Is Frequent Pattern Mining useful in building predictive models?

Thashmee Karunaratne

Department of Computer and Systems Sciences,
Stockholm University, Forum 100, SE-164 40 Kista, Sweden
si-thk@dsv.su.se,

Abstract. The recent studies of pattern mining have given more attention to discovering patterns that are interesting, significant, discriminative and so forth, than simply frequent. Does this imply that the frequent patterns are not useful anymore? In this paper we carry out a survey of frequent pattern mining and, using an empirical study, show how far the frequent pattern mining is useful in building predictive models.

Keywords: Frequent pattern mining, predictive models, chemoinformatics

1 Introduction

The amount of available literature related to frequent pattern mining (FPM) reflects the reason for calling it a focused theme in data mining [1,2]. Due to the evolution of the modern information technology, collecting, combining, transferring and storing huge amounts of data could be done at very low costs. As a consequence more sophisticated methods of mining associations from such data have been introduced, expanding the typical problem of analyzing the associations of commonly found items in a shopping list [3] into discovering interesting and useful patterns from large, complex and dense databases. A frequent pattern in a pattern database may be a set of items, or a subsequence, a sub-tree or a sub-graph that appears across the database with a frequency that is not less than a pre-defined threshold value [2]. A *frequent item set* is a set of items appearing together more frequently, such as bread and butter, for example, in a transaction database [2]. A frequent sequential pattern may be a frequently seen subsequence from a database of an ordered list of items [4], while frequent sub-trees of a database of trees may represent the frequently linked web pages in a weblog database. Frequent sub-graphs in a graph database may be, for example, a more frequently appeared motif in a chemoinformatics database. Such patterns discovered by FPM algorithms could be used for classification, clustering, finding association rules, data indexing, and other data mining tasks.

Ever since the introduction of the first FPM algorithm by Agrawal et.al.[3] in 1993, plenty of approaches related to pattern mining problem are added, including new algorithms as well as improvements to existing algorithms, making the FPM a scalable and a solvable problem [5]. We can find studies dealing with complexity issues, database issues, search space issues, scalability issues of FPM in abundance. After about two

decades since the concept emerged, with thousands of research publications accumulated in the literature, one may wonder whether the problem of FPM is solved at an acceptable level on most of the data mining tasks [2]. On the other hand there is a common criticism about the FPM approaches for (1) producing unacceptably large set of frequent patterns, which limits the usage of the patterns, (2) contains large amount of redundant patters, (3) most of the frequent patterns are not significant, i.e, they do not discriminate between the classes [6, 19]. The arguments of these studies return a fair question; are frequent patterns not useful anymore? In this paper we investigate, with the support of an empirical evaluation using a representative set of pattern mining algorithms, how far the frequent patterns are useful in building predictive models.

The rest of the paper is organized as follows; the next section gives a compact description of the theory of FPM and pioneered related works, including a brief description of the problem we raise. We introduce the empirical study in section 3, together with descriptions of the methods used. The outcome of the study is reported in section 4. Finally section 5 concludes the paper, with some discussion to possible further work.

2 Theory and Related work

Frequent pattern mining problem, when it first attracted the researcher's attention, was purely on frequent item set mining. The work of Agrawal et.al [3] was motivated by the need of analyzing the so called supermarket transaction data for examining the customer behavior in terms of the products they purchased from a supermarket. A frequent set in this framework was the set of items customers often purchased together. Formally, suppose we are given a set of items $\mathcal{I}$. A *transaction* over $\mathcal{I}$ is a couple $T=(tid, I)$, where *tid* is the identifier and $I \subseteq \mathcal{I}$ is some subset of items. For itemset $\mathcal{I}$, a transaction including I is called an *occurrence* of I. $|T(I)|$ is called the frequency of I; and denoted by *frq(I)*. The *support* of I, say *support(I)* is the *frq(I)* over the transaction database. Therefore the problem of frequent item set mining is: given a *minimum support*, determine all item sets I such that $support(I) \geq minimum\ support$. Discovery of frequent itemsets is a two step procedure, the search of frequent patterns and support count. The apriori principle, which is introduced in [3] and substantially improved later, limits the search space using the monotonicity property of the support of sets [5]. The methods using horizontal [3,20] and vertical [21] layouts for organizing the transaction database support efficient scanning of the database. Brief descriptions of various studies related to improving efficiency, scalability, usage of computer memory and so forth, in the frequent itemset mining algorithm could be found in [2, 22].

Despite the optimizations introduced to improve the performance of the frequent itemset mining algorithm, it often generates substantially large amount of frequent itemsets, which limit the possibility of using them meaningfully. To overcome this problem, the concepts of maximal frequent itemsets and closed frequent itemsets were proposed. The *maximal frequent itemset* is a frequent itemset I, which is included in no other frequent itemset. A frequent itemset is called a *closed frequent itemset* if there does not exist a super frequent itemset that has the same support. The extra computational burden related to determining whether a frequent itemset is closed or maximal are being solved by approaches such as keeping the track of the Transaction ID lists, using hash functions [23], or maintaining a frequent pattern tree similar to FP-tree [24]. Maximal

frequent itemset mining approaches use techniques such as vertical bit maps for the Transaction ID list [25], or apriori based pruning for search space [26].

Mining frequent sequences was first introduced by [27], followed by [28,29]. A *sequence* $s=\{t_1..t_i...t_n | t_i \subseteq I\}$ where I is the itemset, is an ordered list of items. Given a sequence database $D=\{s_1,s_2,...,s_m\}$, the number of occurrences of s in D is called the *support* of s in D, and if it exceeds a pre-defined value, s is called a *frequent sequential pattern* in D. Methods of finding frequent sequential patterns in a sequential pattern database include use of sliding windows over the sequences, using regular expressions or canonical transformations and so on. Techniques such as closed [31] and maximal [30] frequent sequences are used for producing a subset of frequent sequences, which could be more meaningful for reuse.

Among the various pattern mining approaches, graphs have attracted the attention of the frequent pattern mining research due to the fact that there is an immense increase in use of graphs in modeling scientific data [18], such as web pages, molecules, social networks, protein interaction networks and image representation in computer vision etc. A graph is a quintuple $G= \{V, E, \lambda, \Sigma\}$, where V is the set of vertices, $E \subseteq V \times V$ is the set of edges and $\lambda: V \cup E \rightarrow \Sigma$ is the labeling function. Further, if (v_i, v_j) is ordered then, the graph is directed; or undirected otherwise. A graph $G_1\{V_1, E_1, \lambda, \Sigma\}$ is a *subgraph* of G if and only if $V_1 \subseteq V$ and $\forall (v_i, v_j) \in V_1, (v_i \times v_j) \in E_1 \Leftrightarrow (v_i \times v_j) \in E$. Two graphs G_1 and G are called *isomorphic* iff G_1 is a subgraph of G and G is a subgraph of G_1. Further, let the database D contains a collection of graphs. The *support* of a graph G in D is the number of occurrences of G in D divided by the number of elements in D. If the support exceeds a pre-defined value, G is a *frequent graph* in D. A graph is a tree when the hierarchical order of the nodes is preserved. Mining trees, graphs and lattices are collectively referred as mining structural patterns [2], which include searching for frequent sub-structures and computing their supports. Algorithms that are found in the literature for frequent sub-structure mining mainly falls into two categories, apriori based and pattern-tree based. The first subgraph miner AGM [32], followed by [33] uses apriori based breadth-first search for discovering candidate subgraphs. The pattern-tree based approaches use a depth-first search strategy [7, 34, 35]. Approaches of discovering maximal [36, 37] and closed [38] graphs have shown that a representable subset of frequent graphs is more useful. An extensive overview of different graph based data mining methods can be found in [39].

The essential drawback of FPM is the generation of overwhelmingly large set of patterns, which leads to problems in storage and subsequent analyses. Methods to limit the huge set of frequent patterns such as, use of constraints such as mining top k patterns [8], pattern compression [9] and summarization [10, 11], using interestingness measures [12,13] and correlations [14, 15], have shown successful. Methods for searching significant patterns [19] and methods for filtering subsets from the frequent patterns [6] also have gained attention in the field. Slightly different approach for finding interesting subgraphs using the maximum description length is discussed in [16]. Studies using slightly different concepts, such as using constraint programming [17], and evolutionary computing [40] for mining interesting patterns are also found among the pattern mining methods. Non-sophisticated algorithms such as [18] use simple data structures that could support improved efficiency.

2.1 Problem description

The efficiency and scalability of the frequent pattern mining algorithms is well studied [2]. Nevertheless, it is obvious that the runtime is not the sole factor that measures the quality of the FPM approach [41]. The common criticism of the FPM algorithms today is the generation of prohibitively large set of frequent patterns [8-19], which limits the avenues of exploring or reusing them. In the classical example of the market basket analysis, finding bread and butter in a frequent itemset supports the retailer to decide the arrangement of shelves to ensure increased convenience to the customer. Therefore, it is equally important that mined patterns could be used in a meaningful way. Limiting the number of discovered patterns using high thresholds may reveal only the common knowledge while low thresholds results an explosion of discovered patterns. This and several other factors such as, existing approaches of FPM concern on efficiency of the mining process and do not give much attention to analyze how to use the discovered patterns effectively, can result a grey region in the field that how and in which ways the FPM is useful.

3 Experimental setup

The experiments are setup for comparing the FPM algorithms in terms of efficiency, scalability and the usefulness of discovered patterns in building predictive models.

3.1 Data sets

Eighteen datasets from the domain of Medicinal Chemistry, which are publicly available [42] and concern modeling tasks of various biochemical activities, such as absorption, distribution, metabolism, excretion and toxicity (ADMET) [43] are included in the experiments. The Table 1 provides the descriptions of these datasets.

3.2 Methods explored

In this study, we have applied an approach for transforming a graph in to a list of itemsets, so that frequent item set mining methods could be used in graph mining, i.e., assume a graph $G=(V, E, \lambda, \Sigma)$ with usual notation (as described in section 2). The *edge list* of G is defined as $L = (v_i, v_j, e_k \mid \forall v_i, v_j \in V \text{ and } e_k \in (v_i, v_j))$. Let $l \in L$, then $\forall l \in L$ are distinct if and only if λ is injective. The edge list L is similar to the Itemset I described in section 2. Even though the itemset mining itself is exponential with respect to the number of items, the generation of the frequent itemset is based on a lexicographical order of items, and therefore, a frequent itemset is generated only once during the mining process, in contrast to subgraphs, which requires growing the search tree in several ways adding nodes or edges. Further, determining whether a graph is a subgraph in another graph requires a subgraph isomorphism test, while determining the same in an itemset is trivial. Therefore the ability of using frequent itemset mining algorithms on graph data is beneficial. Further, if the predictive models built using features from frequent itemset mining algorithms

perform equally well as the graph mining algorithms this transformation would be useful indeed. Brief description the methods we use in this study are given below.

gSpan – **g**raph-based **S**ubstructure **p**atter**n** mining [7], is one of the most popular and most widely compared frequent subgraph mining algorithm. Since gSpan doesn't employ any restriction to the discovered subgraphs other than frequency, the set of discovered subgraphs using gSpan usually grow exponentially, with decreasing threshold values.

GASTON – **G**r**A**ph /**S**equence/ **T**ree extracti**ON** algorithm [37] has a unique feature that, it could discover paths, free trees and rooted trees separately. Similar to gSpan, embedding lists are used to store temporary occurrences of a graph and a join operation is used for refinements of previous legs. New embedding lists are constructed for legs that are not presented in the predecessor graphs. GASTON's main criticism of consuming high memory is due to this process.

GraphSig – GraphSig [19] mines significant patterns from large graph data. The significance of the graphs discovered by GraphSig is tested using a probability measure related to the support of each graph in the graph database. If this p-value lies below a pre-defined threshold, the graph is defined as significant. GraphSig is admired for its ability to mine subgraphs at very low supports.

MoFa – **Mo**lecular **fra**gment miner [34] is an algorithm that can be used for discovering frequent molecular fragments in chemo-informatics databases. MoFa discovers molecular fragments, which best separate molecules that belong to different classes. However, in order to restrict the search space, the algorithm considers only connected substructures.

SUBDUE – SUBDUE [16] uses the minimum description length (MDL) principle to measure the interestingness of the sub-graphs discovered. SUBDUE employs a step-by-step procedure, which starts from single nodes and performs a computationally constrained beam search in order to expand it by another node or edge. Therefore it typically generates a small number of sub-graphs that best compress the dataset.

MFI – **M**aximal **F**requent **I**temset [44] is an approach that uses the maximal frequent itemset mining methods for discovering frequent subgraphs (which are not necessarily connected) from graph databases. MFI algorithm requires transformation of graph data into an edge list as described above. The MAFIA algorithm [25] is used on the edge lists to discover the maximal frequent itemsets. MAFIA computes frequent itemsets using a simple depth-first search strategy over the lexicographic tree and a dynamic ordering on the candidate list in order to remove or rediscover candidates, along with an apriori based stopping condition. Maximality of the discovered frequent itemsets is guaranteed by the superset enumeration.

CPIM (CP) – **C**onstraint **P**rogramming for **I**temset **M**ining [15], uses a correlation measure to discover the significant patterns. Support pruning with respect to the space of classlabels (a bounded PN space) is used together with the two constraints, namely, coverage and support, to find correlated patterns.

In addition to the methods described above, **GASTION-P** (mining frequent paths only) and **SMFI** (mining maximal frequent Itemsets using positive examples in the dataset), are also used for the experiments. The *edgelists* of the graphs of the molecular datasets is used as input data to MFI, SMFI and CP.

All the experiments are carried out using a HP EliteBook, with two Core2 Duo Intel processors 2.40GHz each, and 2.9GB main memory with Ubuntu 10.4.

4 Empirical evaluation

In this study we search for answers to the following questions.

Q1: How efficient are the frequent/significant pattern mining approaches?

This is the most widely studied question in FPM methods. Nevertheless, a brief qualitative comparison of runtime and memory consumption of the methods during our experiments is presented herewith.

gSpan and GASTON was efficient for high thresholds values (50%- 5%). Nevertheless, lower thresholds failed to produce frequent pattern sets within 24 hours. MFI, with our graph transformation method, discovered maximal frequent itemsets in minutes, with almost same efficiency for all seven thresholds values we used between 50% - 1%. In contrast, GraphSig and MoFa were slow miners. GraphSig exhausted the memory for datasets ache and bzr. MoFa did not complete the algorithm for the AMPH1dataset in 24 hours and took about 18 hours to complete the pattern discovery in the ache dataset. GASTON exhausted memory for AMPH1, ache and thr, and did not complete the discovery of features at low frequencies within 24 hours. The results are obvious, since MoFa is shown slow in many experiments [45] and GASTON is often criticized for its high use of memory, due to the fashion of creating the embedding lists. The feature discovery using constraint programming (CP) was slower than MFI, yet quicker than GrapgSig and MoFa, and also quite linear with the decreasing thresholds. GraphSig was the worst in average memory consumption followed by GASTON.

Q2: How scalable are the frequent/significant pattern mining approaches?

It is a known fact that, the frequent pattern mining algorithms scales up with deceasing thresholds. Both gSpan and GASTON could not build the pattern set, on thresholds under 5% within the specified time. Yet MFI and CP with using the *edge list* as the itemset, scaled linearly. The Table 2 provides average number of features discovered under each threshold value, for the AI dataset. For the same dataset GraphSig, SUBDUE and MoFa discovered 155, 50 and 461 patterns respectively.

Table 2. #of patterns discovers with respect to decreasing threshold values for AI dataset

	Thresholds						
Method	0.5	0.4	0.2	0.1	0.05	0.025	0.01
gSpan	1112	2317	51833	1977857	143610000	*	*
GASTON	1110	2315	51831	1977855	17341602	*	*
GASTON-p	260	379	1171	3519	7072	16999	*
SMFI	1	1	5	9	5	6	8
MFI	1	1	6	6	7	7	12
CP	9	13	15	15	15	15	15

Due to the excessive number of patterns discovered by gSpan GASTON and GASTON-p, we omitted those methods during further experiments.

Q3: How far the discovered patterns are useful in building predictive models?

The usefulness of the discovered patterns is tested using both classification and regression models. Two different state-of the-art methods in building predictive models in chemoinformatics, namely, the so-called **selma** parameters [46] and the **ecfi** fingerprints [47] are used for comparison of performance with the FPM methods. These methods are ideal for performance comparison since both Selma and ECFI are molecular feature sets that could be used as feature vectors in building predictive models. Selma calculates a collection of commonly used 2-dimensional molecular descriptors representing molecular size, flexibility, connectivity, polarity, charge, and hydrogen bonding potential. The ecfi fingerprints are topological fingerprints that represent substructural fragments of various sizes and they encode (capture) the 2D or 3D features of the molecule as an array of binary values or counts. Interested readers can refer [48] for further details on the descriptor sets and their usage.

Classification models are built using random forests, support vector machines with the RBF kernel with complexity 2, and the polynomial kernel with complexity 2, and the k-nearest neighbor as implemented in the WEKA data mining toolkit [49]. We considered accuracy as the performance criterion, which is estimated using 10-fold cross-validation. Since the intention of the study is to draw conclusions of the relative performance of the descriptor sets without reference to a specific learning algorithm, we randomly choose one of the three learning algorithms, namely the Support vector machine with non-polynomial kernel. For regression, we have used the support vector machine with the nonlinear polynomial kernel with complexity 2 and the RBF kernel with complexity 2, as implemented in WEKA [49]. The root mean squared error (RMSE) where chosen as performance criterion for the regression tasks, again using 10-fold cross validation. The class labels are used for feature construction in MoFa, GraphSig, SUBDUE, and CP. Parameter optimizations of the algorithms, when required, were done by cross-validation on the training sets, where the optimized parameters were used in the test set. The same training and test folds were used for all methods. Again, we choose one of the learning algorithms randomly.

Results (the average RMSE values) from regression models and the average classification accuracies are presented in Fig. 1 and 2 respectively.

To evaluate the results statistically, a null hypothesis is formed stating that there is no significant difference between the performance values yielded by different feature construction methods. The significance of the differences of the regression errors and classification accuracies is tested by comparing the ranks (relative performance of the methods) using the Friedman test [50]. The Friedman test rejected the null hypothesis for this experiment. Therefore, further tests were conducted to identify pairs of methods for which the difference in performance is significant. The average ranks are used for pair-wise tests for significance, based on the Nemenyi test [50]. In applying this criterion, a method that fails to produce a feature set is assigned the highest rank, which corresponds to the worst performance. Nemenyi test [50] concluded that the pairs corresponding to dark cells in the Table 3 and Table 4 are significantly different in their performance (positive values corresponds to the methods in column label outperforms the methods in the row label, and negative values vice-versa).

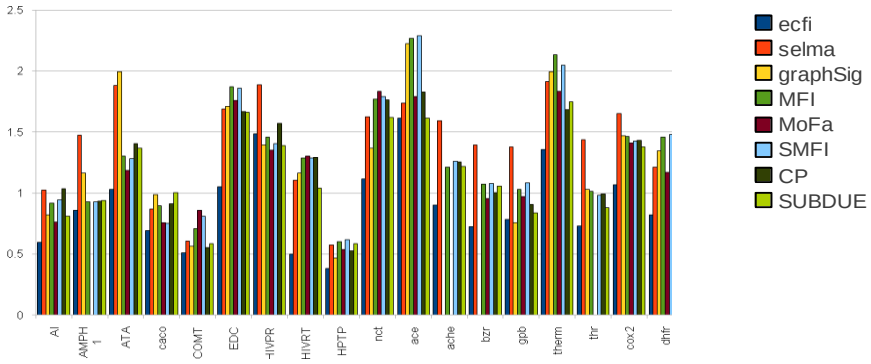

Fig. 1. Average RMSE values of regression models

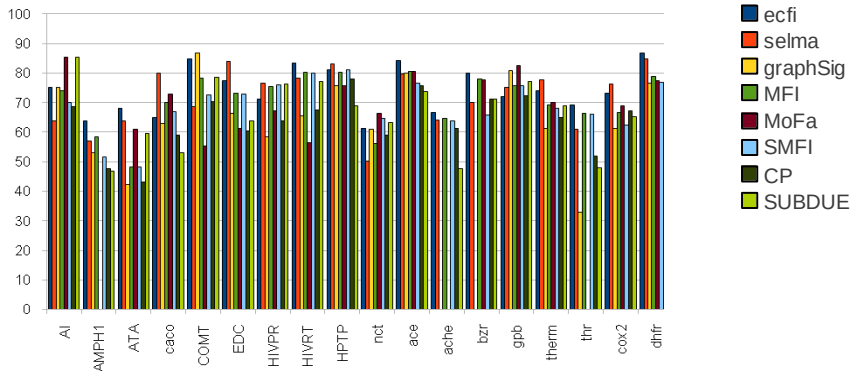

Fig. 2. Performance measures (average classifier accuracies) of classification models

Table 3. Differences of average ranks of performance of regression models.

	ECFI	selma	graphSig	MFI	MoFa	SMFI	CP	SUBDUE
Selma	4.28							
graphSig	3.67	-0.61						
MFI	4.17	-0.11	0.50					
MoFa	3.33	-0.94	-0.33	-0.83				
SMFI	4.28	0.00	0.61	0.11	0.94			
CP	3.33	-0.94	-0.33	-0.83	0.00	-0.94		
SUBDUE	2.22	-2.06	-1.44	-1.94	-1.11	-2.06	-1.11	

The regression models built using ECFI descriptors outperform all the other models except for those generated by SUBDUE. The latter method is however not significantly better than the rest of the substructure discovery methods. The ECFI descriptors based classifier models significantly outperform graphSig and CP. Further, the frequent itemset mining algorithms works equally well as the graph mining methods on graph data. Also, it is worth to note that while some of the graph mining methods had issues in either producing patterns or memory overflow, or delays in discovery process, the frequent itemset mining algorithms discovered patterns for all the data sets with more or less equal (and low) runtime.

Table 4. Differences of average ranks of performance of classification models.

	ECFI	Selma	graphSig	MFI	MoFa	SMFI	CP	SUBDUE
Selma	1.14							
graphSig	3.39	2.25						
MFI	1.06	-0.08	-2.33					
MoFa	2.14	1.00	-1.25	1.08				
SMFI	2.22	1.08	-1.17	1.17	0.08			
CP	3.72	2.58	0.33	2.67	1.58	1.50		
SUBDUE	2.50	1.36	-0.89	1.44	0.36	0.28	-1.22	

5 Conclusions

There has been no argument that frequent pattern mining is a very popular branch of data mining, and several successful approaches related to the field are available to-date. Yet, there exist a gray area in the field created by lack of proper evaluation of the existing methods in terms of how meaningful are the discovered patterns in usage. A proper evaluation of the performance of existing methods is required, similar to that has been carried out during the FIMI workshops in 2003 and 04 [51], to come up with a guidance for the research community who is interested in the field of frequent pattern mining and have a desire to see it as an essential data mining task. In this study we have tried to find some clues to the question of whether the frequent pattern mining is still an interesting topic to explore or not, using an empirical evaluation on some publicly available methods that are proven successful by several studies.

By examining the FPM approaches accumulated in the literature one can easily figure out that studies related to finding more and more efficient FPM algorithms has been carried out before focusing on the real meaning of discovering huge sets of frequent patterns. Without any surprise, the outcome of this experiment urges us to believe that the frequent patterns alone has a very limited applicability and usability in terms of building predictive models, even if the algorithms efficiently mine tens of thousands of patterns in minutes. Therefore, it may not be interesting to see more frequent pattern mining methods which discuss merely the efficiency of the mining. Nevertheless, methods of finding representative subsets of frequent patterns that could be effectively reusable is appealing, since the commonly used interesting pattern mining methods still posses limitations with respect to the discovery of patterns, which is exactly the outcome of this study as well. The FPM methods we used in the experiments did not outperform the state-of-the art methods in chemoinformatics. It is proven that finding maximal patterns and closed

patterns are NP hard [52], yet the maximal frequent itemset mining algorithms in the form that we have used in this study performed equally well on graph data, and were useful especially when sophisticated graph mining methods failed to produce an output. It has been a common case that different implementations of the same method yield different outcomes [22], and some methods could not reproduce the reported results in their original publications, even with the original implementations, and more unfortunately, implementations of most of the theoretically sound methods are not publicly available, limiting the avenues of independent evaluations. The approaches [11, 6, 40] may be interesting to compare with, but the limited access to the original implementations misses avenues for comparisons of such approaches.

As a further study, an analysis similar to this study could be carried out for graph databases with large graphs, and graph databases with unique node graphs, so that one can come up with more general conclusions. It is worthwhile to investigate the pros and cons of using frequent itemset mining on graph data for data mining tasks other than building predictive models. Also, strengths and weaknesses of models built using FPM algorithms can be compared with that of graph kernel based methods [53] and boosting methods [54], even though the approaches of model building of these methods is different from that of FPM, and may not be feasible to directly compare with. Studying the effect of the graph transformation method we used for mining maximal frequent itemsets, on unique node graphs would be beneficial since the edge list of a unique node graph preserves the connectedness of the graph.

References

1. Borgelt C, Graph Mining: An Overview, Proc. 19th GMA/GI Workshop Computational Intelligence, 189-203, Universitätskolleg Bommerholz, Germany 2009
2. Jiawei H, Hong C , Xin D and Yan X, Frequent pattern mining: current status and future directions, Journal of Data Mining and Knowledge Discovery, 1384-5810, 15(1), 2007
3. Agrawal R., Srikant R. and Swami A., Mining Association Rules between Sets of Items in Large Databases, In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Volume 22(2) pp. 207-216,ACM press, 1993
4. Agrawal R. and Srikant R., "Mining sequential patterns," ICDE 1995, pp.3
5. Goethals B., frequent set mining, Data mining and Knowledge discovery Handbook, Chapter 17, pp 377-397
6. Thoma M, Cheng H, Gretton A, Han J, Kriegel H, Smola A J, Le Song, Yu P S, Yan X, and Borgwardt K. , Near-optimal Supervised Feature Selection among Frequent Subgraphs. In Proceedings of SDM'2009, pp.1075~1086
7. Yan X. and Han J., gSpan: Graph-Based Substructure Pattern Mining, In: proceedings of Int. Conf. on Data Mining (ICDM'02), 2002
8. Afrati FN, Gionis A, and Mannila H, Approximating a collection of frequent sets, In: Proceedings of the 2004 ACM SIGKDD, pp 12–19, 2004
9. Vreeken J, van Leeuwen M and Siebes A, Krimp: mining itemsets that compress, Data Mining and Knowledge Discovery, Springer, DOI: 10.1007/s10618-010-0202-x
10. Yan X, Zhou XJ, Han J Mining closed relational graphs with connectivity constraints. In: Proceeding of the 2005 ACM SIGKDD, pp 324–333, 2005
11. Hasan M, and Zaki M J. MUSK: Uniform Sampling of k Maximal Patterns. In Proceedings of SDM'2009. pp.650~661
12. Cheng H, Yan X, Han J, Hsu C (2007) Discriminative frequent pattern analysis for effective classification, In: Proceeding of the ICDE'07, pp 716—725, 2007
13. Hintsanen P. and Toivonen H., Finding reliable subgraphs from large probabilistic graphs, in editors: W.Daelemans, B. Goethals, and K. Morik, Data Min Knowl Disc (2008) 17:3–23

14. Ke Y, Cheng J, and Ng W, Correlated pattern mining in quantitative databases, ACM Trans. Database Syst. 33, 3, Article 14 , September 2008
15. Nijssen S, Guns T and De Raedt L, Correlated itemset mining in ROC space: a constraint programming approach, KDD '2009, Paris,France, pp 647—656, 2009
16. Ketkar N S, Holder L B and Cook D J, Subdue: Compression-Based Frequent Pattern Discovery in Graph Data, in OSDM '05, pp 71-76, ACM Press, 2005
17. De Raedt L, Guns T and Nijssen S, Constraint Programming for Data Mining and Machine Learning, AAAI10, pp 1671-1675, 2010
18. Borgelt C, Simple Algorithms for Frequent Item Set Mining, Advances in Machine Learning II (Studies in Computational Intelligence 263), 351-369, Springer-Verlag, Berlin, 2010
19. Ranu S, and Singh A K. GraphSig: A Scalable Approach to Mining Significant Subgraphs in Large Graph Databases. In Proceedings of ICDE 2009, pp.844-855, 2009
20. Han J, Pei J, Yin Y, Mao R, Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, Data Mining and Knowledge Discovery, 8, 53–87, 2004
21. Zaki M J, Scalable algorithms for association mining, IEEE Transactions of Knowledge and Data Engineering 12:372–390,
22. Goethals B, Survey on Frequent Pattern Mining, 2003, http://adrem.ua.ac.be/~goethals/software/survey.pdf , Last visited: 18/04/2011
23. Zaki M J and Hsiao C J, Efficient Algorithms for Mining Closed Itemsets and their Lattice Structure. IEEE Transactions on Knowledge and Data Engineering, 17(4):462-478. Apr 2005
24. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proceeding of the 2000 ACM-SIGMOD, pp 1–12
25. Burdick D, Calimlim M, Gehrke J, MAFIA: a maximal frequent itemset algorithm for transactional databases. In: Proceeding of the ICDE'01, pp 443–452,2001
26. Bayardo R.J., Efficiently Mining Long Patterns from Databases, In: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, pp 85-93, 1998
27. Agrawal R. and Srikant R., Mining Sequential Patterns, In Proc. of International Conference on Data Engineering, IEEE, 1995
28. Mannila H., Toivonen H., and Verkamo I., Discovering frequent episodes in sequences, In: 1st Int'l Conf. Knowledge Discovery and Data Mining, 1995.
29. Srikant R. and Agrawal R., Mining sequential patterns: Generalizations and Performance improvements, In 5th Int'l Conf. Extending Database Technology, 1996
30. Wang J and Han J, BIDE: Efficient mining of frequent closed sequences, In IEEE Int'l Conf. on Data Engineering, 2004.
31. Luo C and Chung S, Efficient mining of maximal sequential patterns using multiple samples, In: Proceeding of the SDM'05, pp 415–426, 2005
32. Inokuchi A, Washio T, Motoda H, An apriori-based algorithm for mining frequent substructures from graph data. In: Proceeding of the PKDD'00, pp 13–23, 2000
33. Kuramochi M and Karypis G, Frequent subgraph discovery. In: Proceeding of the 2001 international conference on data mining (ICDM'01), pp 313–320, 2001
34. Borgelt C and Berthold M, Mining Molecular Fragments: Finding Relevant Substructures of Molecules,IEEE ICDM 2002, pp. 51-58
35. Huan J, Wang W, Prins J, Efficient mining of frequent subgraph in the presence of isomorphism. In: Proceeding of the ICDM'03, pp 549–552, 2003
36. Nijssen S and Kok J N, A quickstart in frequent structure mining can make a difference. In Proceedings of KDD'2004, pp.647--652
37. Huan J, Wang W, Prins J, Yang J, SPIN: mining maximal frequent subgraphs from graph databases. In Proceedings of KDD'2004. pp.581--586
38. Yan X and Han J, CloseGraph: mining closed frequent graph patterns. In Proceedings of KDD'2003. pp.286~295
39. Cook D.J. and Holder L.B., Mining Graph Data. J. Wiley & Sons, Chichester, United Kingdom 2007
40. Jin N, Young C, and Wang W, GAIA: graph classification using evolutionary computation. In Proceedings of the SIGMOD 2010, pp. 879-890

41. Nijssen S and Kok J N, Frequent subgraph miners: Runtime don't say everything, Proceedings of the MLG 2006, pp 173—180, 2006
42. Data Repository of Bren School of Information and Computer Science, University of California, Irvine, ftp://ftp.ics.uci.edu/pub/baldig/learning/, Last visited: 18/04/2011
43. Stouch T R, Kenyon J R, Johnson S R, Chen X, Doweyko A and Yi Li, in silico ADME/Tox: why models fail, Journal of computer aided Molecular Design, 17: 83-92, 2003, Kluwer
44. T. Karunaratne and H. Boström, "Graph Propositionalization for random forests", *ICMLA'09*, 2009, pp 196-201
45. W¨orlein M., Meinl T., Fischer I., and Philippsen M., A quantitative comparison of the subgraph miners MoFa, gSpan, FFSM, and Gaston. LNCS 3721, pp 392–403, 2005
46. Olsson, T., Sherbukhin, V., SELMA, Synthesis and Structure Administration (SaSA), AstraZeneca R&D Mölndal, Sweden
47. Rogers D and Hahn M, Extended-Connectivity Fingerprints, J. Chem. Inf Model. 2010, 50, 742–754
48. Todeschini R and Consonni V, Handbook of Molecular Descriptors, in Methods and Principles in Medicinal Chemistry, Wiley-VCH; Weinheim 2002
49. Witten I H and Frank E, Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, USA, 2005
50. Garcia S and Herrera F, An Extension on. "Statistical comparisons of classifiers over multiple data sets" for all Pairwise Comparisons, JMLR, 9:2677-2694, 2008
51. Goethals B and Zaki M J, Advances in Frequent Itemset mining implementations: report on FIMI'03, SIGKKD Explorations, 6(1):109-117, 2004
52. Yang G, The complexity of mining maximal frequent itemsets and maximal frequent patterns. In Proceedings of the KDD '04, pp. 344-353.
53. Hinselmann G Ã, Fechner N, Jahn A, Eckert M, Zell A, Graph kernels for chemical compounds using topological and three-dimensional local atom pair environments, Neurocomputing 74, 219–229, 2010
54. Kudo T, Maeda E and Matsumoto Y, An application of boosting to graph classification, In advances in Neural Information processing systems, (NIPS'04), pp 729 – 736, 2004

Table 1. Description of the datasets.

Dataset Name	# compounds in the dataset	#active compounds	#atoms in the largest compound	# atoms in the smallest compound	Average compound size	#node labels
ace	114	57	79	18	42	7
ache	111	55	77	41	56	7
AI	69	35	32	20	24	8
AMPH1	130	65	104	73	87	5
ATA	94	47	34	13	22	5
bzr	163	82	52	23	36	8
caco	100	50	196	6	45	8
COMT	92	46	32	11	20	7
cox2	322	161	48	32	41	8
dhfr	397	199	60	20	41	8
EDC	119	60	43	8	19	7
gpb	66	33	45	22	32	8
HIVPR	113	57	64	26	45	9
HIVRT	101	51	35	19	25	9
HPTP	132	66	50	24	38	9
nct	131	72	44	8	20	8
therm	76	38	94	13	52	6
thr	88	44	101	47	68	5