Graph-based Image Classification by Weighting Scheme

Chuntao Jiang¹ and Frans Coenen²

Abstract Image classification is usually accomplished using primitive features such as colour, shape and texture as feature vectors. Such vector model based classification has one large defect: it only deals with numerical features without considering the structural information within each image (e.g. attributes of objects, and relations between objects within one image). By including this sort of structural information, it is suggested that image classification accuracy can be improved. In this paper we introduce a framework for graph-based image classification using a weighting scheme. The schema was tested on a synthesized image dataset using different classification techniques. The experiments show that the proposed framework gives significantly better results than graph-based image classification in which no weighting is imposed.

1 Introduction

Automated classification of images is an important research area for content-based image retrieval. The usual method to automate image classification is to use lowlevel image features (e.g. colour, shape, texture) as feature vectors. Based on these feature vectors, traditional classification approaches can be employed to train classifiers. However, a prominent disadvantage of such a vector model is that it only uses numerical features and discards the structural information such as the relations between objects. In order to solve this defect, we proposed to make use of the structure of image information together with the numerical features for the

¹ Mr. Chuntao Jiang

The University of Liverpool, Liverpool, L69 3BX, UK , cjiang@csc.liv.ac.uk 2 Dr. Frans Coenen

The University of Liverpool, Liverpool, L69 3BX, UK, frans@csc.liv.ac.uk

purpose of image classification. A simple approach to keep the structural information of an image is to use digital image representation techniques; for instance, Quad-tree [9], Attributed Relational Graphs (ARGs) [21], etc. By modeling images as graphs, the task of image classification becomes one of classifying graphs; in other words the image mining problem is cast into a graph mining application.

Given a collection of images modeled as graphs, we can extract frequent subgraphs (sub-graphs whose number of occurrences is greater than some minimum threshold) using an appropriate graph mining algorithm. It is conjectured that such frequent sub-graphs (patterns) could be useful in classification, although there is no theoretical proof of their effectiveness [5]. Inspired by this fact, we propose a frequent sub-graph based approach to classify graphs. Basically, the idea is simple; after we have extracted the set of frequent sub-graphs, we use these subgraphs to construct feature vectors. The identified feature vectors are then processed using traditional classification techniques.

Due to the characteristics of the image representation technique used, the computational complexity of the frequent sub-graph mining process may become very high; frequent sub-graph mining is recognized as a hard combinatorial problem. For example, quad-tree represented images contain relatively few distinct node and edge labels. So the required (sub) graph isomorphism testing becomes a significant bottleneck within the overall graph (image) mining task. Furthermore, for quad-tree represented images, nodes nearer the root cover larger areas of the image than nodes at the ends of branches. This fact means that the significance of each level of nodes in the quad-tree representation is different. In order to better capture the structural information within images, a weighting factor is needed to reflect the significance of individual nodes and edges.

In the literature, previous work for frequent sub-graph mining assumes that equal importance is allocated to all the frequent sub-graphs among the graph dataset. This is not always appropriate, as in the case of the quad-tree representation described above. In this paper we present an extension of the well known gSpan [23] frequent sub-graph mining algorithm by incorporating a weighting scheme. By combining graph mining approaches with weights, we can significantly reduce the time for searching patterns and at the same time enhance the accuracy of mining applications that can be represented using weighted graphs.

The main contributions of our paper are: (1) a framework for frequent subgraph based image classification and (2) an extension of the gSpan frequent subgraph mining algorithm to include a weighting scheme. The proposed weighting framework is flexible and works for other frequent sub-graph mining algorithms as well. The main benefits of weighted frequent sub-graph mining are enhanced efficiency without compromising classification accuracy with respect to image mining.

The rest of the paper is organized as follows: We start with a short overview of recent work on graph-based image analysis in Section 2. The main components and weighted frequent sub-graph mining framework are described in Section 3.

Section 4 gives experimental results. Some discussion and conclusions are presented in Section 5 and Section 6 separately.

2 Background

2.1 Graph Mining

Graph mining is the process of discovering hidden patterns (frequent sub-graphs) within graph datasets. The graph mining task can be categorized as transaction graph mining or single graph mining. In transaction graph mining the dataset to be mined comprises a collection of small graphs (transactions). The goal is to discover frequent recurring sub-graphs across the dataset. In single graph mining the input of the mining task is one single large graph, and the objective is to find frequent sub-graphs which occur within this single graph. Our weighted graph mining algorithm adopts a transaction graph mining algorithm, because we represent a collection of images as a collection of graphs. In order to explain the rationale of our graph mining algorithm, we start with some definitions.

Definition 1 (labeled graph) A labeled graph can be represented as $G(V, E, L_V, L_E, u)$, where: *V* is a set of vertices, $E \subseteq V \times V$ is a set of edges; L_V and L_E are vertex and edge labels respectively; μ is a label function that defines the mappings $V \to L_V$ and $E \to L_E$.

Definition 2 (sub-graph) Given a pair of graphs $G_1 = (V_1, E_1, L_{V1}, L_{E1}, u_1)$ and $G_2 = (V_2, E_2, L_{V2}, L_{E2}, u_2)$, G_1 is a sub-graph of G_2 , if and only if

 $V_1 \subseteq V_2, \forall v \in V_1, u_1(v) = u_2(v)$, and $E_1 \subseteq E_2, \forall (u,v) \in E_1, u_1(u,v) = u_2(u,v)$ G_2 is also a super-graph of G_1 .

Definition 3 (graph isomorphism) For a pair of graphs $G_1 = (V_1, E_1, L_{V1}, L_{E1}, u_1)$ and $G_2 = (V_2, E_2, L_{V2}, L_{E2}, u_2)$, G_1 is isomorphic to G_2 , if and only if a bijection $\delta : V_1 \to V_2$ exists such that $\forall v \in V_1, u_1(v) = u_2(\delta(v))$, $\forall \alpha, \beta \in V_1, (\alpha, \beta) \in E_1 \Leftrightarrow (\delta(\alpha), \delta(\beta)) \in E_2$ and $\forall (\alpha, \beta) \in E_1, u_1(\alpha, \beta) = u_2(\delta(\alpha), \delta(\beta))$. The bijection δ is an isomorphism between G_1 and G_2

Definition 4 (sub-graph isomorphism) A graph G_1 is sub-graph isomorphic to a graph G_2 , if and only if there exists a sub-graph $g \subseteq G_2$ such that G_1 is isomorphic to g.

Given a database of *n* graphs $GD = \{G_1, G_2, ..., G_n\}$ and a similarity function $\sigma(g,G)$, where *g* and *G* are graphs; then $\sigma(g,G) = 1$ if *g* is isomorphic to a sub-graph of *G*; otherwise $\sigma(g,G) = 0$. The *support* of a graph *g* in *GD*, denoted by $\eta(g,GD)$, is defined as $\eta(g,GD) = \sum_{1 \le k \le n} \sigma(g,G_k)$. The function $\eta(g,GD)$ is therefore a measure of the frequency that a given sub-graph occurs in *GD*. A graph *g* is said to be frequent, if $\eta(g,GD)$ is greater than or equal to some threshold. The frequent sub-graph mining problem is to find all frequent sub-graphs in a graph database *GD*.

Frequent sub-graph mining algorithms have a wide application in bioinformatics, chemical compound analysis, networks, etc. There are various algorithms reported in the literature e.g. AGM [13], FSG [16], FFSM [12], and gSpan [23]. The hard core of graph mining algorithms is the (sub) graph isomorphism checking problem, which is known to be NP-complete [10]. How to efficiently generate sub-graph candidates and calculate the support of each candidate is a key to a successful graph mining algorithm.

2.2 Image Mining

Generally speaking, image mining aims to discover implicit patterns among image databases. The fundamental issue of image mining is how to use low-level (primitive) representations to extract high-level hidden patterns with efficiency and effectiveness. Image mining research mainly focuses on two directions. One is to extract the most discriminative features, which are suitable for traditional data mining techniques. The other involves finding patterns which describe the relations between high-level and low-level image features [11].

A great many image mining techniques exist in the literature such as: object recognition, image indexing and retrieval [2], image clustering and classification [22], association rule mining [1], neural network, etc. In this paper we are only focusing on the image classification task

2.3 Image Representation

Representing images as graphs can maintain the structural information of images. There are a number of techniques for graph based image representation. The main idea of graph based image representation is that the regions of the image, which contain similar properties, are denoted by graph nodes, and the relations between different regions are denoted by graph edges. The node and edge attributes usually describe the characteristics of that region and the relation between regions respectively. A straightforward approach is that of Attributed Relational Graphs (ARGs) [21]. Within ARGs, images are first segmented into objects using a region growing algorithm, the objects are then represented by graph nodes and the relationships between objects are usually employed to facilitate the image mining task.

A quad-tree [9] is a widely used tree structure for representing images. The fundamental idea behind the quad-tree is that any image can be divided into four

quadrants. Each quadrant can be further split into four equal-sized sub quadrants (NW, NE, SW and SE), and so on. The quad-tree decomposition is based on recursively subdividing the sub-images until the imposed limit is met. In this paper, we use a quad-tree to represent each image, which is generated by our *random image generator*³. We use GraphML [3] to represent our graphs (slightly extended to include a class field to support the training stage of the classifier generation process). GraphML is a XML-based file format for graphs.

2.4 Related Work

The research work most directly related to ours is in the area of graph mining based approaches for image analysis. Jiang and Ngo [15] represented images as ARG graphs where the common patterns among images are discovered using an inexact maximum common sub-graph algorithm. Iváncsy [14] modeled the spatial structure of images as graphs instead of using the traditional perceptional features of images, such that the extracted frequent sub-graphs consist of the index structure of the image database. One shortcoming of this model is that the image index structure is only suitable for images comprising well segmented regions. Nowozin et al. [19] proposed a framework of weighted substructure mining with a linear programming boost classifier. The classification results have suggested that weighted frequent pattern mining algorithms are suitable for image classification.

3 Proposed Framework

Given a set of pre-labeled images generated by our random image generator, we model these images as quad-trees, and then output these quad-trees as GraphML formatted graphs. Having represented random images as graphs, we apply our weighted frequent sub-graph mining algorithm to extract weighted frequent sub-graphs. The concept of weighted frequent sub-graph mining algorithms. In the implementation described here we have chosen the well known gSpan algorithm [23] as our base frequent sub-graph mining algorithm because it is efficient, effective and simple to implement. We use weighted support to extract frequent weighted sub-graphs instead of the original simple support value. Finally, when we have discovered the frequent weighted sub-graphs, we use these sub-graphs as features and construct feature vectors for the original images. We then employ an appropriate classifier generator to produce an image classifier. The flowchart of the weighted graph-based image classification framework is illustrated in Figure 1.

³ http://www.csc.liv.ac.uk/~frans/KDD/Software/ImageGenerator/imageGenerator.html



Figure 1 Framework of weighted graph-based image classification

3.1 gSpan

Our weighting scheme is flexible and can be applied to many graph mining algorithms (e.g. AGM, FSG, etc.). We chose the gSpan algorithm because it outperforms many other algorithms e.g. AGM, FSG, and its data structure is simple to implement and integrate with our weighting scheme imposed. gSpan uses a DFS (Depth First Search) lexicographic order to define a minimum code to uniquely represent each graph. This is achieved by building a DFS code tree to model the relations among all graphs. Each node of a DFS code tree represents a DFS code. The $(n+1)^{th}$ level of the tree has nodes which contain DFS codes for *n-edge* graphs. The *n-edge* graphs are produced by one edge growth from the n^{th} level of the tree, significantly reducing the size of the search space. The pseudo-code of gSpan algorithm is described in Figure 2.

3.2 Weighting Scheme

In many real image mining applications, some image "objects" are more important while others are less important. Most existing graph mining algorithms do not take this into account, so that all the edges and sub-graphs are given equal importance. Our approach assumes that some sub-graphs in graphs and some edges and nodes of those sub-graphs have different importance. In this section, we will explain how to tackle the problem of different importance of sub-graphs using a weighting scheme to discover weighted frequent sub-graphs. Some essential definitions must be first introduced.

Definition 5 (weight of edge) Let *e* be an edge of a graph, and $\{g_1, g_2, \dots, g_n\}$ are *n* graphs in graph dataset *GD*. The weight of *e* is defined as $\omega(e) = \tau(e) / \sum_{1 \le i \le n} \kappa(g_i)$, where $\tau(e)$ is the number of occurrences of *e* in *GD*, and $\kappa(g_i)$ is the size of graph g_i , which equals the number of edges in that graph.

Algorithm gSpan (GS, s, c, F)

Input: c = DFS code, GS =graph database, s =min support; Output: F = frequent sub-graph set; 1. F $_1$ = all frequent 1-edge graphs in GS (sorted by DFS) lexicographic order) 2. $F = F_1$ 3. For each edge $e \in F_1$ do 4. $\mathbf{c} = \mathbf{e};$ Subgraph_mining(GS, c, s, F) C = candidate sub-graphs; 1. If $c \neq min(c)$ then return; 2. Insert c into F; 3. C=Ø 4. Scan GS once, find every edge e that c can be right-most extended to frequent c∪e, save $c \cup e$ into C; 5. Sort C in DFS lexicographic order; 6. For each $c \cup e \in C$ do Subgraph_mining(GS, $c \cup e$, s, F); 7. 8. Return; 5. GS = GS - e;6. If |GS| < s break; 7. End For

Figure 2 Pseudo-code for gSpan algorithm

Definition 6 (weight of graph) Let $g_i = \{e_1, e_2, \dots, e_k\}$ be a graph consisting of k edges. The weight of g_i is defined as $\omega(g_i) = \sum_{1 \le j \le k} \omega(e_j)/|g_i|$, where $\omega(e_j)$ is the edge weight, and $|g_i|$ is the size of g_i which equals the number of edges in g_i . This weight is formulated as the average weight of edges in the graph.

Definition 7 (weight factor of sub-graph) A sub-graph's weight factor is defined as the ratio of the sum of graph weights in which this sub-graph occurs to the sum of all graph weights in the graph dataset. This can be formulated as:

$$wfs(g_{sub}) = \frac{|GD| \cap (g_{sub} \subset g_k)}{\sum_{\substack{k=1 \\ |GD| \\ \sum_{k=1} w(g_k)}} w(g_k)}$$

Where, $|\text{GD}| \cap (g_{sub} \subset g_k)$ is the number of graphs in which g_{sub} occurs, |GD| is the number of graphs in graph dataset GD, $w(g_k)$ is the graph weight.

The weight factor derived in this manner is then used to quantify the actual importance of each different sub-graph in a given graph dataset. The weighted support of a sub-graph can then be defined as the product of the number of occurrence of the sub-graph and the weight factor of the sub-graph. This can be formulated as:

$$ws(g_{sub}) = f(g_{sub}) \times wfs(g_{sub})$$

Where, $f(g_{sub})$ is the frequency of a given sub-graph, and $ws(g_{sub})$ is the weighted support of the sub-graph. The goal of weighted frequent sub-graph mining is then to find all the frequent sub-graphs whose weighted support is above a user specified minimum threshold.

Definition 8 (frequent weighted sub-graph) Let β be a sub-graph. If $ws(\beta) \ge threshold$, then β is called a frequent weighted sub-graph, otherwise β is called an infrequent weighted sub-graph.

Theorem 1 If a weighted sub-graph is infrequent, then any super-graph of this sub-graph is also infrequent.

Proof Let α be an infrequent weighted sub-graph, then $f(\alpha) \times wfs(\alpha) < threshold$. Let β be a super-graph of α , i.e. $\alpha \subset \beta$, then $f(\alpha) \ge f(\beta)$. Therefore $wfs(\alpha) \ge wfs(\beta)$. So $f(\beta) \times wfs(\beta) \le f(\alpha) \times wfs(\alpha) < threshold$ thus β is an infrequent weighted sub-graph.

3.3 Image Classification

As described in Section 3.1, we use the weighted frequent sub-graph mining algorithm to extract frequent weighted patterns (sub-graphs) from images. Given these frequent patterns, we build feature vectors upon them. The basic idea is that we use the number of frequent weighted sub-graphs as the number of features for each image. Thus the feature vectors are a two dimensional table. The number of rows corresponds to the number of images and the number of columns corresponds to the number of features. Each feature value can be assigned by either the number of occurrences of that feature in each image or just binary values (1 for the existence of the feature in the image, 0 for the non-existence of the feature in the image). After we built the feature vectors for images, we can exploit a number of existing classification approaches to classifying images.

4 Experimental Results

4.1 Graph Dataset

As noted above we developed a random image generator. We also produced an image processing tool to represent these images as quad-trees, and output these quad-trees as $GraphML^4$ format graphs, which were further used for our graph mining work. Figure 3 gives some sample images generated using our random image generator, noting that they are classified as either "landscape" or "seascape" according to content.

4.2 Implementation

In order to evaluate our weighting scheme, we implemented versions of gSpan with and without the weighting schema. A variety of classifier generators were also used: a decision tree algorithm ([20]), CBA, CMAR, and SVM (Support Vector Machine) classification algorithms for image classification test. CBA (Classification based on Associations) is a classification association rule mining algorithm developed by Liu et al. [18]; CMAR (Classification based on Multiple Association Rules) is another classification association rule mining algorithm developed by Li et al. [17]. We adopted our existing Decision Tree, CBA, and CMAR implementations ([6], [7], [8]) for the experiments. For SVM, we adopted LIBSVM package [4] which employs RBF (Radial Basis Function) kernel.



Figure 3 Sample images by random image generator

⁴ <u>http://graphml.graphdrawing.org</u>

4.3 Results

From the above we tested the performance of weighted gSpan on our randomly generated image dataset and compared the result with our implementation of the original gSpan algorithm. The gSpan implementations were undertaken in Java. Figure 4 illustrates the runtime and the number of discovered frequent sub-graphs as the minimum support threshold is varied from 2% to 30%. Note that the memory consumption for weighted gSpan is much less than gSpan. For gSpan testing, we had to increase the heap size of the JVM (Java Virtual Machine) to 512 Megabytes, in order to run it appropriately; while for weighted gSpan we were able to discover frequent patterns using the default JVM heap size. The runtime in Figure 4(a) shows that weighted gSpan is much faster than gSpan. When the minimum support value is 10%, gSpan cannot complete the search within 10 minutes, but weighted gSpan can even handle the case when the minimum support value is 3% without any difficulty. Figure 4(b) displays the number of frequent sub-graphs discovered by these two algorithms. The number of frequent sub-graphs found using weighted gSpan is much less than that by gSpan. For example, if the support is 9%, gSpan finds 197,460 frequent patterns in 539.6 seconds while weighted gSpan finds 1,130 frequent patterns in 9.8 seconds.



All our experiments were carried out using a 1.86GHZ Intel Core 2 PC with 2GB main memory, running Windows XP. As we mentioned in Section 3.4, there are two methods to calculate the value of each feature. One uses the number of occurrences of each feature in each image (we call this the numerical setting); the other uses the existence of each feature in each image as binary values (we call this the binary setting). We tested our framework on both settings. It appears that

the performance of our framework using the binary setting is much better than that based on numerical settings. Especially using the decision tree algorithm, the classifier runs out of memory using some thresholds. Because of this, we concentrated on the binary setting in experiments to evaluate the effectiveness of classification. Table 1 illustrates the results obtained using a number of different classification algorithms. The first column is the data we used; the next two columns are the search time and support of mining frequent features and the last four columns are the accuracies for the various classification algorithms we adopted. For CBA, CMAR, SVM algorithms, the classification accuracy is tested on ten cross validation. The dataset displayed in the first column of Table 1 can be described by four parameters: (1) D, the number of features extracted, (2) N, the number of images generated by random image generator, (3) C, the number of pre-labeled class, (4) W, indicates that weighted graph mining used. Because we compared our weighted graph mining with graph mining without weightings, we put 'W' as suffix of each dataset in order to differentiate the mining algorithms (for example, for the first two cells in first column of Table 1, 'D3748.N120.C2' and 'D331.N120.C2.W', 'D3748.N120.C2' means feature vector constructed by original graph mining algorithm without weight; 'D331.N120.C2.W' means feature vector constructed by weighted graph mining algorithm using the same dataset, which consists of 120 images with 2 classes).

It is suggested by Table 1 that the performance of our framework is comparable with that of the standard graph mining algorithm based image classifiers with significantly reduced search time and number of features generated under the same condition.



Figure 4 (b) Frequent sub-graphs vs. support for a dataset of 500 random images

5 Discussion

During the testing of our framework, we found that the performance varied with different support values. Choosing the right support value to extract the most discriminating features is still a problem, which merits further research. We also found that classification accuracy based on numerical setting is much lower than that based on binary settings. We are still not sure of the reason for this, and further experimental work is therefore suggested. The irregular support values shown in Table 1 were chosen to enable the original gSpan algorithm to complete searching within tolerable time. We keep the support value as low as possible in order to show the efficiency of our weighted frequent sub-graph mining algorithm.

In our experiments, we only use synthetic data to test our framework, and furthermore, only one method of image representation. It might be better to test our framework on real-life datasets and use different image representation techniques.

Data	Search Time	Support	C4.5	СВА	CMAR	SVM
D3748.N120.C2	12.72s	33%	88.33%	80.83%	77.5%	80.83%
D331.N120.C2.W	2.01s	33%	91.67%	84.17%	75%	83.33%
D3859.N200.C2	26.63s	34%	91%	83.5%	83.5%	84.5%
D328.N200.C2.W	3.20s	34%	100%	82.5%	77%	79%
D1651.N293.C2	22.73s	40%	93.2%	73.4%	81.5%	83.62%
D230.N293.C2.W	3.16s	40%	98.63%	82.89%	77.84%	79.5%
D3125.N400.C2	25.74s	24%	78.5%	86%	90.5%	88%
D270.N400.C2.W	2.75s	24%	86%	87.25%	84.5%	86%
D3602.N500.C2	60.97s	34%	99.2%	80.4%	77.6%	85.4%
D366.N500.C2.W	8.91s	34%	96.4%	81.8%	71.8%	81.4%

Table 1 Classification results on different classifiers by binary valued feature vectors

6 Conclusion

Previous image classification work did not take into account the spatial relations that are evident in images. We believe, intuitively, that the accuracy of image classification can be improved by including spatial relations. We propose a framework of image classification by integrating the gSpan graph mining algorithm, with a weighting scheme. The graph mining algorithm is used to extract discriminative frequent spatial relations in the image. Such frequent patterns can be used to build feature vectors, which can be further processed by classical image mining techniques. We demonstrate that our weighted approach provides efficiency advantages over the non-weighted approach without compromising classification accuracy.

We plan to further extend the work described here by using real-life images and adopting different ways of image representation.

References

- 1. Antonie, M., Zaiane, O. R., and Coman, A.: Application of Data Mining Techniques for Medical Image Classification, In 2nd International Workshop on Multimedia Data Mining (MDM/KDD), San Francisco, CA (2001)
- 2. Babu, G. P. and Mehtre, B. M.: Color Indexing for Efficient Image Retrieval. *Multimedia Tools and Applications*, 327–348 (1995)
- Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M. and Marshall, M. S.: GraphML Progress Report: Structural Layer Proposal, *In Proceedings of 9th International Symposium on Graph Drawing (GD 01)*, pp. 501-512, Austria (2001)
- Chang, C. C. and Lin, C. J.: LIBSVM -- A Library for Support Vector Machines, software available at <u>http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html</u>, Department of Computer Science and Information Engineering, National TAIWAN University (2001)
- Cheng, H., Yan, X., Han, J. and Hsu, C. W: Discriminative frequent pattern analysis for effective classification, In 23rd International Conference on Data Engineering (2007)
- Coenen, F: LUCS KDD implementation of CBA (Classification Based on Associations), http://www.csc.liv.ac.uk/~frans/KDD/Software/CBA/cba.html, Department of Computer Science, The University of Liverpool, UK (2004)
- Coenen, F.: LUCS KDD implementation of CMAR (Classification based on Multiple Association Rules), http://www.csc.liv.ac.uk/~frans/KDD/Software/CMAR/cmar.html, Department of Computer Science, The University of Liverpool, UK (2004)
- Coenen, F.: The LUCS-KDD Decision Tree Classifier Software, http://www.csc.liv.ac.uk/~frans/KDD/Software/DecisionTree/decisionTree.html, Department of Computer Science, The University of Liverpool, UK.(2007)
- 9. Finkel, R. A. and Bentley, J. L.: Quadtrees, *A Data Structure for Retrieval on Composite Keys*, Acta Informatica 4 (1), 1–9 (1974)
- Garey, M. R. and Johnson, D. S.: Computers and Intractability A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, New York (1979)
- 11. Hsu, Wynne, Lee, M. L. and Zhang, J.: Image Mining: Trends and Developments, in *Journal of Intelligent Information System (JISS): Special Issue on Multimedia Data Mining*, Kluwer Academic (2002)
- 12. Huan, J., Wang, W. and Prins, J.: Efficient Mining of Frequent Subgraph in the Presence of Isomorphism, In *Proceedings of the 2003 International Conference on Data Mining (ICDM 03)* (2003)
- Inokuchi, A., Washio, T. and Motoda, H.: An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data, In *Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 00)*, Pages: 13-23 (2000)
- Iváncsy, G., Iváncsy R. and Vajk, I.: Graph Mining-based Image Indexing, In 5th International Symposium of Hungarian Researchers on Computational Intelligence, November, Budapest (2004)
- Jiang, H. and Ngo, C. W.: Image Mining using Inexact Maximal Common Subgraph of Multiple ARGs, In *International Conference on Visual Information Systems(VIS'03)*, Miami, Florida, USA (2003)

- 16. Kuramochi, M. and Karypis, G.: Frequent Subgraph Discovery, In Proceedings of 2001 IEEE International Conference on Data Mining (ICDM 01) (2001)
- Li, W., Han, J. and Pei, J.: CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules, in Proceedings of International Conference of Data Mining (ICDM 2001), pp. 369-376 (2001)
- Liu, B., Hsu, W. and Ma, Y.: Integrating Classification and Association Rule Mining, In the Fourth International Conference on Knowledge Discovery and Data Mining (KDD 98), New York, USA (1998)
- Nowozin, Sebastian, Tsuda, Koji, Uno, Takeaki, Kudo, Taku and BakIr, Gokhan: Weighted Substructure Mining for Image Analysis, In *Proceedings of the 2007 Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, 1-8, IEEE Computer Society, Los Alamitos, CA, USA (2007)
- Quinlan, J. R.: C4.5: Programs for Machine Learning, *Morgan Kaufmann Publishers*, San Francisco, CA, USA (1993)
- Tsai, W. H. and Fu, K. S.: Error-Correcting Isomorphism of Attributed Relational Graphs for Pattern Analysis, *IEEE Transaction on System, Man and Cybernetics*, Vol. 9, pp. 757-768 (1979)
- Vailaya, A., Figueiredo, A. T., Jain, A. K., and Zhang, H. J.: Image Classification for Content-Based Indexing, *IEEE Transactions on Image Processing*, 10(1), 117–130 (2001)
- Yan, X. and Han, J.: gSpan: Graph-based Substructure pattern mining, In Proceedings of 2002 International Conference on Data Mining (ICDM 02) (2002)