A Relative Feature Selection Algorithm for Graph Classification

Yaser Keneshloo and Sasan Yazdani

Abstract. Graph classification is one of the most important research fields in data mining nowadays and many algorithms have been proposed to address this issue. In practice, labeling large or even medium-size graphs is a hard task and we need experts to do so. The biggest challenge in graph classification is extracting a set of proper features from graphs. Since graphs are represented by a complex data structure, this issue has been dealt with for a long time. Previous methods focused on extracting features from a certain class in a dataset. In this paper we propose a new feature selection method that extracts features from each graph rather than extracting them from a certain class in the dataset. We extract only frequent subgraphs as features. These subgraphs are chosen according to their number of occurrences in a graph. Moreover, we proposed a new formula which calculates the minimum number of occurrences required for a subgraph to be considered as frequent. We experimented on five real datasets and reached 7-17% higher accuracy than previously proposed methods.

1 Introduction

In recent years, there has been a great emphasis on graph mining due to its vast application in web, bio-informatics, social sciences and networks. Great number of algorithms has been proposed on graph clustering, graph classification, graph querying, motif finding and finding dense patterns in graph [3, 1, 17, 2]. Among these fields, graph classification recently has been focused on greatly. Labeling graphs is usually time consuming, expensive and requires experts. For example in order to label a chemical compound of an AIDS sample, we need to perform lots of

Yaser Keneshloo · Sasan Yazdani
Iran University of Science and Technology
e-mail: {yaser_keneshloo,sasan_yazdani}@comp.iust.ac.ir

experiments to extract important information required to determine whether the virus is active or not [10]. In intrusion detection systems (IDS) a network analyzer is needed to detect an intrusion based on network logs accumulated over months and years [16]. Therefore if we can build a model to detect these structures we can cut these complexities both in cost and time consumption. The biggest challenge in graph classification is feature extraction, thus the better the features extracted from the graph, the higher the chance of correct classification. For example, in Fig. 1, we presented the output of a Decision Tree classifier used in our experiments on AIDS dataset. As we can see in this figure, at level two of this decision tree we have the chemical compound, $C-1-C-1-C-1-N-1-C$[1]. With a simple threshold value we can easily separate two classes, *CA* and *CI* from each other. In common approaches, a graph dataset is postulated and for each class in the dataset, a set of features known as frequent subgraphs is extracted. For labeling an unlabeled graph, classification is continued by checking whether each class's features are occurred in the unlabeled graph or not. Afterwards, classifier measures the structure similarity between the unlabeled graph and every class in the dataset. Finally, graph's label will be whichever class its structure similarity is closest to. Instead of extracting a feature set of frequent subgraphs for each class in dataset, we extract a set of features for each graph in the dataset and then unlabeled graphs are classified based on their distinct frequent subgraphs. After feature extraction, graphs are represented by a real valued vector which can then be used by any known classifier for classification.

Current approaches in graph classification suffer from some inefficiencies as described below:

1. Initial Parameter: All recent graph classification approaches use an initial parameter known as *Min_Sup*, which is set by user. If this value is not set appropriately, algorithm's performance are affected extremely [21, 9, 13].
2. Dataset size: If a dataset is small or has small number of samples in a specific class, usually previous algorithms are unable to find a proper feature set because they are concentrated to extract features from a class as a whole rather than extracting features from each sample.

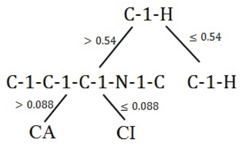


Fig. 1 Separation of two classes using a single rule at level 2

[1] When we show a chemical compound as $C-1-C-1-C-1-N-1-C$ it means the first Carbon joins second with a one bond join and the second is jointed with the third with another one bond join and so on.

In this paper we propose a novel model, *RelativeGraphMiner* (RgMiner), for graph classification which can classify graphs without any starting parameters. To the best of our knowledge, this model is the first model to classify graphs without starting parameters. Moreover, It is the first model to use frequent subgraphs extracted from a single graph as the kernel for classification task. In this approach, by adding a new graph to a dataset, we only need to find frequent subgraphs for this new graph while in previous approaches features must be re-extracted from scratch.

In the following sections, we first point some preliminaries in section 2. Section 3 will focus on related work in graph classification proposed in recent years. Section 4 presents our proposed method. In section 5, we'll present our experimentations on real datasets. Finally section 6, concludes materials proposed in this paper.

2 Preliminary

This section provides basic definitions needed for the rest of this paper. Weighted graph is a common representation in graph classification problems. A weighted graph G can be represented as a quaternary $G = (V, E, l(V), l(E))$ where V is the set of nodes, E is the set of edges and $L(v)$ and $L(E)$ are two functions that label nodes and edges, respectively in G. A subgraph G_s of G, denoted as $G_s \subseteq G$ and is represented as follows:

$$G_s = (V_s, E_s, l_s(V_s), l_s(E_s))$$

Where $V_s \subseteq V, E_s \subseteq E, L_s(V_s) = L(V)$ and $L_s(E_s) = L(E)$. Support of a graph G_s in dataset D is calculated by the number of graphs, $G \in D$ in dataset where $G_s \subseteq G$. We define a new concept of support and show the support of G_s as S_s and define it as number of occurrences of G_s in G. In both definitions G_s is called frequent if number of its occurrences in G is greater or equal to a threshold parameter called *Min_Sup*. G_1 is isomorph with G_2, if there exists a bijection function $f : V(G_1) \rightarrow V(G_2)$ such that for any two adjacent vertex u and v in G_1, $f(u)$ and $f(v)$ are also adjacent in G_2.

3 Related Works

As mentioned before the most important part of classification is extraction of a good feature set. Frequent subgraphs are one of the most common features in graph classification problems. Various methods have been proposed for extracting frequent subgraphs from a graph dataset [12, 13, 9, 15, 21]. All previous methods extract frequent subgraphs from a graph dataset hence they cannot be used for extracting frequent subgraphs, existing in a single graph.

gActive[10] and gSSC[18] used gSpan as their core feature extracting method. In these approaches, for a dataset, all frequent subgraphs are extracted and if a subgraph occurs in a graph G, its corresponding entry in G's feature vector will set to 1, otherwise it will be set to zero. There are other approaches like fragment based approaches [6, 20], kernel based method [19] and topological methods [14]. In fragment based approaches each graph is considered as structures like frequent subgraphs. Kernel based approaches check two graphs similarity using different kernels like random walks, shortest path, cyclic pattern, subtree and graphlet. For detailed discussion refer to [19]. Topological approaches select different characteristics in a single graph like average degree, average clustering coefficient, effective radius and diameter and etc and use them for classification [14].

Our proposed approach doesn't fall in neither of these categories. It takes some of its concept from algorithms like [10] and some from fragment based approaches. Our algorithm is based on extracting frequent subgraphs in a single graph, not a graph dataset. We could see the discriminative power of frequent subgraphs in different categories of application. In [22] the authors used frequent subgraphs as a tool for filtering irrelevant graphs when it is searching for a query graph in a dataset. Grafil attempts to filter out as many irrelevant graphs as possible to apply the graph isomorphism operation, which is proved to be an NP-Complete problem, on small number of graphs. Although frequent subgraphs are powerful feature, to discriminate graphs for all algorithms that used this frequent subgraphs as features, there is another important parameter namely, *Min_Sup*, that should be selected carefully. *Min_Sup* determines the minimum number of occurrences of each subgraph in order to be considered as frequent. So, an important question in these kinds of algorithms is to figure how we can determine a proper value for this parameter. All feature extraction algorithms that use frequent subgraphs as features have an important parameter namely, *Min_Sup*, that should be selected carefully. *Min_Sup* determines minimum number of occurrences of each subgraph in order to be considered as frequent. Therefore, an important issue in these kinds of algorithms is to figure how we can determine the proper value for this parameter. All previous algorithms considered this parameter as user-defined, so if this parameter is set inappropriately, the final result is affected extremely [9, 21]. In order to find the best value for *Min_Sup*, one can tune this parameter by running algorithm several times with different *Min_Sup* values and then pick the best result, which of course is time-consuming.

4 RgMiner Algorithm

In this section first we describe how features are extracted from graphs in a dataset. Then we propose a formula which helps us extract suitable features from graphs with respect to their size and density. Afterwards in section 4.3, we explain how normalized feature vectors can help improving classification results. Finally, we describe how to use extracted features for classification.

4.1 Creating Frequent Subgraphs

In *RgMiner*, each graph is represented by a vector of its frequent subgraphs. This can best be shown by an example. Suppose we want to extract features for the graph in Fig. 2(a). As we can see, Fig. 2(b) and 2(c) are two arbitrary subgraphs extracted from Fig. 2(a), where the subgraph represented in Fig. 2(b) has support value of 1 and the one in Fig. 2(c) has support value of 4, which is the number of its isomorph as shown in Fig. 2(c) through Fig. 2(f). Usually many subgraphs exist in a graph, so finding all of them is time consuming. In order to solve this issue we only find and extend subgraphs which their support are greater than *Min_Sup*. Therefore, subgraphs not satisfying this rule are discarded. Also, in order to decrease *RgMiner*'s complexity even more, we only evaluate subgraphs that are simple walks [5]. A simple walk is a path which nodes can repeat more than once but edges cannot be repetitive. After extending new subgraphs, their support must be calculated in order to find whether they are frequent or not. If a subgraph's support is equal or greater than *Min_Sup* then it is counted as frequent. Also, this subgraph will be considered for next expansions.

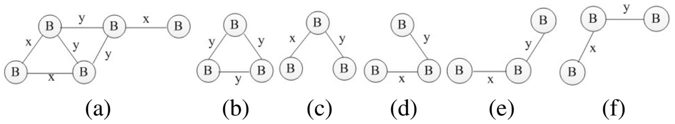


Fig. 2 A simple graph with some of its subgraphs. Graph (b) has support value=1 and graphs (c-f) are isomorph with support value=4.

4.2 Relative_Min_Sup

As described in section 3, *Min_Sup* plays an important role in finding frequent subgraphs. If *Min_Sup* is set inappropriately large, algorithms may not be able to find any frequent subgraph at all. On the other hand if *Min_Sup* is set too small, many frequent subgraphs will be found. In previous approaches, this value is always set by user and if it is not set properly or if the user has a little or no knowledge about the dataset to select this value correctly, classification results will be poor. In order to prevent this kind of problems we proposed *Relative_Min_Sup* which finds the value of *Min_Sup* relatively based on graph's size and density. *Relative_Min_Sup* works based on the following rule: The bigger and denser a graph, the higher its *Min_Sup*.

In recent approaches, first a threshold value $\theta \in [0,1]$ is specified by user for a dataset, then *Min_Sup* value for this dataset is calculated using $\lfloor \theta \times |D| \rfloor$, where $|D|$ shows the number of graphs in dataset. We can use this naïve approach for our problem, i.e. setting a threshold for all of our graphs and multiplying it by the graph size ($\lfloor \theta \times |V| \rfloor$). However, this approach has a big disadvantage: through our experiments on chemical datasets, we figured that determining *Min_Sup* value for each graph does not have a linear relation with its size and density. In the naïve approach this relation is calculated with a linear function. We can show the effectiveness of a feature extracting algorithm using *feature variance*. *feature variance* can be used to show how well an algorithm performs in extracting features for a given graph. In order to calculate *feature variance*, first we need to calculate how many features are extracted from a graph with respect to its size, i.e. $\delta =$ *(number of frequent subgraphs / number of graph nodes)*. Clearly more frequent subgraphs are extracted from bigger graphs. Therefore, to be able to compare two graphs, no matter how big or small, we divided the number of frequent subgraphs by the number of graph's nodes. A dataset $D = \{G_1, \ldots, G_N\}$ can be represented by $\delta_D = \{\delta_1, \ldots, \delta_N\}$, where δ_i is the corresponding feature to size ratio for G_i. We can then define *feature variance* of an algorithm A on dataset D using δ_D as follows:

$$feature\ variance = \gamma_D^A = \max(\delta_D) - \min(\delta_D) \tag{1}$$

This measure can be used to compare how well two feature extraction algorithm are doing on dataset D. A small γ_D^A shows that algorithm A extracts appropriate number of subgraphs proportional to graph's size, while a large γ_D^A means that A is performing poorly on D. We compare *feature variance* of our proposed measure against naïve approach at the end of this section.

To determine a suitable *Relative_Min_Sup* for a graph we follow a simple rule: The sparser a graph, the less the number of frequent subgraphs. With this in mind, a formula should be proposed to determine *Min_Sup* with respect to graph's size and density. In graph theory, density of a graph can be calculated using the following formula:

$$density = \frac{2|E|}{|V|(|V|-1)} \tag{2}$$

For a sparse graph like a chemical compound, density is close to zero while for a dense graph's density is close to one. Because of the denominator, for two large chemical compounds, *density* will be nearly equal and close to zero, which is a big issue. Therefore, density cannot help us in finding a good *Min_Sup* in chemical compounds so we propose a new measure which takes advantage of density's statistics and also resolves the nonlinearity issue:

$$Relative\_Min\_Sup = M_{R_i} = \left\lfloor \frac{|E|}{\log(|V|(|V|-1)/2)} \right\rfloor \tag{3}$$

In this equation, we injected graph's statistics like number of edges and number of nodes to find a proper *Min_Sup*. Moreover, *Relative_Min_Sup* behaves like density, but does not have density's denominator drawback, which is because *Relative_Min_Sup* uses logarithm function to reduce the impact of $|V|$ in denominator. As we can see in Eq. 3, by increasing the number of nodes in a graph, the value of its *Min_Sup* will increase inverse-logarithmically, which is of course in a non-linear manner, so *Relative_Min_Sup* satisfies the condition required for selecting a proper value for *Min_Sup*.

We used PDC-FR dataset, which is described in section 5, to compare the *feature variance* of our approach with naïve approach. For naïve approach the value of *feature variance* was $\gamma_{PDC-FR}^{Naive} = 38.6$, while our approach reached the value of $\gamma_{PDC-FR}^{Relative} = 12.85$, which is clearly better.

4.3 Normalized Feature Vector

In our proposed approach instead of using binary features we used integer valued features, since number of occurrences of a frequent subgraph is very important by itself. Each integer represents the corresponding frequent subgraph's support. But support value of a specific subgraph in a large graph is much greater than its support in a small one, while this specific subgraph may have equal impact in these two graphs' structures. So we need a normalization factor to reduce the effect of this issue. Therefore, in order to get better results, we need to normalize them with respect to their corresponding graph's size, i.e .x_i^j=(S_j/n) where S_j is support of frequent subgraph g_j and n represents number of G_i's nodes.

4.4 Feature Aggregation

Usually there are many frequent subgraphs in large graphs but even for two graphs with a little difference in their structure, there are some distinct subgraphs which make them different. In order to consider all of these features, we use an aggregate feature vector (AFV) for classification. Aggregate feature vector contains all distinct frequent subgraphs occurring in a dataset. By using AFV, each graph in our dataset will have n_{AFV} number of features. So, the feature vector of a graph G_i is created as follows:

$$x_i^j = \begin{cases} \frac{S_j}{n} & \text{if } g_j \subset G_i \text{ and } S_j \geq M_{R_i} \\ 0 & \text{if } g_j \not\subset G_i \text{ or } S_j < M_{R_i} \end{cases}$$

Where M_{R_i} is the calculated relative *Min_Sup* for the graph G_i.

5 Experiments

We used five real world datasets to evaluate *RgMiner*. Table 1 presents more information about these five datasets. All datasets are comprised of graphs, each representing a chemical compound. Each graph node represents a chemical element and each edge shows a bond between two chemical elements in that compound.

Table 1 Real world dataset used for *RgMiner*

Dataset	#Positive	#Total
AIDS	518	1237
PTC-FM	78	201
PTC-MM	67	189
PTC-FR	61	201
PTC-MR	69	193

1. AIDS Antiviral Screen Data[2]. This dataset is gathered by examining tens of thousands of chemical compounds to identify whether they are HIV-active or not. Chemical compounds in this dataset are divided in three categories: Active (CA), Inactive (CI) and Moderately active (CM). We considered CA and CM compounds as negative class and CI as positive class, which are the same settings used in [10]. Finally, to have a fair classification we considered all graphs from CM and CI and randomly chose the same amount of graphs from CA.
2. Predictive-Taxonomy Challenge (PTC): Last four datasets are taken from the challenge proposed in Predictive-Taxonomy website[3]. The goal of this classification challenge was to detect chemical compounds with carcinogenicity activity among them for four types of animal: Male Mouse (MM), Female Mouse (FM), Male Rat (MR) and Female Rat (FR). The compound in this dataset divided in three categories: P: Positive, N: Negative and E: Equivocal. We used only positive and negative data for classification and ignored E samples.

5.1 Classifier Production Methods

In order to evaluate our method and compare it with previous approaches, we employed three different classifiers and compared their results with previous works' best accuracies. All classifiers have been created using Weka 3.6 [7] using 10-fold cross-validation. Classifiers used for comparison are Decision Tree (J48), Random Forest, LibSVM. We used these diverse classifiers in order to have different perspectives on discriminant functions in feature space and also to have a comprehensive evaluation. Also, we built a binary version of our approach called *RgMiner_0-1* which instead of using normalized support and real valued feature vectors, uses binary valued feature vectors. This means, if a frequent subgraph occurs in a graph the corresponding value in its feature vector is set to one, otherwise it is set to zero. We created *RgMiner_0-1* in order to be able to fairly compare our approach with gActive and also, to support two main idea of our approach: first, we wanted to show that if we extract relevant information from each graph rather than extracting them from graphs belonging to a certain class in a dataset, we'll get better results. Second, we wanted to show that by using real values rather than binary ones, we can achieve even higher accuracy. Fig. 3(a) through 3(c) shows final results for these three classifiers based on their accuracy, F-Measure and MCC metrics, respectively. We used gActive and gSSC which are both popular in graph classification. gSSC uses semi-supervised feature extraction algorithm and based on experiments in [11] , best accuracy gained by this method for PTC-FM and PTC-MM datasets are 57% and 57.7%, respectively, while our approach classifies these two datasets with 74.12% and 64.67% accuracy in best case scenario which improves gSSC algorithm by about 17% and 7%. On the other hand, gActive results in [10] for AIDS dataset

[2] Available at: `http://dtp.nci.nih.gov/docs/aids/aids_data.html`

[3] Available at: `http://www.predictive-toxicology.org/ptc/`

shows 67% classification accuracy, while our method reaches 75.9% in accuracy which is an improvement about 9% for this dataset.

Table 2, represents these three classifier's detailed results on each dataset based on different performance metrics we used for evaluating *RgMiner*. Classifiers which work best on each dataset based on different metrics are represented in bold.

Table 2 Result from *RgMiner* using different classifier and metrics

Classifier/Metric		AIDS	PTC-FM	PTC-MM	PTC-FR	PTC-MR
LibSVM	Accuracy	70	66.67	64.55	**69.65**	66.32
	F-Measure	0.68	0.62	0.59	0.63	0.62
	MCC	0.34	0.19	0.09	0.10	0.16
Random Forest	Accuracy	**75.9**	**74.12**	**64.67**	64.65	**67.16**
	F-Measure	**0.759**	0.62	0.61	**0.63**	**0.64**
	MCC	**0.50**	0.20	0.14	**0.11**	**0.22**
J48	Accuracy	69.84	63.68	62.96	60.69	59.06
	F-Measure	0.70	**0.63**	**0.62**	0.61	0.57
	MCC	0.38	**0.21**	**0.15**	0.09	0.05
RgMiner_0-1	Accuracy	75.66	63.68	64.55	71.14	68.39
	F-Measure	0.757	0.60	0.61	0.67	0.65
	MCC	0.50	0.16	0.13	0.20	0.24

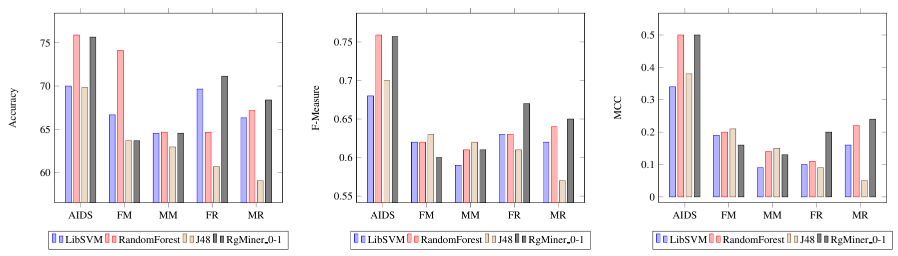
5.2 Discussion

In this section, we describe the result achieved by *RgMiner*. We used different metrics for our evaluation since Accuracy is not a good indicator of behavior of a classifier by itself, especially when datasets are unbalanced. Therefore, we need to use some complementary measures to evaluate our proposed method thoroughly. In addition to accuracy, we used the *F-Measure* and *Matthews Correlation Coefficient (MCC)* to validate our result. MCC is a performance quality measure used in two-class classification problems and is an interesting performance metric in bioinformatics. MCC measure has -1 as its lower bound and +1 as its upper bound, where +1 indicates perfect classification, -1 shows inverse classification, and 0 represents a random classifier. MCC can be measured using Eq. 4.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (4)$$

In Eq. 4, parameters TP, TN, FP, FN stands for true positive, true negative, false positive and false negative extracted from classifier's confusion matrix.

We employed LibSVM which uses C-SVM to build the model and with RBF as its kernel, which is also the default setting in Weka. RBF kernel function maps the original feature space to a higher-dimension feature space and then, the classifier separates two classes in the dataset linearly with a spherical plane. However, RBF kernel function results in poor classification performance. This happens because in our created dataset, number of features dominates the number of samples by about a



(a) Accuracy Measure (b) F-Measure (c) MCC Measure

Fig. 3 Result of classification using *RgMiner* on different classifiers

factor of 14. As stated in [4], in these situations, this method is likely to give us poor results. In these cases since we have an enormous difference between number of samples and features, we do not need yet another kernel function to map our feature space into a higher dimension one. So, it is better to use a linear kernel to do the classification.

Second classifier used for our evaluation was decision tree (J48 in Table 2), which is a supervised rule-based classifier and use simple if-then rules to determine each sample's class. The prominent feature of decision tree among other classifiers is the high interpretability of this classifier. This classifier shows us a good representation to describe our algorithm better. Fig. 1 shows a part of our final decision tree classifier created for AIDS dataset. As we can see in second level of this partial tree, a chemical compound can be classified by a simple rule of checking whether a $C-1-C-1-C-1-N-1-C$ subgraph occurs in its structure. If it does and has a normalized support of equal or greater than 0.088 then the graph is classified as CI, Otherwise graph's label is considered CA. One important characteristic of a decision tree classifier is that no matter how many features used to describe an instance, it only chooses a subset of features that is enough to discriminate existing classes in a dataset against one another. For instance only 11 features were used to build a decision tree classifier for PTC-MR dataset, which compared to 2829, total number of features in AFV for this dataset, is much smaller, which shows our feature extracting approach is performing so well that only 0.4% of features extracted from the dataset is enough for its classification.

Also, we used Random Forest (RF) which technically is an ensemble of classifiers to show that we could reach even better results by incorporating ensemble of classifiers that take advantage of different set of features and samples in building classifiers. Finally, in Table 2 we can see the performance of *RgMiner* using binary values for each feature vector. As we can see, performance of this classifier is near to *RgMiner* but in some datasets it is less preferable. This supports the idea that higher performance can be achieved by simply extracting frequent subgraphs from each graph in a dataset rather than extracting them for each class separately. Using real valued feature vectors represents that relative support does even better than binary values.

Comparing previously proposed approaches with our approach, we can see ours has four great advantages: First of all we use very simple subgraphs instead of using complex ones. This reduces graph isomorphism's cost. Second, we consider all subgraphs no matter how big or dense they are. This fact can be seen in level one and two of the tree shown in Fig. 1. Third, we employ *Relative_Min_Sup* to find a proper minimum number of occurrences of a subgraph, in order to be considered frequent. Finally, we use real valued feature vectors by including the number of occurrences of each subgraph, as an important factor for classification. This way each frequent subgraph has its own impact on classification and is not considered equal to others.

6 Conclusions

In this paper we addressed the problem of graph classification and proposed a new algorithm for this issue. In our proposed algorithm we used the concept of frequent subgraphs in a single graph to create feature sets. Also we proposed an innovative formula to determine the value of *Min_Sup* dynamically based on graph's size and density, finally we normalized the value of supports calculated for each subgraph with respect to graph's size to emphasize on the relative importance of each subgraph. In spite of prior approaches that extract feature sets from and for a certain class in a dataset, we create this feature set from each graph and use it for classification. Through experiments on five real world datasets, we showed that our approach reached up to 17% higher accuracy than prior approaches.

References

1. Aggarwal, C.C.: On classification of graph streams. In: Proceedings of Eleventh SIAM International Conference on Data Mining, SDM 2011, Arizona, Mesa (2011)
2. Aggarwal, C.C., Li, Y., Yu, P.S., Jin, R.: On dense pattern mining in graph streams. PVDLB 3(1), 975–984 (2010)
3. Aggarwal, C.C., Zhao, Y., Yu, P.S.: On clustering graph streams. In: Proceedings of the SIAM International Conference on Data Mining, SDM 2010, Columbus, Ohio, USA (2010)
4. Auria, L., Moro, R.A.: Support Vector Machines (SVM) as a technique for solvency analysis. Discussion Papers of DIW Berlin (2008)
5. Bondy, J.A.: Graph Theory with Applications, pp. 12–21. Elsevier Science Ltd. (1976)
6. Cheng, H., Yan, X., Han, J., Hsu, C.: Discriminative frequent pattern analysis for effective classification. In: Proc. of ICDE, Istanbul, Turkey (2007)
7. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, L.H.: The Weka data mining software: an update. SIGKDD Explor. Newsl. 11(1), 10–18 (2009)
8. He, H., Singh, A.K.: Closure-Tree: An index structure for graph queries. In: ICDE 2006, Atlanta, Georgia (2006)
9. Huan, J., Wang, W., Prins, J.: Efficient mining of frequent subgraph in the presence of isomorphism. In: Proceedings of the 3rd IEEE International Conference on Data Mining, ICDM 2003, Melbourne (2003)

10. Kong, X., Fan, W., Yu, P.S.: Dual active feature and sample selection for graph classification. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, California (2011)
11. Kong, X., Yu, P.S.: Semi-supervised feature selection for graph classification. In: Proceeding of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, Washington, DC (2010)
12. Krishna, V., Suri, N.N.R.R., Athithan, G.: A comparative survey of algorithms for frequent subgraph discovery. Current Science 100(2) (2011)
13. Kuramochi, M., Karypis, G.: An efficient algorithm for discovering frequent subgraphs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1038–1051 (2004)
14. Li, G., Semerci, M., Yener, B., Zaki, M.J.: Graph Classification via Topological and Label Attributes. In: 9th Workshop on Mining and Learning with Graphs (with SIGKDD) (August 2011)
15. Nijssen, S., Kok, J.: A quick start in frequent structure mining can make a difference. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 647–652. ACM (2004)
16. Parveen, P., Weger, Z.R., Thuraisingham, B.M., Hamlen, K.W., Khan, L.: Insider Threat Detection using Stream Mining and Graph Mining. In: IEEE 23rd International Conference on Tools with Artificial Intelligence, ICTAI 2011, Boca Raton, FL (2011)
17. Singh, A.K.: Graph querying, graph motif mining and the discovery of clusters, United State Patent, Santa Barbara, CA (2011)
18. Thoma, M., Cheng, H., Gretton, A., Han, J., Kriegel, H., Smola, A., Song, L., Yu, P.S., Yan, X., Borgwardt, K.M.: Near-optimal supervised feature selection among frequent subgraphs. In: SIAM International Conference on Data Mining (2009)
19. Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, I.R., Borgwardt, K.M.: Graph Kernels. Journal of Machine Learning Research 11, 1201–1242 (2010)
20. Wale, N., Karypis, G.: Comparison of descriptor spaces for chemical compound retrieval and classification. In: Proc. of ICDM, Hong Kong, pp. 678–689 (2006)
21. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. In: ICDM 2002 (2002)
22. Yan, X., Yu, P.S., Han, J.: Substructure similarity search in graph databases. In: SIGMOD Conference (2005)