รวมถึงสิ่งอำนวยความสะดวกที่ตอบโจทย์ผู้บริโภค และอื่นๆ ในการค้นหารูปแบบที่ปรากฏบ่อยและปรากฏอย่างสม่ำเสมอใช้จะต้องกำหนดพารามิเตอร์ 2 ค่า คือ 1) ค่าขีดแบ่งสนันสนุน 2) ค่าขีดแบ่งความสม่ำเสมอ สำหรับพิจารณาเซตรายการที่น่าสนใจ แต่อย่างไรก็ตาม การกำหนดค่าขีดแบ่ง

สนับสนุนโดยที่ไม่ทราบถึงคุณลักษณะของข้อมูลจะเป็นเรื่องยาก จากปัญหาดังกล่าว (Amphawan, K., และคณะ, 2009), (Amphawan, K., และคณะ, 2012), (Amphawan, K., และ Lenca, P., 2015) ได้เสนอขั้นตอนวิธีการค้นหารูปแบบที่ปรากฏบ่อยเคอันดับและปรากฏอย่างสม่ำเสมอ โดยผู้ใช้กำหนดจำนวนผลลัพธ์ที่ต้องการเคอันดับแทนการกำหนดค่าขีดแบ่งสนับสนุน และกำหนดค่าขีดแบ่งความสม่ำเสมอ ซึ่งจะทำให้สามารถหารูปแบบเครูปแบบที่มีค่าสนับสนุนสูงสุดและปรากฏอย่างสม่ำเสมอ และทำการแบ่งฐานข้อมูลรายการออกเป็นส่วนๆ ด้วยค่าขีดบ่งความสม่ำเสมอ ทำให้สามารถลดปริมาณการเปรียบเทียบเซตรายการที่ปรากฏร่วมกันได้ ทำใช้เวลาและพื้นที่การจัดเก็บข้อมูลลดน้อยลง

การค้นหารูปแบบที่ปรากฏบ่อยสุดเคอันดับจากข้อมูลกระแส (Chi-Wing-Wong, R., และ Wai-Chee Fu, A., 2006) ได้เสนอขั้นตอนวิธีการค้นหารูปแบบที่ปรากฏบ่อยจากข้อมูลกระแส Chernoff-based Algorithms และ Lossy Counting Algorithm ในการทดลองของทั้งสองขั้นตอนวิธีช่วยแก้ปัญหาเรื่องเวลาและการใช้พื้นที่หน่วยความจำโดยใช้เทคนิคการเลื่อนหน้าต่างเข้ามาช่วยในการค้นหารูปแบบ เนื่องจากข้อมูลกระแสเป็นข้อมูลที่ไหลเข้ามาเรื่อยๆทำให้สามารถอ่านข้อมูลได้เพียงครั้งเดียวการอ่านข้อมูลแต่ละครั้งจึงมีความสำคัญในการค้นหารูปแบบเป็นอย่างมากนอกจากนี้ยังมีการค้นหารูปแบบที่ปรากฏขึ้นบ่อยและปรากฏอย่างสม่ำเสมอได้ถูกพัฒนาขึ้นอย่างต่อเนื่องในอีกหลายๆ แง่มุม อาทิเช่น การค้นหาเซตรายการที่ปรากฏบ่อยในข้อมูลกระแสจากหลายช่วงเวลา (Giannella, C., และคณะ, 2004) การค้นหาเซตรายการที่ปรากฏสม่ำเสมอในข้อมูลกระแส (Tanbeer, S. K., และคณะ, 2010) การใช้ไดนามิกบิตเวกเตอร์เพื่อค้นหาเซตรายการแบบปิดอย่างรวดเร็ว (Vo, B., และคณะ, 2012) การค้นหาเซตรายที่ปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากฐานข้อมูลรายการ (Sittuchaitaweekul, P., และ Amphawan, K., 2014) การใช้เทคนิคการเลื่อนหน้าต่างในการค้นหาเซตรายการที่ปรากฏบ่อยจากข้อมูลกระแส (Lee, G., และคณะ , 2014) การค้นหาเซตรายการที่ปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอในเซตรายการแบบปิด (Amphawan, K. และ Lenca, P., 2015)

2.3 คุณลักษณะของฐานข้อมูลรายการที่ใช้ในการทดลอง

ข้อมูลที่ใช้สำหรับการค้นหารูปแบบที่ปรากฏบ่อยอันดับแรกและปรากฏอย่างสม่ำเสมอ เป็นชุดข้อมูลที่ได้รับความน่าเชื่อถือ จึงทำให้นักวิจัยอื่นๆ สามารถทราบถึงประสิทธิภาพของการทำงานที่แท้จริงของวิธีการที่นำเสนอได้ โดยสามารถดาวน์โหลดข้อมูลได้จากเว็บไซต์ fimi[1] โดยข้อมูลที่ใช้แบ่งข้อมูลออกได้เป็น 2 ประเภท คือ 1) ฐานข้อมูลจริงทั้งหมด 6 ฐานข้อมูลรายการ ได้แก่ Accidents, Chess, Connect, Mushroom และ Pumsb ที่มีลักษณะของข้อมูลหนาแน่น และ Retail ที่มีลักษณะของข้อมูลเบาบาง 2) ฐานข้อมูลสังเคราะห์ทั้งหมด 2 ฐานข้อมูลรายการ ได้แก่ T10I4D100K และ T40I10D100K ที่มีลักษณะของข้อมูลเบาบาง แสดงในตารางที่ 2.1

[1] http://fimi.ua.ac.be/data/

ตารางที่ 2.1 คุณลักษณะของฐานข้อมูลรายการ

ฐานข้อมูลรายการ	จำนวนรายการ	จำนวนทรานแซกชัน	จำนวนความยาวเฉลี่ยทรานแซกชัน	ลักษณะข้อมูล	ประเภทข้อมูล
Accident	468	340,182	33.8	หนาแน่น	ข้อมูลจริง
Chess	75	3,196	37	หนาแน่น	ข้อมูลจริง
Connect	129	67,556	43	หนาแน่น	ข้อมูลจริง
Mushroom	119	8,124	23	หนาแน่น	ข้อมูลจริง
Pumsb	7,117	49,046	74	หนาแน่น	ข้อมูลจริง
Retail	16,470	88,162	10.3	บางเบา	ข้อมูลจริง
T10I4D100K	1,000	100,000	10	บางเบา	ข้อมูลสังเคราะห์
T40I10D100K	1,000	100,000	40	บางเบา	ข้อมูลสังเคราะห์

จากข้อมูลที่แสดงในตารางที่ 2.1 แสดงให้เห็นถึงจำนวนรายการ จำนวนทรานแซกชัน จำนวนความยาวเฉลี่ยทรานแซกชัน ลักษณะข้อมูล และประเภทข้อมูล โดยฐานข้อมูลรายการ Accident มีรายการทั้งหมด 468 รายการ จำนวนทรานแซกชัน 340,182 ทรานแซกชัน เป็นฐานข้อมูลที่มีลักษณะหนาแน่น และเป็นข้อมูลจริง ซึ่งฐานข้อมูลรายการที่ใช้รวบรวมข้อมูลการเกิดอุบัติเหตุทางจราจรแถบพื้นที่ทางตอนเหนือของประเทศเบลเยี่ยม ในปีคริสต์ศักราช 1991-2000 ซึ่งจัดเก็บโดยสถาบันสถิติแห่งชาติ (National Institute of Statistics, NIS)

ฐานข้อมูลรายการ Chess มีรายการทั้งหมด 75 รายการ จำนวนทรานแซกชัน 3,196 ทรานแซกชัน เป็นฐานข้อมูลที่มีลักษณะหนาแน่น และเป็นข้อมูลจริง และฐานข้อมูลรายการ Connect มีรายการทั้งหมด 129 รายการ จำนวนทรานแซกชัน 67,556 ทรานแซกชัน เป็นฐานข้อมูลที่มีลักษณะหนาแน่น และเป็นข้อมูลจริง เป็นฐานข้อมูลรายการที่จัดเก็บวิธีการเดินหมากในระหว่างการแข่งขันในแต่ละทรานแซกชัน ที่ถูกรวบรวมจาก UCI Machine Learning Repository

ฐานข้อมูลรายการ Mushroom มีรายการทั้งหมด 119 รายการ จำนวนทรานแซกชัน 8,124 ทรานแซกชัน เป็นฐานข้อมูลที่มีลักษณะหนาแน่น และเป็นข้อมูลจริง เป็นฐานข้อมูลรายการที่ใช้สำหรับจัดเก็บข้อมูลคุณลักษณะของสายพันธุ์ของเห็ดแต่ละชนิด

ฐานข้อมูลรายการ Pumsb มีรายการทั้งหมด 7,117 รายการ จำนวนทรานแซกชัน 49,046 ทรานแซกชัน เป็นฐานข้อมูลที่มีลักษณะหนาแน่น และเป็นข้อมูลจริง เป็นฐานข้อมูลรายการที่ใช้สำหรับจัดเก็บข้อมูลการสำรวจสำมโนประชากรและที่อยู่อาศัย

ฐานข้อมูลรายการ Ratail มีรายการทั้งหมด 16,470 รายการ จำนวนทรานแซกชัน 81,162 ทรานแซกชัน เป็นฐานข้อมูลที่มีลักษณะบางเบา และเป็นข้อมูลจริง เป็นฐานข้อมูลรายการที่ถูกจัดเก็บจากตะกร้าสินค้า ณ ร้านค้าในประเทศเบลเยี่ยมตั้งแต่เดือนธันวาคมปีคริสต์ศักราช 1999

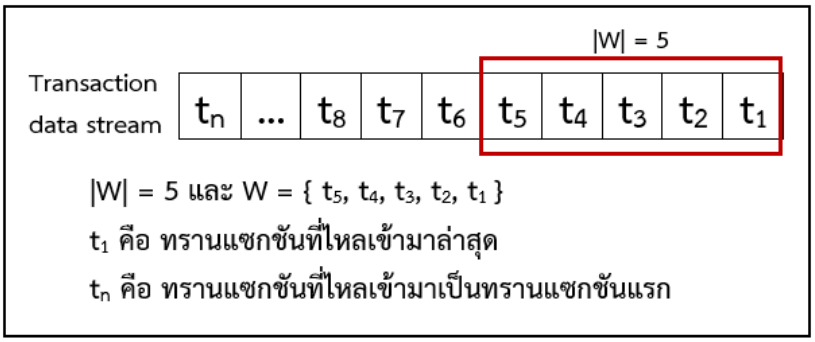
จนถึงเดือนพฤศจิกายนปีคริสต์ศักราช 2000 จึงเป็นข้อมูลที่มีทรานแซกชันมากแต่มีข้อมูลในแต่ละทรานแซกชันน้อย

ฐานข้อมูลรายการ T10I4D100K และฐานข้อมูลรายการ T40I4D100K มีรายการทั้งหมด 1,000 รายการ จำนวนทรานแซกชัน 100,000 ทรานแซกชัน เป็นฐานข้อมูลที่มีลักษณะบางเบา และเป็นข้อมูลสังเคราะห์ เป็นฐานข้อมูลรายการจากการจำลองธุรกรรมการค้าปลีกโดย IBM Generator ลักษณะของชุดข้อมูลคือการศึกษาตามความหนาแน่นในการจำแนนประเภทของชุดข้อมูล โดยชุดข้อมูลจะมีความหนาแน่นเมื่อสร้างชุดข้อมูลที่มีความยาวเป็นจำนวนมาก

บทที่ 3
วิธีดำเนินการวิจัย

จากบทก่อนหน้าได้มีหลายงานวิจัยที่นำเสนอการค้นหารูปแบบที่ปรากฏบ่อยและปรากฏอย่างสม่ำเสมอ ซึ่งการกำหนดค่าขีดแบ่งสนับสนุนเป็นเรื่องที่ทำได้ยากจึงได้มีนักวิจัยได้นำเสนอการค้นหารูปแบบที่ปรากฏบ่อยสุดเคอันดับแรกและปรากฏอย่างสม่ำเมอ โดยการกำหนดผลลัพธ์ที่ต้องการ (k) แทนการกำหนดค่าขีดแบ่งสนับสนุน และมีงานวิจัยหนึ่งที่สนใจเกี่ยวกับการค้นหารูปแบบจากฐานข้อมูลกระแส ซึ่งข้อมูลกระแสเป็นข้อมูลที่มีปริมาณมาก และเป็นข้อมูลที่มีการเพิ่มขึ้นอย่างต่อเนื่องในอัตราที่รวดเร็ว ทำให้การประมวลผลแบบปกติไม่สามารถทำงานได้ในหลายข้อจำกัด จึงจำเป็นต้องออกแบบขั้นตอนวิธีที่อ่านข้อมูลจากข้อมูลกระแสเพียงรอบเดียว และต้องออกแบบโครงสร้างข้อมูลสำหรับจัดเก็บข้อมูลสำหรับการประมวลผลที่ต้องการใช้หน่วยความจำที่น้อยและมีประสิทธิภาพ

ด้วยเหตุนี้งานวิทยานิพนธ์นี้ได้นำเสนอขั้นตอนวิธี TFRIM-DS (Top-k Frequent-Regular Itemsets Miner over Data Streams) สำหรับการค้นหารูปแบบปรากฏบ่อยสุดเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแสด้วยเทคนิคการเลื่อนหน้าต่าง รูปแบบนี้จะพิจารณาความสำคัญหรือความน่าสนใจของรูปแบบในเชิงความถี่และความสม่ำเสมอของการปรากฏ ประกอบกับข้อมูลกระแสเป็นข้อมูลที่มีอัตราการไหลของข้อมูลค่อนข้างมากทำให้การอ่านข้อมูลจึงถูกจำกัดให้อ่านได้เพียงครั้งเดียว และไม่สามารถจัดเก็บข้อมูลทั้งหมดไว้ในหน่วยความจำหรือดิสก์ได้ ดังนั้นจึงจำเป็นที่จะต้องกำหนดความสำคัญของการปรากฏขึ้นของรูปแบบหนึ่งภายใต้การประยุกต์ใช้เทคนิคการเลื่อนหน้าต่างที่ซึ่งจะกำหนดให้ข้อมูลที่ไหลเข้ามาใหม่สุดจะมีความสำคัญมากกว่าข้อมูลที่ไหลเข้ามาก่อนหน้า (กล่าวคือข้อมูลที่มีอายุมากจะมีความสำคัญน้อยเมื่อเทียบกับข้อมูลที่มีอายุน้อยจะมีความสำคัญมาก) โดยเทคนิคการเลื่อนหน้าต่างจะทำการกำหนดขอบเขตของการพิจารณาภายใต้ขนาดของหน้าต่าง Wที่ซึ่งจะทำการพิจารณาเฉพาะทรานแซกชันที่ไหลเข้ามาล่าสุดเป็นจำนวน w ทรานแซกชันเท่านั้น (ดังแสดงในภาพที่ 3.1)



ภาพที่ 3.1 ตัวอย่างการประยุกต์ใช้เทคนิคเลื่อนหน้าต่างกับข้อมูลกระแส

จากภาพที่ 3.1 เราจะสังเกตุได้ว่าเมื่อผู้ใช้ทำการกำหนดขอบเขตของการพิจารณา (ขนาดของหน้าต่าง) เป็น w เทคนิคการเลื่อนหน้าต่างจะทำการพิจารณาเฉพาะทรานแซกชันที่ไหลเข้ามาล่าสุด w ทรานแซกชัน ตัวอย่างเช่น ถ้ากำหนดให้ w มีค่าเท่ากับ 5 เทคนิคการเลื่อนหน้าต่างจะพิจารณา 5 ทรานแซกชันล่าสุดที่เพิ่งไหลเข้ามา (คือ ทรานแซกชันที่ 1 ถึง 5 จากภาพที่ 3.1) โดยทรานแซกชันที่มีอายุ (หมายเลขทรานแซกชัน) เกินกว่าทรานแซกชันที่ 5 จะถูกลบทิ้งจากการพิจารณา ซึ่งจากการดำเนินการดังกล่าว จะทำให้เราทราบถึงความสัมพันธ์ของสิ่งของหรือเหตุการณ์ที่ปรากฏขึ้นล่าสุดตามจำนวนทรานแซกชันที่ผู้ใช้ต้องการพิจารณาได้

3.1 นิยาม

ในงานวิจัยนี้ได้ประยุกต์นิยามพื้นฐานจากที่กล่าวไว้ในบทที่ 2 ในบางส่วน และจากที่กล่าวมาข้างต้นเราจะสามารถกำหนดนิยามเกี่ยวกับรูปแบบที่ปรากฏบ่อยสุดเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแสด้วยเทคนิคการเลื่อนหน้าต่าง ได้ดังนี้

นิยามที่ 3.1 *กำหนดให้* $I = \{i_1, i_2, i_3, \dots, i_n\}$ *เป็นเซตรายการ (items) ที่อาจหมายถึงสิ่งของหรือเหตุการณ์ที่ต้องการหาความสัมพันธ์ โดยเรียก* $i_j \in I$ *ว่า "รายการ"*

นิยามที่ 3.2 *กำหนดให้ เซต* $X = \{i_p, i_{p+1}, \dots, i_q\} \subseteq I$ *เรียกว่า เซตรายการ (set of items, an itemset หรือ a pattern) เมื่อ* X *ประกอบไปด้วยรายการทั้งสิ้น k รายการ จะเรียกว่า k-itemset, k-pattern*

นิยามที่ 3.3 *กำหนดให้* $DS = \{ds_1, ds_2, ds_3, \dots\}$ *เป็นฐานข้อมูลกระแสที่ซึ่งจะประกอบด้วยหลายเซตของทรานแซกชัน โดยแต่ละเซตของทรานแซกชัน* ds_j *จะไหลเข้ามา ณ ช่วงเวลาที่* j *และแต่ละ* $ds_j = \{t_1, t_2, t_3, \dots, t_{|ds_j|}\}$ *จะประกอบไปด้วยเซตของทรานแซกชันที่มีการไหลเข้ามาของแต่ละทรานแซกชันแบบเป็นลำดับ โดยในการได้รับซึ่งข้อมูลจากฐานข้อมูล เราจะสามารถได้รับข้อมูลทีละ* ds_j *ที่ซึ่งแต่ละ* ds_j *อาจมีจำนวนทรานแซกชันไม่เท่ากับ*

นิยามที่ 3.4 *กำหนดให้* w *คือหน้าต่างหรือขอบเขตของการพิจารณาทรานแซกชันจากฐานข้อมูลกระแส และกำหนดให้* $|w|$ *คือจำนวนทรานที่ถูกบรรจุล่าสุดในฐานข้อมูลกระแส*

ตัวอย่างที่ 3.1 *ถ้ากำหนดให้* $|w| = 100$ *แสดงว่าหน้าต่าง* w *จะทำการพิจารณา* 100 *ทรานแซกชันล่าสุดในฐานข้อมูลกระแส ถ้า ณ เวลา* t_1 *มี* ds_1 *ที่บรรจุด้วย* 70 *ทรานแซกชัน เราจะทำการค้นหาผลลัพธ์จาก* ds_1 *ที่มีทั้งสิ้น* 70 *ทรานแซกชัน กล่าวคือ* $w = ds_1(70) = \{t_1, t_2, t_3, \dots, t_{70}\}$ *ต่อมา ณ เวลา* t_2 *มี* ds_2 *ที่บรรจุด้วย* 80 *ทรานแซกชัน นี้จะทำให้หน้าต่าง* w *ทำการพิจารณาข้อมูลจากทั้ง* ds_1 *และ* ds_2 *ที่ซึ่งหน้าต่าง* w *จะประกอบด้วย* 20 *ทรานแซกชันล่าสุดจาก* ds_1 *และ* 80 *ทรานแซกชันจาก* ds_2 *(กล่าวคือ* $w = ds_1(20) + ds_2(80) = \{t_1, t_2, t_3, \dots, t_{20}\} \cup \{t_1, t_2, t_3, \dots, t_{80}\}$)

ต่อมา ณ เวลา t_3 มี ds_3 ที่บรรจุด้วย 50 ทรานแซกชัน นี้จะทำให้หน้าต่าง w ทำการพิจารณาข้อมูลจากทั้ง ds_2 และ ds_3 ที่ซึ่งหน้าต่าง w จะประกอบด้วย 50 ทรานแซกชันล่าสุดจาก ds_2 และ 50 ทรานแซกชันจาก ds_3 (กล่าวคือ $w = ds_2(50) + ds_3(50) = \{t_1, t_2, t_3, \dots, t_{50}\} \cup \{t_1, t_2, t_3, \dots, t_{50}\}$)

นิยามที่ 3.5 *กำหนดให้ $T_W^X = \{t_p, \dots, t_q\}$ เมื่อแต่ละ $t_p \in T_W^X$ เป็นสมาชิกของหน้าต่าง w คือ เซตของทรานแซกชันภายใต้หน้าต่าง w ที่มีเซตรายการ X ปรากฏ*

ดังนั้น ค่าสนับสนุนของเซตรายการ X ภายใต้หน้าต่าง w จะสามารถคำนวณได้จาก $s^X = |T_W^X|$ ซึ่งจะแสดงถึงจำนวน ทรานแซกชันในหน้าต่าง w ที่มีเซตรายการ X ปรากฏ อีกนัยหนึ่งค่าความสม่ำเสมอของเซตรายการ X ภายใต้หน้าต่าง w จะสามารถคำนวณได้จาก $r^X = max(fr^X, rtt_1^X, rtt_2^X, \dots, rtt_{|T_W^X|-1}^X, lr^X)$ เมื่อ

1. fr^X คือ ค่าความสม่ำเสมอสืบเนื่องจากการปรากฏขึ้นครั้งแรกของ X ภายใต้หน้าต่าง w ซึ่งแสดงถึงช่วงของทรานแซกชันจนกระทั่งการปรากฏขึ้นครั้งแรกของ X

2. แต่ละ rtt_u^X คือ ค่าความสม่ำเสมอสืบเนื่องจากการปรากฏขึ้นของ X ณ ทรานแซกชัน t_u และ t_v โดยที่ 1) การปรากฏขึ้น ณ ทรานแซกชัน t_u และ t_v จะเป็นการปรากฏที่ต่อเนื่องกัน โดยจะไม่มีการปรากฏขึ้นของ X ระหว่าง ทรานแซกชัน t_u และ t_v และ 2) ทรานแซกชัน t_u จะปรากฏในหน้าต่าง w ก่อนทรานแซกชัน t_v ที่ซึ่งจะแสดงถึงช่วงของทรานแซกชันระหว่างทรานแซกชัน t_u และ t_v ที่ซึ่ง X ไม่ปรากฏในหน้าต่าง w โดย rtt_p^X สามารถคำนวณได้จาก $rtt_u^X = v - u$

3. lr^X คือ ค่าความสม่ำเสมอสืบเนื่องจากการปรากฏขึ้นครั้งสุดท้ายของ X ภายใต้หน้าต่าง w ซึ่งแสดงถึงช่วงของทรานแซกชันที่ไม่มี X ปรากฏขึ้นจากการปรากฏครั้งสุดท้ายไปจนกระทั่งทรานแซกชันสุดท้ายของหน้าต่าง w โดย lr^X สามารถคำนวณได้จาก $lr^X = |w| - t_q$ เมื่อ t_q คือทรานแซกชันสุดท้ายภายใต้หน้าต่าง w ที่มี X ปรากฏ

ตัวอย่างที่ 3.2 *ถ้าเซตรายการ X ปรากฏขึ้นครั้งแรก ณ ทรานแซกชัน t_{10} จะทำให้ค่า fr^X มีค่าเท่ากับ 10*

ตัวอย่างที่ 3.3 *ถ้าเซตรายการ X ปรากฏในทรานแซกชัน t_{10} และปรากฏขึ้นอีกครั้งหนึ่งในทรานแซกชัน t_{30} โดยในระหว่างทรานแซกชัน t_{10} และ t_{30} จะไม่มีเซตรายการ X ปรากฏ ดังนั้น rtt_{10}^X จะสามารถคำนวณได้เป็น $rtt_{10}^X = 30 - 10 = 20$*

ตัวอย่างที่ 3.4 *ถ้ากำหนดให้หน้าต่าง w มีขอบเขตการพิจารณาเท่ากับ 100 ทรานแซกชัน และ t_{90} คือทรานแซกชันสุดท้ายภายใต้หน้าต่าง w ที่มี X ปรากฏ ดังนั้น ค่า lr^X จะสามารถคำนวณได้จาก $lr^X = 100 - 90 = 10$*

จากตัวอย่างที่ 3.2, 3.3 และ 3.4 ค่าความสม่ำเสมอ r^X จะสามารถคำนวณได้เป็น $r^X = max(10,20,10) = 20$ ที่ซึ่งจะสามารถบ่งบอกได้ว่าเซตรายการ X จะปรากฏขึ้นอย่างน้อย 1 ครั้งในทุกๆ 20 ทรานแซกชันภายใต้หน้าต่าง w

นิยามที่ 3.6 *กำหนดให้ k คือจำนวนรูปแบบที่ผู้ใช้ต้องการค้นหา, $|w|$ คือจำนวนทรานแซกชันภายใต้ขอบเขตการการพิจารณา*

นิยามที่ 3.7 *กำหนดให้ σ_r คือ ค่าขีดแบ่งความสม่ำเสมอ (Regularity threshold)*

นิยามที่ 3.8 *เซตรายการ X ใดๆ จะเป็นรูปแบบปรากฏบ่อยสุดเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแสก็ต่อเมื่อ*

1. การปรากฏขึ้นของ X มีความสม่ำเสมอ (กล่าวคือ ค่าความสม่ำเสมอ r^X ของ X จะต้องมีค่าน้อยกว่าหรือเท่ากับค่าขีดแบ่งความสม่ำเสมอ σ_r)

2. จะต้องไม่มีเซตรายการ Y มากกว่า $k-1$ รายการ ที่ซึ่งเซตรายการเหล่านั้นปรากฏอย่างสม่ำเสมอและมีค่าสนับสนุนมากกว่า X

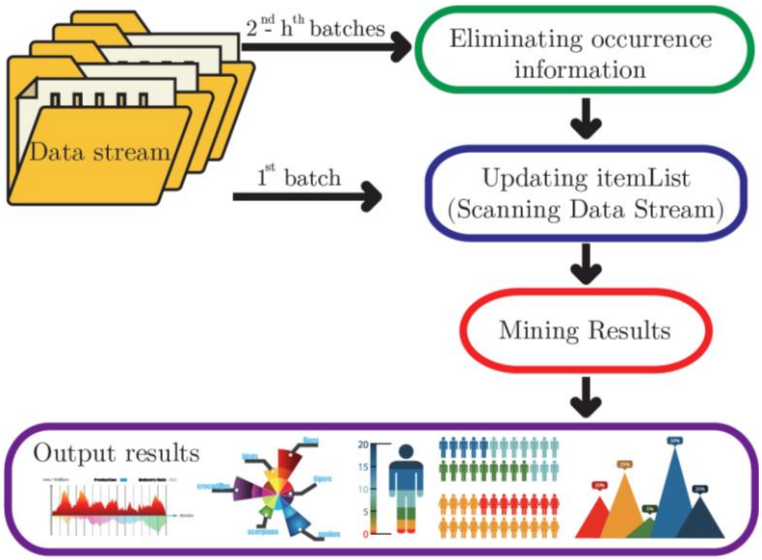
ปัญหาการค้นหารูปแบบปรากฏบ่อยสุดเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแสจะเป็นการค้นหา k เซตรายการที่ปรากฏอย่างสม่ำเสมอและมีค่าสนับสนุนสูงสุดจากหน้าต่างขณะนั้นๆ ตามที่ผู้ใช้กำหนด

3.2 วิธีดำเนินงานวิจัย

จากนิยามข้างต้น ผู้วิจัยได้นำเสนอขั้นตอนวิธี TFRIM-DS (Top-k Frequent-Regular Itemsets Miner over Data Streams) สำหรับการค้นหารูปแบบปรากฏบ่อยสุดเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแสด้วยเทคนิคการเลื่อนหน้าต่าง โดยขั้นตอนวิธี TFRIM-DS จะประกอบด้วย 3 ขั้นตอนหลัก (ดังแสดงในภาพที่ 3.2) คือ

1. การสร้างลิสต์ของเซตรายการและการจัดเก็บเซตรายการที่ปรากฏขึ้นในฐานข้อมูลกระแส
2. การค้นหาผลลัพธ์ (รูปแบบปรากฏบ่อยสุดเคอันดับแรกและปรากฏอย่างสม่ำเสมอ) จาก itemList ที่สร้างขึ้นและอัพเดทในขั้นตอนที่ 1
3. การจัดการ itemList เพื่อลบข้อมูลที่ไม่มีความสำคัญออกจากการพิจารณา (ลบข้อมูลสืบเนื่องจากการเลื่อนของหน้าต่าง) ตามลำดับ

โดยจากขั้นตอนทั้ง 3 เมื่อมีเซตของทรานแซกชันไหลเข้ามาใหม่ ขั้นตอนวิธี TFRIM-DS จะดำเนินการซ้ำในขั้นตอนที่ 3, 1 และ 2 ตามลำดับ โดยระหว่างการค้นหารูปแบบ TFRIM-DS ได้ประยุกต์ใช้บิตเวกเตอร์ (Bit-vector) เพื่อใช้ในการจัดเก็บข้อมูลการปรากฏขึ้นของเซตรายการหนึ่งๆ



ภาพที่ 3.2 ขั้นตอนวิธี TFRIM-DS

3.2.1. บิตเวกเตอร์ และเทคนิคการแทนที่บิตในบิตเวกเตอร์ (Bit-vector and the technique for reusing on bit of bit-vector)

ณ ปัจจุบัน บิตเวกเตอร์ถูกประยุกต์ใช้ในการค้นหารูปแบบที่ปรากฏบ่อยจากฐานข้อมูลอย่างแพร่หลาย (Dong & Hun, 2007; Song et al., 2008) โดยบิตเวกเตอร์หนึ่งๆ จะถูกใช้สำหรับจัดเก็บข้อมูลการปรากฏขึ้นของเซตรายการหนึ่งๆ ที่ซึ่งบิตเวกเตอร์จะประกอบด้วยลิสต์ของบิตที่ต่อเนื่องกัน แต่ละเซตรายการจำนวนของบิตเวกเตอร์จะเท่ากับ w บิต (เท่ากับจำนวนของขอบเขตการพิจารณา, w) บิตเวอร์เตอร์ที่ใช้จัดเก็บรายการ x จะได้เป็น $BV^x = \{b_1, b_2, b_3, \dots, b_w\}$ แต่ละ $b_j \in BV^x$ ที่แสดงให้เห็นถึงจำนวนทรานแซกชันการปรากฏขึ้นในหน้าต่างการพิจารณาของรายการ x ที่ j^{th} โดยบิตในลำดับที่ j^{th} จะใช้สำหรับจัดเก็บข้อมูลการปรากฏขึ้นของเซตรายการนั้นๆ ที่จะปรากฏหรือไม่ปรากฏในทรานแซกชันที่ j^{th} ถ้าเซตรายการนั้นๆปรากฏในทรานแซกชัน j^{th} จะทำให้บิตในลำดับที่ j^{th} จะมีค่าเป็น 1 แต่ในทางกลับกัน ถ้าเซตรายการนั้นๆ ไม่ปรากฏในทรานแซกชัน j^{th} จะทำให้บิตในลำดับที่ j^{th} จะมีค่าเป็น 0

ตัวอย่างที่ 3.5 *ฐานข้อมูลรายการ ds_1 ที่ประกอบไปด้วย 20 ทรานแซกชัน (ดังแสดงในภาพที่ 3.3) ขนาดของขอบเขตการพิจารณาเท่ากับ 16 ช่วงของพิจารณารายการในฐานข้อมูลรายการ ds_1 จะเริ่มทำการพิจารณาจากทรานแซกชันที่ $t_5, t_6, \dots, t_{20}$ ซึ่งเป็นทรานแซกชันที่ปรากฏล่าสุดในฐานข้อมูลรายการ เมื่อพิจารณาเซตรายการ "a" ที่ปรากฏในทรานแซกชันที่*

$t_5, t_6, t_7, t_9, t_{12}, t_{13}, t_{15}$ และ t_{16} *ตามลำดับ บิตเวกเตอร์ของเซตรายการ* "a" *คือ* $\{11101001, 10110000\}$ *ซึ่งจะถูกจัดเก็บเป็นเลขฐานสิบคือ* $BV^a = \{233, 176\}$

ds_1			
tid	set of items	tid	set of items
1	a, e	11	e
2	d, e, f	12	a, b, d
3	a, d	13	a, b, c, d
4	c, e, f	14	e, f
5	a, b, d, e	15	a, b, d, f
6	a, b, d	16	a, b, c, d
7	a, b, c	17	e
8	e	18	d, f
9	a, b, d, f	19	c
10	b, c, d	20	b, c, d, e

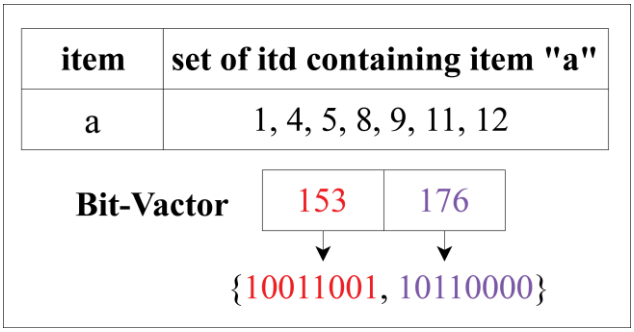
ds_2	
tid	set of items
1	a, c, e, f
2	d, e, f
3	c, f
4	a, c, f
5	a, c, e, f

ภาพที่ 3.3 ฐานข้อมูลกระแสที่ประกอบไปด้วย 2 ชุดข้อมูล $DS = \{ds_1, ds_2\}$

เมื่อมีข้อมูลกระแสเข้ามาใหม่ ds_d บิตเวกเตอร์ในชุดแรกจะถูกลบออกบางส่วน และถูกแทนที่ตัวบิตเวกเตอร์ที่เข้ามาใหม่จากฐานข้อมูลกระแสที่ ds_d

ตัวอย่างที่ 3.6 *ข้อมูลกระแสที่เข้ามาใหม่* ds_2 *ที่ประกอบไปด้วย 5 ทรานแซกชัน (ดังแสดงในภาพที่ 3.3) ขอบเขตการพิจารณาของ* ds_1 *จะเริ่มจากทรานแซกชันที่* $t_{10}, t_{11}, \dots, t_{20}$ *และ* ds_2 *จะเริ่มจากทรานแซกชันที่* $t_1, t_2, \dots, t_5$ *ดังนั้น 5 บิตแรกของเซตรายการ* "a" *คือ* "11101" *ที่เกิดขึ้นในทรานแซกชันที่* $t_5 - t_9$ *จะถูกลบออกจาก* BV^a *และแทนที่ด้วย* "10011" *ที่เกิดขึ้นในทรานแซกชันที่* $t_1 - t_5$ *จาก* ds_2 *ดังนั้น* BV^a *จะถูกอัพเดทให้เป็น* $\{153,176\}$ *(ในรูปแบบเลขฐานสองจะได้เป็น* $\{10011001, 10110000\}$*)*

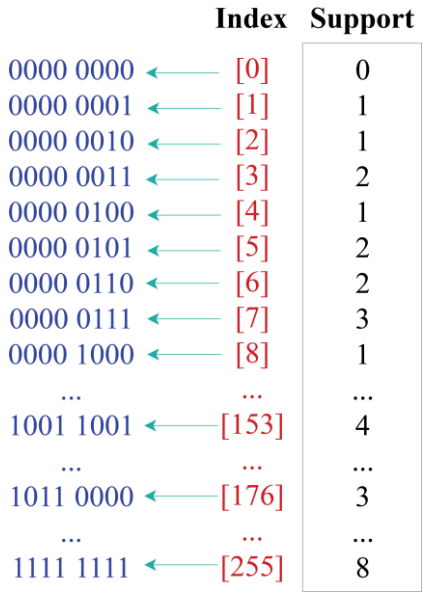
จากการใช้บิตเวกเตอร์ ทำให้สามารถค้นหาจำนวนครั้ง (ความถี่) ของการปรากฏได้จากการนับจำนวนบิตที่อยู่ในชุดของบิตเวกเตอร์ที่มีค่าเท่ากับ 1 โดยจำนวนบิตที่ใช้ในการจัดเก็บบิตเวกเตอร์ของรายการ/เซตรายการหนึ่งๆ จะมีจำนวนเท่ากับจำนวนทรานแซกชันในฐานข้อมูล



ภาพที่ 3.4 ตัวอย่างของบิตเวกเตอร์ ของเซตรายการ “a”

จากการใช้บิตเวกเตอร์ในการจัดเก็บข้อมูลการปรากฏขึ้นของรายการ/เซตรายการหนึ่งๆ Vo, B. และคณะ (Vo, B., และคณะ, 2012) ได้เสนอการนับค่าสนับสนุนจากบิตเวกเตอร์ที่มีประสิทธิภาพ ที่ซึ่งจะทำการพิจารณาข้อมูลการปรากฏที่ถูกบรรจุอยู่ในแต่ละไบต์ของบิตเวกเตอร์ โดยนำแต่ไบต์ในบิตเวกเตอร์มาทำการ look up ใน ตาราง Look-Up Table (ซึ่งแสดงในภาพที่ 3.5) แล้วเก็บรวบรวมเป็นค่าสนับสนุนของรายการนั้นๆ

ตัวอย่างที่ 3.6 *เซตรายการ ‘a’ ปรากฏในฐานข้อมูลที่ประกอบด้วย 16 ทรานแซกชันดังภาพที่ 3.4 โดยปรากฏในทรานแซกชันที่* $t_1, t_4, t_5, t_8, t_9, t_{11}$ *และ* t_{12} *เมื่อทำการคำนวณค่าสนับสนุนจะได้เป็น* $S^a = Look - up(153) + Look - up(176) = 4 + 3 = 7$ *จะเห็นได้ว่าค่าสนับสนุนของเซตรายการ ‘a’ เท่ากับ 7*



ภาพที่ 3.5 ตาราง Look-up Table

3.2.2 ขั้นตอนวิธี TFRIM-DS

3.2.2.1 การสร้างลิสต์ของเซตรายการและการจัดเก็บเซตรายการที่ปรากฏขึ้นในฐานข้อมูลกระแส

ในขั้นตอนแรก Updating itemList จะทำการสร้างลิสต์ (เรียกว่า itemList) เพื่อใช้ในการจัดเก็บรายการและข้อมูลที่สำคัญ โดยแต่ละสมาชิกของลิสต์จะใช้เก็บข้อมูลสำหรับรายการหนึ่งๆ ที่ ซึ่งจะประกอบด้วย

1. รายการ (i)
2. บิตเวกเตอร์ของรายการ i (BV^i)
3. ค่าสนับสนุนของรายการ i (S^i)
4. ค่าความสม่ำเสมอของรายการ i (r^i)
5. หมายเลขของทรานแซกชันสุดท้ายที่รายการ i ปรากฏ (lo^i)

จากนั้นจะทำการตรวจสอบจำนวนทรานแซกชันในฐานข้อมูลกระแสว่ามีค่าน้อยกว่าหรือเท่ากับขนาดของหน้าต่าง (ขอบเขตของการพิจารณาทรานแซกชันจากฐานข้อมูลกระแส, w) โดยแบ่งเงื่อนไขการพิจารณาออกเป็น 2 กรณีดังนี้

1. ถ้าจำนวนทรานแซกชันทั้งหมดในฐานข้อมูลกระแส ($|DS|$) น้อยกว่าหรือเท่ากับขนาดของหน้าต่าง (ขอบเขตของการพิจารณาทรานแซกชันจากฐานข้อมูลกระแส, w) ขั้นตอนวิธี Updating itemList จะทำการกำหนดขอบเขตการพิจารณาทรานแซกชันจากฐานข้อมูลกระแส โดยขอบเขตที่กำหนดจะเริ่มที่ทรานแซกชันแรก

2. ถ้าจำนวนทรานแซกชันทั้งหมดในฐานข้อมูลกระแส ($|DS|$) มากกว่าขนาดของหน้าต่าง w ขั้นตอนวิธี Updating itemList จะทำการกำหนดขอบเขตการพิจารณาทรานแซกชันจากฐานข้อมูลกระแสเท่ากับทรานแซกชันในลำดับที่ w โดยนับจากข้างท้าย (สามารถคำนวณได้จาก $|DS_1| - w + 1$)

เมื่อทำการพิจารณาขอบเขตของการพิจารณาทรานแซกชันเสร็จสิ้น TFRIM-DS จะทำการอ่านข้อมูลจากฐานข้อมูลกระแสทีละทรานแซกชัน (โดยอ่านแบบเรียงลำดับ) โดยเมื่อทำการอ่านทรานเซกชัน t_p จากนั้นจะทำการพิจารณาแต่ละรายการ i_j ที่ปรากฏในทรานแซกชัน t_p แล้วทำการอัพเดทข้อมูลใน itemList ดังนี้

1. ทำการอัพเดท BV^i ด้วย หมายเลขทรานแซกชัน p
2. ทำการเพิ่มค่าสนับสนุนของรายการ s_i^j ขึ้น 1 ค่า
3. ทำการอัพเดทค่าความสม่ำเสมอของรายการ r_i^j ด้วยการตรวจสอบระยะห่างของการ

ปรากฏขึ้นของรายการ i_j จากครั้งล่าสุด (สามารถคำนวณได้จาก $r^i = \max(r^i, \mathrm{q}^i - lo^i)$)

4. ทำการจัดเก็บหมายเลขทรานแซกชันล่าสุดที่มีรายการ i_j ปรากฏ ($lo^i = \mathrm{q}^i$)

เมื่อทำการพิจารณาข้อมูลในแต่ละทรานแซกชันจากฐานข้อมูลกระแสเสร็จสิ้น TFRIM-DS จะทำการคำนวณหาค่าความสม่ำเสมอของทุกรายการ i_j ใน itemList (สำหรับรายการ i_j หนึ่งๆ สามารถอัพเดทค่าความสม่ำเสมอโดย $r^i = \max(r_j^i, |DS| - lo_j^i)$) ถ้าค่าความสม่ำเสมอของรายการ i มีค่ามากกว่าค่าขีดแบ่งความสม่ำเสมอที่ผู้ใช้กำหนด รายการนั้นจะถูกกำหนดให้เป็น “รายการที่ปรากฏไม่สม่ำเสมอ (irregular items)” จากนั้นในขั้นตอนสุดท้าย จะทำการเรียงลำดับรายการใน itemList ตามค่าสนับสนุนโดยเรียงลำดับจากมากไปหาน้อย (ดังแสดงในภาพที่ 3.6)

Algorithm 1 *Updating-itemList*

Input: $ds_d = \{t_1, t_2, \ldots, t_{|ds_d|}\}$, σ_r, w, k
Output: A list named *itemList* containing all of single items with their occurrence information in the current window

1: create *itemList*
2: create an entry for each item $i_j \in I$ in the *itemList*
3: **if** $|ds_d| \leq w$ **then**
4: $p = 1$
5: **else**
6: $p = |ds_d| - w + 1$
7: **for** each transaction $t_q \in ds_d$ (start from t_p to $t_{|ds_d|}$) **do**
8: **for** each item $i_j \in t_q$ **do**
9: $s^{i_j} \leftarrow s^{i_j} + 1$
10: $r^{i_j} \leftarrow max(r^{i_j}, q - lo^{i_j})$
11: $BVR^{i_j} \leftarrow BVR^{i_j} \cup q$
12: $lo^{i_j} \leftarrow q$
13: **for** each item $i_j \in$ *itemList* **do**
14: $r^{i_j} \leftarrow max(r^{i_j}, |ds_d| - lo^{i_j})$
15: **if** $r^{i_j} > \sigma_r$ **then**
16: mark i_j as an irregular item
17: sort *itemList* by support descending order

ภาพที่ 3.6 ขั้นตอนวิธี Updating itemList

3.2.2.2. การค้นหารูปแบบที่ปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอ

ขั้นตอนวิธี Mining results จะทำการค้นหารูปแบบที่มีจำนวนรายการมากกว่า 1 รายการ ที่ซึ่งปรากฏร่วมกันบ่อยและสม่ำเสมอ โดยการทำงานจะเริ่มจากการสร้างลิสต์ ที่เรียกว่า top-k list จาก itemList ที่สร้างขึ้นจากขั้นตอนก่อนหน้า โดยถ้า itemList มีรายการที่ถูกระบุว่าเป็นรายการที่ปรากฏสม่ำเสมอ (regular item) เป็นจำนวนมากกว่าหรือเท่ากับ k ขั้นตอนวิธี Mining results จะ

ทำการเลือกรายการที่ปรากฏสม่ำเสมอ ที่มีค่าสนับสนุนสูงสุด k ลำดับแรกมาเก็บไว้เป็นสมาชิกของ top-k list โดยทำการจัดเก็บแบบเรียงลำดับจากค่าสนับสนุนจากมากไปน้อย จากนั้นจะทำการกำหนดให้ค่าสนับสนุน s^k ให้มีค่าเท่ากับค่าสนับสนุนของรายการในลำดับที่ k เพื่อใช้สำหรับลดทอนปริภูมิสถานะ (หมายเหตุ ถ้า itemList มีจำนวนรายการที่ถูกระบุว่าเป็นรายการที่ปรากฏสม่ำเสมอน้อยกว่า k รายการ ขั้นตอนวิธี Mining results จะทำการจัดเก็บเซตรายการดังกล่าวทั้งหมดไว้ในเป็นสมาชิกของ top-k list โดยทำการจัดเก็บแบบเรียงลำดับจากค่าสนับสนุนจากมากไปน้อยเช่นกัน)

ในการค้นหารูปแบบที่ปรากฏร่วมกันบ่อยและสม่ำเสมอจะทำการพิจารณาแต่ละเซตรายการ X ใน top-k list จากนั้นทำการพิจารณาแต่ละรายการ Y ใน top-k list (ที่ซึ่งเซตรายการ $X \neq Y$) โดยที่ ถ้าเซตรายการ X และ Y มี prefix items ที่เหมือนกัน (กล่าวคือ X และ Y มีรายการเหมือนกัน แต่แตกต่างกันเฉพาะรายการสุดท้ายเท่านั้น) ขั้นตอนวิธี Mining results จะทำการรวมเซตรายการ X และ Y เข้าด้วยกันเพื่อสร้างเป็นเซตรายการ $Z = X \cup Y$ จากนั้นจะทำการอินเทอร์เซกชันเซตของทรานแซกชันที่มี X และ เซตของทรานแซกชันที่มี Y ปรากฏ ($BV^X \cap BV^Y$) เพื่อค้นหาเซตของทรานแซกที่มี X และ Y ปรากฏร่วมกัน และทำการเก็บไว้ใน BV^Z ที่ซึ่งจะทำให้สามารถคำนวณค่าสนับสนุนและค่าความสม่ำเสมอของรายการ Z ได้ โดยหลังจากทำการอินเทอร์เซกชันแล้ว ถ้าค่าความสม่ำเสมอของเซตรายการ Z มีค่าน้อยกว่าหรือเท่ากับค่าขีดแบ่งความสม่ำเสมอ และค่าสนับสนุนของเซตรายการ Z มีค่ามากกว่าค่าสนับสนุนของเซตรายการลำดับที่ k ใน top-k list ขั้นตอนวิธี Mining-results จะทำการกำหนดให้เซตรายการในลำดับที่ k ไม่เป็นผลลัพธ์และลบเซตรายการในลำดับที่ k ออกจาก top-k list จากนั้นจะทำการสร้างการจัดเก็บข้อมูลของเซตรายการ Z และเพิ่มรายการ Z เข้าไปใน top-k List (เพิ่มโดยเรียงลำดับตามค่าสนับสนุนจากมากไปน้อย) (ดังแสดงในภาพที่ 3.7)

หลังจากทำการพิจารณาทุกคู่ของเซตรายการใน top-k list แล้ว เราจะได้เซตรายการ k รายการที่ปรากฏบ่อยและสม่ำเสมอถูกบรรจุอยู่ใน top-k list ที่ซึ่งสามารถส่งต่อให้กับนักวิเคราะห์ข้อมูลเพื่อทำการสร้างกราฟหรือทำการวิเคราะห์เพิ่มเติมต่างๆ ได้

Algorithm 2 *Mining-results*

Input: *itemList*, k, σ_r
Output: *top-k list*

1: *top-k list* $= i_1, i_2, \ldots, i_k$ where $i_1, i_2, \ldots, i_k \in$ *itemList*
2: $s^k \leftarrow s^K$ where K is the k^{th} itemset in the *top-k list*
3: **for** each entry of $X = \{i_a, \ldots, i_b, i_c\}$ in *top-k list* **do**
4: **for** each entry of $Y = \{i_d, \ldots, i_e, i_f\}$ in *top-k list* **do**
5: **if** $i_a, \ldots, i_b = i_d, \ldots, i_e$ **then**
6: $Z \leftarrow X \cup Y$
7: $BV^Z \leftarrow BV^X \cap BV^Y$
8: $s^Z \leftarrow support(BV^Z)$ and $r^Z \leftarrow regularity(BV^Z)$
9: **if** $r^Z \leq \sigma_r$ and $s^Z \geq s^k$ **then**
10: *top-k list* $\leftarrow$ *top-k list* $- K$
11: *top-k list* $\leftarrow$ *top-k list* $+ Z$
12: $s^k \leftarrow s^K$ where K is the current k^{th} itemset in the *top-k list*

ภาพที่ 3.7 ขั้นตอนวิธี Mining results

3.2.2.3 การจัดการ itemList เพื่อลบข้อมูลที่ไม่มีความสำคัญออกจากการพิจารณา

เมื่อข้อมูลกระแสมีการอัพเดท (เรียกว่า ข้อมูลกระแสใหม่ สามารถเขียนแทนด้วย ds_d ซึ่งจะหมายถึงข้อมูลกระแสที่มีการอัพเดทในครั้งที่ d) ขั้นตอนวิธี Eliminating occurrence information จะทำการลบข้อมูลเซตรายการที่ถูกจัดเก็บใน top-k list ออก และทำการยกเลิกการระบุรายการที่ถูกระบุว่า “เซตรายการที่ปรากฏไม่สม่ำเสมอ” ออก เนื่องจากเซตรายการที่เป็นเซตรายการที่ปรากฏไม่สม่ำเสมออาจเป็นเซตรายการที่ปรากฏสม่ำเสมอเมื่อมีการเลื่อนหน้าต่างการพิจารณา เมื่อมีการเพิ่มขึ้นของทรานแซกชันที่ต้องทำการพิจารณา ที่ซึ่งจะทำให้หน้าต่างของการพิจารณาทรานแซกชันมีการเปลี่ยนแปลง โดยการพิจารณาข้อมูลกระแสที่ถูกอัพเดทจะต้องลบทรานแซกชันที่มีถูกพิจารณาลำดับแรกๆ ออกจากการพิจารณา จากนั้นทำการเพิ่มทรานแซกชันที่มีการอัพเดทไปยังหน้าต่างของการพิจารณาทรานแซกชัน

โดยการอัพเดทของข้อมูลกระแสครั้งหนึ่งๆ เราจะทราบถึงจำนวนทรานแซกชัน $|ds_d|$ ที่ต้องทำการพิจารณาเพิ่มขึ้น ที่ซึ่งจะสามารถแบ่งการพิจารณาออกเป็น 2 กรณี ดังนี้

1. ถ้าจำนวนทรานแซกชันในฐานข้อมูลกระแสใหม่ ($|ds_d|$) มีจำนวนมากกว่าหรือเท่ากับขนาดของหน้าต่าง w ขั้นตอนวิธี Eliminating occurrence information จะทำการรีเซตบิตเวกเตอร์ใน itemList ทั้งหมดให้มีค่าเป็น 0

2. ถ้าจำนวนทรานแซกชันทั้งหมดในฐานข้อมูลกระแส ds_d ($|ds_d|$) มีจำนวนน้อยกว่าขนาดของหน้าต่าง w ขั้นตอนวิธี Eliminating occurrence information จะทำการรีเซต ds_d แรกของบิตเวกเตอร์ใน itemList ให้มีค่าเป็น 0 และข้อมูลการปรากฏขึ้นของทรานแซกชันในส่วนที่เหลืออยู่

ใน itemList จะถูกคำนวณหาค่าสนับสนุนและค่าความสม่ำเสมอทั้งหมด จากนั้นทำการอัพเดทข้อมูลค่าสนับสนุนและค่าสม่ำเสมอของแต่ละรายการลงใน itemList (ดังแสดงในภาพที่ 3.8)

หลังจากลบข้อมูลที่ไม่มีความสำคัญออกจาก itemList เสร็จสิ้นแล้วในขั้นตอนต่อไปจะดำเนินการซ้ำในขั้นตอนที่ 1 และ 2 ตามลำดับ

Algorithm 3 *Eliminating-occurrence-information*

Input: $ds_d = \{t_1, t_2, \ldots, t_{|ds_d|}\}$, σ_r, w, k, *top-k list*, *itemList*
Output: *itemList*, *top-k list*

1: *top-k list* $\leftarrow \emptyset$
2: **for** each irregular item $i_j \in$ *itemList* **do**
3: unmark i_j
4: **if** $|ds_d| \geq w$ **then**
5: **for** each item $i_j \in$ *itemList* **do**
6: reset all bits in BV^{i_j} to be 0
7: **else**
8: **for** each item $i_j \in$ *itemList* **do**
9: reset the first $|ds_d|$ bits in BV^{i_j} to be 0
10: $s^{i_j} \leftarrow$ *support*(BV^{i_j}) and $r^{i_j} \leftarrow$ *regularity*(BV^{i_j})
11: **if** $r^{i_j} > \sigma_r$ **then**
12: mark i_j as an irregular item

ภาพที่ 3.8 ขั้นตอนวิธี Eliminating occurrence information

ตัวอย่างที่ 3.7 *กำหนดให้ข้อมูลกระแส* $DS = \{ds_1, ds_2\}$ *ของฐานข้อมูลรายการ ณ ช่วงเวลาที่* t_1 *และช่วงเวลาที่* t_2 *ดังแสดงในภาพที่ 3.3, กำหนดให้ขอบเขตการพิจารณา* $w = 16$, *ค่าขีดแบ่งความสม่ำเสมอ* $\sigma_r = 3$ *และจำนวนรูปแบบที่ต้องการค้นหา* $k = 6$ *ขั้นตอนวิธี TFRIM-DS จะทำการสร้าง itemList ที่มีเซตรายการทั้งหมด 6 รายการที่ประกอบไปด้วยรายการ* $\{a, b, c, d, e\}$ *และ* $\{f\}$ *จากนั้นทำการพิจารณาข้อมูลกระแส* ds_1 *ที่มีขนาดเท่ากับ 20 ทรานแซกชัน ซึ่งมีขนาดใหญ่กว่าขอบเขตการพิจารณาที่กำหนดไว้ ดังนั้นการพิจารณารายการจาก* ds_1 *ซึ่งสามารถคำนวณได้จาก* $|ds_1| - W + 1 = 20 - 16 + 1 = 5$ *ดังนั้นการพิจาณาทรานแซกชันใน* ds_1 *จะเริ่มจากทรานแซกชันที่* t_5 *ไปจนถึงทรานแซกชันที่* t_{20} *ซึ่งจะพิจารณาทรานแซกชันทั้งหมดใน* ds_1 *ทั้งหมด 16 ทรานแซกชันจาก* ds_1

จากนั้นจะเริ่มพิจารณาแต่ละทรานแซกชันโดยเริ่มจากทรานแซกชันที่ t_5 *ที่ประกอบไปด้วยรายการ* $\{a, b, d, e\}$ *แล้วทำการจัดเก็บข้อมูลลงใน itemList โดยทำการอัพเดทค่าสนับสนุนของเซตรายการ* s^a, s^b, s^d *และ* s^e *ให้มีค่าเท่ากับ 1, ค่าสนับสนุนของเซตรายการ* r^a, r^b, r^d *และ* r^e *ให้มีค่าเท่ากับ 1, หมายเลขทรานแซกชันสุดท้ายที่เซตรายการปรากฏ* lo^a, lo^b, lo^d *และ* lo^e *ให้มีค่าเท่ากับ 1 (เซตรายการที่ปรากฏจะอยู่ในตำแหน่งทรานแซกชันแรกของหน้าต่างพิจารณา) และบิตเวกเตอร์ของเซตรายการ* BV^a, BV^b, BV^d *และ* BV^e *ให้มีค่าเท่ากับ* $\{128, 0\}$ *ดังแสดงในภาพที่ 3.9*

i	s	r	ls	BV
a	1	1	1	{128, 0}
b	1	1	1	{128, 0}
c	-	-	-	{0, 0}
d	1	1	1	{128, 0}
e	1	1	1	{128, 0}
f	-	-	-	{0, 0}

{10000000 00000000}

{00000000 00000000}

ภาพที่ 3.9 itemList หลังจากอ่านข้อมูลทรานแซกชันที่ t_5 จาก ds_1

จากนั้นทำการพิจารณาเซตรายการที่ทรานแซกชันที่ $t_6, \ldots, t_{20}$ จาก ds_1 และทำการอัพเดทข้อมูลลงใน itemList ตามขั้นตอนที่กล่าวขั้นต้นจนครบทุกทรานแซกชัน เมื่อจัดเก็บข้อมูลครบทรานแซกชันแล้ว จากภาพที่ 3.10 จะเห็นได้ว่ารายการ "f" ถูกระบุให้เป็น "เซตรายการที่ปรากฏไม่สม่ำเสมอ" เนื่องจากมีค่าสนับสนุน r^f เท่ากับ 6 ซึ่งมีค่ามากกว่าค่าขีดแบ่งสนับสนุน σ_r ที่มีค่าเท่ากับ 3 ในขั้นตอนสุดท้ายของขั้นตอน Updating itemList จะทำการเรียงลำดับรายการที่มีค่าสนับสนุนจากมากไปน้อย (ดังแสดงในภาพที่ 3.10)

i	s	r	ls	BV
b	11	2	16	{237, 181}
d	10	3	16	{205, 181}
a	9	3	14	{233, 180}
c	8	3	16	{37, 147}
e	6	3	16	{144, 73}
f	5	6	14	{9, 100}

{11101101 10110101}

{11001101 10110101}

{11101001 10110100}

{00100101 10010111}

{10010010 01001001}

{00001001 01100100}

Mark

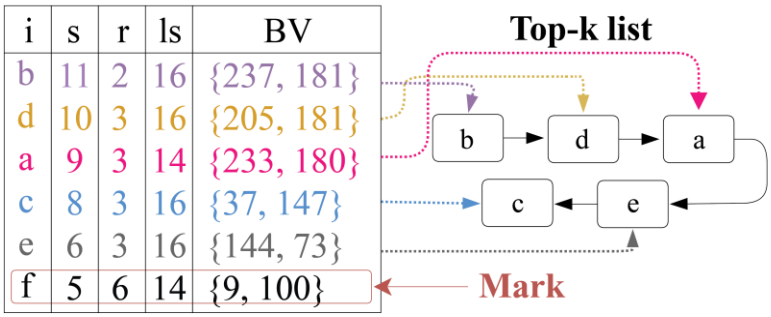
ภาพที่ 3.10 itemList หลังจากอ่านข้อมูลทรานแซกชันที่ $t_5 - t_{20}$ จาก ds_1

ต่อมาในขั้นตอนวิธี Mining results รายการ "b", "d", "a", "c" และ "e" ซึ่งเป็น 5 รายการแรกที่เป็นรายการที่ปรากฏบ่อยใน itemList จะถูกเพิ่มเข้าไปใน top-k list และเรียงลำดับจากค่าสนับสนุนจากมากไปน้อย (ดังแสดงในภาพที่ 3.11) จากนั้นทำการพิจารณาเซตรายการที่มีค่าสนับสนุนสูงสุดสองรายการแรกจาก top-k list คือ เซตรายการ "b" และเซตรายการ "d" เมื่อนำสองเซตรายการมาทำการจับคู่กับจะได้เป็นเซตรายการ "bd" จากนั้นบิตเวกเตอร์ BV^b และ BV^d จะทำการอินเตอร์เซกชันกันเพื่อคำนวณค่าสนับสนุนของเซตรายการ $s^{bd} = 10$, ค่าความสม่ำเสมอของเซตรายการ $r^{bd} = 3$ และบิตเวกเตอร์ของเซตรายการ $BV^{bd} = \{208, 181\}$ ตามลำดับ เซตรายการ "bd" จะถูกจัดเก็บลงใน top-k list เนื่องจากค่าความสม่ำเสมอของเซตรายการ r^{bd} มีค่าเท่ากับค่าขีด

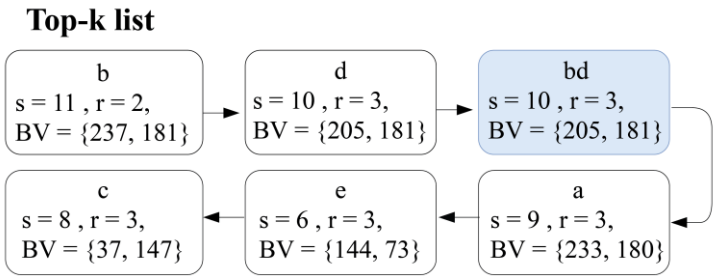
แบ่งความสม่ำเสมอ และจำนวนเซตรายการที่ปรากฏบ่อยและปรากฏอย่างสม่ำเสมอมีจำนวนน้อยกว่าค่าที่ผู้ใช้กำหนดไว้ (ดังแสดงในภาพที่ 3.12)

จากนั้นทำการพิจารณาเซตรายการในลำดับต่อมาคือ เซตรายการ "b" และเซตรายการ "a" เมื่อนำสองเซตรายการมาทำการจับคู่กับจะได้เป็นเซตรายการ "ba" จากนั้นบิตเวกเตอร์ BV^b และ BV^a จะทำการอินเทอร์เซกชันกันเพื่อคำนวณค่าสนับสนุนของเซตรายการ $s^{ba} = 9$, ค่าความสม่ำเสมอของเซตรายการ $r^{ba} = 3$ และบิตเวกเตอร์ของเซตรายการ $BV^{ba} = \{233, 180\}$ ตามลำดับ เซตรายการ "ba" จะถูกจัดเก็บลงใน top-k list *เนื่องจากค่าความสม่ำเสมอของเซตรายการ r^{ba} มีค่าเท่ากับค่าขีดแบ่งความสม่ำเสมอ และเซตรายการ "e" จะถูกลบออกจาก top-k list เนื่องจากมีค่าสนับสนุนเกินลำดับที่ k*

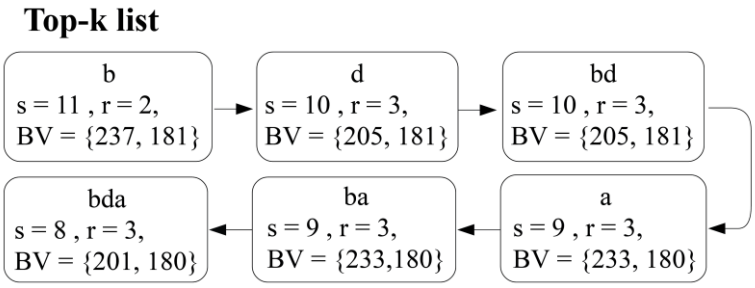
จากนั้นทำการพิจารณาเซตรายการในลำดับต่อมาที่มีขนาด 3 รายการคือ เซตรายการ "bd" และเซตรายการ "ba" เมื่อนำสองเซตรายการมาทำการจับคู่กับจะได้เป็นเซตรายการ "bda" จากนั้นบิตเวกเตอร์ BV^{bd} และ BV^{ba} จะทำการอินเทอร์เซกชันกันเพื่อคำนวณค่าสนับสนุนของเซตรายการ $s^{bda} = 8$, ค่าความสม่ำเสมอของเซตรายการ $r^{ba} = 3$ และบิตเวกเตอร์ของเซตรายการ $BV^{bda} = \{210, 180\}$ ตามลำดับ เซตรายการ "bda" จะถูกจัดเก็บลงใน top-k list *เนื่องจากค่าความสม่ำเสมอของเซตรายการ r^{bda} มีค่าเท่ากับค่าขีดแบ่งความสม่ำเสมอ และค่าสนับสนุน s^{bda} มีค่าเท่ากับค่าสนับสนุนในลำดับสุดท้ายใน top-k list เมื่อพิจารณาจนครบทุกคู่ของเซตรายการใน top-k list แล้วจะได้ผลลัพธ์ดังภาพที่ 3.13*



ภาพที่ 3.11 การสร้าง top-k list สำหรับจัดเก็บผลลัพธ์



ภาพที่ 3.12 การจับคู่เซตรายการ "b" และเซตรายการ "d" และเพิ่มเซตรายการ "bd" ใน top-k list

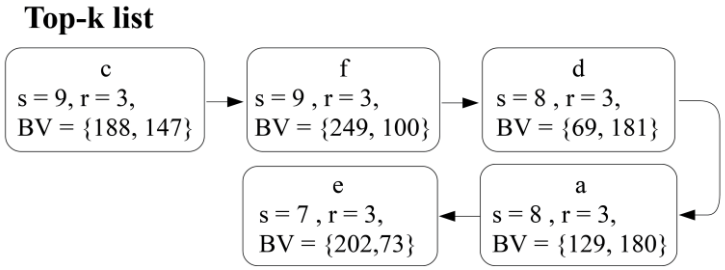


ภาพที่ 3.13 top-k list *หลังจากการค้นหารูปแบบที่ปรากฏบ่อยและสม่ำเสมอจากทรานแซกชันที่* $t_5 - t_{20}$ *จาก* ds_1

เมื่อข้อมูลกระแสมีการเพิ่มขึ้นของทรานแซกชันที่ต้องทำการพิจารณา (ดังแสดงในภาพที่ 3.3) ที่ซึ่งจะทำให้หน้าต่างการพิจารณาทรานแซกชันมีการเปลี่ยนแปลงในการพิจารณาข้อมูลจาก ds_2 *ขั้นตอนวิธี Eliminating occurrence information จะพิจารณาข้อมูล* ds_2 *ที่ประกอบไปด้วยจำนวนทรานแซกชันทั้งหมด 5 ทรานแซกชัน ดังนั้นทรานเซกชันของ* ds_2 *ที่มีจำนวนทรานแซกชันทั้งหมดเท่ากับ 5 ทรานแซกชันจะถูกรวมเข้ากับทรานแซกชันของ* ds_1 *ที่มีจำนวนทรานแซกชันเท่ากับ 11 ทรานแซกชัน และก่อนที่จะพิจารณาเซตรายการของ* ds_2 *จะต้องทำการอัพเดท 5 บิตแรกที่ปรากฏในไบต์แรกบิตเวกเตอร์ใน itemList ให้มีค่าเท่ากับ "00000" จากนั้นทำการคำนวณค่าสนับสนับ และค่าความสม่ำเสมอของทุกเซตรายการใน itemList (ดังแสดงในภาพที่ 3.14) จากนั้นขั้นตอนทั้งหมดข้างต้นจะถูกทำซ้ำในขั้นตอนต่อไป จากภาพที่ 3.15 แสดงให้เห็น* top-k list *ที่มีเซตรายการที่ปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจาก* ds_1 *และ* ds_2 *เมื่อมีข้อมูลกระแสเข้ามาใหม่ขั้นตอนวิธี TFRIM-DS จะทำซ้ำในทุกขั้นตอนที่กล่าวข้างต้น*

i	s	r	ls	BV	
b	7	2	16	{5, 181}	→ {00000101 10110101}
d	7	2	16	{2, 181}	→ {00000101 10110101}
a	5	3	14	{1, 180}	→ {00000001 10110100}
c	5	3	16	{4, 147}	→ {00000100 10010011}
e	4	3	16	{2, 73}	→ {00000010 01001001}
f	4	3	14	{1, 100}	→ {00000001 01100100}

ภาพที่ 3.14 itemList หลังจากทำการอัพเดท 5 บิตแรกในไบต์แรกของบิตเวกเตอร์



ภาพที่ 3.15 top-k list หลังจากทำการค้นหารูปแบบที่ปรากฏบ่อยและปรากฏอย่างสม่ำเสมอจาก ds_1 และ ds_2

บทที่ 4
ผลการวิจัย

ในบทนี้จะกล่าวถึงการวิเคราะห์และทดสอบประสิทธิภาพของขั้นตอนวิธี TFRIM-DS (Top-k Frequent-Regular Itemsets Miner over Data Streams) ในการค้นหารูปแบบที่ปรากฏบ่อยและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแสด้วยเทคนิคการเลื่อนหน้าต่าง โดยในการวิเคราะห์ประสิทธิภาพจะทำการวิเคราะห์ความซับซ้อนเชิงเวลาและหน่วยความจำของขั้นตอนวิธี TFRIM-DS แต่ในส่วนของการทดสอบประสิทธิภาพจะดำเนินการกับชุดข้อมูลจริงและชุดข้อมูลสังเคราะห์ (ที่ซึ่งเป็นชุดข้อมูลที่ได้รับความนิยมในการทดสอบประสิทธิภาพของขั้นตอนวิธีสำหรับการค้นหารูปแบบที่น่าสนใจ) โดยจะทำการพิจารณาเกี่ยวกับเวลาและหน่วยความจำที่ใช้ในการประมวลผล

4.1 การวิเคราะห์ประสิทธิภาพของขั้นตอนวิธี

การวิเคราะห์ความซับซ้อนการคำนวณ (Complexity analysis) ของขั้นตอนวิธี TFRIM-DS จะพิจารณาในเชิงเวลาและหน่วยความจำที่ใช้ในการประมวลผลดังนี้

ข้อเสนอที่ 4.1 ความซับซ้อนเชิงเวลาของขั้นตอนวิธี TFRIM-DS คือ $O((n) + (nw) + (2k\ |BV|))$ โดยที่ 1) n คือ จำนวนรายการทั้งหมดในเซต I, 2) w คือ ขอบเขตของทรานเซกชันที่ทำการพิจารณา, 3) k คือ จำนวนผลลัพธ์ที่ต้องการ และ 4) $|BV|$ คือ จำนวนไบต์สูงสุดในบิตเวกเตอร์ สามารถคำนวณได้จาก $w/8$ ตามลำดับ

พิสูจน์ข้อเสนอที่ 4.1 ในการพิจารณากลุ่มของทรานแซกชัน ds_j ที่ปรากฏในฐานข้อมูลกระแส ขั้นตอนวิธี TFRIM-DS จะเริ่มจากการลบข้อมูลการปรากฏของแต่ละรายการใน itemList ที่ซึ่งข้อมูลการปรากฏที่ถูกลบจะเป็นข้อมูลการปรากฏในทรานแซกชันที่อยู่ในหน้าต่างการพิจารณา โดยขั้นตอนลบข้อมูลปรากฏจะใช้การประมวลผลเท่ากับจำนวนรายการทั้งหมดที่ทำการพิจารณา (กล่าวคือจะใช้การประมวลผลเท่ากับ $|I| = n$) จากนั้นจะทำการอ่านแต่ละทรานแซกชัน t_p ที่ปรากฏในฐานข้อมูลกระแส ds_j เพื่อทำการจัดเก็บข้อมูลการปรากฏของแต่ละรายการที่ปรากฏในทรานแซกชัน t_p โดยในกรณีเลวร้ายสุดจะต้องทำการอ่านทรานแซกชันเป็นจำนวนทั้งสิ้น w ทรานแซกชัน โดยที่แต่ละทรานแซกชันสามารถมีรายการที่ปรากฏขึ้นได้สูงสุดเท่ากับ $|I| = n$ ดังนั้น การอ่านทรานแซกชันจาก ฐานข้อมูลกระแส ds_j จะใช้การประมวลผลสูงสุดเท่ากับ $w \times n$ ต่อไป ขั้นตอนวิธี TFRIM-DS จะทำการเลือกเฉพาะรายการที่มีค่าสนับสนุนสูงสุด k รายการ แล้วทำการพิจารณาแต่ละคู่ลำดับของรายการใน k รายการที่ถูกเลือก เพื่อทำการพิจารณาเซตรายการที่มีจำนวนรายการที่เพิ่มขึ้น โดยจำนวนการพิจารณาคู่ลำดับของรายการทั้งหมดที่เป็นไปได้จะมีจำนวนทั้งสิ้น 2^k รายการ โดยที่ในการพิจารณาแต่ละคู่ลำดับของรายการจะต้องทำการอินเทอร์เซกชันบิตเวกเตอร์เพื่อทำการคำนวณค่าสนับสนุนและค่าสม่ำเสมอที่ซึ่งจะใช้การประมวลผลสูงสุดเท่ากับ $|BV|$ (เมื่อ $|BV|$ คือขนาดสูงสุดของไบต์ที่ถูกบรรจุอยู่ในบิตเวกเตอร์ สามารถคำนวณได้จาก $w/8$) โดยจากข้างต้นสามารถกล่าวได้ว่า ขั้นตอนวิธี TFRIM-DS จะใช้การประมวลผลคู่ลำดับของรายการทั้งหมดที่เป็นไป

ได้เท่ากับ $2^k \times |BV|$ ท้ายสุด เราสามารถสรุปได้ว่า ความซับซ้อนเชิงเวลาของขั้นตอนวิธี TFRIM-DS คือ $O((n) + (nw) + (2^k |BV|))$

ข้อเสนอที่ 4.2 ความซับซ้อนเชิงหน่วยความจำของขั้นตอนวิธี TFRIM-DS คือ $O((n|BV|) + (k|BV|))$ โดยที่ 1) n คือ จำนวนรายการทั้งหมดในเซต I, 2) k คือ จำนวนผลลัพธ์ที่ต้องการ และ 3) $|BV|$ คือ จำนวนไบต์สูงสุดในบิตเวกเตอร์ สามารถคำนวณได้จาก $w/8$ (เมื่อ w คือ ขอบเขตของทรานเซกชันที่ทำการพิจารณา) ตามลำดับ

พิสูจน์ข้อเสนอที่ 4.2 การค้นหารูปแบบที่ปราฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแสด้วยขั้นตอนวิธี TFRIM-DS จะทำการสร้าง itemList ที่จะบรรจุไปด้วย n รายการ และบรรจุไปด้วยบิตเวกเตอร์ของแต่ละรายการที่มีขนาดสูงสุดเป็น $|BV|$ (กล่าวคือ ขนาดสูงสุดของบิตเวกเตอร์คือจำนวนไบต์สูงสุดในบิตเวกเตอร์ สามารถคำนวณได้จาก $w/8$ เมื่อ w คือ ขอบเขตของทรานเซกชันที่ทำการพิจารณา) ดังนั้น itemList จะใช้พื้นที่ในการจัดเก็บข้อมูลทั้งหมด $n \times |BV|$ นอกจากนั้น ในระหว่างการประมวลผล ขั้นตอนวิธี TFRIM-DS จะทำการจัดเก็บรายการ (เซตรายการ) ที่ปราฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอไว้ใน top-k list โดยใน top-k list จะบรรจุไปด้วย k เซตรายการ และบรรจุไปด้วยบิตเวกเตอร์ของแต่ละเซตรายการที่ขนาดสูงสุดเท่ากับ $|BV|$ ดังนั้น top-k list จะใช้พื้นที่ในการจัดเก็บข้อมูลทั้งหมด $k \times |BV|$ ท้ายสุด จากข้างต้น เราสามารถสรุปได้ว่า ขั้นตอนวิธี TFRIM-DS จะมีการใช้พื้นที่ในการจัดเก็บข้อมูลระหว่างการประมวลผลสูงสุดเท่ากับ $O((n|BV|) + (k|BV|))$

4.2 ผลการทดลอง

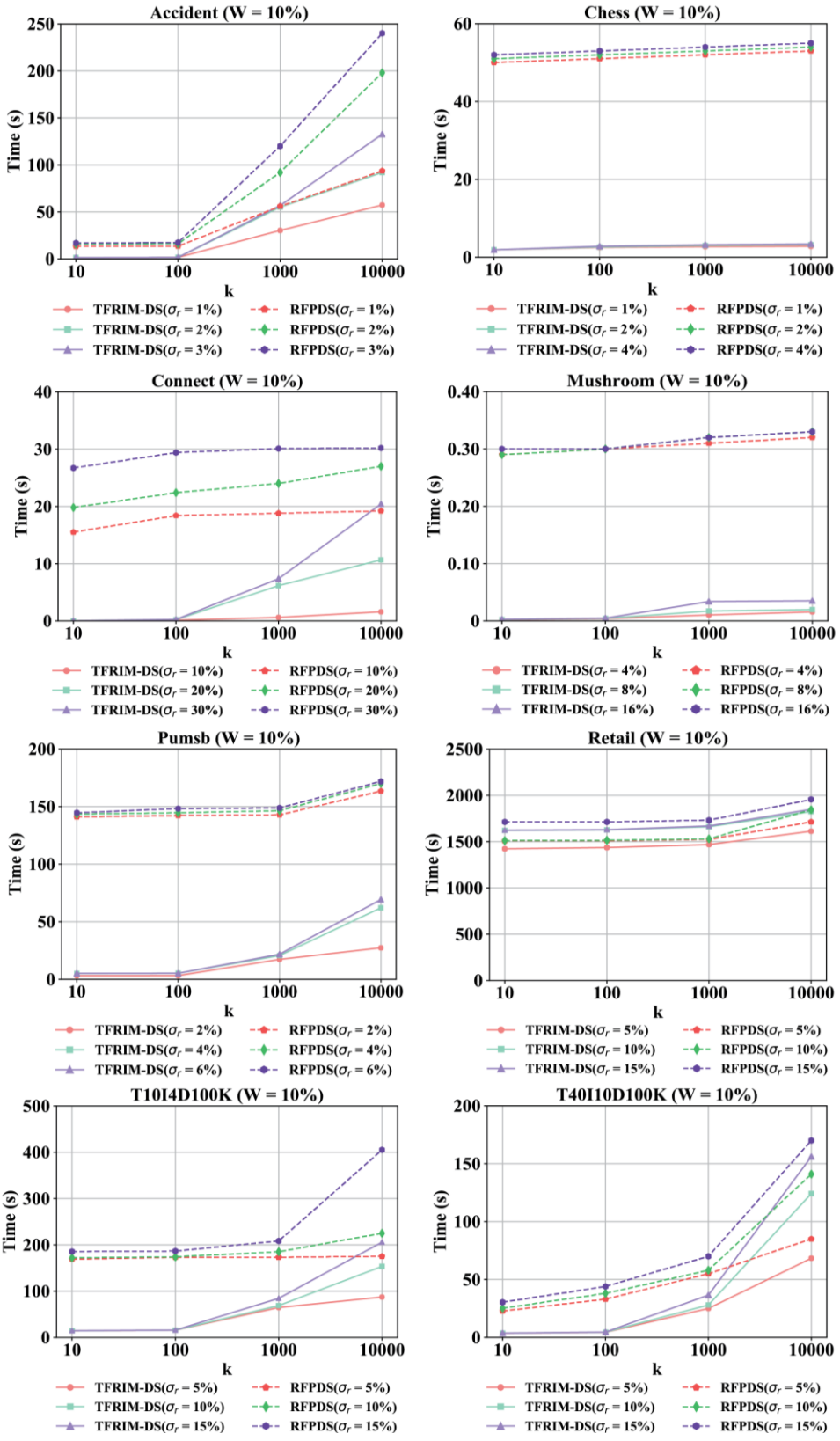
นอกเหนือจากการวิเคราะห์ความซับซ้อนของขั้นตอนวิธี TFRIM-DS ผู้วิจัยได้ทำการทดสอบประสิทธิภาพของขั้นตอนวิธี TFRIM-DS เพื่อทดสอบการใช้เวลาและหน่วยความจำในการประมวลผลจริง โดยการทดลองจะดำเนินการกับ 8 ชุดข้อมูลที่ได้รับนิยมในการทดสอบประสิทธิภาพของขั้นตอนวิธีสำหรับค้นหาเซตรายการที่น่าสนใจ (ดังแสดงคุณลักษณะของข้อมูลในตารางที่ 2.1) นอกจากนั้น ผู้วิจัยได้ทำการเปรียบเทียบประสิทธิภาพการทำงานของขั้นตอนวิธี TFRIM-DS กับขั้นตอนวิธีใกล้เคียง แต่ด้วยเนื่องจาก ของขั้นตอนวิธี TFRIM-DS เป็นขั้นตอนวิธีแรกที่ถูกเสนอขึ้นเพื่อค้นหาเซตรายการที่ปราฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแส จึงทำให้ไม่สามารถเปรียบเทียบกับขั้นตอนวิธีที่ค้นหารูปแบบในลักษณะเดียวกันได้ แต่อย่างไรก็ตาม ผู้วิจัยได้ทำการค้นหาขั้นตอนวิธีที่ทำการค้นหารูปแบบที่มีความใกล้เคียงกัน คือ ขั้นตอนวิธี RFPDS (Regular Frequent Patterns from Data Streams) ที่ซึ่งเป็นขั้นตอนวิธีสำหรับค้นหาเซตรายการที่ปราฏบ่อยและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแส เมื่อดำเนินการเปรียบเทียบดังกล่าว ผู้วิจัยได้ทำการกำหนดให้ทั้งสองขั้นตอนวิธีใช้พารามิเตอร์ที่เหมือนกันคือ 1) ค่าขีดแบ่งสม่ำเสมอ และ 2) ขอบเขตของหน้าต่าง (ทรานแซกชัน) ที่ทำการพิจารณาแต่ทั้งสองขั้นตอนวิธีจะมีพารามิเตอร์ที่ต่างกันคือ ขั้นตอนวิธี TFRIM-DS จะต้องการจำนวนผลลัพธ์ที่ผู้ใช้ต้องการ แต่สำหรับขั้นตอนวิธี RFPDS จะต้องการค่าขีดแบ่งสนับสนุน โดยจากความต้องการดังกล่าว ผู้วิจัยได้กำหนดให้ค่าขีดแบ่งสนับสนุนสำหรับขั้นตอนวิธี RFPDS มีค่าเท่ากับค่าสนับสนุนของเซตรายการลำดับที่ k^{th} ที่ค้นหาได้จาก

ขั้นตอนวิธี TFRIM-DS ซึ่งจากการกำหนดดังกล่าวจะทำให้ทั้งสองขั้นตอนวิธีสามารถค้นหาผลลัพธ์ที่เหมือนกันได้

ขั้นตอนวิธีทั้งสองถูกเขียนด้วยภาษ C++ และ การทดลองบนเครื่องคอมพิวเตอร์ที่มีความเร็ว CPU 3.0 GHz., Memory 8 GB และระบบปฏิบัติการ Windows 10 ในการทดลองได้กำหนดค่าขีดแบ่งความสม่ำเสมอ σ_r ให้ค่าอยู่ในระหว่าง 1% ถึง 30% ตามลักษณะของฐานข้อมูลรายการ, จำนวนผลลัพธ์ k ให้มีค่าเป็น 10, 100, 1,000 และ 10,000 ตามลำดับ, และขนาดของหน้าต่างการพิจารณา w ให้มีขนาดเท่ากับ 10%, 20%, 50% และ 80% ของจำนวนทรานแซกชันในฐานข้อมูล

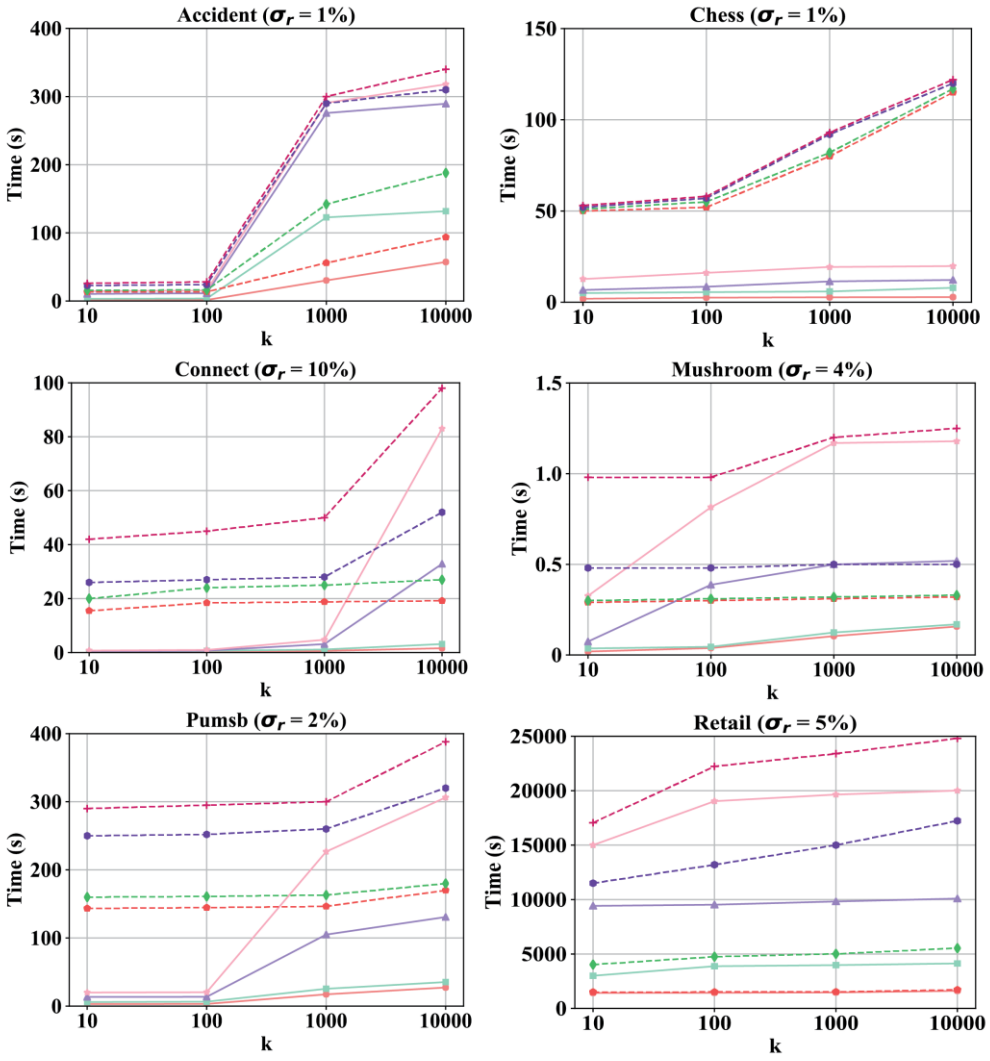
4.2.1 เวลาที่ใช้ในการประมวลผล

ภาพที่ 4.1 แสดงเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธี TFRIM-DS และขั้นตอนวิธี RFPDS ภายใต้การกำหนดขนาดของหน้าต่างให้มีค่าเท่ากับ 10% ของจำนวนทรานแซกชันทั้งหมดในแต่ละฐานข้อมูล และการกำหนดค่าขีดแบ่งสม่ำเสมอแบบผันแปรที่ซึ่งมีค่าอยู่ระหว่าง 1% ถึง 30% ของจำนวนทรานแซกชันทั้งหมดในแต่ละฐานข้อมูล (หมายเหตุ การกำหนดค่าขีดแบ่งสม่ำเสมอในการทดลองกับแต่ละฐานข้อมูลจะกำหนดตามความหนาแน่นของการปรากฏขึ้นของรายการ/เซตรายการในแต่ละฐานข้อมูล โดยจะทำการกำหนดให้มีค่าเท่ากับงานวิจัยก่อนหน้า อาทิเช่น การค้นหารูปแบบที่ปรากฏอย่างสม่ำเสมอจากฐานข้อมูลกระแส (Tanbeer, S., K., และคณะ, 2010), การค้นหารูปแบบที่ปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอแบบปิด (Amphawan, K., และคณะ, 2015) และ การใช้เทคนิคการเลื่อนหน้าต่างในการค้นหารูปแบบที่ปรากฏบ่อยและปรากฏอย่างสม่ำเสมอจากฐานข้อมูลกระแสโดยใช้การจัดเก็บแบบตั้ง (Kumar, G., V., และคณะ, 2012) โดยจากภาพที่ 4.1 จะสามารถสังเกตุได้ว่าเวลาในการประมวลผลของทั้งสองขั้นตอนวิธีจะเพิ่มขึ้นตามจำนวนผลลัพธ์ k ที่เพิ่มขึ้นและเพิ่มขึ้นตามค่ากำหนดค่าขีดแบ่งสม่ำเสมอที่เพิ่มขึ้น เนื่องจากจำนวนของเซตรายการที่ปรากฏบ่อยและสม่ำเสมอมีจำนวนเพิ่มขึ้นนอกจากนั้นยังสามารถสังเกตุได้อีกว่าขั้นตอนวิธี TFRIM-DS สามารถประมวลผลได้เร็วกว่าขั้นตอนวิธี RFPDS เนื่องจากการจัดเก็บข้อมูลของขั้นตอนวิธี TFRIM-DS โดยใช้บิตเวกเตอร์ในการจัดเก็บข้อมูลสามารถนำกลับมาใช้ซ้ำเมื่อมีข้อมูลเข้ามาใหม่ซึ่งช่วยในการรักษาข้อมูลที่จำเป็นได้อย่างมีประสิทธิภาพ

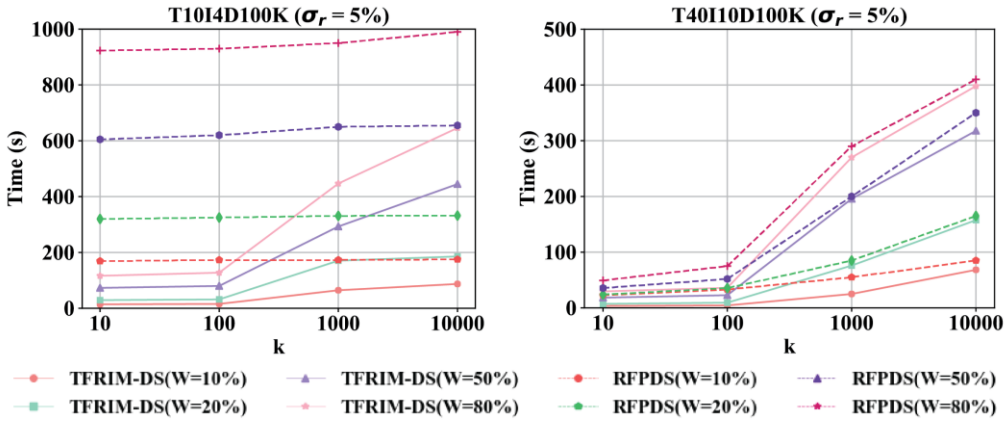


ภาพที่ 4.1 เวลาที่ใช้ในการประมวลผลของ ขั้นตอนวิธี TFRIM-DS และขั้นตอนวิธี RFPDS โดยมีค่าขีดแบ่งความสม่ำเสมอที่แตกต่างกัน

ภาพที่ 4.2 แสดงเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธี TFRIM-DS และขั้นตอนวิธี RFPDS ภายใต้การกำหนดขนาดของหน้าต่างมีขนาดเท่ากับ 10%, 20%, 50% และ 80% ของจำนวนทรานแซกชันในฐานข้อมูล และค่าขีดแบ่งความสม่ำเสมอที่กำหนดไว้ซึ่งมีค่าอยู่ระหว่าง 1% ถึง 5% ของจำนวนทรานแซกชันทั้งหมดในแต่ละฐานข้อมูล โดยจากภาพที่ 4.2 จะสามารถสังเกตุได้ว่าเวลาในการประมวลผลของทั้งสองขั้นตอนวิธีจะเพิ่มขึ้นตามจำนวนผลลัพธ์ k ที่เพิ่มขึ้นและเพิ่มขึ้นตามขนาดของหน้าต่างการที่เพิ่มขึ้น เนื่องจากขอบเขตการพิจารณาที่เพิ่มขึ้นทำให้เซตรายการที่ได้มีจำนวนมาก ในการค้นหารูปแบบปรากฏบ่อยและสม่ำเสมอจึงใช้เวลานานขึ้น นอกจากนั้นยังสามารถสังเกตุได้อีกว่าขั้นตอนวิธี TFRIM-DS สามารถประมวลผลได้เร็วกว่าขั้นตอนวิธี RFPDS เนื่องจากขั้นตอนวิธี TFRIM-DS สามารถอ่านข้อมูลซ้ำใน itemList โดยอ่านข้อมูลจากฐานข้อมูลเพียงหนึ่งครั้ง ส่วนขั้นตอนวิธี RFPDS จำเป็นต้องมีการอ่านข้อมูลเดิมจากฐานข้อมูลซ้ำทุกครั้งที่มีข้อมูลเข้ามาใหม่จึงทำให้ต้องมีการประมวลผลใหม่ตั้งแต่เริ่มต้น



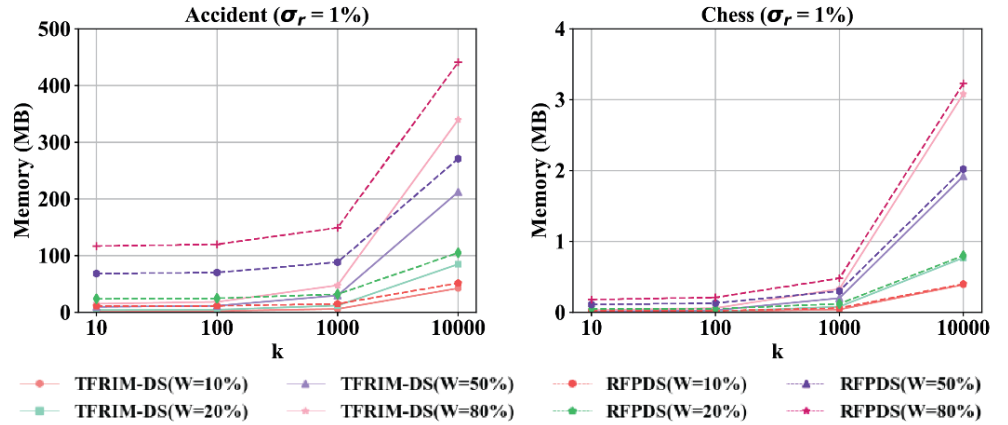
ภาพที่ 4.2 เวลาที่ใช้ในการประมวลผลของ ขั้นตอนวิธี TFRIM-DS และขั้นตอนวิธี RFPDS โดยมีขนาดของหน้าต่างการพิจารณาที่แตกต่างกัน



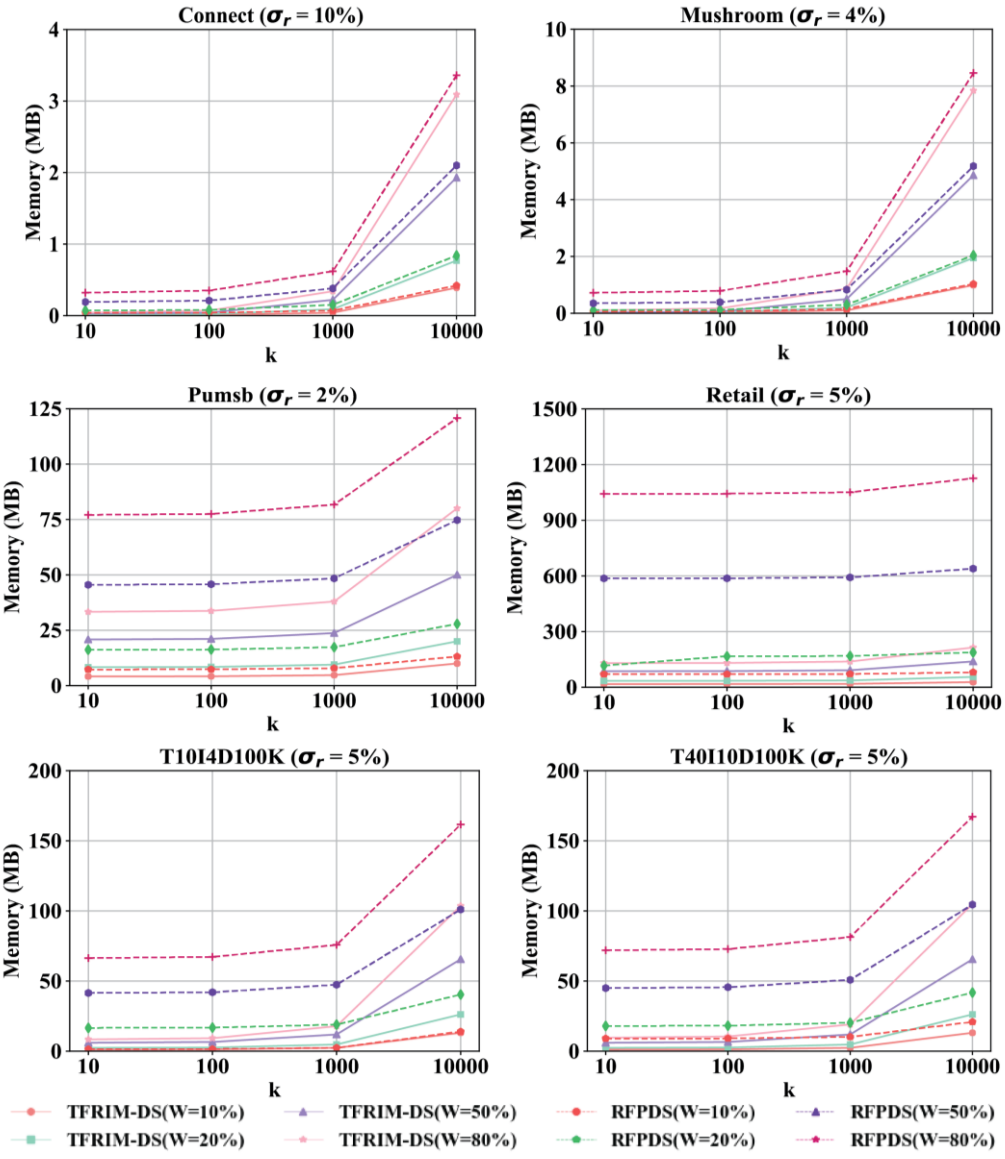
ภาพที่ 4.2 เวลาที่ใช้ในการประมวลผลของ ขั้นตอนวิธี TFRIM-DS และขั้นตอนวิธี RFPDS โดยมีขนาดของหน้าต่างการพิจารณาที่แตกต่างกัน (ต่อ)

4.2.2 พื้นที่หน่วยความจำที่ใช้ในการจัดเก็บข้อมูล

ภาพที่ 4.3 แสดงการใช้พื้นที่หน่วยความจำในการจัดเก็บข้อมูลของขั้นตอนวิธี TFRIM-DS และขั้นตอนวิธี RFPDS ภายใต้การกำหนดขนาดของหน้าต่างมีขนาดเท่ากับ 10%, 20%, 50% และ 80% ของจำนวนทรานแซกชันในฐานข้อมูล และค่าขีดแบ่งความสม่ำเสมอที่กำหนดไว้ซึ่งมีค่าอยู่ระหว่าง 1% ถึง 5% ของจำนวนทรานแซกชันทั้งหมดในแต่ละฐานข้อมูล โดยจากภาพที่ 4.3 จะสามารถสังเกตุได้ว่าการใช้พื้นที่หน่วยความจำในการจัดเก็บข้อมูลของทั้งสองขั้นตอนวิธีจะเพิ่มขึ้นตามจำนวนผลลัพธ์ k ที่เพิ่มขึ้นและเพิ่มขึ้นตามขนาดของหน้าต่างการที่เพิ่มขึ้น เนื่องจากขอบเขตการพิจารณาที่เพิ่มขึ้นทำให้เซตรายการที่ได้มีจำนวนมาก ในการค้นหารูปแบบปรากฏบ่อยและสม่ำเสมอจึงใช้พื้นที่หน่วยความจำในการจัดเก็บข้อมูลเพิ่มขึ้น นอกจากนั้นยังสามารถสังเกตุได้อีกว่าขั้นตอนวิธี TFRIM-DS ใช้พื้นที่หน่วยความจำในการจัดเก็บข้อมูลน้อยกว่าขั้นตอนวิธี RFPDS เนื่องจากขั้นตอนวิธี TFRIM-DS ใช้บิตเวกเตอร์ในการจัดเก็บข้อมูลที่ซึ่งข้อมูลที่ถูกจัดเก็บจะเท่ากับ $|DS|/8$



ภาพที่ 4.3 พื้นที่หน่วยความจำที่ใช้ในการจัดเก็บข้อมูลของ ขั้นตอนวิธี TFRIM-DS และขั้นตอนวิธี RFPDS



ภาพที่ 4.3 พื้นที่หน่วยความจำที่ใช้ในการจัดเก็บข้อมูลของ ขั้นตอนวิธี TFRIM-DS และขั้นตอนวิธี RFPDS (ต่อ)

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

วิทยานิพนธ์ฉบับนี้ได้นำเสนอการพัฒนาการค้นหารูปแบบที่ปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแส ภายใต้กรอบของหน้าต่างการพิจารณา ด้วยขั้นตอนวิธี TFRIM-DS (Top-k Frequent-Regular Itemsets Miner over Data Streams) โดยการค้นหาผลลัพธ์ ผู้ใช้ต้องเป็นผู้กำหนดค่าขีดแบ่งความสม่ำเสมอ σ_r , จำนวนผลลัพธ์ที่ต้องการ k และขอบเขตหน้าต่างการพิจารณา w ขั้นตอนวิธี TFRIM-DS ได้นำเทคนิคการเลื่อนหน้าต่างมาใช้ในการพิจารณาแต่ละทรานแซกชันที่ไหลเข้ามาจากฐานข้อมูลกระแส โดยทำการอ่านข้อมูลจากฐานข้อมูลกระแสเพียงครั้งเดียว และได้มีการประยุกต์ใช้บิตเวกเตอร์ในการจัดเก็บข้อมูลทรานแซกชันของการปรากฏขึ้นของแต่ละเซตรายการ และเมื่อมีข้อมูลเข้ามาใหม่บิตที่จัดเก็บข้อมูลที่ไม่มีความสำคัญ (ทรานแซกชันที่เก่าสุด) ในบิตเวกเตอร์ถูกนำมาใช้ซ้ำตามจำนวนของทรานแซกชันที่เข้ามาใหม่ ที่ซึ่งช่วยในเรื่องของการลดเวลาในการประมวลผลและลดการใช้พื้นที่หน่วยความจำในการจัดเก็บข้อมูล เมื่อเปรียบเทียบขั้นตอนวิธี TFRIM-DS กับขั้นตอนวิธี RFPDS (Regular Frequent Patterns from Data Streams) สรุปได้ว่า ขั้นตอนวิธี TFRIM-DS มีการใช้เวลาในการประมวลผลและพื้นที่หน่วยความจำในการจัดเก็บข้อมูลน้อยกว่า ขั้นตอนวิธี RFPDS เนื่องจากขั้นตอนวิธี RFPDS ทำการจัดเก็บข้อมูลด้วยหมายเลขทรานแซกชันที่เซตรายการปรากฏ และเมื่อมีทรานแซกชันเข้ามาใหม่ขั้นตอนวิธี RFPDS จำเป็นต้องอ่านข้อมูลเดิมซ้ำทุกครั้ง และต้องทำการประมวลผลทั้งหมด

5.2 ปัญหาและข้อจำกัดที่พบจากการวิจัย

จากการทดลองการค้นหารูปแบบที่ปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแสด้วยเทคนิคการเลื่อนหน้าต่าง ด้วยฐานข้อมูลรายการที่เป็นฐานข้อมูลจริงและฐานข้อมูลสังเคราะห์ เป็นฐานข้อมูลรายการที่มีลักษณะหนาแน่นและบางเบา จึงพบว่าเมื่อฐานข้อมูลมีขนาดใหญ่ หรือมีลักษณะบางเบาจะใช้ระยะเวลาในการประมวลผลและพื้นที่หน่วยความจำในการจัดเก็บข้อมูลเป็นจำนวนมาก เนื่องจากเครื่องคอมพิวเตอร์ที่ใช้ในการทดลองมีหน่วยความจำไม่เพียงพอต่อความต้องการ ขั้นตอนวิธี TFRIM-DS จึงไม่สามารถทำการทดลองได้ในทุกฐานข้อมูลรายการ

5.3 ข้อเสนอแนะ

ในงานวิทยานิพนธ์นี้ได้นำเสนอการค้นหารูปแบบที่ปรากฏบ่อยเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแสด้วยเทคนิคการเลื่อนหน้าต่าง ซึ่งการค้นหารูปแบบที่ปรากฏบ่อยและปรากฏอย่างสม่ำเสมอสามารถนำมาประยุกต์ใช้กับธุรกิจที่มีการเพิ่มขึ้นของทรานแซกชันเป็นจำนวนมากจากฐานข้อมูลกระแสที่มีการไหลเวียนของข้อมูลอย่างต่อเนื่องในหลายแง่มุม ที่ซึ่งสามารถนำขั้นตอนวิธีไปพัฒนาหรือปรับปรุงให้มีประสิทธิภาพมากยิ่งขึ้น และสามารถนำประยุกต์ใช้ได้ในหลายๆ

ด้าน อาทิเช่น ใช้ในการประกอบการตัดสินใจ การวางแผนกลยุทธ์ การพยากรณ์การขายสินค้าตามฤดูกาล กระตุ้นการขายสินค้า การเลือกทำเลในการลงทุนธุรกิจ การเลือกทำเลด้านอสังหาริมทรัพย์ รวมไปถึงทางด้านการแพทย์ การประเมินการใช้ยาที่เหมาะสมกับผู้ป่วย การวิเคราะห์พฤติกรรมของผู้ป่วยที่ส่งผลต่อโรคที่เป็น พฤติกรรมการดำรงค์ชีวิตของผู้สูงอายุ เด็ก ผู้พิการ หรือผู้ป่วยจิตเวชในการป้องกันอัตรายที่จะเกิดขึ้น

บรรณานุกรม

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules in large databases. in *Proceedings of the 20th International Conference on Very Large Data Bases,* pp. 487-499.

Amir, A., & Levy, A. (2012). Approximate period detection and correction. In *Proceedings of the 19th International Symposium on Siring Processing and Information Retrieval, Vol.7608, pp. 1-15.*

Amphawan, K., Lenca, P., & Surarerks, A. (2009). Mining top-k periodic frequent patterns without support threshold. in *Proceedings of the 3rd International Conference on Advances in Information Technology*, Vol. 55, pp. 18–29.

Amphawan, K., Lenca, P., & Surarerks, A. (2012). Mining top-k regular-frequent itemset using database partitioning and support estimation. in *Expert Systems with Applications*, Vol.39, pp. 1924-1936.

Amphawan, K., & Lenca, P. (2013). Mining top-k frequent/regular patterns based on user-given trade-off between frequency and regularity. in *Advances in Information Technology, Vol. 409, pp. 1-12.*

Amphawan, K., & Lenca, P. (2015). Mining top-k frequent-regular closed patterns. *Expert Systems with Applications*, Vol. 42, No. 21, pp. 7882-7894.

Amphawan, K., Lenca, P., Jitpattanakul, A.& Surarerks, A. (2016) Mining high utility itemsets with regular occurrence. in *Journal of ICT Research and Applications,* Vol.10, No.2, pp.153-176.

Calders, T., Dexters, N., Gillis, J.M., J., & Goethals, B. (2014). Mining frequent items in a stream. in *Information Systems,* Vol. 39, pp. 233-255.

Chen, H. (2014). Mining top-k frequent patterns over data streams sliding window. In *Journal of Intelligent Information Systems*, Vol. 42, pp. 111-131.

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. Vol. 29, No. 2, pp. 1-2.

Kiran, R. & Kitsuregawa, M, (2015). Discovering chronic-frequent patterns in transaction databases. in *Databases in Networked Information Systems,* Vol.8999, pp. 12-26.

Kumar, G., V. & Kumari, V., V. (2012). Sliding window technique to mine regular frequent patterns in data streams using vertical format. in *2013 IEEE International Conference on Computational Intelligence and Computing Research, pp.1-4.*

Lam, H.T., & Calders, T. (2010). Mining top-k frequent items in a data stream with flexible sliding windows. in *Proceeding of the 16th ACM SIGKDD international conference on Knowled e discovery and data mining*, pp. 283-292.

Lee, G., Yun, U., & Ryu, K.H. (2014). Sliding window based weighted maximal frequent pattern mining over data streams. in *Expert Systems with Applications*, Vol.41, pp. 694-708.

Sittichaitaweekul, P., & Amphawan, K. (2014). Enhancing quality of results on Top-k Frequent-Regular Pattern mining. in *proceedings of 1st international conference on Engineering Science and Innovative Technology (ESIT-2014)*, pp.64.

Sreedevi, M., & Reddy, L. (2013). Mining regular closed patterns in transactional databases. In *Proceedings of the 7th International Conference on Intelligent Systems and Control,* pp. 399-413.

Tanbeer, S.K., Ahmed, C.F., Jeong, B., & Lee, Y. (2008). Efficient frequent pattern mining over data streams. in *Proceedings of the 17th ACM conference on Information and knowledge management*, pp. 1447-1448.

Tanbeer, S. K., Ahmed, C. F., Jeong, B. -S., & Lee, Y. -K. (2009). Discovering periodic frequent patterns in transactional databases. in *Proceedings of the 13th PacificAsia Conference on Advances in Knowledge Discovery and Data Mining*, pp. 242-253.

Tanbeer, S.K., Ahmed, C.F., & Jeong, B.-S (2010). Mining Regular Patterns in incremental transaction databases. In *Proceeding of the 12th International Asia-Pacific Web Confercence,* pp. 375-377.

Tanbeer, S.K., Ahmed, C.F., & Jeong, B.-S (2010). Mining Regular Patterns in Data Streams. in International Conference on *Database Systems for Advanced Application on Database Systems for Advanced Applicaions,* pp. 399-413.

Vo, B., Hong, T., & Le, B. (2011) Dynamic bit vector : An effice approach for mining frequent itemsets. in *Scientific Research and Essays*, Vol. 6(25). pp. 5358-5368.

Vo, B., Hong, T., & Le, B. (2012). DBV-Miner : A Dynamic Bit-Vector apporch for fast mining frequent closed itemsets. in *Expert Systems with Applications,* Vol. 39, pp. 7196-7206.

Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 3, pp. 372-390.

ภาคผนวก

ภาคผนวก ก

เอกสารรับรองผลการพิจารณาจริยธรรมการวิจัยในมนุษย์

บันทึกข้อความ

ส่วนงาน สำนักงานอธิการบดี กองบริหารการวิจัยและนวัตกรรม โทร. ๒๕๖๑ - ๒๕๖๒
ที่ ศธ ๖๒๐๐/ ๐๖๓๐๙ **วันที่** ๑๒ กรกฎาคม พ.ศ. ๒๕๖๑
เรื่อง ขอแจ้งรับรองวิทยานิพนธ์ที่ส่งมาขอรับการพิจารณาจริยธรรมการวิจัยในมนุษย์ มหาวิทยาลัยบูรพา

เรียน นางสาวธาชินี มีเสมา (นิสิตระดับบัณฑิตศึกษา คณะวิทยาการสารสนเทศ)

ตามที่ท่านได้ส่งเอกสารวิทยานิพนธ์เพื่อขอรับการพิจารณาจริยธรรมการวิจัยในมนุษย์ ในหัวข้อวิทยานิพนธ์เรื่อง การค้นหารูปแบบปรากฏบ่อยสุดเคอันดับแรกและปรากฏอย่างสม่ำเสมอจากข้อมูลกระแสด้วยเทคนิคการเลื่อนหน้าต่าง นั้น

บัดนี้ คณะกรรมการพิจารณาจริยธรรมการวิจัยในมนุษย์ มหาวิทยาลัยบูรพา ได้พิจารณาตามวิธีดำเนินการมาตรฐาน (Standard Operating Procedures, SOP) ฉบับที่ ๑ พ.ศ. ๒๕๖๐ ที่ได้ประกาศใช้เมื่อวันที่ ๙ มกราคม พ.ศ. ๒๕๖๐ แล้วว่า วิทยานิพนธ์ดังกล่าวไม่ได้ทำการศึกษาวิจัยในมนุษย์ จึงเห็นสมควรให้ดำเนินการวิจัยได้

จึงเรียนมาเพื่อโปรดทราบ

วิทวัส แจ้ง

(ผู้ช่วยศาสตราจารย์ ดร.วิทวัส แจ้งเอี่ยม)
ประธานคณะกรรมการพิจารณาจริยธรรมการวิจัยในมนุษย์
มหาวิทยาลัยบูรพา

ภาคผนวก ข

การเผยแพร่ผลงานวิทยานิพนธ์

Mining top-*k* frequent-regular itemsets from data streams based on sliding window technique

Tashinee Mesama and Komate Amphawan
Computational Innovation Laboratory, Faculty of Informatics, Burapha University, Chonburi, 20131, Thailand
Email: t.mesama@gmail.com, komate@gmail.com

***Abstract*—Frequent-regular itemset mining has achieved a great attention and applied in several applications. In this framework, an itemset that frequently and regularly occurs in a database is identified as interesting. However, without prior knowledge, the setting appropriate support and regularity thresholds to measure interestingness of itemsets is quite difficult. This may lead to none, only few or overwhelm of generated results causing users cannot further take advantages from these itemsets. In addition, mining interesting itemsets over data streams is a challenging task on various domains. Therefore, to cope with these issues, we here propose an approach to mine top-*k* frequent-regular itemsets over data streams. To mine such itemsets, the concept of sliding window is applied in which recent occurrences are considered to be more important than the former occurrences. An efficient single-pass algorithm, called *TFRIM-DS*, is also introduced to mine a set of k itemsets that regularly occur and have highest support in the current considered window. In addition, a bit-vector with a reuse technique is applied and designed to efficiently maintain occurrence information of each itemset. Experiments were conducted and showed efficiency of our proposed *TFRIM-DS* to mine top-*k* frequent-regular itemsets over sliding window of data streams.**

***Keywords*—Frequent patterns, Regular patterns, top-k patterns, Bit-vector, Data stream;**

I. Introduction

Recently, mining interesting itemsets over a data stream becomes an attractive research topic in data mining community. This can be applied in a wide range of applications such as traffic analysis, web click stream analysis, network intrusion detection, network traffic analysis, telecommunication call records analysis, mobile commerce analysis, etc. In real-world applications, a data stream is a continuous sequence of data having high arrival rate, huge volume of incoming data, changing on characteristics of data and so on. With these properties, mining interesting itemsets over a data stream is a highly complex and challenging task and the traditional algorithms based on the variation of *Apriori* [1], *FP-growth* [2] and *ECLAT* [3] algorithms cannot efficiently discover interesting itemsets over a data stream. Thus, there are various approaches focusing on increasing computational performance and designing measures to get different kind of itemsets (to get different and alternative knowledge) from mining interesting itemsets over a data stream.

Currently, frequent-regular itemset mining (*FRIM*) [4] has achieved a great attention and applied in several applications. In this approach, a support and a regularity thresholds are given from users to measure interestingness of itemsets. Then, an itemset that frequently and regularly occurs in a database (*i.e.* occur more frequent and more regular than the user-given support and regularity thresholds) is identified as an interesting itemset. Since the great interest of *FRIM*, *FRIM* has several extensions such as mining itemsets with approximate regularity (periodic) of occurrence [5], [6], mining (frequent) regular itemsets from incremental database (data stream) [7], [8], mining regular closed itemsets [9], mining chronic-frequent itemsets [10], mining high-utility regular itemsets [11], etc.

However, without prior knowledge, the users may suffer from the setting appropriate support and regularity thresholds leading to the situation about none, only few or overwhelm of results generated. These situations makes users cannot further take advantages from these itemsets. Therefore, to cope with this issue, the task of top-*k* frequent-regular itemsets mining over a transactional database (*TFRIM*) [12], [13], [14] is then introduced to control the number of generated results. However, the input database *TFRIM* usually contains a static number of transactions causing *TFRIM* cannot efficiently discover the itemsets from dynamic database. Moreover, to the best of our knowledge, there is no train of thought to discover top-k frequent-regular itemsets over a data stream and there is a room to develop efficient methods for this approach.

Thus, in this paper, we here introduce a new single-pass algorithm, called *TFRIM-DS* (*Top-k Frequent-Regular Itemsets Miner over Data Streams*), to discover a set of k itemsets that regularly occur and have highest support in the current data stream. The concept of sliding window is applied to scope constant amount of recent transactions where the recent occurrences are considered to be more important than the former occurrences. Moreover, the bit-vector structure and a technique for reusing bits of bit-vector are applied and designed to efficiently maintain occurrence information of each item/itemset and to increase computational performance. Experiments were conducted on benchmark datasets to observe efficiency of *TFRIM-DS* and the results show that *TFRIM-DS* can efficient in both computational time and memory usage based on variations of parameter setting.

II. Problem definition

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a finite set of items. A set $X \subseteq I$ is a set of items or a k-itemset if X has k items. A data stream $DS = \{ds_1, ds_2, \ldots\}$ is a infinite set of batches of transactions. In *DS*, each batch $ds_d = \{t_1, t_2, \ldots, t_{|ds_d|}\}$ contains a sequence of transactions. Each transaction $t_j \in ds_d$ is a tuple of $\langle j, Y \rangle$ where j is a unique transaction identifier

978-1-5386-4804-9/18/$31.00 @2018 IEEE

(also called *tid*) and $Y \subseteq I$ is a set of items occurs in the transaction t_j. If $X \subseteq Y$, it then can be inferred that X occurs in t_j or t_j contains X. For an itemset X, there is an associated set of ordered tids of transactions in *DS* containing X, defined as $T^X = \{p, \ldots, q\}$. The support of X expresses the frequency of occurrence of X, denoted as $s^X = |T^X|$. The regularity of X, $r^X = max(fr^X, rt_u^X, \ldots, rt_v^X, lr^X)$, is the maximum consecutive transactions that X does not occur in database, where $fr^X = p - o$ is the number of transactions from the first transaction t_o of the current consider window to the first occurrence of X in the transaction t_p, each $rt_v^X = v - u$ is the number of transactions between two consecutive occurrences of X in transactions t_u and t_v, and $lr^X = w - q$ is the number of transaction from the last occurrence of X in the transaction t_q to the end of window w, respectively.

Given a data stream $DS = \{ds_1, ds_2, \ldots\}$, a number of user-required results k, a regularity threshold σ_r and a size of window w. The top-*k* frequent-regular itemsets mining is to discover k itemsets having highest support and regular occurrence in the latest window.

III. Proposed method

In this section, we here describe the concept of the bit-vector used for maintaining occurrence information of each item/itemset and then introduce all details of *TFRIM-DS* algorithm. Lastly, for the sake of understanding, an example of *TFRIM-DS* on a data stream is given, respectively.

A. Bit-vector and the technique for reusing on bits of bit-vector

In itemset mining, a bit-vector is commonly used for maintaining occurrence information of each item/itemset. For an itemset X, its bit-vector is a sequence of w bits (of window size w) used for maintaining tid of transactions in the window containing X, denoted as $BV^X = \{b_1, b_2, \ldots, b_w\}$. Each bit $b_j \in BV^X$ indicates the occurrence of X in the j^{th} transaction of the current considered window where b_j is 1 if X occur in the j^{th} transaction, otherwise, b_j is 0, respectively. For example, from the transactional database of ds_1 containing 20 transactions as in Fig. 1 and the size of window is set to be 16, the gap of occurrence of each item to be considered is $t_5, t_6, \ldots, t_{20}$. Based on the above gap, the item 'a' occurs in transactions $t_5, t_6, t_7, t_9, t_{12}, t_{13}, t_{15}$ and t_{16}, respectively. Then, the bit-vector of 'a' can be defined as $BV^a = \{174, 153\}$ (can be viewed as bytes of binary numerical system as $\{11100001, 10110000\}$).

ds1			
tid	set of items	tid	set of items
1	a, e	11	e
2	d, e, f	12	a, b, d
3	a, d	13	a, b, c, d
4	c, e, f	14	e, f
5	a, b, d, e	15	a, b, d, f
6	a, b, d	16	a, b, c, d
7	a, b, c	17	e
8	e	18	d, f
9	a, b, d, f	19	c
10	b, c, d	20	b, c, d, e

ds2	
tid	set of items
1	a, c, e, f
2	d, e, f
3	c, f
4	a, c, f
5	a, c, e, f

Fig. 1: A data stream $DS = \{ds_1, ds_2\}$ containing two batches

With the new coming data stream ds_d, some of first bits in bit-vector of an itemset will be removed and new bits for the new coming transactions will be inserted. For example, with the new coming of ds_2 containing five transactions, the gap of transactions to be considered will change to be $t_{10}, t_{11}, \ldots, t_{20}$ of ds_1 followed by $t_1, t_2 \ldots, t_5$ of ds_2. Therefore, for the item 'a', the first five bits *i.e.* '11100' for its occurrence between transactions $t_5 - t_9$ of ds_1 is removed from BV^a. In addition, the new five bits *i.e.* '10011' for its occurrence between transactions $t_1 - t_5$ of ds_2 is inserted at tail of BV^a. However, to reduce the cost of removing and insertion of bits in bit vectors, the reusing method is designed and applied. With the reusing method, the first five bits of '11100' is replaced by the new five bits '10011'. Thus, the BV^a is updated to be $\{153, 176\}$ (*i.e.* $\{10011001, 10110000\}$ in binary numerical system) in which the first five bits is for the occurrence of 'a' in $t_1, t_2, \ldots, t_5$ of ds_2 and the remaining bits is for the occurrence of 'a' in $t_{10}, t_{11}, \ldots, t_{20}$ of ds_1, respectively.

B. TFRIM-DS algorithm

As shown in Fig. 2, transactions of ds_1 in the current considered window is sequentially read in order to maintain occurrence information of each item (called this step as *Updating-itemList*). Then, the step of *Mining-results* is executed to find the complete set of results from the current window and then maintained in the top-*k* list. On the other hand, for the next batch, the out-of-scope occurrence information (*i.e.* the occurrence in transactions that are not in the considered window) of items in the *itemList* and top-*k* list are then eliminated (also called this step as Eliminating-occurrence-information). Then, the *Updating-itemList* and the *Mining-results* are then performed.

Algo. 1 gives all details on *Updating-itemList* step. For the first data stream ds_1, a simple list called *itemList* is created with entries of items to be considered. But for other data streams, it is no need to create the *itemList* since the list is already created from the first data stream. Next, the number of transactions in the new coming data stream ds_d (*i.e.* $|ds_d|$) is checked up with the user-given window size w. If $|ds_d|$ is not greater than w, the *Updating-itemList* will start to consider from the first transaction of ds_d. Otherwise, it will start to at the p^{th} transaction (*i.e.* $p = |ds_d| - w + 1$) of ds_d. Each transaction $t_p \in ds_d$ is sequentially read and each item $i_j \in t_p$ is considered. The support s^{i_j} is increased by 1. The regularity r^{i_j} is updated by $r^{i_j} \leftarrow max(r^{i_j}, q - lo^{i_j})$. The bit-vector BV^{i_j} and the last occurrence lo^{i_j} are updated by tid p of t_p. After scanning all transactions, each item $i_j \in$ *itemList* is then considered again. The regularity r^{i_j} is thus updated by $r^{i_j} \leftarrow max(r^{i_j}, |ds_d| - lo^{i_j})$. Last, if r^{i_j} is greater than the user-given regularity threshold σ_r, the item i_j is thus identified and marked as an irregular item. Then, i_j and all of its supersets are then ignored from the computation. Last, items in the *itemList* is sorted by descending of support.

The "*Mining results*" is executed to mine the complete set of top-*k* frequent-regular itemsets (see Algo. 2). The *top-k list* is first initialized. Then, the first k entries of the *itemList* (*i.e.* the k items having highest support and regular occurrence) are copied and inserted into the *top-k list* by support descending order. The support s^k is set to be equal to the support s^K (where the itemset K is the k^{th} itemset in the *top-k list*)

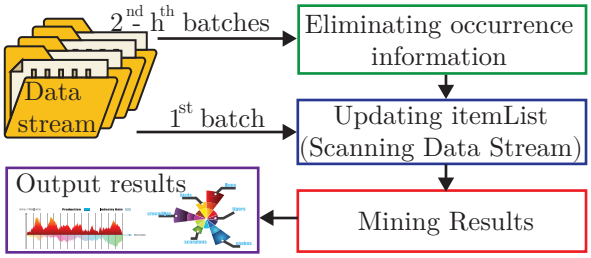


Fig. 2: Process of mining top-k frequent-regular itemsets from data streams

Algorithm 1 *Updating-itemList*

Input: $ds_d = \{t_1, t_2, \ldots, t_{|ds_d|}\}$, σ_r, w, k
Output: A list named *itemList* containing all of single items with their occurrence information in the current window

create *itemList*
create an entry for each item $i_j \in I$ in the *itemList*
if $|ds_d| \leq w$ **then**
  $p = 1$
else
  $p = |ds_d| - w + 1$
for each transaction $t_q \in ds_d$ (start from t_p to $t_{|ds_d|}$) **do**
  for each item $i_j \in t_q$ **do**
    $s^{i_j} \leftarrow s^{i_j} + 1$
    $r^{i_j} \leftarrow max(r^{i_j}, q - lo^{i_j})$
    $BVR^{i_j} \leftarrow BVR^{i_j} \cup q$
    $lo^{i_j} \leftarrow q$
for each item $i_j \in$ *itemList* **do**
  $r^{i_j} \leftarrow max(r^{i_j}, |ds_d| - lo^{i_j})$
  if $r^{i_j} > \sigma_r$ **then**
    mark i_j as an irregular item
sort *itemList* by support descending order

Algorithm 2 *Mining-results*

Input: *itemList*, k, σ_r
Output: *top-k list*

top-k list $= i_1, i_2, \ldots, i_k$ where $i_1, i_2, \ldots, i_k \in$ *itemList*
$s^k \leftarrow s^K$ where K is the k^{th} itemset in the *top-k list*
for each entry of $X = \{i_a, \ldots, i_b, i_c\}$ in *top-k list* **do**
  for each entry of $Y = \{i_d, \ldots, i_e, i_f\}$ in *top-k list* **do**
    if $i_a, \ldots, i_b = i_d, \ldots, i_e$ **then**
      $Z \leftarrow X \cup Y$
      $BV^Z \leftarrow BV^X \cap BV^Y$
      $s^Z \leftarrow$ *support*(BV^Z) and $r^Z \leftarrow$ *regularity*(BV^Z)
      if $r^Z \leq \sigma_r$ and $s^Z \geq s^k$ **then**
        top-k list $\leftarrow$ *top-k list* $- K$
        top-k list $\leftarrow$ *top-k list* $+ Z$
        $s^k \leftarrow s^K$ where K is the current k^{th} itemset in the *top-k list*

Algorithm 3 *Eliminating-occurrence-information*

Input: $ds_d = \{t_1, t_2, \ldots, t_{|ds_d|}\}$, σ_r, w, k, *top-k list*, *itemList*
Output: *itemList*, *top-k list*

top-k list $\leftarrow \emptyset$
for each irregular item $i_j \in$ *itemList* **do**
  unmark i_j
if $|ds_d| \geq w$ **then**
  for each item $i_j \in$ *itemList* **do**
    reset all bits in BV^{i_j} to be 0
else
  for each item $i_j \in$ *itemList* **do**
    reset the first $|ds_d|$ bits in BV^{i_j} to be 0
    $s^{i_j} \leftarrow$ *support*(BV^{i_j}) and $r^{i_j} \leftarrow$ *regularity*(BV^{i_j})
    if $r^{i_j} > \sigma_r$ **then**
      mark i_j as an irregular item

which can act as a support threshold to filter out candidate itemsets that cannot be the k itemsets with highest support and regular occurrence (Note that if the *top-k list* contain less than k itemsets, the support s^k is set to be 0). Next, an entry with an itemset $X = \{i_a, \ldots, i_b, i_c\}$ in *top-k list* is considered and merged with another itemset $Y = \{i_d, \ldots, i_e, i_f\}$ having the same prefix (*i.e.* $i_a, \ldots, i_b = i_d, \ldots, i_e$) in order to generate to be the itemset $Z \leftarrow X \cup Y$. Then, the bit-vector BV^X and BV^Y are then intersected and collected into BV^Z. The support s^Z and the regularity r^Z are thus calculated from the BV^Z by looking-up the look-up table (as in [14]). If the support s^Z is greater than the support s^k and the regularity r^Z is not greater than the regularity threshold σ_r, the itemset Z is thus identified as a top-k frequent-regular itemset. Then, the itemset K (*i.e.* the k^{th} itemset with lowest support in the set of the top-k frequent-regular itemsets) is eliminated from the top-k list. A new entry for the itemset Z (with BV^Z, s^Z, r^Z) is created and inserted into the top-k list by descending order of support. Therefore, the support s^k is re-assigned by the support of the current k^{th} itemset in the top-k list. The merging process is repeatedly performed on all pairs of itemsets in the top-k list. Last, we then get then complete set of top-k frequent-regular itemsets contained in the top-k list.

Last, whenever a new data stream ds_d (where $d \geq 2$) is coming, the process of *Eliminating-occurrence-information* is performed to unmark all irregular items (marked from the previous computation) and to remove unnecessary occurrence information in bit-vectors of items from the *itemList*. First, the *top-k list* is empty since we cannot guarantee that the top-k frequent-regular itemsets from the current window will be the top-k frequent-regular itemsets of the next window or not. Then, each irregular item $i_j \in$ *itemList* is unmarked. Then, the number of transactions of ds_d (*i.e.* $|ds_d|$) is considered. If $|ds_d|$ is more than the window size w, all of bits in each bit-vector BV^{i_j} (where $i_j \in$ *itemList*) is set to be all of 0s. Otherwise, only the first $|ds_d|$ bits is set to be all 0s and the support s^{i_j} and the regularity r^{i_j} are then recalculated. If the regularity r^{i_j} is greater than the regularity threshold σ_r, the item i_j is thus marked as an irregular item. At the end of *Eliminating-occurrence-information*, the *itemList* contains all single items with occurrence information under the scope of the window.

C. Example of TFRIM-DS

Given a data stream $DS = \{ds_1, ds_2\}$ of transactional databases at time t_1 and t_2 as in Fig. 1, the size of window $w = 16$, a regularity threshold $\sigma_r = 3$, and a value of $k = 6$. *TFRIM-DS* will mine six itemsets regularly occurring and having highest support in the latest 16 transactions of *DS*.

i	s	r	ls	BV
a	1	1	1	{128, 0}
b	1	1	1	{128, 0}
c	-	-	-	{0, 0}
d	1	1	1	{128, 0}
e	1	1	1	{128, 0}
f	-	-	-	{0, 0}

{10000000 00000000}

{00000000 00000000}

Fig. 3: *itemList* after firstly scanning of transaction t_5 of ds_1

i	s	r	ls	BV
b	11	2	16	{237, 181}
d	10	3	16	{205, 181}
a	9	3	14	{233, 180}
c	8	3	16	{37, 147}
e	6	3	16	{144, 73}
f	5	6	14	{9, 100}

{11101101 10110101}

{11001101 10110101}

{11101001 10110100}

{00100101 10010111}

{10010010 01001001}

{00001001 01100100}

Mark

Fig. 4: *itemList* after scanning transaction t_5 - t_{20} of ds_1

i	s	r	ls	BV
b	11	2	16	{237, 181}
d	10	3	16	{205, 181}
a	9	3	14	{233, 180}
c	8	3	16	{37, 147}
e	6	3	16	{144, 73}
f	5	6	14	{9, 100}

Top-k list

b → d → a

c ← e

Mark

Fig. 5: Creating Top-k list

Top-k list

b: s = 11 , r = 2, BV = {237, 181}

d: s = 10 , r = 3, BV = {205, 181}

bd: s = 10 , r = 3, BV = {205, 181}

a: s = 9 , r = 3, BV = {233, 180}

e: s = 6 , r = 3, BV = {144, 73}

c: s = 8 , r = 3, BV = {37, 147}

Fig. 6: Merging of 'b' and 'd' adn insertion of 'bd'

Top-k list

b: s = 11 , r = 2, BV = {237, 181}

d: s = 10 , r = 3, BV = {205, 181}

bd: s = 10 , r = 3, BV = {205, 181}

a: s = 9 , r = 3, BV = {233, 180}

ba: s = 9 , r = 3, BV = {233,180}

bda: s = 8 , r = 3, BV = {201, 180}

Fig. 7: Top-k list after mining of $t_5 - t_{20}$ of ds_1

i	s	r	ls	BV
b	7	2	16	{5, 181}
d	7	2	16	{2, 181}
a	5	3	14	{1, 180}
c	5	3	16	{4, 147}
e	4	3	16	{2, 73}
f	4	3	14	{1, 100}

{00000101 10110101}

{00000101 10110101}

{00000001 10110100}

{00000100 10010011}

{00000010 01001001}

{00000001 01100100}

Fig. 8: *itemList* after updating the first five bits of the first byte in bit-vectors with '00000'

Top-k list

c: s = 9, r = 3, BV = {188, 147}

f: s = 9 , r = 3, BV = {249, 100}

d: s = 8 , r = 3, BV = {69, 181}

a: s = 8 , r = 3, BV = {129, 180}

e: s = 7 , r = 3, BV = {202,73}

Fig. 9: Top-k list after mining of $t_{10} - t_{20}$ of ds_1 and $t_1 - t_5$ of ds_2

First, the *itemList* for all of single items is created. As the number of transactions contained in ds_1 is 20 which is greater than the size of window ($w = 16$), then only latest 16 transactions of ds_1 are thus considered (*i.e.* the transactions $t_5, t_6, \ldots, t_{20}$ of ds_1). Then, the transaction $t_5 = \{a, b, d, e\}$ of ds_1 is firstly read and the item 'a', 'b', 'd' and 'e' are then considered to update its essential information in the *itemList*. For example, as shown in Fig. 3, the support s^a, s^b, s^d and s^e are set to be 1, the regularity r^a, r^b, r^d and r^e are set to be 1, the last occurrence lo^a, lo^b, lo^d and lo^e are also set to be 1 (*i.e.* these items occur in the first transaction of the window) and the bit-vector BV^a, BV^b, BV^d and BV^e are set to be $\{128, 0\}$ (these vectors can be viewed in the binary numeral system as $\{10000000, 00000000\}$). Noted that as the size of window is set to be 16, each bit-vector of each item contains 2 bytes of 8 consecutive bits. Next, each transaction of $t_6, \ldots, t_{20} \in ds_1$ is thus sequentially scanned in the same manner as above. Then, the item 'f' is thus marked as an irregular item and eliminated from mining process since its regularity r^f is greater than σ_r. Last, all items in the *itemList* are sorted in support descending order as shown in Fig. 4.

Next, the first unmarked 5 items in the top-k list, items 'b', 'd', 'a', 'c' and 'e', are thus copied and inserted into the top-k list by descending order of support (as shown in Fig. 5). Then, the item 'b' (the first item) in the top-k list is first considered and sequentially merged with items 'd', 'a', 'c' and 'e' in the top-k list. For the merging of the item 'b' and 'd' to generate itemset 'bd', the bit-vector BV^b and BV^d are thus intersected to calculate the support $s^{bd} = 10$, the regularity $r^{bd} = 3$, and to collect $BV^{bd} = \{208, 181\}$, respectively. Since the regularity r^{bd} is not greater than σ_r and the top-k list currently contains only 5 itemsets (less than the value of k), the itemset 'bd' is directly inserted into the top-k list by descending order of support (see Fig. 6). The item 'b' is thus merged with 'a', 'c' and 'e' in the same manner as above (also for all of items/itemsets in the top-k list). After considering and merging on all items and itemsets in the top-k list, we gain the complete set of top-k frequent-regular itemsets contained in the top-k list as shown in Fig. 7.

With the new coming data stream ds_2, the number of transactions in ds_1 and ds_2 to be considered is identified. Since ds_2 contains 5 transactions, then all of 5 transactions of ds_2 are then considered incorporated with latest 11 transactions of ds_1. However, before reading each transaction of ds_2, the occurrence information of each item in the *itemList* is updated by setting the first five bits in the first byte of each bit-vector to be '00000' (Noted that to save computational time, this applies the concept of reusing of bits in bit-vector instead of removing the first five bits for transactions t_5, t_6, t_7, t_8, t_9 of ds_1 and then adding new five bits for transactions t_1, t_2, t_3, t_4, t_5 of ds_2).

The support and regularity of each item in the *itemList* is also recalculated (as shown in Fig. 8).

Next, all of steps as above are thus repeated to mine the completed set of results *i.e.* (*i*) each transaction in ds_2 is sequentially read, (*ii*) irregular items are thus identified, marked and eliminated from considering, (*iii*) top-*k* list is emptied and six regular items with with highest support are copied to the top-*k* list, and (*iv*) each item in the top-*k* list is considered and merged with other items in the top-*k* list to generate complete set of top-*k* frequent-regular itemsets, respectively. After considering all of above steps, the *itemList* and the top-*k* list are then contained information of items and top-*k* frequent-regular itemsets as in Fig. 9.

IV. Experimental studies

In this section, we here report experimental studies of the proposed *TFRIM-DS* algorithm. To the best of our knowledge, our approach is the first one on mining top-*k* frequent-regular itemsets over a data stream. Thus, there is no comparative studies with previous algorithms on mining of such itemsets. However, we then conduct comparative studies between the proposed *TFRIM-DS* and the *RFPDS* algorithm [15] for mining frequent-regular itemsets from data streams using sliding window technique. Since the aim of these two algorithms is different, they then require dissimilar parameters *i.e. TFRIM-DS* needs a regularity threshold σ_r, a number of required results k and a size of window w but *RFPDS* demands a regularity threshold σ_r, a support threshold σ_s and a size of window w, respectively. Based on this difference, we then set the value of the support threshold σ_s (required by *RFPDS*) to be equal to support of the k^{th} itemset with highest support which causes these two algorithms can discover similar results.

Four well-known datasets from http://fimi.ua.ac.be/data/ with different characteristics, *i.e. Chess*, *Mushroom*, *Retail* and *T10I4D100K*, are used in our experiments. *Chess* and *Mushroom* are real dense datasets containing $3,196$ and $8,124$ transactions. Meanwhile, *Retail* is a real sparse dataset and *T10I4D100K* is a synthetic sparse dataset containing $88,162$ and $100,000$ transactions, respectively. The three parameters are set in the same manner as in [8], [13], [14], *i.e.* the number of required results k, the regularity threshold σ_r, and the size of window w are set in range $10 - 10,000$, $1\% - 15\%$ and $10\% - 80\%$ of total number of transactions, respectively.

Two experiments are conducted to investigate *TFRIM-DS*'s computational time. As in Fig. 10, the computational time of *TFRIM-DS* and *RFPDS* on a variation of regularity threshold and a fixed of window size are observed. From the figure, the runtime of both algorithms increase as the value of k (support threshold) increases. With more results to be mined (higher value of k or low support threshold), both algorithms need more time to consider and mine results. Similarly, the runtime of both algorithms also increase as the regularity threshold increases. On high regularity threshold, there is a higher chance that each item/itemset will meet the threshold and there are more and more items/itemsets to be considered. Moreover, it can be observed that *TFRIM-DS* is faster *RFPDS* in all experiments. It is because *TFRIM-DS* can aviod repeating scan data stream (expensive cost on computation) by scanning data stream once. In addition, *TFRIM-DS* uses bit-vector with a new reuse technique which can help to efficiently maintain essential information. Thus, thanks to both strategies which can help *TFRIM-DS* to save computational cost. Meanwhile, *RFPDS* need to repeatedly scan data stream whenever the data stream is updated and it does not use any efficient data structure to hold essential information. On each updating of data stream, *RFPDS* need to mine results from scratch.

Besides, Fig. 11 shows the computational time of both algorithms on a variation of window size and a fixed regularity threshold. From the figure, it can be seen that the runtime of both algorithms increase as the size of window increases. With the increasing of window size, the number of transactions to be considered is increased. This causes both algorithms need to send more time to read more transactions, to maintain ore occurrence information and to intersect more tids during mining process. In addition, *TFRIM-DS* significantly outperforms *RFPDS* in all experiments (Thanks to both strategies mentioned above). From above, we can observe that the value of regularity threshld, k, and size of window can affect to increasing/decreasing of computational time.

To investigate on memory usage of both algorithms, experiments based on a fixed value of regularity threshold to be lowest value of each dataset (Noted that as the low value of regularity threshold there are higher itemsets to be considered which can help to see higher memory consumption), a variation of window size and a variation of k are conducted. As shown in Fig. 12, we can see that the memory usage of both algorithms increase as the value of k increases. Obviously, higher value of k (higher support threshold) causes both algorithms need to maintain more candidate itemsets in memory during mining process. Similarly, as the window size w increases, the memory usage of both alogorithms are also increased. The reason is that with higher w, both algorithms need to maintain more occurrence information of each item/itemset. Last, it can be observed that *TFRIM-DS* uses less than *RFPDS* since *TFRIM-DS* can take an advantage from using of bit-vector to save memory consumption.

V. Conclusion

In this paper, we have introduced an approach for mining top-*k* frequent-regular itemsets from data streams. The sliding window technique is applied to scope a set of recent transactions to be considered. A single-pass algorithm named *TFRIM-DS* is presented to efficiently discover such itemsets. A bit-vector with a reuse of bit-vector technique is applied and designed to efficiently maintain occurrence information of each item/itemset during mining process. Experiments on real and synthetic datsets were done to show that *TFRIM-DS* can efficiently on both computational time and memory usage to mine top-*k* frequent-regular itemsets from data streams.

References

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *VLDB*, 1994, pp. 487–499.

[2] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 2000, pp. 1–12.

[3] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New algorithms for fast discovery of association rules," in *Proceedings of the 3rd Int. Conf. on Knowledge Discover and Data Mining*, 1997, pp. 283–296.

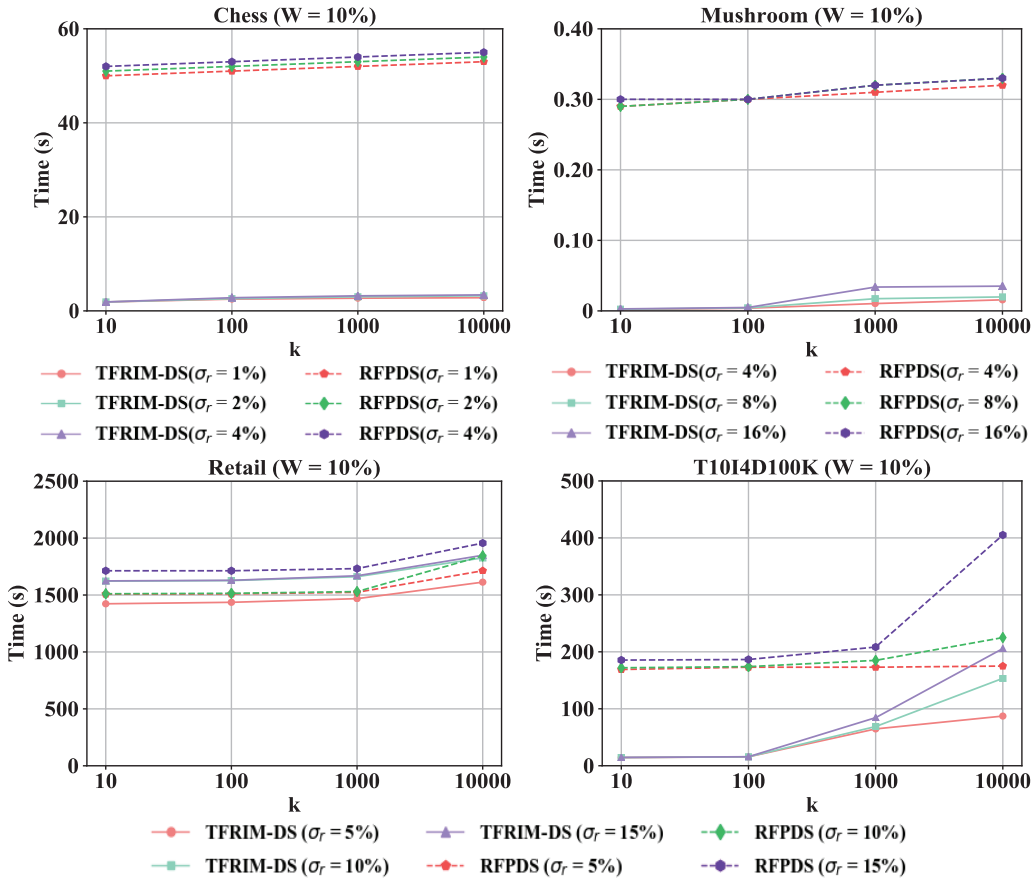


Fig. 10: Computational time of *TFRIM-DS* and *RFPDS* on a varation of regularity threshold

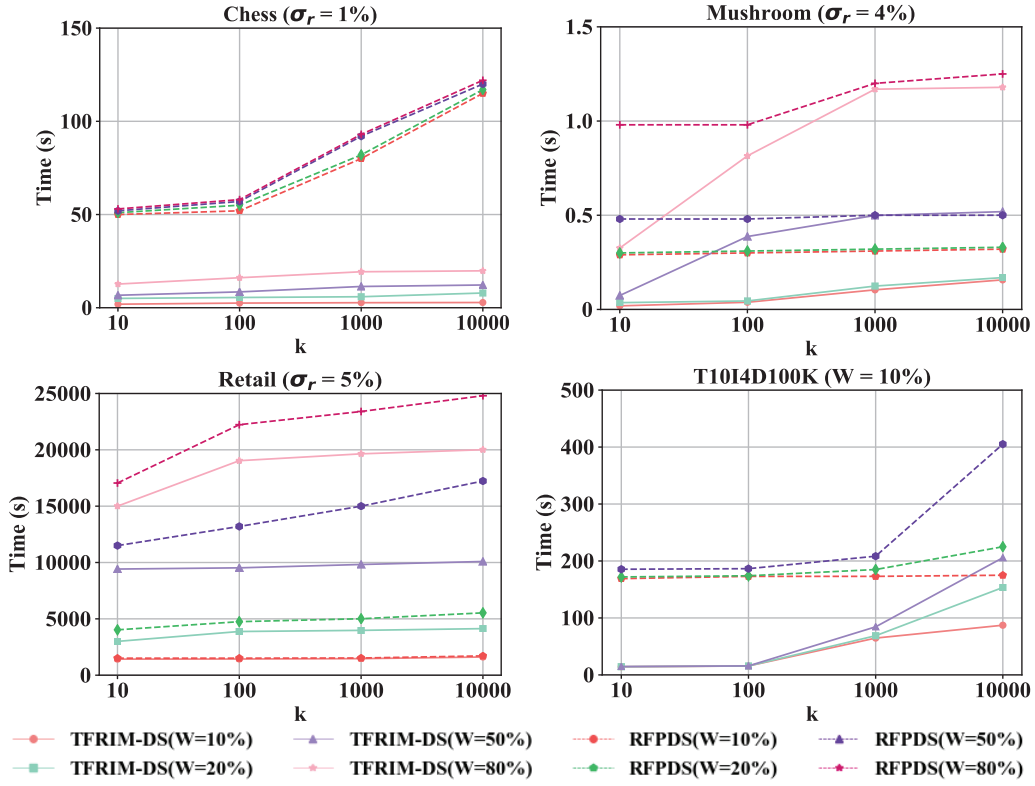


Fig. 11: Computational time of *TFRIM-DS* and *RFPDS* on a varation of window size

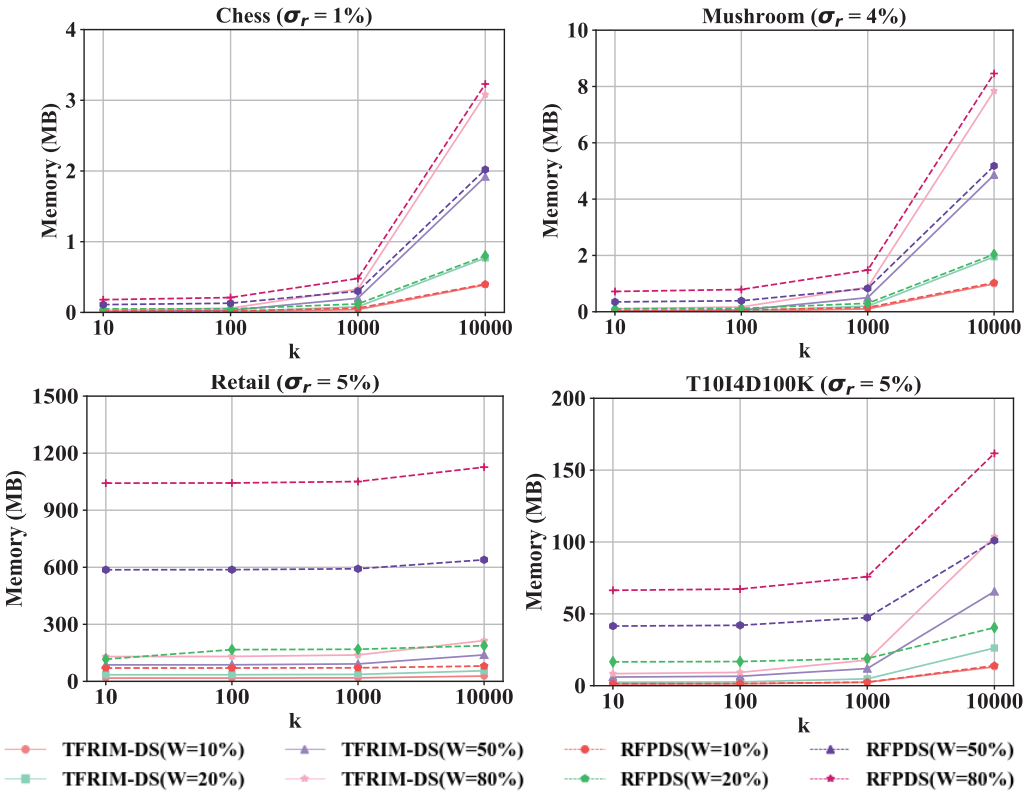


Fig. 12: Memory usage of *TFRIM-DS* and *RFPDS*

[4] S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong, and Y.-K. Lee, "Discovering periodic-frequent patterns in transactional databases," in *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, 2009, pp. 242–253.

[5] K. Amphawan, A. Surarerks, and P. Lenca, "Mining periodic-frequent itemsets with approximate periodicity using interval transaction-ids list tree," in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, January 9-10, Phuket, Thailand*. IEEE Computer Society, 2010, pp. 245–248.

[6] A. Amir and A. Levy, "Approximate period detection and correction," in *Proceedings of the 19th International Symposium on String Processing and Information Retrieval, October 21-25, Cartagena de Indias, Colombia*, ser. Lecture Notes in Computer Science, vol. 7608. Springer, 2012, pp. 1–15.

[7] S. K. Tanbeer, C. F. Ahmed, and B.-S. Jeong, "Mining regular patterns in incremental transactional databases," in *Proceedings of the 12th International Asia-Pacific Web Conference, April 6-8, Buscan, Korea*, 2010, pp. 375–377.

[8] ——, "Mining regular patterns in data streams," in *Proceedings of the 15th International Conference on Database Systems for Advanced Applications, April 1-4, Tsukuba, Japan*, ser. Lecture Notes in Computer Science, vol. 5981, 2010, pp. 399–413.

[9] M. Sreedevi and L. Reddy, "Mining regular closed patterns in transactional databases," in *Proceedings of the 7th International Conference on Intelligent Systems and Control, Jan 4-5, Coimbatore, India*, 2013, pp. 380–383.

[10] R. Kiran and M. Kitsuregawa, "Discovering chronic-frequent patterns in transactional databases," in *Databases in Networked Information Systems*, ser. Lecture Notes in Computer Science, 2015, vol. 8999, pp. 12–26.

[11] K. Amphawan, P. Lenca, A. Jitpattanakul, and A. Surarerks, "Mining high utility itemsets with regular occurrence," *Journal of ICT Research and Applications*, vol. 10, no. 2, pp. 153–176, 2016.

[12] K. Amphawan, P. Lenca, and A. Surarerks, "Mining top-k periodic-frequent patterns without support threshold," in *Proceedings of the 3rd International Conference on Advances in Information Technology*, vol. 55, 2009, pp. 18–29.

[13] K. Amphawan and P. Lenca, "Mining top-k frequent/regular patterns based on user-given trade-off between frequency and regularity," in *Advances in Information Technology*, ser. Communications in Computer and Information Science, 2013, vol. 409, pp. 1–12.

[14] ——, "Mining top-k frequent-regular closed patterns," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7882–7894, 2015.

[15] G. V. Kumar and V. V. Kumari, "Sliding window technique to mine regular frequent patterns in data streams using vertical format," in *2012 IEEE International Conference on Computational Intelligence and Computing Research*, 2012, pp. 1–4.

Gmail **Tashinee Mesama <t.mesama@gmail.com>**

Fwd: ICAICTA-2018 notification for paper 74

1 ข้อความ

Komate Amphawan <komate@gmail.com> 10 กรกฎาคม 2561 15:34
ถึง: Tashinee Mesama <t.mesama@gmail.com>

Komate Amphawan

Computational Innovation Laboratory (CIL)
Faculty of Informatics, Burapha University
Chonburi, 20131, Thailand

---------- Forwarded message ----------
From: **ICAICTA-2018** <icaicta2018@easychair.org>
Date: Tue, Jul 10, 2018 at 3:32 PM
Subject: ICAICTA-2018 notification for paper 74
To: Komate Amphawan <komate@gmail.com>

Dear Komate Amphawan

We are pleased to inform you that your paper with paper id 74 entitled Mining top-k frequent-regular itemsets from data streams based on sliding window technique has been accepted with Major Revision for “Oral Presentation” at the 2018 – 5th International Conference on Advance Informatics: Concepts, Theory and Applications (ICAICTA-2018) which will be held on August 14-17, 2018 at Beyond Hotels & Resorts, Krabi, Thailand. Therefore, we cordially invite you to attend ICAICTA-2018 for presenting your paper.

Reviewers' comments are listed at the end of this message. It is strongly recommended to the author to revise the paper following reviewer's comments in camera-ready submission.
You MUST strictly adhere to the following requirements:
1. At least one author of each paper must register as Regular no later than the final camera-ready paper submission due date. The registration rates are as follows: http://services.informatics.buu.ac.th/icaicta2018/registration.php
2. Your paper have been accepted with a condition that the paper must be revised according to the reviewer comments and complete response to reviewer form. Failing to do so may exclude your paper from the program.
3. The final camera ready paper is due on July 25, 2018. This is a firm deadline for the production of the proceedings. Please submit your paper using your EasyChair author account via Submission Page. The page limit is 6 page(s) and is strict. You can refer to the Final Manuscript format as follows: http://services.informatics.buu.ac.th/icaicta2018/format
Please note that ICAICTA-2018 conference is technical co-sponsorship by IEEE Thailand Section. Therefore, all accepted and presented papers should have the similarity score lower than 25% to be submitted for inclusion in IEEEXplore® Digital Library. Please also be aware of self-plagiarism.

Best regards
ICAICTA 2018 Organizing Committee

---------- REVIEW 1 ----------
PAPER: 74
TITLE: Mining top-k frequent-regular itemsets from data streams based on sliding window technique
AUTHORS: Tashinee Mesama and Komate Amphawan

Relevance to the conference: 5 (Excellent)
Novelty and originality: 2 (Limited / Minor variations on a well investigated subject.)
Technical content and scientific rigour: 2 (Limited / Marginal work and simple contribution. Some flaws.)
Quality of presentation: 2 (Limited / Substantial revision work is needed.)

---------- Detailed comments ----------
This paper proposes algorithm to mine top-k frequent-regular item-set from data stream based on sliding window technique. Some experiments are performed to elaborate the behavior of the proposed algorithm in terms of computational time dan space.
The authors claim that this paper is the first paper that focus on mine top-k frequent-regular item-set from data stream based on sliding window and so the author said that the comparison can not be done with other algorithm.
But actually as far as The Reviewer know, there are lots of algorithm to mine top-k frequent-regular item-set from data stream either using sliding window technique or not. Some examples are as follows :
1. S. K. Tanbeer, C. F. Ahmed, and B.-S. Jeong, “Mining regular patterns in incremental transactional databases,” in Proceedings of the 12th International Asia-Pacific Web Conference, April 6-8, Buscan, Korea, 2010, pp. 375–377.
2. ——, “Mining regular patterns in data streams,” in Proceedings of the 15th International Conference on Database Systems for Advanced Applications, April 1-4, Tsukuba, Japan, ser. Lecture Notes in Computer Science, vol. 5981, 2010, pp. 399–413.

So it is very recommended that the author elaborate the difference between the proposed algorithm with other similar work. The experiments comparing the related work above (either using sliding window technique or not) with the proposed algorithm are also needed.

---------- REVIEW 2 ----------
PAPER: 74
TITLE: Mining top-k frequent-regular itemsets from data streams based on sliding window technique
AUTHORS: Tashinee Mesama and Komate Amphawan

Relevance to the conference: 4 (Very good)
Novelty and originality: 3 (Good / Some interesting ideas and results on a subject well investigated.)
Technical content and scientific rigour: 4 (Very good / Solid work of notable importance.)
Quality of presentation: 3 (Good / Readable, but revision is needed in some parts.)

---------- Detailed comments ----------
* avoid claiming that to the best ot the author's knowledge, there is no solution to a particular problem.
* good that example was shown but the steps are quite complicated and remains difficult to comprehend despite having the example.
* why was computational time and memory usage used to evaluate performanc? What about accuracy-related measurements?
* why wasn't a comparison with existing work done? Difficult to justify the proposed method.
* Spelling: netword
* references are out of date. What about papers from Nguyen & Nguyen (2016)?
* check language

---------- REVIEW 3 ----------
PAPER: 74
TITLE: Mining top-k frequent-regular itemsets from data streams based on sliding window technique
AUTHORS: Tashinee Mesama and Komate Amphawan

Relevance to the conference: 4 (Very good)
Novelty and originality: 3 (Good / Some interesting ideas and results on a subject well investigated.)
Technical content and scientific rigour: 3 (Good / Valid work but limited contribution.)
Quality of presentation: 3 (Good / Readable, but revision is needed in some parts.)

---------- Detailed comments ----------
This paper propose an approach to mine top-k frequent regular itemsets over data streams. Although it is claimed to be the first approach FRIM over data streams and traditional algorithms are less efficient, its efficiency (computational time and memory usage) is neccessary to be compared to traditional algorithms.

Description in subsection III.A. about bit-vector calculation is still confusing. Authors provide 20 transactions with window 16 (t5..t20), how we get BV^a={174,153} ? Since b(i) represents bit for transaction i, why there are only two elements of BV^a ? How to calculate 174 and 153? Please give better explanation in this section.

There are some typos, e.g. netword, tradditional, acheived, intertest, etc.