# Balanced Parallel Algorithm for Mining Frequent Pattern from Data Stream

**Dr. Zakria Mahrousa**[*]
**Dr. Dima Mufti Alchawafa**[**]
**Hasan Kazzaz**[***]

**(Received 10 / 5 / 2020. Accepted 14 / 6 / 2020)**

## □ ABSTRACT □

The frequent pattern mining methods play very important role to generate association rules from massive data stream such as include customer click streams, network monitoring data, etc. The continuous, unbounded and high-speed characteristics of massive data stream are a huge challenge for the current frequent pattern mining approach. The complexities related to finding frequent itemset for mining association rules from a massive data stream in this work can be minimized by using modified FP-growth algorithm and parallelizing the mining task with MapReduce technique in Hadoop framework, improves performance by using balanced load technique, which exploits correlations among transactions. In this paper, we introduce (**B**alanced **P**arallel **G**raph **F**requent **P**attern **BPGFP**-growth), a modified FP-growth with one-pass scan based on directed graph, Hadoop framework, partitioning and balancing load strategy in order to reduce the execution time for the massive dynamic database and the volume of data exchanged between computational nodes (computers). The algorithm was tested, our experimental results demonstrated that the proposed algorithm could scale well and efficiently process large dynamic datasets. In addition, it achieves improvement in memory consumption to store frequent patterns and time complexity.

**Keywords:** Association Rule Mining (ARM), Frequent itemset, FP-growth, Directed Graph, Massive Data Stream, MapReduce, Hadoop, Partitioning data.

[*]**Associate Professor - Department of Computer Engineering, Faculty of Electrical and Electronic Engineering, University of Aleppo, Aleppo, Syria.**
[**]**Assistant Professor - Department of Computer Engineering, Faculty of Electrical and Electronic Engineering, University of Aleppo, Aleppo, Syria.**
[***]**Postgraduate Student (PhD.) - Department of Computer Engineering, Faculty of Electrical and Electronic Engineering, University of Aleppo, Aleppo, Syria.**

# خوارزمية متوازنة تفرعية للتنقيب عن النماذج المتكررة في تيار بيانات مستمرة

**د. زكريا مهروسة***

**د. ديما مفتي الشوافعة****

**حسن قزاز*****

**(تاريخ الإيداع 10 / 5 / 2020. قُبِل للنشر في 14/ 6 / 2020)**

## □ ملخّص □

تلعب خوارزميات التنقيب عن العناصر المتكررة (Frequent Itemset) دوراً هاماً في إيجاد قواعد الترابط (Association Rules) من تيار بيانات مستمرة (Data Stream) مثل: البيانات الناتجة عن تتبع سلوك الزبائن، ومراقبة الشبكات، إلخ. تشكل الطبيعة المستمرة وغير المحدودة والسرعة العالية لتيار البيانات تحدياً كبيراً للعديد من الخوارزميات الحالية في مجال التنقيب عن النماذج المتكررة. بهدف تخفيض درجة تعقيد عملية إيجاد العناصر المتكررة من تيار بيانات مستمرة نقترح في هذا البحث تطوير خوارزمية FP–growth وتوزيع عملية التنقيب من خلال البنية MapReduce على أكثر من حاسب في الإطار هادوب Hadoop وباستخدام طريقة فعّالة من أجل موازنة الأحمال بين العقد الحسابية، وإيجاد الترابط بين مداولات (Transactions) قاعدة البيانات. حيث تم اقتراح خوارزمية (**B**alanced **P**arallel **G**raph **F**requent **P**attern **BPGFP**–growth) وهي خوارزمية مُطورة عن FP–growth تقوم بمسح البيانات لمرة واحدة فقط، وتعتمد على الغراف الموجه (Directed Graph) والهادوب وطريقة لموازنة وتقسيم البيانات من أجل تخفيض الزمن اللازم لإيجاد العناصر المتكررة وحجم البيانات المتبادلة بين العقد الحسابية. تم اختبار الخوارزمية المقترحة على قواعد بيانات قياسية، وأثبتت النتائج قدرة الخوارزمية على القيام بعملية التنقيب في قواعد البيانات المتغيرة. وتخفيض كبير في معدل استهلاك الذاكرة، بالإضافة إلى تخفيض التعقيد بالنسبة إلى الزمن.

**الكلمات المفتاحية:** التنقيب عن قواعد الترابط، مجموعات العناصر المتكررة ، FP–growth، تيار البيانات المستمرة، الغراف الموجه، البيانات الضخمة، MapReduce، Hadoop، تقسيم البيانات.

* أستاذ مساعد – قسم هندسة الحواسيب، كلية الهندسة الكهربائية والالكترونية، جامعة حلب، حلب، سورية.

** مدرس – قسم هندسة الحواسيب، كلية الهندسة الكهربائية والالكترونية، جامعة حلب، حلب، سورية.

*** طالب دراسات عليا (دكتوراه) – قسم هندسة الحواسيب، كلية الهندسة الكهربائية والالكترونية، جامعة حلب، حلب، سورية.

## مقدمة:

تعتبر عملية إيجاد العناصر المتكررة من أجل التنقيب عن قواعد الترابط، مرحلة أساسية في عملية تحليل البيانات الضخمة المستمرة والناتجة عن العديد من المصادر مثل شبكات التواصل الاجتماعي والتجارة الالكترونية، إلخ. يتم إيجاد مجموعة العناصر المتكررة من خلال دراسة مداولات تيار البيانات والتي تكون على شكل مجموعة من الأحداث غير المتكررة والمترابطة مع بعضها البعض[1]. صممت البنية البرمجية التفرعية الموزعة MapReduce من قبل Google، حيث تقوم تقنية Map بتقسيم البيانات إلى أجزاء (مفتاح Key / قيمة Value). ومن ثم ترسل مخرجات Map إلى مرحلة Shuffle لترتيبها وارسالها الى تقنية Reduce لتجميعها[2]. أما هادوب Hadoop فهو عبارة عن منصة برمجية مفتوحة المصدر مكتوبة بلغة Java تستخدم البنية MapReduce. ويستخدم بشكل واسع في معالجة وتحليل قواعد البيانات الضخمة الموزعة عبر تجمعات من الحواسيب(عنقودCluster)[3][4]. قدمت العديد من الخوارزميات التي تقوم بالتنقيب عن مجموعات العناصر المتكررة في قواعد البيانات الثابتة (Static Datasets)، والتي تتميز بحجمها المحدود والثابت مع الزمن، على عكس قواعد البيانات المتغيرة (Dynamic Datasets) والتي تتصف بمايلي: الأستمرارية (Continuity): أي تدفق تيار البيانات بمعدل مرتفع وبشكل متواصل، والصلاحية (Expiration): ويقصد بها قراءة تيار البيانات لمرة واحدة فقط، واللامحدودية (Infinity): أي الحجم غير الثابت لتيار البيانات[5] [6]. لتصميم خوارزمية قادرة على التعامل مع تيار البيانات، لابد من الأخذ بعين الاعتبار عدد من النقاط. أولاً: التخلص من المسح المتكرر للبيانات. ثانياً: التكيف مع المعدل المرتفع والمتغيّر لتدفق البيانات[5]. ثالثاً: يجب أن تتوافق نتائج خوارزمية التنقيب مع التغيّر المستمر الحاصل في تيار البيانات مع الأخذ بعين الاعتبار نتائج عمليات التنقيب الخاصة بالبيانات السابقة لاستنتاج العناصر أو النماذج المتكررة للبيانات الحالية أي عملية معالجة تدريجية (Incremental Process) [7].

## أهمية البحث وأهدافه:

الهدف الأساسي لهذا البحث تطوير خوارزمية FP-growth لتحسين أدائها وإيجاد حلّ للسلبيات الناتجة من عملية التنقيب عن العناصر المتكررة وهي الزمن والمساحة التخزينية الكبيرة وتمكينها من التعامل مع تيار بيانات مستمرة. حيث سيتم الاعتماد على استخدام منصة هادوب لتصميم خوارزمية تفرعية BPGFP-growth، تقوم بمسح وحيد لقاعدة البيانات، وتعتمد على الغراف الموجه لضغط قاعدة البيانات بشكل كبير من أجل توفير الزمن اللازم لعملية التنقيب والمساحة التخزينية المطلوبة، والعمل على اقتراح استراتيجية لموازنة وتقسيم الأحمال بين العقد الحسابية بهدف تخفيض كلفة التواصل بينها ومعدل عمليات الدخل والخرج.

**1- إيجاد العناصر المتكررة في تيار بيانات Finding frequent itemset in data stream**: يُعرف تيار البيانات المستمرة بأنه عبارة عن سلسلة من البيانات المرتبة زمنياً، والتي تتدفق بمعدل مرتفع ومستمر[8]. توصف عملية إيجاد العناصر المتكررة في تيار البيانات كمايلي: إذا كان لدينا مجموعة من العناصر $I=\{I_1,I_2,I_3\ldots I_n\}$ حيث $n$: تمثل العدد الكلي للعناصر، وإذا كان لدينا مجموعة من المداولات $SD=\{T_1,T_2,\ldots,T_m,..\}$ علماً أن $T_i$ $(i \in [1..m..])$، حيث تمثل $m$: عدد المداولات الحالي، بفرض أن كل مداولة تحوي مجموعة من العناصر المنتمية إلى $I$ فإن المجموعة الجزئية $X \subseteq I$ تُسمى بمجموعة عناصر(Itemset) أو نموذج(Pattern)[1]. كما يسمى النموذج $X$ بـ

k–itemset أي يحوي $K$ عنصر يتكرروا معاً في المداولات. وتعرف قيمة الدعم للنموذج $X$ والتي يرمز لها بـ $sup(X)$ بأنها عدد المداولات التي تتضمن $X$. ونقول عن $X$ أنه عنصر مكرر إذا وفقط إذا كان الدعم له $sup(X)$ أكبر أو يساوي الدعم الأصغري (قيمة عددية أكبر من الصفر معرفة من قبل المستخدم ويرمز لها بـ min–sup). على سبيل المثال إذا كان $X$ يحوي عنصرين $I_1$، $I_2$ وكانت قيمة الدعم له4 فهذا يعني أن العنصرين قد تكرر وجودهما معاً في4 مداولات. أما قواعد الترابط فتأخذ الشكل $X \Rightarrow Y$، حيث إن $X \subseteq I$ ، $Y \subseteq I$، $X \neq \emptyset$، $Y \neq \emptyset$، $X \cap Y \neq \emptyset$. ومقدار الثقة (Confidence) لأي قاعدة هو $\frac{sup(X \cup Y)}{sup(X)}$ . والمشكلة الأساسية في عملية التنقيب عن قواعد الترابط، هو إيجاد جميع القواعد التي تمتلك قيمة دعم أكبر من قيمة الدعم الأصغري ومقدار ثقة أكبر من عتبة الثقة الأصغرية أيضاً[9].

**2–الدراسة المرجعية والأعمال السابقة Related work:** هناك عدة خوارزميات تختص بإيجاد النماذج المتكررة، في هذه الفقرة سوف نناقش الخوارزميات الأساسية للتنقيب عن العناصر المتكررة، ثم الخوارزميات الخاصة بالبيانات الضخمة.

**2–1 الخوارزميات الأساسية للتنقيب عن البيانات:** يمكن تقسيم الخوارزميات الأساسية التسلسلية للتنقيب على حسب طبيعة عملها إلى قسمين أساسيين:

**الخوارزميات المعتمدة على توليد العناصر المرشحة:** مثل خوارزمية Apriori والتي تستخدم طريقة التوليد والاختبار (Generate and Test) أي تولد العناصر المرشحة ثم تختبر فيما إذا كانت تمثل تكراراً. تعتمد هذه الخوارزمية استراتيجية البحث بالعرض (Breadth First Search) لذلك تعتبر مكلفة من حيث الزمن والمساحة التخزينية [9].

**الخوارزميات الغير المعتمدة على توليد العناصر المرشحة:** مثل خوارزمية FP–growth والتي تعتبر من أفضل وأسرع الخوارزميات في مجال إيجاد قواعد الترابط، تعتمد على فرضية فرّق تسد (Divide and Conquer) وتتمتع بالمزايا التالية: يتم فيها مسح قاعدة البيانات لمرتين فقط، كما يتم تحويل قاعدة البيانات إلى بنية شجرية FP tree حجمها دائماً أصغر من حجم قاعدة البيانات الأصلية[10] [11] . كل الخوارزميات السابقة تعاني من فشل عملية التنقيب عند التعامل مع قواعد البيانات الضخمة. بسبب الزيادة الكبيرة في المساحة الذاكرية اللازمة لعملية المعالجة، والمسح المتكرر لقاعدة البيانات.

**2–2 الخوارزميات الخاصة بالتنقيب عن البيانات الضخمة:** يوجد العديد من الخوارزميات القادرة على التعامل مع البيانات المتغيرة الضخمة أو تيار البيانات. من أوائل الخوارزميات في هذا المجال، الخوارزمية التسلسلية (معتمدة على عقدة حسابية واحدة) المقترحة في البحث[12]، حيث تم اعتماد بنية معطيات مضغوطة ديناميكية للتنقيب عن النماذج المتكررة في قاعدة بيانات متغيرة. قام (Tanbeer. et al.) بتقديم خوارزمية للتنقيب عن مجموعات العناصر المتكررة بالاعتماد على مفهوم النافذة المنزلقة(Sliding Window)[13]. أدى التطور السريع لشبكة الانترنت، والزيادة المضطردة في أجهزة الهاتف المحمول إلى زيادة غير مسبوقة في حجم البيانات، مما يعني فشل الخوارزميات السابقة التسلسلية في التنقيب عن النماذج المتكررة. لذلك تم التوجه إلى استخدام البنى البرمجية الموزعة، من أجل تصميم خوارزميات للتنقيب عن النماذج المتكررة. كما عملت العديد من الأبحاث على معالجة مشكلة التوازن في الأحمال بين العقد الحسابية في بيئة العمل الموزعة، إن مشكلة التوازن في الأحمال يقصد بها قيام بعض العقد الحسابية بإيجاد النماذج المتكررة للعناصر ذات التكرار المرتفع وبالتالي زيادة زمن عملية التنقيب بشكل كبير في بعض العقد، مقابل انخفاضها في عقد حسابية أخرى. وتنتج هذه المشكلة نتيجة الطريقة المتبعة في تقسيم العناصر المتكررة بين العقد الحسابية. الخوارزمية المقدمة في البحث [14]، هي عبارة عن خوارزمية Apriori موزعة تعتمد على البنية MapReduce والمسح المتكرر لقاعدة البيانات لتوليد مجموعة العناصر المتكررة. أما الخوارزمية المقترحة في البحث

[15] والتي اعتمدت على تفريع خوارزمية FP-growth من خلال البنية MapReduce، من أجل بناء شجرة FP-tree خاصة بكل عقدة حسابية. حيث تتألف الخوارزمية السابقة من ثلاث مراحل تفرعية (MapReduce) ومرحلة تسلسلية. في المرحلة الأولى والثانية يتم تقسيم قاعدة المعطيات على عدة عقد حسابية، وحساب تكرار كل عنصر بشكل تفرعي لتخزينها في قائمة تسمى F-List. أما في المرحلة الثالثة والتي تنفذ على عقدة حسابية واحدة يتم تجميع القوائم F-List لكل العقد الحسابية في قائمة I-List مرتبة تنازلياً. ومن ثم تُقسم حسب عدد العقد الحسابية إلى مجموعات، تحوي كل منها عدد متساوي من العناصر تسمى بـ G-List، ويتم تقسيم المداولات بناءً عليها. لاتأخذ هذه الخوارزمية مفهوم ترابط العناصر أثناء عملية التقسيم ضمن نفس المجموعة بعين الأعتبار، مما يؤدي الى زيادة حجم البيانات المتبادلة بين العقد الحسابية، بالإضافة الى مشكلة عدم التوازن في الأحمال بين العقد الحسابية (Imbalanced load). في المرحلة الرابعة والخامسة يتم تطبيق خوارزمية FP-growth وتجميع العناصر المتكررة الناتجة عن كل عقدة حسابية. قام Zhou et al بتحليل مشكلة عدم توازن الأحمال بين العقد في الخوارزمية السابقة PFP، وقدم خوارزمية تتألف من مرحلتين تفرعيتين ومرحلة تسلسلية واحدة، بالإضافة الى طريقة من أجل تقسيم القائمة I-list ترتكز على فرضيتين: الأولى يتم حساب عدد عمليات التكرار التعاودية خلال تنفيذ خوارزمية FP-growth لإيجاد شجرة النموذج الشرطية (Conditional pattern tree)(قاعدة بيانات جزئية تشكل مسارات مرتبطة في شجرة النموذج الشرطية) لكل عنصر من أجل تحديد الحمل الناتج عن هذا العنصر. أما الفرضية الثانية يتم حساب موقع كل عنصر في القائمة I-list من أجل معرفة أطول مسار للنموذج المتكرر في شجرة النموذج الشرطية[16]. لايمكن للاقتراح السابق حلّ مشكلة عدم توازن الأحمال بين العقد الحسابية عند التعامل مع تيار بيانات مستمرة، بسبب التغيير المستمر في مواقع العناصر في القائمة F-list مع وصول بيانات جديدة. الخوارزمية المقترحة في المرجع[17] عملت على تفريع خوارزمية FP-growth من خلال ثلاث مراحل تفرعية ومرحلة تسلسلية. وقامت الخوارزمية في هذا المرجع بتجميع المداولات عن طريق إيجاد درجة الترابط بينها لتجميعها ضمن تجمع واحد وبمساعدة خوارزمية مطورة عن خوارزمية k-means تسمى k-means++، وعلى الرغم من فعالية الحلّ المقترح في هذا البحث إلا أن عمليات المقارنة تستغرق زمن كبير عندما تكون قاعدة البيانات ضخمة. كل الخوارزميات السابقة تتعامل مع قواعد بيانات ضخمة ذات حجم ثابت، فتقوم بإيجاد تكرار العناصر الموجودة في قاعدة البيانات وحذف العناصر غير المتكررة من خلال مسح قاعدة البيانات مرات متعددة، وهذه الطريقة لا تناسب عملية التنقيب عن النماذج المتكررة في تيار بيانات مستمرة، حيث إن تكرار العناصر يتغير مع وصول تيار بيانات جديد. فالعناصر غير المتكررة ممكن أن تصبح متكررة والعكس صحيح[5][8]. عملت الخوارزمية BPFPMS المقترحة في البحث[8] على حلّ مشاكل الخوارزميات السابقة، وهي عبارة عن خوارزمية تفرعية مشتقة من خوارزمية FP-Growth، تعتمد على بنية معطيات ديناميكية تسمى CPS-tree تقوم بمسح قاعدة البيانات لمرة واحدة فقط، الا أن حجم الشجرة المقترحة يصبح كبيراً أيضاً عند التعامل مع تيار بيانات مستمرة[8]. كما تم اقترح طريقة لموازنة الأحمال بين العقد الحسابية، وذلك بالاعتماد على عمق هذا العنصر في شجرة CPS-tree، مع تجاهل حجم البيانات المتبادلة بين العقد.

**3-خوارزمية FP-growth الأساسية FP-growth algorithm:**

تعمل خوارزمية FP-growth الأساسية وفق الخطوات التالية:

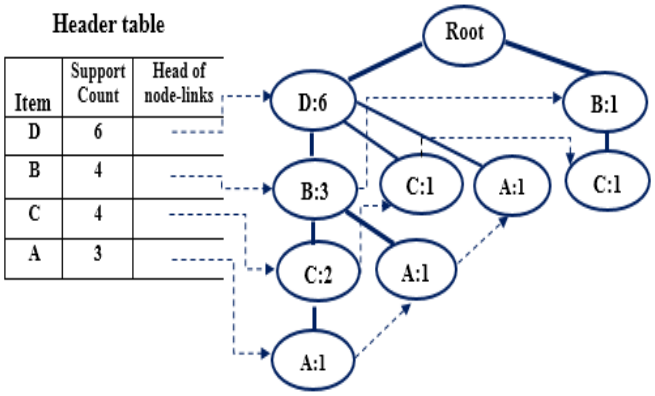
1. المرور الأول لقاعدة البيانات لإيجاد 1-itemsets (مجموعة تتضمن كل عنصر بشكل منفرد) مع تكرارها، وتخزينها في اللائحة H.

2. إيجاد العناصر المتكررة، وتُرتيبها تنازلياً بحسب قيمة تكرارها.
3. المرور الثاني على قاعدة المعطيات والذي يتضمن ترتيب عناصر كل مداولة وفق اللائحة H، والبدء ببناء شجرة النموذج المتكرر FP–tree.
4. التنقيب في شجرة FP tree [10] [11].

المثال التالي يوضح آلية عمل خوارزمية FP–growth حيث يبين الشكل(1) تيار بيانات مستمرة SD ، مع افتراض أن عتبة التكرار min–sup=3.

Time line

| TID | Transactions |
|---|---|
| $T_1$ | A, D |
| $T_2$ | A, B, D |
| $T_3$ | C, D |
| $T_4$ | A, B, C, D |
| $T_5$ | B, C |
| $T_6$ | D |
| $T_7$ | B, C, D |
| ..... | ..... |




**الشكل (1): شجرة FP–Tree.**

يوضح الشكل(1) شجرة FP–Tree التي تمثل قاعدة المعطيات السابقة. مع الأخذ بعين الأعتبار أن خوارزمية FP–growth غير قادرة على التعامل مع تيار البيانات المستمرة عن ورود بيانات جديدة بسبب المسح المتكرر للبيانات وحذف البيانات غير المتكررة.

**4–الخوارزمية المحسنةImproved algorithm:** قمنا باقتراح خوارزمية مطورة عن خوارزمية FP–growth تقوم بمسح وحيد لتيار البيانات. واستبدال بنية الشجرة بالغراف، حيث تتألف الخوارزمية المقترحة GFP–growth من مرحلتين أساسيتين:

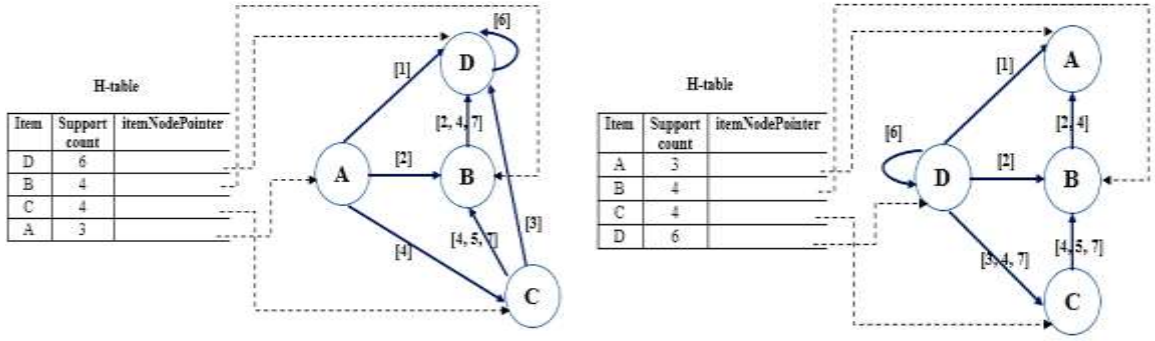
1. بناء الغراف FP–Graph.
2. التنقيب عن مجموعات العناصر أو النماذج المتكررة.

تم الاستعاضة عن شجرة FP–Tree بالغراف الموجه FP–Graph لتسريع الخوارزمية المقترحة، وتخفيض المساحة الذاكرية المطلوبة لتخزين العناصر المتكررة. تركز الخوارزمية المقترحة على تخفيض عدد عقد الغراف اللازمة لتخزين عناصر أول مجموعة (عنصر واحد 1–itemsets) ضمن مداولات قاعدة البيانات. يحوي الغراف عدد عقد مساوي لعناصر أول مجموعة (عنصر واحد 1–itemsets). كل حافة (Edge) $E_{ij}$ بين عقدتين (Vertices) $V_i$ و$V_j$ تمثل جزء من مداولة (Sub path)، ومن أجل تخزين هذا الجزء عند وروده في أكثر من مداولة تم تزويد كل حافة بقائمة TransId–list وذلك لتخزين المداولات المشتركة بتلك الحافة. يتألف كل سطر في جدول العناصر المتكررة H–table

من ثلاثة حقول، الأول مُعرف العنصر (itemId)، والثاني تكرار العنصر (Frequency count) ، أما الثالث مؤشر لعقدة هذا العنصر في الغراف (ItemRetrievalPointer). كل عقدة في الغراف تتألف من حقلين، الأول مُعرف العنصر، والثاني عبارة عن قائمة (Parent-list) والتي تضم مؤشر آباء هذه العقدة (ParentNodePointer). ذكرنا سابقاً بأن الخوارزمية المقترحة تتألف من مرحلتين، الفقرات التالية توضح المراحل السابقة الذكر بالتفصيل مع مثال توضيحي.

**4-1 بناء الغراف Construction of FP-Graph:** يُشكل الغراف بنية مضغوطة لتخزين عناصر أول مجموعة (1-itemset)، كل سطر في جدول العناصر المتكررة H-table يحوي مؤشر (ItemRetrievalPointer) لعقدة هذا العنصر في الغراف. يتم تخزين تكرار النموذج اللاحق باستخدام قائمة(Parent-list) بدلاً من حفظ تكرار العنصر ضمن العقدة لزيادة سرعة عملية التنقيب. يتم بناء الغراف FP-Graph من خلال مسح وحيد لقاعدة البيانات. حيث تقسم عملية بناء الغراف إلى:

**طور إدخال المداولات إلى الغراف Insertion phase:** حيث يتم مسح تيار البيانات، لادخال المداولات إلى الغراف بالاعتماد على ترتيب معرف مسبقاً (ترتيب هجائي للعناصر على سبيل المثال). وبشكل متوازي يتم بناء جدول H-table. وبعد الانتهاء من ادخال مداولات تيار البيانات، يتم ترتيب عناصر جدول H-table تنازلياً وفق قيم تكرارها. الشكل(a-2) يُبين آلية الإدخال في الخوارزمية المقترحة حيث يتم قراءة تيار البيانات المستمرة في الشكل(1)، وادخالها إلى الغراف بالإعتماد على ترتيب ابتدائي للعناصر في جدول H-table وهو {A, B, C, D..}. كل مداولات قاعدة المعطيات يتم تمثيلها في الغراف من خلال Parent-list والتي تحوي مؤشرات للعقد الآباء لهذه العقدة ParentNodePointer (حيث تمثل عقدة عنصر ما $I_i$ عقدة أب لعنصر آخر $I_j$ إذا كان تكراره ضمن H-table أكبر أو يساوي تكرار العنصر$I_j$ ويتواجدان ضمن نفس المداولة)، بالإضافة إلى TransId-list التي تحوي أرقام المداولات وتستخدم لتعليم (Tagging) الحواف بين عقد الغراف. عدد عقد الغراف يساوي عدد العناصر المتكررة والمكونة من عنصر واحد (frequent 1-itemsets). تحوي قاعدة البيانات السابقة على 4 عناصر وبالتالي مخطط الغراف يحوي أربع عقد كما هو موضح في الشكل(a-2). المداولة الأولى (A, D) ذات المُعرف أو الرقم $T_1$، تخزن ضمن الغراف، حيث إن العقدة D تشير إلى العقدة الأب A ويتم تعليم الحافة بينهما بـ $T_1$، أما مؤشر ParentNodePointer الخاص بالعقدة A يساوي null. المداولة الثانية (A, B, D) ذات المُعرف $T_2$، تخزن ضمن الغراف، حيث إن العقدة D تشير إلى العقدة الأب B ويتم تعليم الحافة بينهما بـ $T_2$، العقدة B تشير إلى العقدة الأب A ويتم تعليم الحافة بينهما بـ $T_2$، أما مؤشر ParentNodePointer الخاص بالعقدة A يساوي null. وهكذا.




**(a): طور ادخال المداولات الى الغراف.** **(b): طور إعادة بناء الغراف.**

**الشكل(2): مراحل بناء مخطط الغراف FP-Graph لقاعدة البيانات السابقة.**

يتم تهيئة العقد وربطها مع المؤشر ItemRetrievalPointer: من خلال الخطوات من 1 إلى 2.3.4 في الشيفرة الزائفة الموضحة لاحقاً. وحفظ المداولات في الغراف: الخطوات من 2.4 إلى 2.7 في الشيفرة الزائفة الموضحة لاحقاً أيضا.

**Algorithm1:** Procedure of Construct FP-Graph.
**Input**: Stream dataset **SD** and initial frequent item header table **H-table**.
**Output**: **FP-Graph,** sorted **H-table .**

1. TransId = 0.// Initialize a TransId.
2. **For** each transaction T ∈ SD **do**.
2.1 ParentNodePointer = null. // Initialize a Pointer.
2.2 transId++.
2.3 Sort transactions according to a predefined order.
2.4 **For** each item i ∈ T **do**.
2.4.1 Update H-Table.// increase the item's support count by one.
2.4.2 ItemRetrievalPointer = scan H-Table and get the graph pointer.
2.4.3 **If** ItemRetrievalPointer != null **then** //Get the graph pointer from scanning H-Table.
2.4.4 return ItemRetrievalPointer.
**Else**
2.4.5 Construct new FP-Graph node.
2.4.6 Insert the ItemRetrievalPointer address into H-Table.
//If the ItemRetrievalPointer != null, the next step is modifying ParentNodePointer and TransId for tagging pattern path with TransId.
2.5 **If** Parent-list != null **then** // Item node has parent.
2.5.1 ParentNodePointer=Add Prefix Item Parent Node.
**Else**
2.5.2 Construct new Parent-list and then Add Prefix Item Parent Node in it.
2.6 Add this TransId to TransId-list.
3. Sort H-Table in frequency-descending order.
4. Return FP-Graph, sorted H-Table.

**طور إعادة بناء الغراف Restructuring phase:** بعد ترتيب عناصر جدول H–table تنازلياً في المرحلة السابقة، نقوم بإعادة بناء الغراف بالاعتماد على جدول H–table المرتب بهدف ضغط الغراف وتسريع عملية التنقيب عن مجموعات العناصر المتكررة من خلال إيجاد المسارات المشتركة بين مداولة تيار البيانات المستمرة، كما يوضح الشكل (2–b). حيث يتم إعادة ترتيب كل مسار (سلسلة من الحواف التي تملك نفس رقم المداولة) ضمن الغراف وإعادة ادخاله مرة ثانية إلى الغراف. تبدأ عملية إعادة بناء الغراف انطلاقاً من عقدة العنصر الموجود أسفل الجدول H–table، تمتلك العقدة D في الشكل(2–a) أربع مؤشرات ParentNodePointer يشيران إلى العقد A، B، C، والعقدة D نفسها. وبقيمة حافة [$T_1$]، [$T_2$]، [$T_3$, $T_4$, $T_7$]، [$T_6$] على الترتيب. مسار المداولة $T_1$ هو (A, D) وحسب جدول H–table بعد الترتيب الموضح في الشكل(2–b) يتم إعادة ترتيب المسار ويصبح (D,A) وادخاله مرة ثانية إلى الغراف. مسار المداولة $T_2$ هو (A, B, D) وحسب الجدول H–table المرتب يتم إعادة ترتيب المسار ويصبح (D,B,A) وادخاله مرة ثانية إلى الغراف وهكذا. والشيفرة الزائفة لعملية إعادة بناء الغراف.

| **Algorithm2:** Procedure of Restructuring FP-Graph.<br>**Input**: **FP-Graph** and sorted header table **H-table**.<br>**Output**: Restructured **FP-Graph.** |
|---|
| 1. **For** each path $P_i$ in FP-Graph **do**<br>1.1 **If** $P_i$ is not sorted then<br>1.2 Extract and sort $P_i$ according to sorted H-table.<br>1.3 Reinsert $P_i$ into FP-Graph.<br>2. return Restructured FP-Graph**.** |

**4-2 التنقيب عن العناصر المتكررة Frequent itemset mining:** بعد إعادة بناء الغراف بالطريقة الموضحة سابقاً، يتم البدء بالمرحلة الثانية للتنقيب عن العناصر المتكررة الموضحة في الشيفرة الزائفة لاحقاً، حيث يتم توليد العناصر المتكررة بدون الحاجة لإيجاد الشجرة الشرطية (كما في FP-growth) لزيادة سرعة الخوارزمية. يتم استخدام خوارزمية البحث من الأسفل إلى الأعلى في جدول H-table لكل العناصر التي تحقق عتبة الدعم الدنيا min-sup من أجل عبور كل عقد الغراف(الخطوة1 من من الشيفرة الزائفة التالية). لإيجاد النموذج المتكرر من خلال الانطلاق من العقدة التي تمثل هذا النموذج، ثم يتم الانتقال إلى آباء هذه العقدة حتى نصل إلى عقدة مؤشر الأب ParentNodePointer لها يساوي null. ومن أجل إيجاد النماذج المتكررة الشرطية لكل عنصر نستخدم اللائحة TransId-list. في خوارزمية التنقيب عن العناصر المتكررة، يتم الوصول إلى ParentNodePointer من القائمة Parent-list لكل عقد الغراف. ثم يتم استرجاع قيم TransId من القائمة TransId-list من أجل كل ParentNodePointer لتوليد النماذج الشرطية(الخطوات من 1.1 حتى 1.3.1 من الشيفرة الزائفة التالية). بعد توليد النماذج الشرطية CP يتم حساب قيمة الدعم لها من أجل القيام بحذف CP والتي لاتحقق عتبة الدعم الدنيا min-sup. ليتم بعدها إيجاد مجموعات النماذج المتكررة النهائية.سنقوم بالتنقيب عن مجموعات العناصر المتكررة لمخطط الغراف الموضح في الشكل(b-2)، مع افتراض أن min-sup=3.

| **Algorithm3:** Procedure of Frequent itemset mining**.**<br>**Input**: Restructured **FP-Graph, min-sup** and **H-table**.<br>**Output**: **Frequent patterns FP.** |
|---|
| 1. **For** each item from bottom H-table which have support greater than min-sup.<br>1.1 Access the FP-Graph node of that Item-id.<br>1.2 FP = {∅}. // Initialize a FP set.<br>1.3 **For** each ParentNodePointer **do**<br>1.3.1 Skip and get next ParentNodePointer, If TransId-list is empty.<br>1.3.2 **For** each TransId in TransId-list **do**<br>1.3.3 Find conditional patterns CP.<br>1.3.4 Delete CP based on min-sup.<br>1.3.5 FP = CP ∪ FP.<br>1.3.6 Find all combinations of FP. |

**الجدول (1): النماذج المتكررة Frequent pattern .**

| Conditional patterns | Conditional patterns based on min-sup | Frequent pattern |
|---|---|---|
| A: {D} \| {B, D} \| {C,B,D} | A: {D} | {A}, {A, D} |
| C: {D} \|{B, D}\|{B}\|{B} | C: {B} | {C}, {C, B} |
| B: {D}\|{D}\|{D} | B: {D} | {B}, {B, D} |
| D: { } | D: { } | {D} |

تبدأ عملية التنقيب من العنصر الموجود أسفل الجدول H-table والذي يحقق min-sup=3، يتم الانطلاق من سطر العنصر A إلى الغراف FP-Graph. تمتلك العقدة A ثلاث مؤشرات ParentNodePointer تشير إلى العقدة D، B، C وبقيمة حافة [$T_1$]، [$T_2$]، [$T_4$] على الترتيب. من أجل المداولة $T_1$ يتم الانتقال من العقدة A إلى العقدة D. والمداولة $T_2$ يتم الانتقال من العقدة A إلى العقدة B ثم D أيضاً. أما المداولة $T_4$ يتم الانتقال من العقدة A إلى العقدة C ثم B ثم D. وبالتالي النماذج الشرطية الخاصة بالعقدة A: {D},{B, D},{C,B,D} وبنفس الطريقة يتم استنتاج النماذج الشرطية لباقي العناصر. بعد ذلك يتم تطبيق عتبة الدعم الدنيا من أجل حذف النماذج الشرطية غير المتكررة (العمود الثاني من الجدول(1))، ليتم بعدها توليد النماذج المتكررة بشكلها النهائي (العمود Frequent pattern من الجدول(1)) من خلال إيجاد كل المجموعات الممكنة من النماذج الشرطية المتكررة.

**5-الخوارزمية المحسنة التفرعية Parallelized improved algorithm**:

قمنا في هذا البحث باقتراح خوارزمية BPGFP-growth، وهي خوارزمية تفرعية متوازنة معتمدة على الغراف الموجه في بيئة حوسبة تفرعية بالاعتماد على إطار هادوب والبنية MapReduce لمعالجة البيانات بشكل متزامن وفعّال وموزع. ومن أجل تخفيض كلفة التواصل ومعدل عمليات الدخل والخرج بين العقد الحسابية قمنا بتعديل الطريقة المتبعة في البنية MapReduce لتجميع المداولات واقتراح استراتيجية لإيجاد الترابط بين المداولات المتشابهة لتجميعها ضمن تجمع واحد بالاعتماد على المفاهيم التالية: Jaccard similarity و MinHash و LSH والتي سوف يتم شرحها في الفقرات التالية.

**5-1 مقياس Jaccard similarity**: عملية التجميع في الخوارزمية المقترحة تتم بالاعتماد على إيجاد درجة الترابط (Correlation Degree) بين المداولات بهدف تسريع عملية التنقيب. يستخدم مقياس Jaccard similarity الموضح في العلاقة (1) من أجل إيجاد مدى التطابق والاختلاف بين مجموعة A و B من البيانات. تتراوح قيمة مقياس Jaccard بين القيمة 0 و 1، ومع اقتراب القيمة إلى 1 يزداد التطابق بين البيانات.

$$JS(A,B)=\frac{|A\cap B|}{|A\cup B|} \qquad (1)$$

تستغرق عملية إيجاد درجة الترابط زمناً كبيراً عندما تكون أبعاد البيانات ضخمة[18].

**5-2 طريقة MinHash**: يقدم MinHash طريقة سريعة من أجل حساب مقدار التشابه بين مجموعتين ضخمتين من البيانات. تستخدم طريقة MinHash لتخفيض أبعاد مجموعات البيانات الضخمة وتحويلها الى مجموعات أصغر تسمى بالتواقيع (Signatures)، ويتم إيجادها من خلال توليد مصفوفة تسمى مصفوفة الخصائص (Characteristic matrix) أعمدتها تمثل مداولات قاعدة البيانات (m) أما أسطرها تمثل عناصر قاعدة البيانات (n). حيث يتم ضبط

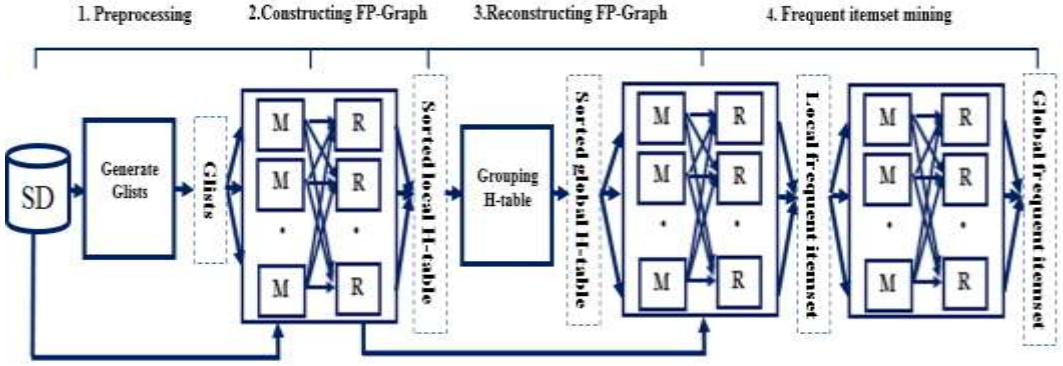
الموقع (n,m) بالقيمة 1 في حال انتماء العنصر الى المداولة وإلا يتم ضبط الموقع بالقيمة 0. وبالاعتماد على مصفوفة الخصائص وتوابع الهاش يتم توليد مصفوفة التواقيع. إن تابع الهاش h(x) هو عبارة عن تابع يستخدم من أجل تمثيل مجموعة من العناصر $T=\{x_1, x_2,..x_n\}$ من خلال قيمة صحيحة واحدة تساوي أصغر قيمة بين قيم الهاش لعناصر المجموعة T كما توضح المعادلة (2)، حيث يتم استبدال عناصر المداولات بأعداد صحيحة من 0 حتى N− 1 لتطبيق تابع الهاش الذي يأخذ الشكل $h_i(x)= (x+1) \bmod N-1$ لبناء مصفوفة التواقيع. تحتوي مصفوفة التواقيع على نفس عدد الأعمدة في مصفوفة الخصائص، أما عدد الأسطر يكون أقل وذلك لتخفيض أبعاد مجموعات البيانات.

$$h_{min}(T) = x, \text{Where } h(x) = \text{Min}_{i=1}^{n}\left(h(x_i)\right) \qquad (2)$$

بالاعتماد على مصفوفة التواقيع يتم إيجاد درجة الترابط بين أي مداولتين [18].

**5-3 تجميع البيانات من خلال Locality-Sensitive Hashing LSH**: تستخدم طريقة LSH من أجل تسريع عملية تجميع المداولات ضمن حزم (bands)، حيث يتم تقسيم مصفوفة التواقيع الى عدة حزم b تحوي كل منها r سطر، حيث تعود كل حزمة الى تابع هاش معين (من الممكن استخدام نفس التوابع المعرفة في مرحلة MinHash). تعمل توابع الهاش على تقسيم فضاء المداولات الى تجمعات أو صناديق (Buckets) تحتوي أزواج المداولات المرشحة (Candidate pairs)لأن تكون مداولات متشابهة[17] [18].

**5-4 تطبيق الخوارزمية المقترحة تفرعياً**: تتألف الخوارزمية المقترحة من ثلاث مراحل أساسية ومرحلة معالجة أولية كما يوضح الشكل(3).




**الشكل (3): المخطط العام للخوارزمية التفرعية المقترحة BPGFP-growth**

**1. المعالجة الأولية:** في البداية يتم ومن خلال عقدة حسابية واحدة العمل على إيجاد مراكز تجمعات المداولات المتشابهة في تيار البيانات من خلال طريقة الأختيار العشوائي (المرجع[19]). حيث يتم اختيارQ مجموعة من مداولات تيار البيانات، ومن ثم يتم إيجاد المجموع الكلي للمسافات بين كل مداولتين من أجل كل مجموعة. وفي النهاية يتم اختيار المجموعة ذات المجموع الكلي الأكبر كمراكز لتقسيم البيانات. حيث يتم وضع كل مداولة في المجموعة المختارة ضمن قائمة تسمى بـ $G_i$-List يتم تميزها من خلال معرف خاصG-id. ومن أجل ضمان توزيع الأحمال بشكل متوازن بين العقد الحسابية يتم حساب الحمل المتوقع لكل تجمع (عدد المداولات) من خلال اختيار عينة من المداولات وإيجاد المركز الأقرب لكل منها بواسطة مقياسJaccard (الفقرة 5-1). وفي المرحلة التالية(التفرعية) يتم من خلال التابع Map (M في الشكل(3)) بعد تحميل قوائم $G_i$-List في الذاكرة الرئيسية(الخطوة2 من الشيفرة

الزائفة)، إيجاد تجمعات المداولات المتشابهة وفق مراكز التجمعات الناتجة عن المرحلة السابقة(الخطوات3حتى6) وبالاعتماد على مفهوم MinHash و LSH. والشيفرة الزائفة لخوارزمية التجميع المقترحة.

| |
|---|
| **Algorithm:** Procedure of LSH-partitioning**.**<br>**Input**: Stream Data **$SD_i$** and item groups **$G_i$-List**.<br>**Output**: transactions corresponding to each Gid. |
| 1. **Function** Map(key offset, value $SD_i$)<br>2. Load $G_i$-List and applying MinHash on it.<br>3. **For** each transaction T in $SD_i$ **do**<br>3.1 Applying the MinHash function on T.<br>4. Generating Signature Matrix M(l×n).<br>5. **For** each columns $c_i$ in sigMatrix **do**<br>5.1 divide ci into b bands with r rows;<br>5.2 Hashbucket = HashMap(each band of ci());<br>6. **IF** at least one band of ci and $G_i$-List is hashed into the same bucket **then** ci and $G_i$-List are considered similar.<br>6.1Output(key $G_{id}$, value $SD_i(G_{id})$); //$SD_i(G_{id})$: transactions corresponding to each Gid. |

**2. بناء الغراف:** بعد تجميع كل المداولات التي تعود إلى نفس المجموعة عن طريق التابع Map، تنتقل مخرجات البنية Map إلى المرحلة Shuffle التي تقوم بترتيبها حسب المفتاح (رقم المجموعة)، ومنها إلى البنية Reduce (R في الشكل(3)) والتي تعمل على بناء الغراف الخاص بكل مجموعة كما هو موضح في الفقرة (4-1)، وإيجاد جدول Sorted local H-table الخاص بتلك المداولات أيضاً ومن ثم ارساله إلى العقدة المركزية (Master). ومن الممكن للعقدة الحسابية الواحدة أن تقوم ببناء الغراف الخاص بأكثر من تجمع واحداً تلو الأخر، حيث يتم التحكم بذلك من خلال عدد المداولات بكل تجمع من أجل تحقيق الموازنة بين أحمال العقد الحسابية.

**3. إعادة بناء الغراف الخاص بكل عقدة:** يتم تجميع كل الجداول Sorted local H-table الواردة من العقد الحسابية في جدول واحد عام يسمى Sorted global H-table ومن ثم يتم ارساله إلى العقد من أجل إعادة بناء الغراف الخاص بكل عقدة لإيجاد الغراف النهائي عن طريق التابع Map الخاص بالمرحلة التفرعية الثانية.

**4. التنقيب عن العناصر المتكررة باستخدام خوارزمية GFP-growth:** يقوم التابع reduce الخاص بالمرحلة التفرعية الثانية بالتنقيب ضمن الغراف من أجل إيجاد قواعد الترابط الخاصة بتلك المداولات والتي تسمى Local frequent itemset، كما هو موضح في الفقرة (4-2). والمرحلة التفرعية الأخيرة تعمل على تجميع نتائج كل العقد الحسابية الناتجة في المرحلة السابقة، لإيجاد مجموعات العناصر المكررة النهائية Global frequent itemset.

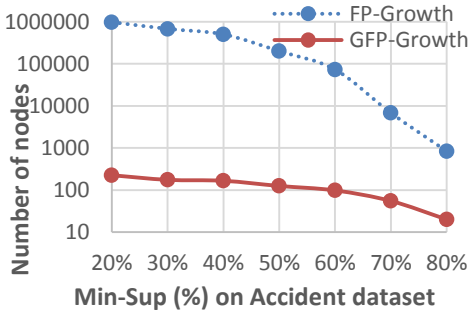
**6-النتائج التجريبية Experimental results:** تم بناء خوارزمية BPGFP-growth المقترحة بلغة الـ Java وبيئة Hadoop، وتم تطبيق هذه الخوارزمية والحصول على النتائج التجريبية من خلال عنقود هادوب مكون من خمس عقد، عقدة أب (Master) وأربع عقد أبناء (Slaves). كل العقد تمتلك نفس البنية البرمجية والمادية، نظام التشغيل Ubuntu 16.4 LTS-64Bit، ومعالج Core i3 2.30GHz وذاكرة RAM 4GB. واصدار Hadoop-2.8.0، JDK1.8.0_181. سيتم مقارنة الخوارزمية المقدمة مع خوارزمية FP-growth[10] و BPFP-growth [16]

و FiDoop–DP [17] و BPFPMS [8]، لإثبات فعالية هذه الخوارزمية عند التعامل مع قواعد البيانات الضخمة، وتوفير الزمن اللازم لعملية التنقيب والمساحة التخزينية المطلوبة، وتخفيض التعقيد الحسابي والاتصال بين عقد التجمع، وذلك عند تطبيق عتبات min–sup مختلفة.

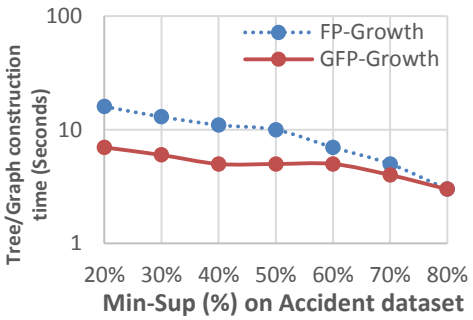
**الجدول (2): خصائص قواعد المعطيات المستخدمة.**

| Dataset | Trans | Avg. Length | Items |
|---|---|---|---|
| Accident | 340183 | 34 | 468 |
| T40I10D100K | 100.000 | 10 | 942 |
| Kosarak | 990.002 | 8 | 41270 |

استخدمت مجموعة من قواعد معطيات قياسية معتمدة في أبحاث سابقة، تم تحميلها من موقع FIMI ( FIMI'04 2004)[20]، لإجراء الاختبارات. الجدول(2) يبين أهم خصائص قواعد المعطيات المستخدمة في اختبار الخوارزميات وهي على الترتيب عدد المداولات، متوسط طول المداولة، عدد العناصر. في البداية سوف يتم مقارنة النتائج التجريبية الخاصة بالخوارزمية المقترحة المحسنة GFP–growth مع FP–growth الأساسية وعلى عقدة حسابية واحدة لإثبات مدى فعالية الخوارزمية المحسنة، باستخدام قاعدة المعطيات Accident وبتطبيق عتبات دعم مختلفة من20%إلى80%.



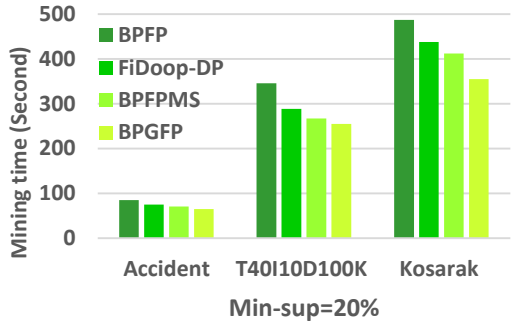

**الشكل (5): عدد عقد الغراف/الشجرة الخاصة بقاعدة معطيات Accident.**



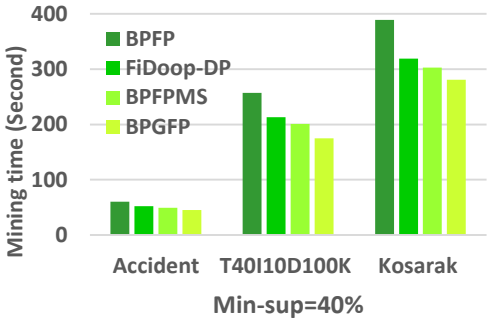

**الشكل (6): الزمن اللازم لبناء الغراف/ الشجرة الخاصة بقاعدة معطيات Accident.**

من الشكل(5) نجد أن عدد عقد FP–Graph في الخوارزمية المقترحة أقل بشكل كبير وملحوظ من عدد عقد شجرة FP–Tree في خوارزمية FP–growth. الشكل(6) يبين أن زمن بناء الغراف في الخوارزمية المقترحة أقل من الزمن اللازم لبناء الشجرة في خوارزمية FP–growth وخصوصاً عند عتبات الدعم المنخفضة. سوف يتم مقارنة الخوارزمية التفرعية المقترحة BPGFP–growth مع خوارزمية FiDoop–DP [17] لمقارنة تأثير طريقة تقسيم البيانات على زمن التنفيذ في كلا الخوارزميتين وخوارزمية BPFP–growth [16] لمقارنة تأثير طريقة موازنة الأحمال على زمن التنفيذ في كلا الخوارزميتين و BPFPMS [8] لمقارنة فعالية البنية المستخدمة في كلا الخوارزميتين. يبين الشكل(7) و(8) أن الخوارزمية المقترحة تستغرق زمن تنقيب (عند عتبة دعم 20% و40% على الترتيب) أقل بين الخوارزميات

الأخرى ، وخصوصاً عند التعامل مع البيانات الضخمة مثل Kosarak، ويرجع السبب في ذلك نتيجة طريقة التنقيب المتبعة في الغراف بالإضافة الى استراتيجية تقسيم وموازنة الأحمال بين العقد الحسابية.

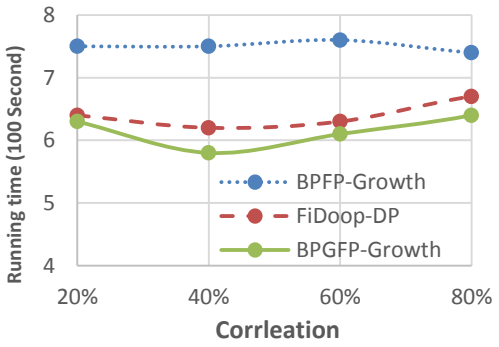



**الشكل (7): الزمن اللازم للقيام بعملية التنقيب ضمن قواعد المعطيات السابقة عند عتبة دعم 20%.**

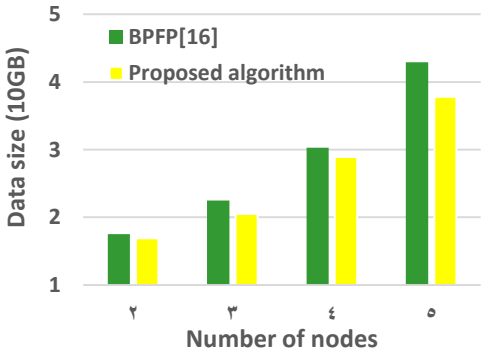



**الشكل (8): الزمن اللازم للقيام بعملية التنقيب ضمن قواعد المعطيات السابقة عند عتبة دعم 40%.**

تم ضبط معامل الترابط عند تقسيم العناصر المتكررة والمداولات بالقيم التالية (0.20, 0.40, 0.60, 0.80) وذلك من أجل تقييم تأثير معامل الترابط على أداء الخوارزمية المقترحة. الشكل(9) يوضح أن الخوارزمية المقترحة أكثر حساسية لمعامل ترابط البيانات من خوارزمية BPFP-growth [16] وهناك شبه تطابق مع خوارزمية FiDoop-DP [17]، وأن أفضل قيمة لمعامل الترابط هي القيمة 0.4. فإذا كانت قيمة معامل الترابط منخفضة فهذا يؤدي إلى تقسيم العناصر بصورة عشوائية ويصعب تقسيم العناصر المتكررة الى مجموعات مستقلة عند ضبط قيمة معامل الترابط بقيمة كبيرة. أما الشكل(10) يوضح زيادة حجم البيانات المتبادلة مع زيادة عدد العقد الحسابية من 2 إلى 4، بالإضافة إلى أن حجم البيانات المتبادلة بين العقد الحسابية في الخوارزمية المقترحة أقل من خوارزمية BPFP-growth.

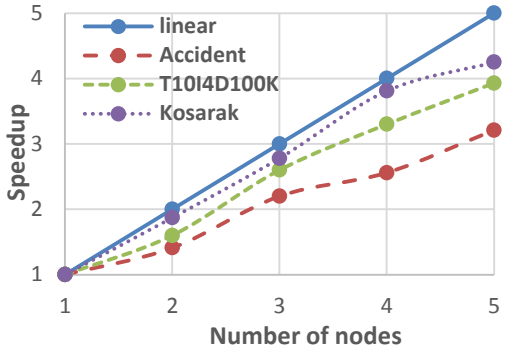



**الشكل (9): تأثير معامل الترابط على زمن التنفيذ.**




**الشكل (10): حجم البيانات المتبادلة بين العقد الحسابية.**

ولتقييم أداء الخوارزمية التفرعية المقترحة سوف يتم استخدام مقياس التسارع Speedup [15] الموضح في المعادلة(3)، ويستخدم من أجل تقييم سرعة الخوارزمية التفرعية مقارنة بالخوارزمية التسلسلية المطبقة على عقدة واحدة.




**الشكل (11): مقياس Speedup للخوارزمية المقترحة.**

$$\text{Speedup}=S(n)=\frac{T_1}{T_n} \quad (3)$$

حيث إن $T_1$: تمثل زمن تنفيذ الخوارزمية على عقدة واحدة، أما $T_n$: تمثل زمن تنفيذ الخوارزمية وباستخدام نفس قاعدة المعطيات على n عقدة حسابية. تم التنقيب عن العناصر المتكررة في قواعد المعطيات السابقة، باستخدام الخوارزمية المقترحة وانطلاقاً من عقدة حسابية واحدة حتى خمس عقد. من الشكل(11) نجد أن تسارع الخوارزمية المقترحة يزداد بشكل شبه خطي مع زيادة عدد العقد، وخصوصاً عند زيادة حجم قاعدة البيانات. وصلت قيمة التسارع من أجل قاعدة البيانات Kosarak الى 4.250 أي 85% (4.250/5=0.85) من قيمة التسارع المثالي. بشكل عام من الصعب الوصول إلى التسارع المثالي نتيجة كلفة الاتصال بين العقد الحسابية[15].

## الاستنتاجات والتوصيات:

تم في هذا البحث دراسة خوارزميات التنقيب عن العناصر المتكررة وبشكل خاص تطويرات FP-growth التفرعية، حيث تم دراسة إيجابيات وسلبيات كل منها. ومن ثم تم اقتراح خوارزمية تفرعية جديدة BPGFP-growth، معتمدة على الغراف الموجه، وتقوم بمسح وحيد لقاعدة البيانات لتمكينها من التعامل مع تيار بيانات مستمرة وتوفير الزمن اللازم لعملية التنقيب والمساحة التخزينية المطلوبة، والعمل على توزيع المهام الحسابية (من خلال استراتيجية لتوزيع الأحمال وتقسيم المداولات عن طريق إيجاد الترابط بينها) بين تلك العقد مما يحقق تخفيض التعقيد الحسابي والاتصال بين عقد التجمع. وعند تطبيق الخوارزمية المقدمة على قواعد بيانات قياسية أثبتت النتائج التجريبية فعالية وسرعة تلك الخوارزمية عند تطبيق عتبات دعم مختلفة بالإضافة إلى القدرة على التعامل مع قواعد المعطيات بمختلف أنواعها وأحجامها. وقد ظهر تحسن في الأداء خاصة عند التعامل مع قواعد البيانات ذات الأحجام الضخمة. كل تقيمات الأداء السابقة توضح بأن النموذج المقترح والمكون من هادوب والبنية MapReduce واستراتيجية تقسيم وتوزيع الأحمال حقق تحسن ملحوظ في الدقة، والفعالية، وقابلية التوسع.

## References:

[1]. Yan X ·, Zhang J ·, Xun Y ·, Qin X. 2015," A parallel algorithm for mining constrained frequent patterns using MapReduce". Springer, published online 17 November 2015.

[2]. Yang X.Y., Liu Z., Fu Y. 2010 "MapReduce as a programming model for association rules algorithm on Hadoop", In: 2010 3rd International conference on information sciences and interaction sciences (ICIS), pp 99–102.

[3]. Apache Hadoop. http://hadoop.apach e.org/. Accessed 20 Feb 2020.

[4]. Dahdouh K., Dakkak A., Oughdir L. and Messaoudi F., 2019. 'Large‐scale e-learning recommender system based on Spark and Hadoop'. Springer, Journal of big data (2019).

[5]. N. Jiang and L. Gruenwald, "Research issues in data stream association rule mining," ACM Sigmod Record, vol. 35, no. 1, pp. 14–19, 2006.

[6]. S. Moens, E. Aksehirli, and B. Goethals, "Frequent itemset mining for big data," in Big Data, 2013 IEEE International Conference on. IEEE, 2013, pp. 111–118.

[7]. S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong, and Y.-K. Lee, "Efficient frequent pattern mining over data streams," in Proceedings of the17th ACM conference on Information and knowledge management. ACM, 2008, pp. 1447–1448.

[8]. X. Fu X., L. Shi and J. Li,2017." Balanced Parallel Frequent Pattern Mining Over Massive Data Stream", IEEE, 2017 IEEE Third International Conference on Big Data Computing Service and Applications, pp. 50-59.

[9]. Agrawal, R., & Srikant, R. 1994. "Fast algorithm for mining association rules in large databases". In Proceedings of 20th VLDB conference (pp. 487–499).

[10]. Han J., Pei J. and Yin .2000 'Mining frequent patterns without candidate generation', in Proceedings of the ACM-SIGMOD Conference Management of Data, Vol. 29, No. 2, pp.1–12.

[11]. Han, J., Pei J. and Mao 2004 'Mining frequent patterns without candidate generation: a frequent-pattern tree approach', Data Mining and Knowledge Discovery,Vol.8, pp.53–87.

[12]. Y. Chi, H. Wang, P. S. Yu, and R. R. Muntz, "Moment: Maintaining closed frequent itemsets over a stream sliding window," in Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on. IEEE, 2004, pp. 59–66.

[13]. S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong, and Y.-K. Lee, "Sliding window-based frequent pattern mining over data streams," Information sciences, vol. 179, no. 22, pp. 3843–3865, 2009.

[14]. M.-Y. Lin, P.-Y. Lee, and S.-C. Hsueh, "Apriori-based frequent itemset mining algorithms on mapreduce," in Proceedings of the 6th international conference on ubiquitous information management and communication. ACM, 2012, p. 76.

[15]. Li H., Wang Y., Zhang D., Zhang M., and Chang E. Y., 2008," PFP: Parallel FP-Growth for Query Recommendation". In Proceedings of the 2008 ACM conference.

[16]. Le Zhou, Zhiyong Zhong, Jin Chang, Junjie Li, Huang, J.Z., Shengzhong Feng ,2010. "Balanced parallel FP-Growth with MapReduce". Information Computing and Telecommunications, 2010 IEEE Youth Conference, pp. 243-246.

[17]. Xun Y., Zhang J., Qin X. and Zhao X.,2016,"FiDoop-DP: Data Partitioning in Frequent Itemset Mining on Hadoop Clusters", IEEE, 1045-9219 (c) 2016 IEEE.
[18]. J. Leskovec, A. Rajaraman, and J. D. Ullman, 2014, "Mining of massive datasets" . Cambridge University Press.
[19]. W. Lu, Y. Shen, S. Chen, and B. C. Ooi, 2012, "Efficient processing of knearest neighbor joins using mapreduce," Proceedings of the VLDB Endowment, vol.5, no.10, pp. 1016 1027.
[20]. FIMI 2004, FIMI Repository [online] http://fimi.ua.ac.be/data/.