Private Maximal Frequent Digraphs Mining Based on Spanning Tree Algorithm

Abstract : Graph mining is a significant field which has it applications in various fields such as computer and social networks, chemical industry and in bioinformatics. Variety of mining techniques are implemented in a graph to mine the useful information. The graph mining framework has been proposed in which all the frequent tree patterns from a graph database are mined initially and then the maximal frequent subgraphs from trees are constructed. The proposed method allows the user to assimilate well-develop techniques from maximal itemsets and information gained in graph mining into a new minimum spanning tree algorithm. A novel maximal spanning tree-based graph mining algorithm is proposed to mine only maximal frequent subgraphs from a large graph database. An extensive analysis of the proposed algorithm has been made to analyze its performance on graph data sets with different characteristics. The efficiency of the algorithm is also confirmed by two benchmark datasets. The experimental result shows the better performance of the proposed method in comparison with the existing graph mining algorithm.

IndexTerms **- Graph Mining, frequent pattern mining, maximal itemset, maximal frequent subgraph, spanning tree**

I. INTRODUCTION

In recent days, Graph mining has multiplied its significance and responsiveness which is considered as one of the powerful techniques for mining the data represented as a graph. The idea behind the technique is to mine the graph structures from a single or a collection of graph. Frequent subgraph mining is one of the most frequently used method in graph mining. As several fields such as computer networks, social networks, chemical bonding and bioinformatics model its complex data by employing the graph data structure which consists of set of nodes and edges. Theses nodes labeled and it stores additional information related to the dataset. The main advantage of representing this type of complex data into a graph structure is that the relationships are visualized in an efficient and understandable way than the representing them in the relational tables.

Data mining techniques are applied on the graph structure to index the graphs, to find frequent patterns, to find exact matches and even to partition the graph. Normally, the subgraphs are segregated into numerous classes and the data mining techniques are applied on the graph based data that intuitively depend on the targeted class. Another general graph mining method that is used in pattern matching is the Subgraph isomorphism with a basis of substructure matching. Furthermore, the mining measures define the characteristics of the patterns to be mined similar to conventional data mining.

Frequent subgraph mining involves discovering interesting patterns in graphs. This become the most vital process as it has various application in real world problems in social and computer networks, chemical molecules, map of roads in a country, etc. Thus the graph mining is anticipated to investigate graph data to determine interesting, surprising, and useful patterns, that can be used to understand the structure and relationship of data which helps in decision making. These graphs extracted from the real-life can be categorized as either connected or disconnected and the sample graphs are shown in Figure 1. The connected graph is closely packed as there is a path from one vertex to all the other vertex. A disconnected graph is not connected completely and thus in Figure 1, vertex A cannot be reached from the other vertices by following the edges. Only connected graphs are considered for the proposed work. The graphs can also be categorized as directed and undirected graphs based on the directions of the edges in the graph. The edges are bidirectional in the undirected graph whereas it is unidirectional in directed graphs. The mining algorithms are deliberated to work for any of the graph types based on the characteristics of the underlying application.

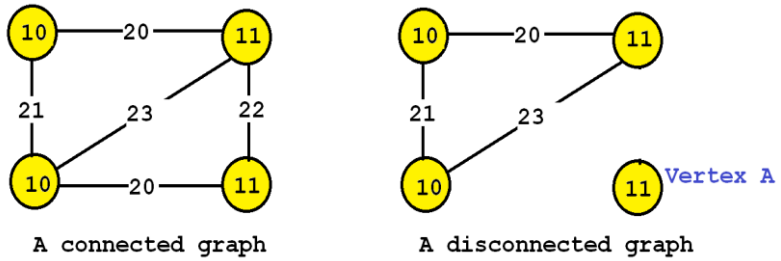
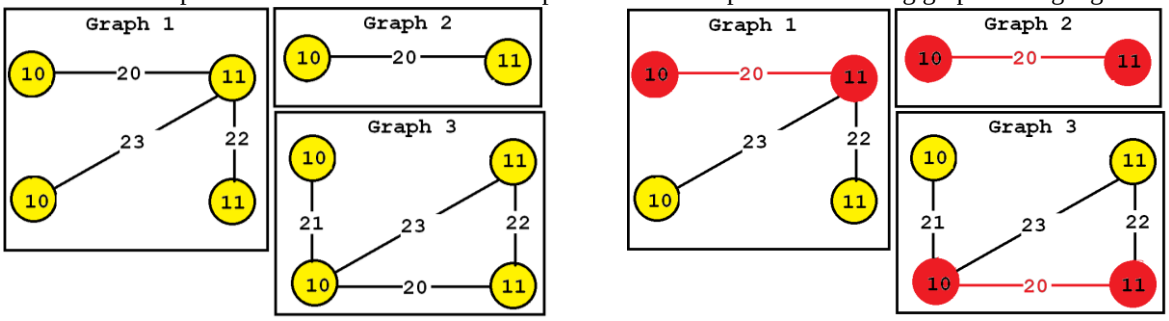


Figure 1. Types of graphs

Identifying the frequent subgraph from the set of graphs are termed as frequent subgraph mining (FSM). For mining frequent subgraphs, the user has to provide two parameters such as a set of graphs and the integer variable minimum support threshold (minsup). The algorithm processes the input graphs and the set of subgraphs are provided as an output in which its appearance count is equal to or greater than the minsup. The input graphs and the output frequent subgraphs are given in Figure 2 as an illustration with the minsup = 3. By applying the FSM algorithm on the input graphs, the output obtained is the set of all subgraphs appearing in at least 3 input graphs. Figure. 2(a) id the three input graphs and the Figure. 2(b) is the frequent subgraphs with minsup = 3 which are shown in red color.

Basically choosing a value for the minsup is another significant issue in which the higher the values, lower the subgraphs and vice versa. Thus the paper aims at proposing a novel algorithm maximal spanning tree-based graph mining to mine only maximal

frequent subgraphs of large graph dataset is suggested. The efficiency of the algorithm is also confirmed by a benchmark Cancer and HIV data set. The experimental result shows the better performance compare with existing graph mining algorithm.



a) Sample Set of Input Graphs **b) Frequent Subgraph with minsup = 3**

Figure 2. Frequent Subgraph Mining Illustration

II. LITERATURE SURVEY

The literature survey has been made for frequent subgraph mining. Most of the techniques in literature targets on static graphs. Unfortunately, the modern applications like social network based graphs uses large set of evolving dynamic graphs. Thus employing existing techniques on the dynamic datasets provides an infeasible result due to its high computational cost (Abdelhamid et al., 2017). Thus identifying frequent subgraphs from the graph become most significant research area and if the underlying data are sensitive, publishing information regarding the frequent patterns or characteristics holds significant risk with respect to privacy.

L. Sweeney (2002) and A. Machanavajjhala et al. (2006) suggested a formal defense model named k-anonymity with a set of accompanying policies for deployment. The method inspects re-identification attacks that can be identified on releases that follow to k-anonymity unless associated policies are appreciated. C. Dwork (2006) introduced a new concept called differential privacy, which, intuitively, captures the increased risk to one's privacy incurred by contributing in a database. It can achieve any desired level of privacy under this measure. R. Bhaskar et al. (2010) suggested two proficient techniques for determining the top K frequent patterns from a critical data set. These algorithms incorporate differential privacy that guarantees meaningful privacy in the presence of indiscriminate external information. The authors also introduced a utility measure that computes the accuracy of generated top K output frequent pattern. A survey on various FSM algorithms was presented by Jiang et al., (2013).

Based on the concept of differential privacy, many authors suggested several variations in FSM. This type of mining frequent subgraphs offers valuable information in a variety of applications (E. Shen and T. Yu, 2013; F. McSherry and K. Talwar, 2007; N. Li et. al, 2012). Maximal frequent subraph mining is another example for graph mining in which the supgraphs are identified in such a way that none of its super graph are frequent. An algorithm named MARGIN is a proposed to mine the maximal frequent subgraphs that traverse among promising nodes of the search space along the "border" of the uncommon and frequent subgraphs which diminishes the number of candidate patterns in the search space (Thomas et al., 2010).

An approach was suggested that begins by truncating long transactions, trading off errors introduced by the truncation with those introduced by the noise added to guarantee privacy (C. Zeng, et al., 2012; S. Xu, et al., 2015). A transaction splitting based differentially private frequent itemset mining algorithm, which is referred to as DP-Apriori was proposed. In particular, a smart weight based splitting method was anticipated to divide long transactions into sub-transactions whose cardinality is no more than a specified number of items (X. Cheng, et al., 2015; J. Han, et al., 2000). A differentially private frequent itemset mining algorithm based on the FP-growth algorithm, which is referred to as PFP-growth was proposed which consists of preprocessing and mining phases. In the preprocessing phase, to improve the utility and privacy tradeoff, a novel smart splitting method is employed to transform the database (S. Su et al., 2015; J. Lee and C. Clifton, 2014; L. Bonomi and L. Xiong, 2013).

The solution for the problem of designing a differentially private algorithm for mining maximal frequent sequences, which can not only achieve high data utility and a high degree of privacy, but also provide high time efficiency was suggested by C. Xiang et al., (2015). The author introduced new differentially private algorithm, which is referred to as DP-MFSM. A computationally efficient algorithm, called FSG, for finding all frequent subgraphs in large graph data sets was suggested by M. Kuramochi and G. Karypis (2004) and S. Chen and S. Zhou, (2013). A two-step filter-and-refinement approach that is suitable to massive parallelization within the scalable MapReduce computing mode was proposed (Lin et al., 2014). A platform for differentially private analysis of weighted datasets was given by D. Proserpio, et al., (2014). A novel differentially private FGM algorithm, which is referred to as DFG was introduced by S. Xu et al., (2016). However, these techniques adopt that the data structure of the mining task is small enough to fit in the main memory of a computer (Bhuiyan and Al Hasan, 2015). Unfortunately, the real-world graph data propagates in size and as well as quantity and thus the assumption does not hold in reality.

III. PROPOSED METHODOLOGY

A new spanning tree algorithm has been proposed that mines only maximal frequent subgraphs. The method first mines all frequent trees from a general graph database and then reconstructs all maximal subgraphs from the mined trees. The proposed method enables us to integrate well-developed techniques from mining maximal itemsets and knowledge gained in graph mining into a new algorithm. The framework of the proposed method is versatile. Depending on the particular tree mining algorithm, the search can be either breadth-first or depth-first (preferred due to its better memory utilization).

3.1 Frequent Subgraph Mining (FSM)

A graph G can be signified as a set of coordinates representing vertex V and edges E (V, E). Each vertex is associated with a label, which is drawn from a set of vertex labels. The size of a graph is defined to be the number of edges in this graph. A graph is called an i-graph if its size is i. Take the graph G1. Here consider that each edge in a graph is undirected and not associated with a label. However, the proposed solution can be extended to the case of graphs with directed and labeled edges. In FSM, to count the support of graphs, a basic operation is to determine whether a graph is a subgraph of another graph. As the computational problem of determining the subgraph (subgraph isomorphism problem) is known to be NP-complete, FSM problem is also NP-complete. FSM takes as input a graph database consisting of the records (e.g., trajectory graphs) contributed by a group of individuals, and produces the output as frequent subgraphs and their supports. However, releasing such results directly may pose considerable threat to individuals' privacy. In this paper, differential privacy is adopted to solve this problem, and study the differentially private FSM problem. In the context of differentially private FSM, we consider two graph databases as neighboring databases if they differ by at most one input record (i.e., graph). By applying differential privacy in FSM, adversaries cannot infer the presence or absence of any individual record from the results. Therefore, the privacy of individuals can be protected through the results.

3.1.1 Binary Estimation Method

Given the candidate *i*-subgraphs, to estimate the number of frequent *i*-subgraphs, a simple method is to perturb the support of each candidate subgraph, and count the number of candidate subgraphs which have noisy support above the threshold. Inspired by the method proposed for estimating the maximum size of frequent itemsets, a binary estimation method is proposed to estimate the number of frequent *I* subgraphs. Both of the binary estimation method and the method proposed in [8] use the idea of binary search to reduce the amount of added noise. It has $O(|C_i| \cdot \lceil log_2|C_i| \rceil)$ time complexity, where $|C_i|$ is the number of candidates *I* subgraphs. First obtain the noisy support of the candidate *i*-subgraph with the *m*-th largest support, where $m = \lfloor (C_i| - 1)/2 \rfloor$. If this noisy support is larger than the threshold, it means the candidate *i*-subgraphs in the upper half are all above the threshold, and we only need to consider the candidate *i*-subgraphs in the lower half in the next iteration. Similarly, if this noisy support is smaller than the threshold, we only need to further search the candidate *I* subgraphs in the upper half in the next iteration. This process continues until the number of candidates *i*-subgraphs that have noisy support above the threshold is determined.

Algorithm 1 Binary Estimation
Input: Candidate *i*-subgraphs *Ci*; Privacy budget ϵb; Threshold θ;
Output: The number of frequent *i*-subgraphs *ni*;

1: *low* ← 0; *high* ← |*Ci*| - 1;
2: **while** *low* ≤ *high* **do**
3: *m* ← ⌊(*low* + *high*) / 2⌋;
4: *sm* ← get support of the candidate *i*-subgraph with the
m-th largest support;
5: *nsm* ← *sm* + *Lap*(⌈log2|*Ci*|⌉/ ϵb);
6: **if** *nsm* == θ **then**
7: *return* |*Ci*| - *m*;
8: **else if** *nsm* > θ **then**
9: *high* = *m* - 1;
10: **else if** *nsm* < θ **then**
11: *low* = *m* + 1;
12: **end if**
13: **end while**
14: *return* |*Ci*| - 1 - *high*;

In this method, like the binary search, we only need to obtain the support of at most ⌈log2|*Ci*|⌉ candidate *i*-subgraphs. Based on the sequential composition property it can see our binary estimation method satisfies ϵb-differential privacy.

3.1.1 Conditional Exponential Method

The estimated number of frequent i-subgraphs, to identify frequent i-subgraphs from candidate i-subgraphs, a simple method is to directly use exponential mechanism to select frequent graphs from candidate subgraphs. However, this simple method might result in poor performance. The reason is explained as follows. For real-world datasets, the number of real frequent subgraphs is typically much smaller than that of candidate subgraphs generated by using the downward closure property, which makes the selections of real frequent subgraphs inaccurate. To solve this problem, a conditional exponential method has been proposed.

Algorithm 2 Conditional Exponential Method
Input: Candidate *i*-subgraphs *Ci*; Privacy budget ϵc; Threshold θ; The number of frequent *i*-subgraphs *ni*;
Output: Frequent *i*-subgraphs *Fi*;

1: $\epsilon c1 \leftarrow \beta \epsilon c$, $\epsilon c2 \leftarrow (1-\beta)\epsilon c$;
2: **for** *j* from 1 to *ni* **do**

3: $Ci' \leftarrow \emptyset$;
4: **for** each subgraph g in Ci **do**
5: $nsg = sg + Lap(2ni\epsilon c1)$;
6: **if** $nsg \geq \theta$ **then**
7: add g into Ci';
8: **end if**
9: **end for**
10: **if** Ci' is not empty **then**
11: $gj \leftarrow$select a subgraph from Ci' such that Pr[Selectingsubgraph g] $\propto \exp(\epsilon c2 \times sg2ni)$;
12: remove gj from Ci;
13: add gj into Fi;
14: **end if**
15: **end for**
16: *return* Fi;

The main idea of this method is to use the noisy support of candidate subgraphs to prune those obviously infrequent candidate subgraphs, such that the candidate set can be considerably reduced and the probabilities of selecting real frequent subgraphs can be significantly improved.

3.2 Differentially Private Subgraph Mining (DFG)

In the first phase of DFG, a frequent subgraph identification approach (referred to as FI1) is employed to privately identify frequent subgraphs in order of increasing size. A binary estimation method is proposed and employed for computing the number of frequent subgraphs for a certain size along with the conditional exponential method for improving the accuracy of identified frequent subgraphs. In the second phase of DFG, a lattice-based noisy support computation approach is devised to compute the noisy support of identified frequent subgraphs. In this approach, two methods, namely count accumulation and error-aware path construction, are proposed to improve the accuracy of the noisy supports. For the privacy budget ϵ, we divide it into three parts: $\epsilon 1$, $\epsilon 2$ and $\epsilon 3$. In particular, $\epsilon 1$ is used to estimate the maximum size Mg of frequent subgraphs, $\epsilon 2$ is used in the FI1 approach, and $\epsilon 3$ is used in the NC2 approach.

3.3 Lattice-Based Noisy Support Computation

A simple method is to uniformly assign the privacy budget to the support computation of each frequent subgraph and directly perturb their supports. However, it causes the amount of added noise to be proportional to the number of frequent subgraphs. To reduce the amount of added noise, we devise a lattice based noisy support computation (NC2) approach. In the NC2 approach, we first build a directed lattice based on the identified frequent subgraphs. In the lattice, each node vg is associated with a frequent subgraph g. A directed edge from node vg1 to node vg2 is introduced if graph g2 can be expanded to graph g1 by adding one edge. An example of a lattice formed by frequent subgraphs. In the lattice, there are several directed paths. For ease of presentation, it can be said that a graph g is on a path p if g's corresponding node vg is contained in p, and the depth of g on p is the number of nodes from the first node in p to vg. Then a count accumulation method is proposed to compute the noisy support of the graphs on a given path in the lattice. Moreover, to construct a set of paths that can cover all nodes in the lattice, an error-aware path construction method is presented.

3.3.1 Count Accumulation Method

For a path in a given lattice, we propose a count accumulation method to compute the noisy support of the graphs on this path. The main idea is to leverage the inclusion relations between the discovered frequent subgraphs and the parallel composition property of differential privacy to improve the accuracy of the results.

3.3.2 Error-aware Path Construction

Given a set of paths which cover all the nodes in the lattice, we can utilize the count accumulation method to obtain the noisy support of the graphs on each path. In addition, we can use the noise variance to measure the accuracy of the noisy supports.

3.4 Spanning Tree Subgraph Mining

The framework combines tree mining and subgraph mining; we first find all frequent trees from a graph database and then reconstruct the group of frequent subgraphs from the mined trees. There are two important components in the framework. The first is a graph partitioning method through which we group all frequent subgraphs into equivalence classes based on the spanning trees they contain. The second important component is a set of pruning procedures that is intended to eliminate the partitions completely or partially to find maximal frequent ones only.

3.4.1 Tree-based Equivalence Classes

A subtree of an undirected graph G is an acyclic connected subgraph of G. A subtree T is a spanning tree of G if T contains all nodes in G. Given a graph G, there are many spanning trees and the maximal one is defined, according to a total order defined on trees, and call it the canonical spanning tree of G. The frequent subgraph mining is theoretically broken into two steps:

(1) mine all the frequent trees from a graph database and (2) for each such frequent tree T, find all frequent subgraphs whose canonical spanning trees are isomorphic to T. Maximal frequent subgraphs can be identified among the set of frequent ones. The first step is skipped in the following discussion for two reasons. The current subgraph mining algorithms can be easily tailored to find only trees from a graph database by limiting the topology of the patterns.

3.4.2 Global and Local Maximal Subgraphs

These techniques ("pruning" techniques) dynamically remove a set of frequent subgraphs that cannot be maximal from a search space. To that end, a frequent subgraph G is defined to be a locally maximal if it is maximal in its equivalence class i.e. G has no frequent supergraph(s) that share the same canonical spanning tree as G; a subgraph is defined as globally maximal if it is maximal frequent in a graph database. Clearly, every global maximal subgraph must be locally maximal but not every local maximal subgraph is necessarily globally maximal. The proposed pruning techniques aim to avoid enumerating subgraphs which are not locally maximal. Not surprisingly, the problem of finding all locally maximal frequent subgraphs can be transformed to the well-known maximal frequent itemset mining problem. Each candidate edge is considered as an item in which the joining operation can be viewed as the union operation for itemsets and each local maximal subgraph corresponds to a maximal frequent itemset in its search space. Hence, we advocate the following pruning techniques, which are partially adapted from the maximal itemset mining and partially developed in the graph mining context, for maximal frequent subgraph mining.

An important technique related to the efficiency of the bottom-up pruning is the so-called dynamic reordering technique, which works in two ways. First, it trims infrequent candidate edges from the tail of a graph to reduce the size of the search space (an edge candidate can become infrequent after several iterations since other edges are incorporated into the patterns). Second, it rearranges the order of the elements in the tail according to their support value. However, without the spanning tree framework, applying dynamic ordering is very difficult in any of the current subgraph mining algorithms, which intrinsically have a fixed order of adding edges to an existing pattern for various performance considerations.

IV. EXPERIMENTAL ANALYSIS

The experiments are conducted on a PC with Intel Core i5 CPU (3.0GHz) and 8GB RAM. Due to the randomness of the algorithms, we run every algorithm ten times and report the average results. In all these experiments, the relative threshold (i.e., the fraction of records that contain a pattern) is used. The absolute threshold can be obtained by multiplying the relative threshold by the number of input records. Three publicly available real datasets are used in our experiments. Specifically, dataset Cancer includes structures of human tumor cell lines, dataset HIV contains molecules tested against HIV virus. In the experiments, two widely used metrics are employed to evaluate the performance of the algorithms. The first is F-score, which is used to measure the utility of discovered frequent subgraphs. The second is Relative Error (RE), which is used to measure the error with respect to the true supports.

The f-score value for the proposed and various existing algorithms such as differentially oprivate frequent subgraph mining (DFG), Distributed Frequent Pattern Mining (DFPM) for the cancer dataset is shown in Table 1. The f-score is measured by varying various user threshold as 0.4, 0.45, 0.5, 0.55 and 0.6. The corresponding graph is given in Figure 3.

Table 4.1: Comparison of f-score for cancer dataset with varied user threshold

Algorithms	User Threshold				
	0.4	**0.45**	**0.5**	**0.55**	**0.6**
DFG	0.71	0.79	0.81	0.88	0.91
DFPM	0.21	0.28	0.31	0.34	0.39
MST-GM	0.75	0.81	0.83	0.89	0.94

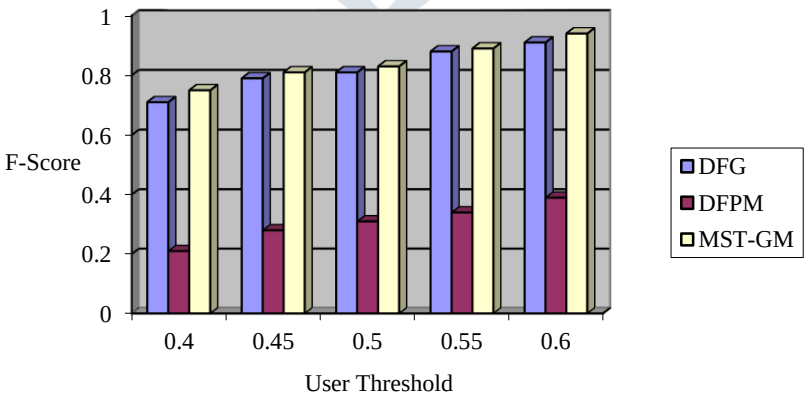


Figure 3. Comparison of F-Score for cancer dataset with varied user threshold

The relative error value for the proposed and various existing algorithms for the cancer dataset is shown in Table 2. The relative error is measured by varying various user threshold as 0.4, 0.45, 0.5, 0.55 and 0.6. The corresponding graph is given in Figure 4.

Table 4.2: Comparison of relative error for cancer dataset with varied user threshold

Algorithms	User Threshold				
	0.4	**0.45**	**0.5**	**0.55**	**0.6**
DFG	0.021	0.023	0.026	0.027	0.029
DFPM	0.16	0.14	0.1	0.08	0.06
MST-GM	0.014	0.015	0.017	0.018	0.019

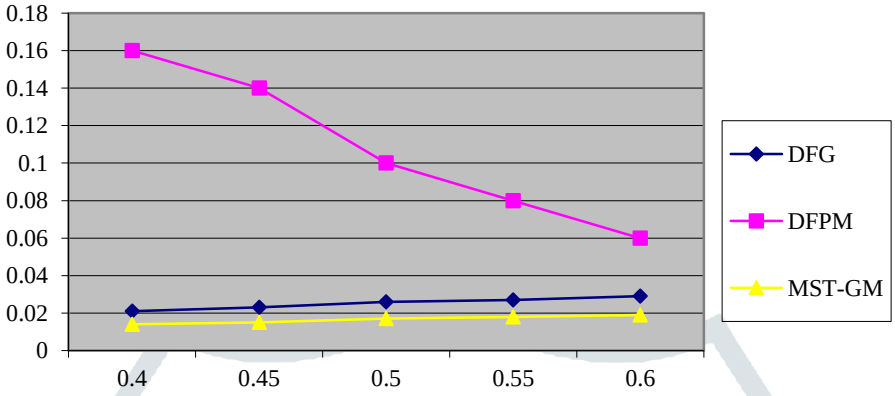


Figure 4. Comparison of relative error for cancer dataset with varied user threshold

The f-score value for the proposed and various existing algorithms for the HIV dataset is shown in Table 3. The f-score value is measured by varying various user threshold as 0.4, 0.45, 0.5, 0.55 and 0.6. The corresponding graph is given in Figure 5.

Table 4.3: Comparison of F-Score for HIV dataset with varied user threshold

Algorithms	User Threshold				
	0.4	**0.45**	**0.5**	**0.55**	**0.6**
DFG	0.81	0.84	0.89	0.91	0.92
DFPM	0.25	0.28	0.31	0.39	0.42
MST-GM	0.83	0.87	0.91	0.93	0.95

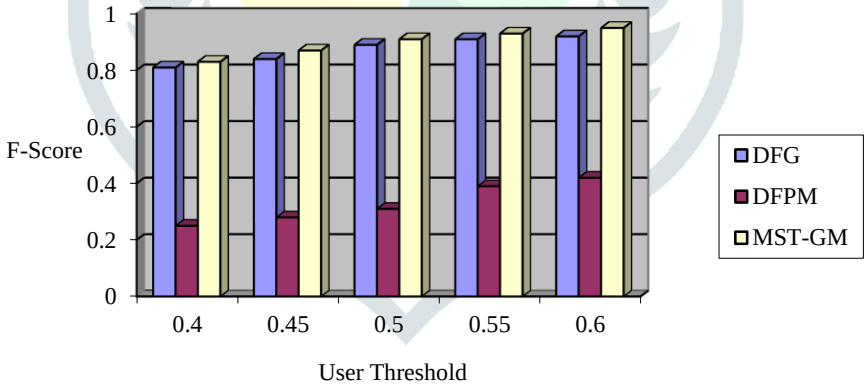


Figure 5. Comparison of F-Score for the HIV dataset with varied user threshold

The relative error value for the proposed and various existing algorithms for the HIV dataset is shown in Table 4. The relative error is measured by varying various user threshold as 0.4, 0.45, 0.5, 0.55 and 0.6. The corresponding graph is given in Figure 6.

Table 4.4: Comparison of relative error for HIV dataset with varied user threshold

Algorithms	User Threshold				
	0.4	**0.45**	**0.5**	**0.55**	**0.6**
DFG	0.12	0.08	0.06	0.043	0.045
DFPM	0.024	0.026	0.028	0.029	0.029
MST-GM	0.011	0.013	0.018	0.021	0.025

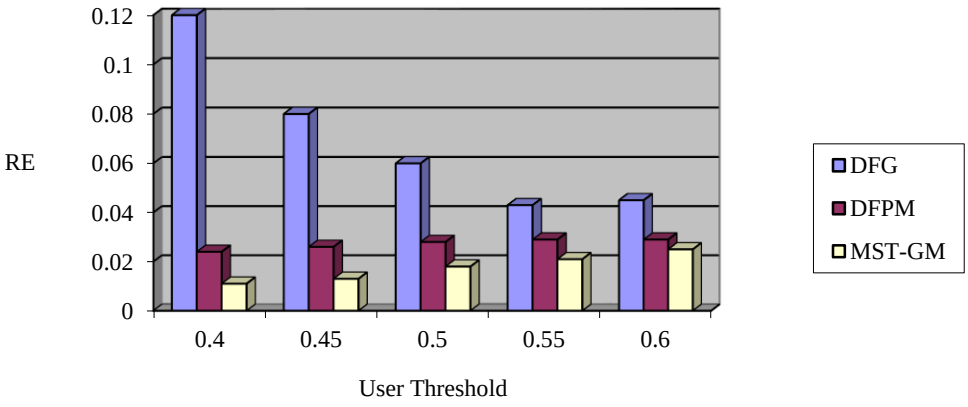


Figure 6. Comparison of F-Score for HIV dataset with varied user threshold

The f-score value for the proposed and various existing algorithms for the cancer dataset is shown in Table 5. The f-score is measured by varying the Top-K itemset as 20, 40, 60, 80 and 100. The corresponding graph is given in Figure 7.

Table 4.5: Comparison of f-score for cancer dataset with varied top-k itemsets

Algorithms	Top-K				
	20	**40**	**60**	**80**	**100**
DFG	0.91	0.89	0.85	0.81	0.80
DFPM	0.40	0.35	0.28	0.27	0.26
MST-GM	0.93	0.91	0.88	0.85	0.82

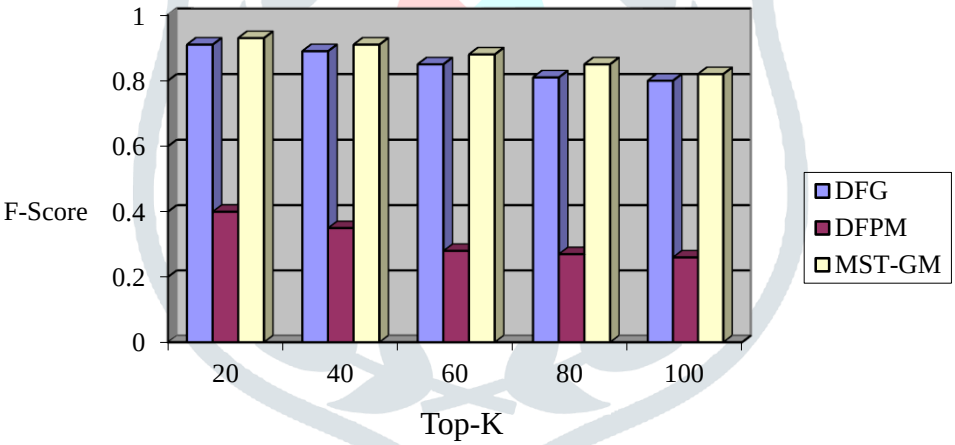


Figure 7. Comparison of f-score for cancer dataset with varied top-k itemsets

The f-score value for the proposed and various existing algorithms for the cancer dataset is shown in Table 6. The f-score is measured by varying the Top-K itemsets as 20, 40, 60, 80, 100. The corresponding graph is given in Figure 7.

Table 4.6: Comparison of f-score for HIV dataset with varied top-k itemsets

Algorithms	Top-K				
	20	**40**	**60**	**80**	**100**
DFG	0.93	0.91	0.84	0.82	0.81
DFPM	0.42	0.37	0.29	0.28	0.27
MST-GM	0.95	0.93	0.91	0.88	0.85

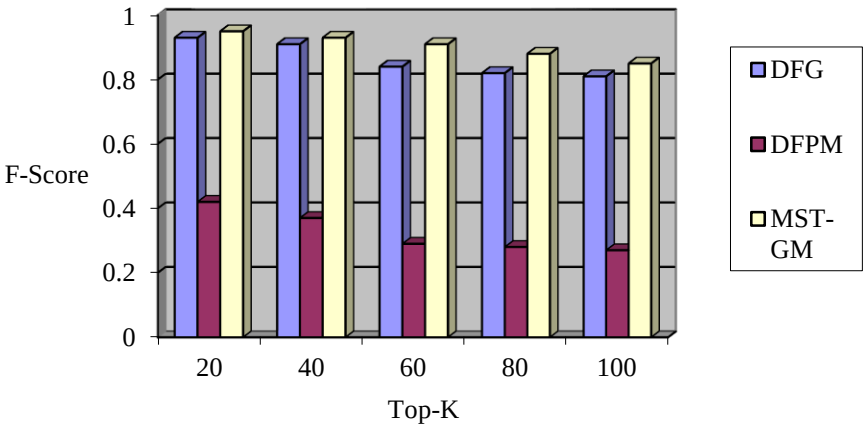


Figure 8. Comparison of f-score for HIV dataset with varied top-k itemsets

The runtime for the proposed and various existing algorithms for the cancer dataset is shown in Table 7. The runtime is measured by varying the Top-K itemsets as 20, 40, 60, 80, 100. The corresponding graph is given in Figure 9.

Table 4.7: Comparison of Runtime for cancer dataset with varied top-k itemsets

Algorithms	Top-K				
	20	**40**	**60**	**80**	**100**
DFG	102	145	168	189	203
DFPM	98	123	144	165	189
MST-GM	73	96	104	112	123

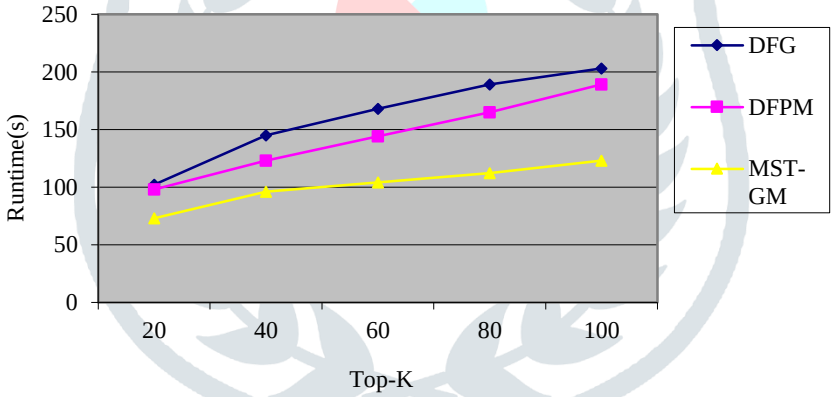


Figure 9. Comparison of Runtime for cancer dataset with varied top-k itemsets

The runtime for the proposed and various existing algorithms for the HIV dataset is shown in Table 8. The runtime is measured by varying the Top-K itemsets as 20, 40, 60, 80, 100. The corresponding graph is given in Figure 10.

Table 4.8: Comparison of Runtime for HIV dataset with varied top-k itemsets

Algorithms	Top-K				
	20	**40**	**60**	**80**	**100**
DFG	112	134	167	181	195
DFPM	88	91	124	137	144
MST-GM	65	76	82	104	112

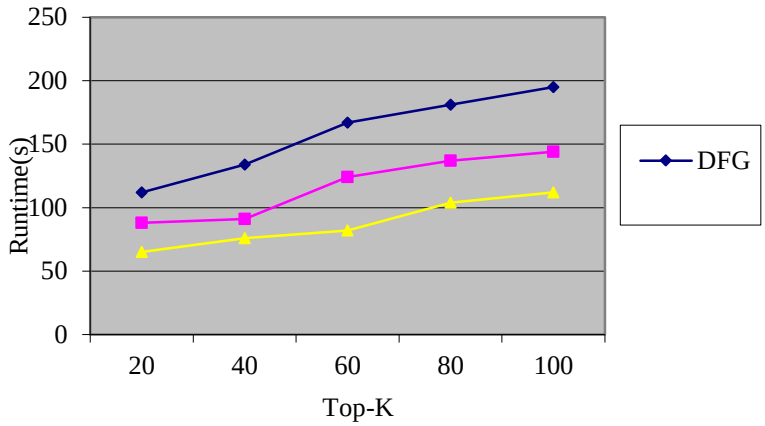


Figure 10. Comparison of Runtime for HIV dataset with varied top-k itemsets

From the experimental analysis, it is clear that the proposed method produces better results for f-score, relative error and runtime for both cancer and HIV datasets.

V. CONCLUSION

The graph mining framework in which we first mine all frequent tree patterns from a graph database and then construct maximal frequent subgraphs from trees. The propose method enables us to integrate well-develop techniques from maximal itemsets and information gained in graph mining into a new algorithm called minimum spanning tree. A novel maximal spanning tree-based graph mining algorithm to mine only maximal frequent subgraphs of large graph databases, perform an extensive analysis of the propose algorithm and analyze how its performance on graph data sets with different characteristics. The efficiency of the algorithm is also confirmed by a benchmark data set. The experimental result shows the better performance compare with existing graph mining algorithm. The maximal subgraphs will help us to investigate demanding applications such as finding structure patterns from proteins in the future. Another future research is speeding up the FAS mining process, where some kind of vectoral or parallel processing could be used for creating more efficient sub-graph mining algorithms. In the future we hope to include other innovative ways of browsing and analyzing the lattice of fragments, and we want to improve scalability where possible.

REFERENCES

[1] Bhuiyan, M.A. and Al Hasan, M., 2015. An iterative MapReduce based frequent subgraph mining algorithm. IEEE Transactions on Knowledge and Data Engineering, 27(3), pp.608-620.
[2] L. Sweeney, “k-anonymity: A model for protecting privacy,” Int. J. Uncertain. Fuzziness Knowl.-Base Syst, 2002.
[3] Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, “l-diversity: Privacy beyond k-anonymity,” in ICDE, 2006.
[4] A. Dwork, “Differential privacy,” in ICALP, 2006.
[5] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta, “Discovering frequent patterns in sensitive data,” in KDD, 2010.
[6] Jiang, C., Coenen, F. and Zito, M., 2013. A survey of frequent subgraph mining algorithms. The Knowledge Engineering Review, 28(1), pp.75-105.
[7] E. Shen and T. Yu, “Mining frequent graph patterns with differential privacy,” in KDD, 2013.
[8] F. McSherry and K. Talwar, “Mechanism design via differential privacy,” in FOCS, 2007.
[9] N. Li, W. Qardaji, D. Su, and J. Cao, “Privbasis: frequent itemset mining with differential privacy,” in VLDB, 2012, pp. 305–316.
[10] Thomas, L.T., Valluri, S.R. and Karlapalem, K., 2010. Margin: Maximal frequent subgraph mining. ACM Transactions on Knowledge Discovery from Data (TKDD), 4(3), p.10.
[11] C. Zeng, J. F. Naughton, and J.-Y. Cai, “On differentially private frequent itemset mining,” in VLDB, 2012.
[12] S. Xu, S. Su, X. Cheng, Z. Li, and L. Xiong, “Differentially private frequent sequence mining via sampling-based candidate pruning,” in ICDE, 2015.
[13] X. Cheng, S. Su, S. Xu, and Z. Li, “Dp-apriori: A differentially private frequent itemset mining algorithm based on transaction splitting,” Computers & Security, 2015.
[14] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation,” in SIGMOD, 2000.
[15] S. Su, S. Xu, X. Cheng, Z. Li, and F. Yang, “Differentially private frequent itemset mining via transaction splitting,” TKDE, 2015.
[16] J. Lee and C. Clifton, “Top-k frequent itemsets via differentially private fp-trees,” in KDD, 2014.
[17] L. Bonomi and L. Xiong, “A two-phase algorithm for mining sequential patterns with differential privacy,” in CIKM, 2013.
[18] C. Xiang, S. Su, S. Xu, P. Tang, and Z. Li, “Differentially private maximal frequent sequence mining,” Computers & Security, 2015.

[19] M. Kuramochi and G. Karypis, "An efficient algorithm for discovering frequent subgraphs," TKDE, 2004.
[20] S. Chen and S. Zhou, "Recursive mechanism: towards node differential privacy and unrestricted joins," in SIGMOD, 2013.
[21] Lin, W., Xiao, X. and Ghinita, G., 2014, March. Large-scale frequent subgraph mining in MapReduce. In 2014 IEEE 30th International Conference on Data Engineering (pp. 844-855). IEEE.
[22] D. Proserpio, S. Goldberg, and F. McSherry, "Calibrating data to sensitivity in private data analysis: A platform for differentially private analysis of weighted datasets," in VLDB, 2014.
[23] S. Xu, S. Su, L. Xiong, X. Cheng, and K. Xiao, "Differentially private frequent subgraph mining," in ICDE, 2016.
[24] Abdelhamid, E., Canim, M., Sadoghi, M., Bhattacharjee, B., Chang, Y.C. and Kalnis, P., 2017. Incremental frequent subgraph mining on large evolving graphs. IEEE Transactions on Knowledge and Data Engineering, 29(12), pp.2710-2723.