

Graph Mining based on a Data Partitioning Approach

Son N. Nguyen¹, Maria E. Orlowska^{1,2}, Xue Li¹

¹School of Information Technology and Electrical Engineering
The University of Queensland, QLD 4072, Australia

²Faculty of Information Technology
Polish-Japanese Institute of Information Technology, 02-008 Warsaw, Poland

{nnson, maria, xueli}@itee.uq.edu.au

Abstract

Existing graph mining algorithms typically assume that the dataset can fit into main memory. As many large graph datasets cannot satisfy this condition, truly scalable graph mining remains a challenging computational problem. In this paper, we present a new horizontal data partitioning framework for graph mining. The original dataset is divided into fragments, then each fragment is mined individually and the results are combined together to generate a global result. One of the challenging problems in graph mining is about the completeness because of the complexity of graph structures. We will prove the completeness of our algorithm in this paper. The experiments will be conducted to illustrate the efficiency of our data partitioning approach.

Keywords: Subgraph, graph mining, algorithm.

1 Introduction

Discovering subgraph patterns from graph datasets is of great importance in many application domains (e.g., chemical analysis, common protein structure as well as common structure in XML documents). Since the research problem of discovering frequent subgraphs was introduced, many mining algorithms such as Gaston (Nijssen and Kok 2004), gSpan (Yan and Han 2002) have been proposed for efficiently finding frequent subgraphs from the graph datasets. To the best of our knowledge, the most of existing techniques are memory-based approaches. As a result, they are facing the scalability challenge when the input graph dataset cannot be held in the main memory.

In this paper, we present a new horizontal data partitioning framework for graph mining. The original dataset is divided into fragments, and then the fragments are mined individually and combined together for a global result. Compared with a well-known approach PartMiner, proposed by Wang et al. (2006), our solution can guarantee the completeness and correctness of the

mined result. We point out that, in general the algorithm PartMiner cannot find the correct complete set of frequent subgraphs in original dataset D , that is, it violates the correctness condition. Formally, we give proofs and counter examples.

The remainder of the paper is organized as follows. Section 2 gives the formal definition of the graph mining problem. Section 3 briefly discusses the related work. A data partitioning approach for graph mining is introduced in Section 4. Furthermore, we prove the completeness of our algorithm in Section 5. The experiments are conducted in Section 6. Finally, we conclude this paper in Section 7.

2 Graph mining problem

In graph mining, undirected labelled graphs are commonly considered. A graph is defined as follows.

Definition 1: A undirected labelled graph is represented by a 4-tuple $G = (V, E, L, l)$ where;

V is a set of vertices; $E \subseteq V \times V$ is a set of edges;

L is a set of labels;

$l: V \cup E \rightarrow L$, l is a function assigning labels to the vertices and edges.

A graph G is connected if a path exists between any two vertices in V . We focus on connected graph only. The size of a graph is the number of edges in it, and a graph G with k edges is called a graph of size k or k -edge graph.

Definition 2: A graph G_1 is *isomorphic* to graph G_2 if there exists a bijective function $f: V(G_1) \rightarrow V(G_2)$ such that $\forall u \in V(G_1), l_{G_1(u)} = l_{G_2(fu)}$ and $\forall (u, v) \in E(G_1), (f(u), f(v)) \in E(G_2)$ and $l_{G_1(u,v)} = l_{G_2(f(u), f(v))}$

An *automorphism* of G is an *isomorphism* from G to G .

Definition 3: A graph G_1 is *subgraph* of G_2 if all vertices and edges of G_1 and their labels are belonged to G_2 , i.e., $V(G_1) \subseteq V(G_2)$ and $E(G_1) \subseteq E(G_2)$.

A set of all subgraphs of G is denoted as $PS(G)$.

A *subgraph isomorphism* from G_1 to G_2 is an isomorphism from G_1 to a *subgraph* of G_2 .

A graph dataset is a set of tuples $(gid; G)$, where gid is a graph identifier and G is an undirected labelled graph. Size of graph dataset is a number of graphs in dataset.

Given a graph dataset D , the *support* of a graph G is the number of graphs in D that are supergraphs of G , denoted $support_D(G)$.

Problem statement: Given a graph dataset D , a minimum *support* (min_sup). The *frequent subgraph mining* is to discover the complete set of subgraphs which have support in D is no less than min_sup , denoted as $P(D)$.

3 Related Work

There many well-known frequent pattern mining algorithms on subgraphs such as Gaston (Nijssen and Kok 2004), gSpan (Yan and Han 2002), ADIMine (Wang et al. 2004).

Recently, a partition-based algorithm is proposed by Wang et al. (2006), namely PartMiner. PartMiner is developed on the basis of the memory-based graph mining algorithm Gaston. With the aims of solving memory limitation and improving performance, PartMiner partitions original dataset into units first, then Gaston technique is applied to mine the units locally. After that the merging phase is taken to discover complete mining result of the original dataset. In addition, some optimization techniques are applied in the partitioning phase and combining phase.

4 Our partitioning approach

Partitioning approach is firstly applied to frequent itemset mining by Savasere, Omiecinski and Navathe (1995). It is a horizontal partitioning approach to a transaction dataset. Another smart partition technique has been proposed to improve the overall performance by Nguyen and Orłowska (2005). This paper can be regarded as an extension to these approaches to the graph mining.

4.1 Horizontal partitioning framework

The horizontal partitioning frame work is based on the following principle.

A fragment $F \subseteq D$ of the graph dataset is defined as any subset of the graphs contained in the dataset D . Further, any two different fragments are non-overlapping. *Local support* for a subgraph is the fraction of graphs (percentage) containing that subgraph in a fragment. Assume that the *support* is a percentage instead of number of graphs as formal definition in Section 2. A *Local frequent* subgraph is a subgraph whose local support in the fragment is no less than the minimum support which is a percentage in this case. *Global support*, *Global frequent pattern* are defined as above except they are in the context of the entire dataset. The goal is to find all *Global frequent subgraphs*.

Lemma 1: If G is a frequent subgraph in dataset D , which is partitioned into k fragments F_1, F_2, \dots, F_k , then G must be frequent in at least one of the k fragments.

Proof: Proved by contradiction.

Assume that G is a frequent subgraph in dataset D , but not in any of the k fragments, and prove that G must not be a frequent subgraph in D (because the total support for the entire dataset is smaller than the minimum support which is a percentage). As a result, the proof is done by contradiction.

The partitioning framework divides D into k fragments, denoted as $F_i, i = 1, \dots, k$. The framework first scans fragment F_i in the main memory at a time using current memory-based techniques such as Gaston algorithm, for $i = 1, \dots, k$, to find the set of all *Local frequent subgraphs* in F_i , denoted as $R(F_i)$, and the number of frequent subgraphs in F_i , denoted as $|R(F_i)|$. Then, by taking the union of all $R(F_i), i = 1, \dots, k$, a set of candidate subgraphs over D is constructed, denoted as C^G . Based on *Lemma 1*, C^G is a superset of the set of all *Global frequent subgraphs* in D . Moreover, denoted F^G as the intersection of all $R(F_i), i = 1, \dots, k$, F^G is a set of subgraphs already known as *Global frequent subgraphs* (because of frequent in all fragments). Finally, the algorithm scans each fragment for the second time to calculate the support of each *Global candidate* in $(C^G - F^G)$ and to find the all *Global frequent subgraphs*. **Therefore, it guarantees the completeness and correctness of frequent subgraph mining result.**

The outline of our partitioning graph mining algorithm is described as follows:

Algorithm: PartGraphMining()

Input: Graph dataset: D ; min_sup – given the support threshold; k – given number of fragments

Output: The complete frequent subgraphs set in D

Begin

//Phase one

1. Partition dataset D into k fragments (using partitioning algorithm): F_1, F_2, \dots, F_k , every fragment can be loaded into the main memory.

2. Applying the Gaston (or gSpan) subroutine to find the *Local frequent subgraphs* for each $F_i, i = 1 \dots k$;

For $i = 1$ to k : Call Gaston (or gSpan) routine
to get $R(F_i)$

3. Computing the union of all $R(F_i)$:

$$C^G = \bigcup \{ R(F_1), R(F_2), \dots, R(F_k) \}$$

4. Computing the intersection of all $R(F_i)$:

$$F^G = \bigcap \{ R(F_1), R(F_2), \dots, R(F_k) \};$$

F^G is the set of already frequent subgraphs in D .

//Phase two

5. Scan dataset D again (from the first fragment to last fragment) to verify the *Global candidates* set $(C^G - F^G)$ if they are frequent or not; and output the total *Global frequent subgraphs*.

End

4.2 Characteristics of graph fragment

Our main goal is to find an approach to partitioning dataset but not too ‘expensive’ for data fragmentation. Obviously, fragments that have many ‘dissimilar’ graphs (graphs with small or empty common subgraphs) generate a small number of *Local frequent subgraphs*. In this paper we call them ‘*dissimilar fragments*’. We want to partition the original graph dataset into the ‘dissimilar’ fragments. As a result, at the second phase of **PartGraphMining()** algorithm, the number of *Global candidates* ($C^G - F^G$) is reduced as well as the number of already frequent subgraphs (F^G) is increased by taking union and intersection among *Local frequent subgraphs* of fragments.

4.3 Proposed data partitioning solution

The incremental clustering algorithm is our idea for partitioning process. Our technique is based on traditional K-mean algorithm, but there is only one scan through out the dataset, i.e., it is incremental processing. The concept of this technique is illustrated as Figure 1 for the case: number of fragments is 3.

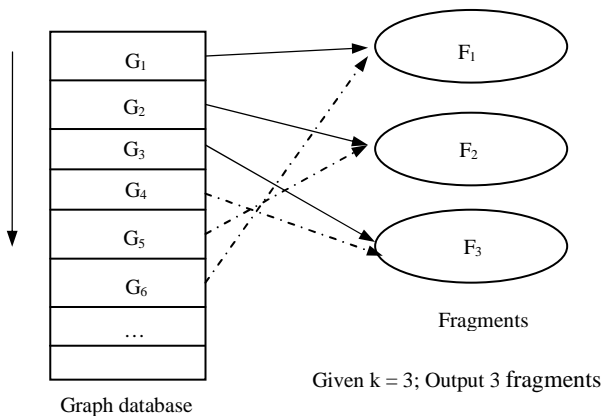


Figure 1: The incremental processing

The size of each fragment, i.e. the number of graphs in fragments, is controlled to keep the same size condition among fragments. The reason is to distribute as much uniform as possible among fragments.

Furthermore, in order to get ‘*dissimilar fragments*’, we define two concepts: Centroid of fragment and similarity function between graph and fragment centroid.

Definition 4: Centroid of graph fragment is a set of all 1-edge subgraphs in the fragment, denoted as $C_i = \{I_1, I_2, \dots, I_n\}$ where I_k represents a 1-edge subgraph. Additionally, each element in C_i has its associated weight which is its number of occurrences in the fragment; $\{w_1, w_2, \dots, w_n\}$

Definition 5: Similarity function between a graph and a fragment centroid, is denoted $\text{SimGraph}(S_i, C_j)$ and defined as follows;

$\text{SimGraph}(G_i, C_j) \rightarrow \mathbb{R}^+$; Calculation of this function:

1. Let S_i^* is the set of 1-edge subgraphs in the graph G_i
2. Let S be the intersection between S_i^* and C_j ,

$$S = S_i^* \cap C_j$$

3. If $S = \emptyset$ then $\text{SimGraph}(S_i, C_j) = 0$. Otherwise, $S = \{I_1, I_2, \dots, I_m\}$ with the corresponding weights $\{w_1, w_2, \dots, w_m\}$ in fragment C_j , respectively, therefore $\text{SimGraph}(S_i, C_j) = w_1 + w_2 + \dots + w_m$

The overview of our data partitioning algorithm is given as follows.

Algorithm: IncPartDBGraph ()

Input: Graph dataset: D ; k – given number of fragments

Output: The k fragments: F_1, F_2, \dots, F_k

Begin

1. Assign the first k graphs to k fragments, and initialize the all fragment Centroids: $\{C_1, C_2, \dots, C_k\}$
2. Consider the next k graphs. These k graphs are assigned to k different fragments. These operations are done based on the following criteria: (i) the **minimum similarity** between the new graph and the suitable fragments; (ii) the sizes of these fragments are controlled to keep the balance.
3. Repeat step 2 till all graphs in D are partitioned.

End

5 Completeness and Correctness issues

The completeness and correctness are challenging to all data mining algorithms. As can be seen in section 4.1, our data partitioning solution guarantees the completeness of the frequent patterns discovery. Compared with PartMiner approach (Wang et al. 2006), we find out that the later violates the correctness condition. We now discuss this finding in more detail.

5.1 PartMiner approach

Figure 2 shows the basic framework of the PartMiner approach. It consists of two phases. At the first phase, a graph partitioning algorithm is used to split every graph in the dataset into smaller subgraphs. Then the subgraphs are grouped into units. The second phase applies an existing memory-based graph mining algorithm (used Gaston algorithm) to discover the frequent subgraphs in each unit. The set of frequent subgraphs in each unit are then merged via a merge-join operation to recover the complete set of frequent subgraphs.

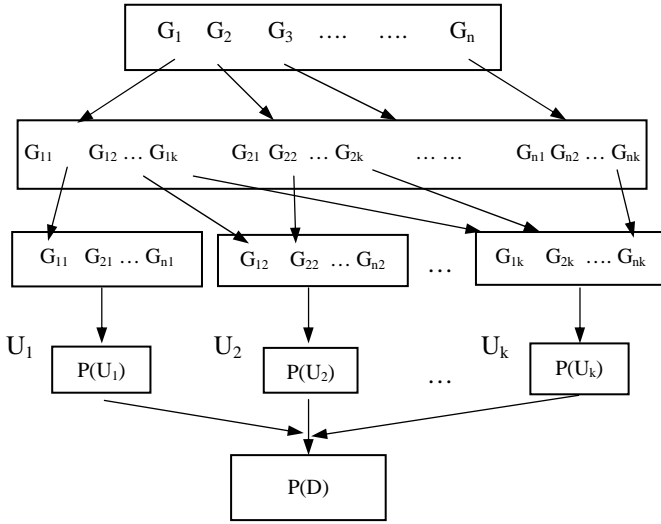


Figure 2: PartMiner partitioning framework

Dividing Graph Dataset into Units

PartMiner has adopted an approach that repeatedly bi-partitions each of the graphs in the dataset. For example, a graph G is first divided into 2 subgraphs G_1 and G_2 , then G_1 (G_2) is again further divided into 2 subgraphs G_{11} and G_{12} (G_{21} and G_{22}). This bi-partitioning process yields a total of 4 subgraphs for G . By applying this bi-partitioning procedure on each of the graphs G_i in the dataset, we can have four subgraphs G_{i1} , G_{i2} , G_{i3} , and G_{i4} for each G_i . Each of the subgraphs G_{ij} , $1 \leq i \leq 4$, can be grouped into one unit U_j .

Mining Frequent Subgraphs in Units

PartMiner approach uses the Gaston algorithm to find the set of frequent graphs in the units. After all $P(U_i)$, $1 \leq i \leq k$, are discovered, the combining step is applied to get full frequent subgraphs in the original dataset.

Combining Frequent Subgraphs

At this point, the set of frequent subgraphs in the units have been computed. In PartMiner, a merge-join operation is defined to recover the complete set of frequent subgraphs in original graph dataset. The idea behind the merge-join operation is illustrated detail in PartMiner paper.

Completeness of PartMiner approach

Completeness means all frequent subgraphs in the dataset will be discovered independent from the partitioning technique. The completeness of PartMiner approach is claimed by three theorems as follows:

Theorem 1: *The set of subgraphs of a graph G (i.e. $PS(G)$) with size of n , $n \geq 2$, can be fully recovered by recursively applying the merge-join operation on its bi-partitioned subgraphs G_1 and G_2 .*

Proof: By an induction method.

“Base Case: $n = 2$. This is trivially true as shown in PartMiner paper.

Induction Step: Suppose H_n is true, the authors want to show that H_{n+1} is also true. If H_n is true, it is able to recover all subgraphs of a graph G of size n . Now one have a graph G' of size $n + 1$. Graph G' is partitioned into two subgraphs. Let G_1 denote the partition of size n , and let G_2 denote the partition of size $i - 2$, $3 < i < n$ (see Figure 3). All subgraphs from G_1 (because G_1 is of size n) can be recovered. Hence, the only missing subgraphs are those involving the edge $(v_1; v_2)$ in G_2 . These subgraphs are formed by the joining of the subgraphs of G_1 and G_2 , which share one of the common edges $(v_2; v_3); \dots; (v_2; v_i)$ marked as grey in Figure 3. This step in fact is included in the merge-join operation. In other words, if H_n is true, then H_{n+1} must be true”.

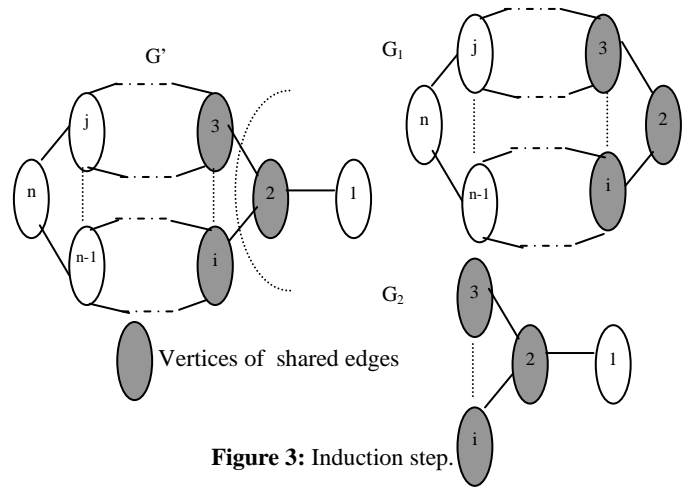


Figure 3: Induction step.

Theorem 2: *(Apriori property) If a graph G is frequent in graph dataset D , all of its subgraphs are frequent in D .*

Proof: It is obviously according to subgraph definition.

Theorem 3: *Let D be a graph dataset that has been divided into k smaller units U_i , $k \geq 2$, $1 \leq i \leq k$. If the complete set of frequent subgraphs $P(U_i)$ in each unit U_i , $1 \leq i \leq k$ is already known, the complete set of frequent subgraphs $P(D)$ in D can be determined.*

Proof: Also by induction method. This theorem is a key point to prove the completeness of PartMiner. However, we found that there is not a correct proof. We will come back with this statement in our following section.

5.2 Critical observations

After above review on PartMiner approach, we now show that in general, PartMiner provides incorrect mining results. Particularly, we shall prove that the Proof of Theorem 1 in PartMiner is incorrect and the proof of Theorem 3 in PartMiner is incorrect either.

Claim 5.1 (Incorrectness of Proof for Theorem 1)

We think that the induction method is not suitable to prove Theorem 1. In the induction step, under the assumption that Theorem 1 is true for graphs with size n (H_n), the authors try to show that Theorem 1 is also true for graphs with size $n+1$ (H_{n+1}).

An example is used for this purpose, as shown in Figure 3. However, this example is just a special case and is thus not general enough to be representative of all possible situations. The reasons are:

It is assumed that the difference between G' (i.e., size $n+1$) and G (i.e., size n) is that there is a new vertex v_1 in G' because the authors claim that “*the only missing subgraphs are those involving the edge (v_1, v_2) in G_2* ”. After all, in Figure 3, v_1 is only connected to v_2 .

However, the size of a graph is determined by the number of edges instead of the number of vertices. In other words, G' could also be constructed by adding a new edge between two existing vertices in G that are not originally connected. Note that this case can not be explained by Figure 3.

Furthermore, in Figure 3, G' is carefully partitioned so that G_1 is of size n and G_2 is of size $i-2$ with $3 < i < n$. Compared to the definition of Theorem 1, this is also a special case of partition. In fact, the authors of PartMiner should show that H_{n+1} is true for any two G_1 and G_2 resulted from any bipartition of G' .

Counter example 1: Finally, an example is shown below (Figure 4) where G' is of size $n+1=3$. It is not sure whether Theorem 1 is true on G' given the base case of induction method with $n=2$. Note that G_1 is of size $n+1=3$ and G_2 is of size $n=2$. In the paper, the authors claim that G_1 is of size n and G_2 is of size $i-2$ with $3 < i < n$, which is impossible in this case, no matter how G' is bi-partitioned.

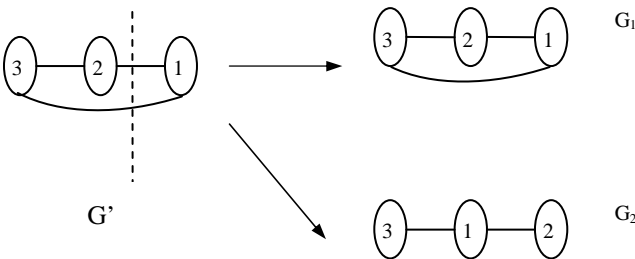


Figure 4: Counter example 1 of induction proof for Theorem 1

Correct proof: We prove by contradiction.

Given a graph G partitioned by the method proposed in PartMiner approach into two subgraphs G_1 and G_2 . Suppose the complete set of subgraphs of G_1 , $PS(G_1)$ and the complete set of subgraphs of G_2 , $PS(G_2)$, are given. We can recover the complete set of subgraphs of G by applying the merge-join operation in PartMiner.

Assume that there is a subgraph of G , denoted gs , but cannot recover from $PS(G_1)$ and $PS(G_2)$. Therefore, there is at least one 1-edge subgraph of gs , but cannot recover from the set of 1-edge subgraphs of $PS(G_1)$ and the set of 1-edge subgraphs of $PS(G_2)$. It is impossible because according to PartMiner’s bi-partitioning techniques, the set of 1-edge subgraphs of $PS(G_1)$ and the set of 1-edge subgraphs of $PS(G_2)$ contains complete set 1-edge subgraphs of original graph G . As a result, the proof of Theorem 1 is done.

Claim 5.2 (Incorrectness of Proof for Theorem 3)

We see there is no relationship between Theorem 1 and Theorem 3. Because, Theorem 1 is about the subgraphs of only one graph G , meanwhile Theorem 3 is about the frequent subgraphs of unit which is a set of graphs. Moreover, the set of frequent subgraphs in original dataset D is totally deferent from the set of subgraphs in each graph of D . We cannot generalize from only one graph (Theorem 1) to a set of graphs (Theorem 3). Therefore, the base case of induction step ($k=2$) in the proof of Theorem 3 is incorrect. As a result, the induction step for $k > 2$ is failed as well.

Correct proof

Assume that the min_sup threshold is a number of graphs (not percentage threshold). A graph dataset has been divided into k smaller units U_i , $k \geq 2$, $1 \leq i \leq k$. If the complete set of frequent subgraphs $P(U_i)$ with min_sup/k threshold (according to PartMiner algorithm) in each unit U_i , $1 \leq i \leq k$ is already known, the complete set of frequent subgraphs $P(D)$ in D can be determined by the merge-join operation.

We prove by contradiction.

Assume that there is a frequent subgraph of $P(D)$, denoted fgs , but cannot determine from $P(U_i)$, $k \geq 2$, $1 \leq i \leq k$. Therefore, there is at least one 1-edge subgraph of fgs which is frequent in D , denoted as f_1gs , but cannot determine from the set of 1-edge subgraphs of $P(U_1)$, $P(U_2)$, ... and $P(U_k)$. As a result:

$$support_D(f_1gs) \geq min_sup \quad (\text{for the whole original dataset } D)$$

In every unit U_i , $1 \leq i \leq k$:

$$support_{U_i}(f_1gs) < \frac{min_sup}{k}; \text{ because it is infrequent.}$$

Let taking sum of all support of f_1gs in all k units:

$$support_{U_1..U_2...U_k}(f_1gs) < \frac{min_sup}{k} * k = min_sup$$

However, $support_{U_1..U_2...U_k}(f_1gs) \geq support_D(f_1gs) \geq min_sup$

As a result, the proof of Theorem 3 is done by contradiction.

Claim 5.3 (Incorrectness of PartMiner mining result)

According to Theorem 3, PartMiner decreases the minimum support to $\frac{\min_sup}{k}$ when mining individually every unit. It guarantees the complete frequent subgraphs of original dataset which can be determined. However, according to merge-join operation, there is no correct pruning technique to remove the infrequent subgraph in D which is frequent in units (because of decreasing the minimum support).

For example, we have to determine all frequent 1-edge subgraphs in D . **It cannot be the union of all frequent 1-edge subgraphs in all units (because of different minimum support)**. There is only way to verify it by scanning all dataset D to count the frequency, but it will violates the idea of partitioning techniques. As a result, after taking the merge-join operation, the final PartMiner mining result may contain the infrequent subgraphs in whole original dataset.

Counter example 2: We consider the following simple example. Let given a graph dataset D contained 2 graphs (G_1, G_2), and a number of units $k = 2$, given $\min_sup = 2$. Assume that there is no frequent pattern at all. All edges in two graphs are different

According to PartMiner algorithm, at the first phase, each graph $G_i, i = 1, \dots, 2$, in D is partitioned into 2 subgraphs G_{i1}, G_{i2} . After that all subgraphs $G_{i1}, i = 1, \dots, 2$ are grouped into unit U_1 , all subgraphs $G_{i2}, i = 1, \dots, 2$ are grouped into unit U_2 .

At the second phase, the frequent subgraphs in 2 units are discovered by applying Gaston algorithm with $\min_sup / k \text{ threshold} = 1$ (in this case). That means all subgraphs in U_1 and U_2 are frequent. Then, the merge-join operation will discover all subgraphs of G_1, G_2 which are frequent in D , this mining result is incorrect. There is no pruning technique at all.

In conclusion, we have proved that the PartMiner's theorems have some weakness. Therefore, PartMiner approach violates the completeness and correctness of graph mining result.

6 Experiments

In this section, the experiments are conducted using two real data sets from gSpan work (Yan and Han 2002), namely Chemical_340 and Compound_420. Those data sets are the same one used in (Kuramochi and Karypis 2001). The Chemical_340 data set contains 340 graphs. The average size is 27.4 in terms of the number of edges, and 27 in terms the number of vertices. Because the number of edges is very close to that of vertices, this data set is sparse. In contract, the Compound_420 data set contains 420 graphs and it is dense.

Our goal is to compare the cardinality of the outputs from two phases of the PartGraphMining algorithm; at the *Local level* and the *Global level*, before and after applied data pre-processing. Firstly, data set is partitioned into 2

fragments; secondly the gSpan routine (Yan and Han 2002) is applied to find *Local frequent subgraphs* ($R(F_i)$) for each fragment. Subsequently, union of these $R(F_i)$ generates the *Global candidates* (C^G) and intersection of these $R(F_i)$ generates the common candidates which are already frequent.

Resulting figures for each data set are represented in following template Table 1. The 2nd and 3rd columns' names indicate two techniques for data preparation: **Sequential** fragments correspond to loading clusters with original graph data, and the **Clustering** fragments are the pre-processed data as presented by our incremental clustering method described in section 4.3.

The data sets used are indicated on the top of each table segment. Note that this figure is presented by showing its two components; for example, **372 + (71)** indicate that there are **372** candidates to be checked and **71** common candidates don't need an additional check (already known as *Global frequent subgraphs*).

	Sequential	Clustering
Chemical_340		
1-fragment: 190 Frequent Subgraphs		
2 fragments		
$R(F_1)$	223	239
$R(F_2)$	291	175
C^G	372 + (71)	88 + (163)
Compound_420		
1-fragment: 923 Frequent Subgraphs		
2 fragments		
$R(F_1)$	2,167	368
$R(F_2)$	12,552	1,996
C^G	14,337 + (191)	1,686 + (339)

Table 1: The figures with a threshold 0.2

	Sequential	Clustering
Chemical_340		
1-fragment: 844 Frequent Subgraphs		
2 fragments		
$R(F_1)$	1,425	965
$R(F_2)$	999	777
C^G	1,398 + (513)	224 + (759)
Compound_420		
1-fragment: 15,966 Frequent Subgraphs		
2 fragments		
$R(F_1)$	226,206	16,022
$R(F_2)$	116,813	17,526
C^G	341,765 + (627)	3,094+ (15,227)

Table 2: The figures with a threshold 0.1

As can be seen from Table 1 and 2, there are big benefits from the intelligent data pre-processing. Further, to discuss the impact of threshold level, let us denote the cardinality of *Global checked subgraph* set as $|C^G|$. As a result, $|C^G|$ is reduced for all data sets for all support thresholds. For example, if Chemical_340 data set is partitioned into 2 fragments, $|C^G|$ decreases from 372 for Sequential to 88 for Clustering with the support threshold 0.2 and its value reduces significantly from 1,398 to 224 with the support threshold 0.1 for Clustering.

This reduction is also present when considering other data sets as well. Its value reduces from **14,337 (Sequential)** to **1,686 (Clustering)** with the threshold **0.2** for data set Compound_420. More significantly, there is a huge decrease from **341,765** to **3,094** with the threshold **0.1** for the same data set Compound_420.

Moreover, another interesting and encouraging trend can be found in the growth of the number of *common candidates* among $R(F_i)$ for fragmented data sets. For example, this common number increases, from **71 (Sequential)** to **163 (Clustering)** for Chemical_340 with the threshold **0.2**, from **191 (Sequential)** to **339 (Clustering)** for Compound_420 with the threshold **0.2**. Especially, there is extremely gap for Compound_420 with the threshold **0.1** from **627** to **15,227**. If we reduce the support threshold to **0.05** or less, the differences will go further because of the huge number of candidates generated.

In conclusion, the figures from these two tables show that the proposed incremental *Clustering* pre-processing technique can dramatically improve the partition-based approach to graph mining. It is delivered in form of two strongly related benefits; reduction of the number of *Global candidates* requiring the final check and increase of the *common candidates* numbers that are already frequent for phase two of PartGraphMining algorithm.

7 Conclusion

This paper presents a new horizontal data partitioning framework for graph mining. We prove the completeness and correctness of mining result generated by our data partitioning approach. In addition, we show some drawbacks of a well-known PartMiner approach by demonstrating that in general, it fails to mine the correct complete set of frequent subgraphs from an input graph dataset, and the proofs of the basic theorems in the PartMiner algorithm are incorrect.

This paper gives new algorithms and the proof of the completeness of the discovered frequent subgraphs. We have done experiments to illustrate the efficiency of our data partitioning approach. More experiments with very large real data sets will be given in our future work to show the performance of our data partitioning method for graph mining. In addition, we are also designing new partitioning technique for improving the performance of mining frequent subgraphs.

8 References

- Kuramochi M., Karypis G. (2001): An efficient algorithm for discovering frequent subgraphs. *Proc. IEEE International Conf. on Data Mining (ICDM'01)*, 2001
- Nguyen S. and Orłowska M. (2005): Improvements in the Data Partitioning Approach for Frequent Itemsets Mining. *Proc. of European Conf. on Principles and Practice of Knowledge Discovery in Database (PKDD'05)*, 2005, Springer
- Nijssen S. and Kok J. (2004): A quickstart in frequent structure mining can make a difference. *ACM SIGKDD Conf. on Knowledge Discovery in Data (KDD'04)*, 2004, ACM Press
- Savasere A., Omiecinski E. and Navathe S. (1995): An efficient algorithms for mining association rules in large database. *Proc. 21th Int. Conf. Very Large Data Bases*, Swizerland, 1995
- Wang J., Hsu W., Lee M. and Sheng C. (2006): A Partition-based Approach to Graph Mining. *IEEE Int. Conf. on Data Engineering (ICDE06)*, 2006
- Wang C., Wang W., Pei J., Zhu Y. and Shi B. (2004): Scalable mining of large disk-based graph databases. *ACM SIGKDD Conf. on Knowledge Discovery in Data (KDD'04)*, 2004, ACM Press
- Yan X. and Han J. (2002): gSpan: Graph-based substructure pattern mining. *Proc. IEEE International Conf. on Data Mining (ICDM'02)*, 2002