Graph Pyramid Embedding in Vector Space

Seyedeh Fatemeh Mousavi[a], Mehran Safayani[b], Abdolreza Mirzaei[b]
Department of Electrical and Computer Engineering
Isfahan University of Technology
Isfahan, Iran
[a]sf.mousavi@ec.iut.ac.ir, [b]{safayani, mirzaei}@cc.iut.ac.ir

Abstract— **Graph-based representation has an effective and extensive usage in pattern recognition due to represent properties of entities and binary relations at the same time. But a major drawback of graphs is lack of basic and essential mathematical operations required in many algorithms of pattern recognition. To overcome this problem, graph embedding in vector space enables classical statistical learning algorithms to be used on graph-based input patterns by providing a feature vector for each graph. The aim of this paper is to propose a new generic framework of graph embedding based on ideas from multiresolution theory. The main idea is mapping image pyramid from field of image processing to graph pyramid in graph domain. To this end, we suggest a summarization algorithm that can be used on graphs with continuous node labels. Finally we use resulted graph pyramid for a hierarchical embedding. In an experimental evaluation, we will show the advantages of this new approach in the context of classification problems.**

Keywords- graph-based representaion; graph embedding; multiresolution theory; graph pyramid; classification

I. INTRODUCTION

Statistical pattern recognition provides many algorithmic tools for various applications by using vectors as a formalism of objects representation. The reason is that the vector space resulted from this representation method is flexible and controllable, so the basic operators are well defined. Nevertheless, the nature of patterns in some applications such as bioinformatics and chemistry, document analysis and images classification are not only dependent on features but also structural relations between them. In this situation, the vector-based representation are faced with two serious limitations. First, vectors always represent different objects with some fixed features regardless of their complexity. Second, there is no possibility in modeling of relations between different parts of an object.

In contrast, structural pattern recognition try to solve the mentioned drawbacks related to feature vectors by means graph-based representations. In fact, graphs in addition to represent features as nodes, can easily show their relationships with edges. On the other hand they do not suffer from the constant size limitation and with variable number of nodes and edges; they can simply cope with the complexity of objects, although this powerful representation increases the complexity of algorithms. For example, comparison between two objects in the graph domain namely graph isomorphism has exponential complexity. Moreover due to little mathematical structures in graph domain, even elementary operations for many algorithms such as sum and product cannot be performed on graphs. As a result there are just some algorithms for processing of graphs. These algorithms are usually limited to distance and its derivatives.

In recent years, several efforts have been made to take advantage from both the universality of graphs for pattern representation and the computational convenience of vectors for pattern recognition [1, 2]. An interesting approach which has attracted the attention of many researchers, is known as graph embedding in vector space. Graph embedding by explicit mapping each graph to a real vector space makes all existing algorithmic tools in statistical approach available for graphs. However, the problem of finding adequate vector representations for graphs is absolutely non-trivial [3]. The present paper provides a generic framework for embedding which can be improved available methods. In the next subsections after an overview of related works, we explain motivation of this new framework.

A. Relataed works

Different graph embedding procedures have been proposed in the literature so far. The first family is based on frequencies of appearance of specific knowledge-dependent substructures, capturing topology information (including histogram of nodes degree, number of nodes and edges) and content of graph via its labels. The writers [4] have recently introduced a method for labeled graphs based on statistics of the node labels and the edges between them, based on their similarity to a representatives set. The introduced method in [5] is considered the graph information in several levels of topology, structural and attributes; also it is used fuzzy logic for embedding to reduce sensitivity of numerical data against noise. Spectral embedding of graph is another extensive family based on extract features from graph by eigen-decomposition of adjacency and Laplacian matrices [6]. An interesting method is introduced in [7]. In this method, spectral matrix of the Laplacian of a graph is applied to construct symmetric polynomials that its coefficients are used as graph features. Finally according to dissimilarity representations, the third family represents a graph by its dissimilarities from a set of prototypes [3].

Each of these families has advantages and disadvantages. For example, the first group are able to use domain knowledge for embedding by finding substructures, but it is faced with the computational challenges for exploring them. On the other hand, feature extraction via labels and adjacency matrix, strongly reduces the complexity since it is not dependent on finding substructures, however embedding relies on labels. Spectral methods provide a solid theoretical insight into the meaning of extracted features. The main problem of spectral methods is that they have more sensitivity to structural errors. Also the majority of these methods are used for the unlabeled graphs or labeled graphs with strongly limited label

978-1-4799-5487-2/14/$31.00 ©2014 IEEE

alphabets. The third family represent a graph using its dissimilarity to a prototype set. Since this approach using graph edit distance, it can handle arbitrary graphs but computation of distances is not cost effective.

B. Motivation and contribution

The mentioned problems about existing methods will be more remarkable by increasing graph size. As graph gets larger, possible noise and distortion of the structure and content of the original graph can be increase. Moreover, whenever the graph information increases, the processing of all its elements and safe embedding of the graph in vector space, will be complicated. Hence a generic framework for embedding that improves any existing methods is required. The goal of the framework is to provide easy access to a variety of information in the graph. A graph consists of subgraphs with different sizes and contents which they have features of different sizes and contents too. Furthermore each of these subgraphs can represent a special part which is similar in graphs come from same class. As a result, any analysis procedure that is applied only at a single scale may miss information at the other scales. The solution is to carry out analyses at all scales simultaneously [8]. This case can remind the multiresolution theory in the field of signal and image processing.

Multiresolution theory is concerned with representation and analysis of signals (or images) at more than one resolution. The appeal of such an approach is obvious, features that might go undetected at one resolution may be easy to spot at another [9]. Image pyramid is a strong and simple structure for representation of images with more than one resolution. It is a collection of decreasing resolution images arranged in the shape of a pyramid, to extract features and structures of interest and to attenuate noise. One of the aspects of multiresolution in graph domain, can be seen in graph summarization. Graphs are large in many applications and is almost impossible to understand the information encoded in the graphs by mere visual inspection. Therefore effective graph summarization methods are required to help users extract and understand the information [10]. Among various summarization algorithms, those which provide resolution control of summarized graph can be played a key role in the graph domain. So if we have a summarization algorithm, which provide resolution control of summarized graph, we can provide a set of graphs in pyramid-shaped with different abstract levels. This purpose can be achieved by sequential summarization of input graph. This condition allows the users to move to larger summaries with more details or smaller summaries with less details using a slider.

The present paper at the first step conveys the concept of image pyramid in signal and image processing domain to graph pyramid in the graph domain. In literature, one can find similar works for special applications such as image segmentation [11] and to find a suboptimal tour in traveling salesperson problem [12]. But what we are going to do is defining graph pyramid based on suggested summarization algorithm. Samples of these multilevel summarization are hierarchical graph matching with the purpose of reducing computational complexity [13] and graph visualization by applying hierarchically clustering of graph [14]. In the next step we are going to introduce the generic framework of embedding by considering a basic method. We will name it "graph pyramid embedding in vector space". In this framework, first a graph pyramid is built, then embedding is done at several levels instead of single embedding of original graph level. It is expected that the extracted features from different resolution levels can provide interesting information and more comprehensive for embedding than selected features from only one level. Time complexity of the framework, illustrates its flexibility to maintain an appropriate balance between time embedding and classification accuracy.

The rest of this paper is organized as follows: after introducing primary concepts and basic used method in the next section; we will define the graph pyramid in Section 3. To construct this pyramid we propose a summarization method which can be applied on different graphs by selecting suitable graph clustering algorithm. Then we define our generic framework based on an embedding basic method. In Section 4 we will design a scenario from definition of embedding and offer results of classification experiments on dataset consists of relatively large graphs. Finally Section 5 contains our concluding remarks.

II. BASIC CONCEPTS

A. Graph

Definition. Let L_v and L_E be a finite or infinite label set for nodes and edges, respectively. A graph g is a four-tuple $g = (V, E, \mu, \gamma)$, where V is the finite set of nodes, $E \subseteq V \times V$ is a set of edges, $\mu: V \rightarrow L_v$ is node labeling function and $\gamma: E \rightarrow L_E$ is edge labeling function.

B. Graph edit distance

The main idea of graph edit distance is defining dissimilarity or distance of graphs with minimum amount of distortion required to transform one graph into another [15]. So in the first step, distortion model should be defined. A standard set of edit operations is defined by insertion, deletion and substitution of nodes/edges. For example substitution one node with another is equivalent to change label of a node. For each pair of graphs there is a sequence of edit operations or edit paths to transform one graph into another. The differences between two graphs is modeled using applied edit operations. Due to the large number of edit paths between two graphs and higher strength modification of some edit operations, a special cost is assigned to each edit operation. So, given a set of edit paths and a function of edit cost, dissimilarity for a pair of graphs is defined as minimum cost of edit path in the set.

Definition. If $g_1 = (V_1, E_1, \mu_1, \mathcal{V}_1)$ and $g_2 = (V_2, E_2, \mu_2, \mathcal{V}_2)$ be source and destination graphs respectively, the graph edit distance between g_1 and g_2 is defined as

$$d(g_1, g_2) = \min_{(e_1,\dots,e_k)\in E(g_1,g_2)} \sum_{i=1}^{k} c(e_i) \qquad (1)$$

Where $E(g_1, g_2)$ is set of edit paths which transform g_1 into g_2. The cost function c shows the amount strength of edit operation e_i.

In this paper, in order to compute the graph edit distance, an efficient method in [16] is selected which is based on a fast suboptimal bipartite optimization procedure.

C. Basic method: Disimilarity graph embedding

The method based on dissimilarity [3] due to its proven solutions and the ability to use on different types of graphs, is usually used as a reference method to compare with the new embedding method. In present paper we are trying to improve it with our embedding framework.

Definition. Assume a graph domain $\mathcal{G}$, namely the set of all graphs over the label alphabets L_v and L_E, is given. If $\mathcal{T} = \{g_1, \dots, g_N\}$ is a training set with N graphs and $\mathcal{P} = \{p_1, \dots, p_n\} \subseteq \mathcal{T}$ is a prototype set with n $\leq N$ graphs, the mapping $\varphi: \mathcal{G} \to \mathbb{R}^n$ is defined as following function:

$$\varphi(g) = \big(d(g, p_1), \dots, d(g, p_n)\big) \qquad (2)$$

The problem of prototypes selection in this method is critical issue. Intuitively, prototypes must be simultaneously include high information and low redundancy. Although there are several selectors, we suffice ourselves to a fairly powerful method, Spanning Prototype Selector (SPS), in order to show the achieved improvement in experiments. SPS considers all distances to the prototypes selected before. The first prototype is set median graph i.e. the graph whose its sum of distances to all other graphs in set is minimal. Every further prototype is the graph with the largest distance to the set of previously selected prototypes.

III. GRAPH PYRAMID EMBEDDING

The image pyramid offers a flexible and convenient multiresolution format that mirrors the multiple scales of processing in the human visual system [8]. The pyramid are computed in an iterative fashion [9]. With considering the original image as base level J and applying successive filtering and down sampling on level j ($j = J, \dots, 2$), it will be obtained approximate image of level $j-1$. It will be simple to define graph pyramid, if a suitable alternative to above procedure within the graph domain can be found. This section formally defines the graph pyramid and how to make it, then uses constructed pyramid for generic embedding framework.

A. Graph pyramid

Graph pyramid is a collection of graphs with different abstract levels. As one can be seen in Fig. 1, the base level of the pyramid contains the available original graph and the top most level of the pyramid contains the most abstract level. As moving up, abstract level of graph will increase. Suppose a regular pyramid for graph $g = (V, E, \mu, \gamma)$. The number of nodes is reduced with a factor of $1/\lambda$, $\lambda > 1$, by moving to next upper layer. It makes that the number of created levels is equal to $\lceil \log_\lambda |V| \rceil$. The base level J has $|V|$ nodes, where $J = \lceil \log_\lambda |V| \rceil$, intermediate level j has $\left\lceil \frac{|V|}{\lambda^{J-j}} \right\rceil$ nodes where $0 \leq j \leq J$. This pyramid is composed of $J+1$ resolution levels from $|V|$ to 1, but we can limit ourselves to P reduced resolution approximations from original graph, because usually high abstract levels such as a graph with only one node does not give us much information. So graph pyramid is truncated to $P+1$ levels where $j = J-P, \dots, J-2, J-1, J$.

A system can be designed to create a graph pyramid with summarization algorithm as a core component, just similar to image pyramid construction system. This system

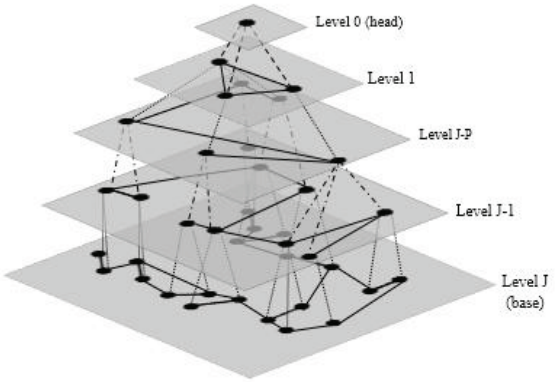


Figure 1. Graph pyramid

can be seen in Fig. 2. By selecting the appropriate summarization algorithm, level $j-1$ abstract output can be created from the level j input graph. The original graph and its P higher abstractions can be directly accessed. Most of summarization algorithms first produce super nodes by grouping similar nodes and relabeling them as needed and then form abstract graph structure by inserting super edges between produced super nodes. So the number of nodes will reduce with factor of $1/\lambda$.

In general, any summarization algorithm which is able to control the resolution of the summarized graph can be used to build the pyramid. Depending on graph type and its labels, selected options from the set of existing algorithms for summarization will be different. The purpose of existing algorithms typically is finding groups and communities in social networks, improving the quality of web search, customers clustering, and etc. Almost all features are used as the labels on the nodes and edges in such graphs, are discrete. So if we face with a graph containing continuous labels, we should find a way to summarize it. In the next subsection we propose an algorithm for this purpose.

B. Summarization algorithm

What we apply as the summarization algorithm is using a clustering algorithm [17], and then inserting edges between produced super nodes. So regardless of the time taken to insert super edges or super loops between super nodes, the time complexity of summarization algorithm is equals to the time complexity of used clustering algorithm.

With little attention in clustering algorithms can be realized that most of them are doing clustering based on an affinity matrix $|V| \times |V|$. The elements of this matrix are the weight on the edges. In fact these values are the similarity of source and target nodes of each edge. Graph clustering is to cluster the nodes of a graph, such that sum of weights on all edges connecting different clusters is minimized. The greater weight will increase the possibility of being in a cluster, and if no edge exists between two nodes (the weight is equal to zero) or existing edge have little similarity, they are not likely be member of a cluster. Therefore, each cluster can be considered as a super node.

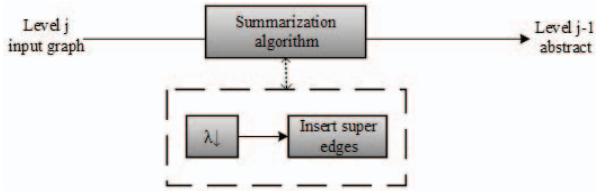


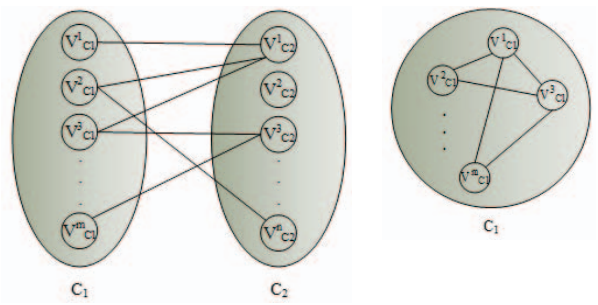
Figure 2. A simple system to create graph pyramid

Affinity matrix has a key role in clustering algorithm. The right choice of affinity matrix results to a suitable clustering for every kind of graph. An interesting point in obtaining such matrix is that, the graphs representation, depends on prior knowledge and assumptions about dataset. For example, in the case of the COIL-15 dataset (Section 5), it can be guessed that the coordinates of nodes that are closer together are more similar. Therefore the amount of similarity on the edges can be obtained by defining a similarity measure on the coordinate's distance of the nodes labels. In our experiments, let nodes s and t, radial basis function kernel is used to evaluate the amount of similarity (3). The main problem with this function is to determine the value of the parameterθ, however, if properly set up, it will represent the distribution of similarities very well. Standard deviation of the distances between all pairs of label nodes is a good estimation of this parameter. This measure only takes presence or absence of edges into account (i.e. binary relations) and further structural similarity does not consider in the calculation. But because it is applied on all graphs in classification experiments, there is less need to a more comprehensive and more complex measure.

$$A_{ts} = exp\left(\frac{-|(x_t, y_t) - (x_s, y_s)|^2}{2\theta^2}\right) \qquad (3)$$

After producing the super nodes, we should insert super edges. Consider two disjoint super nodes $C_1 = \{v_{C_1}^1, v_{C_1}^2, \dots, v_{C_1}^m\}$ and $C_2 = \{v_{C_2}^1, v_{C_1}^2, \dots, v_{C_2}^n\}$. If the ratio of total number of existing edges between two super nodes to total number of possible edges between them $\{(v_{C_1}^i, v_{C_2}^j) | i = 1,2, \dots, m \; and \; j = 1,2, \dots, n\}$, be greater than a specific threshold t_{edge}, an edge is inserted between two super nodes. According to Fig. 3a, the number of possible edges is equal to the number of edges of a complete bipartite graph with two separated set C_1 and C_2 $(n \times m)$.

In this summarization method, the nodes are also permitted to have super loop. It is necessary to have more accurate approximation and to reconstruct original graph. In this case, the number of possible edges between all pair of nodes $(v_{C_1}^i, v_{C_1}^j)$ where $i, j = 1,2, \dots, m$ is equal to the number of edges in a complete graph with node set C_1 (Fig. 3b). We can add the number of possible loops in C_1 which is equal to $|C_1|$ to prior amount, this condition depends on whether underlying graph contains loop or not. If you want the graph keep its simple shape, the threshold for inserting super loop t_{loop} should be usually greater than the selected threshold for inserting super edge t_{edge}. In our experiments $t_{loop} = 0.5$ and $t_{edge} = 0$ are considered.



(a) (b)

Figure 3. Possible number of edges between two different super nodes (a) or in a super node.

In the previous paragraphs we described how to build a graph structure. Some of the missing information in summarization process can be maintained with appropriate labeling mechanism for nodes and edges of summarized graph. Various labels can be assigned to the nodes and edges which are dependent on its application and the importance of graph reconstruction for us. Also these labels are important to make affinity matrix graph of lower level in order to create higher abstract levels. On COIL-15 dataset, simply average coordinates in each super node, are considered as a label for that, and edges are remained without labels.

C. *Generic framework*

Embedding several abstract levels beside original graph level can be useful. It causes to exploit the available information at different resolution levels to select interesting features of the content and structure of the graph. In formally, given an input graph $g_J = (V_J, E_J, \mu_J, \gamma_J)$, P abstract graphs which form a graph pyramid, is produced by considering a reduction factor. Then the graph related to each level is mapped to vector space and finally the resulted vectors are placed in a final vector next to each other. This vector is assigned to input graph g_J. Fig. 4 shows the final vector. The framework also permits us to select several levels among constructed levels of graph pyramid. This reduces the time of embedding. Now, by considering the basic method presented in Section 2, our goal is to apply the above approach based on the basic method on large graph data. Although every existing embedding method can be considered to improve.

Let us the graph domain $\mathcal{G}_J$ is given, that J is equal to the least possible number of nodes in input graphs. By constructing graph pyramid with similar λ for each graphs $g \in \mathcal{G}_J$, the resulted graphs from each level of $J - i$ for $1 \le i \le P$ is supposed as a new graph domain of $\mathcal{G}_{J-i}$. So if $\mathcal{T}_J = \{g_J^1, g_J^2, \dots, g_J^N\} \subseteq \mathcal{G}_J$ be a training set with N graphs from domain $\mathcal{G}_J$, training sets corresponding to $\mathcal{T}_J$ are obtained for new domains as follows:

$$\mathcal{T}_{J-i} = \{g_{J-i}^1, g_{J-i}^2, \dots, g_{J-i}^N\} \subseteq \mathcal{G}_{J-i} \quad 1 \le i \le P \qquad (4)$$

By selecting prototypes set for each graph domain

$$\forall 1 \le i \le P \quad \mathcal{P}_{J-i} = \{p_{J-i}^1, p_{J-i}^2, \dots, p_{J-i}^{n_{J-i}}\} \subseteq \mathcal{T}_{J-i} \; and \; n_{J-i} \le N, \qquad (5)$$

mapping $\varphi: \mathcal{G}_J \to \mathbb{R}^{n_J} \times \mathbb{R}^{n_{J-1}} \dots \times \mathbb{R}^{n_{J-P}}$ is defined by function

$$\varphi(g_J) = (d(g_J, p_J^1), \dots, d(g_J, p_J^{n_J}), \dots, d(g_{J-i}, p_{J-i}^1), \dots, d\left(g_{J-i}, p_{J-i}^{n_{J-i}}\right), \dots, d(g_{J-P}, p_{J-P}^1), \dots, d(g_{J-P}, p_{J-P}^{n_{J-P}})). \qquad (6)$$

It is important to note that graph edit distance is a powerful and flexible concept and using this concept in graph embedding makes embedding framework more powerful. Nevertheless the major problem in using this concept is its exponential time complexity for calculating distances. To overcome this problem, several suboptimal algorithms has been presented with polynomial complexity

Vector of g_{J-P}	…	Vector of g_{J-1}	Vector of g_J

Figure 4. Final vector of graph pyramid embedding

$O(n^3)$ in the number of nodes of involved graphs so far [16]. Yet by increasing the size of graphs, use of these algorithms in embedding take a lot of time. So summarized graph embedding has remarkable affect in reducing time for mapping a graph to vector.

For our generic framework, time complexity for transform an input graph (g_J) to a feature vector is equal to total time to create P successive abstract levels of input graph, and calculate the distance between graph corresponding to each level (g_{J-i} $that$ $0 \leq i \leq P$) with selected prototypes of the same level ($\mathcal{P}_{J-i} = \left\{p_{J-i}^1, p_{J-i}^2, \dots, p_{J-i}^{n_{J-i}}\right\}$). If $O(S(g_i))$ is time complexity of summarization algorithm related to graph g_i, embedding time equals with

$$\sum_{i=0}^{P-1} O(S(g_i)) + \sum_{i=0}^{P} n_{J-i} \times O((\frac{n}{\lambda})^3). \qquad (7)$$

First term can be ignored when a suitable algorithm with low complexity is selected.

For example, in the case of COIL-15 dataset we set $\lambda = 2$, and then determine the values of J and P parameters based on the minimum and maximum number of nodes that exists among graphs. Here we set $J = 4$ and $P = 2$. So we obtain first two abstract level of pyramid for each graph. By placing graphs from same level into a set, three graph domain $\mathcal{G}_4$, $\mathcal{G}_3$, $\mathcal{G}_2$ will be produced. Now depending on importance of time and accuracy in desired application, embedding could be achieved by selecting arbitrary prototype set of every domain.

IV. EXPERIMENTS

The goal of our experiments is to show the efficiency of the proposed generic framework in term of accuracy and time complexity. In order to better representation of this improving, k nearest neighbor (*k*-NN) which is very simple classifier, has been used. We show that how our framework can increase the power of weak classifiers by an efficient embedding.

A. Dataset

Though there are many large graphs in real world, only few experimental datasets exist which contain such graphs that can be used for classification tasks. We choose COIL-100 dataset from IAM graph database repository [18]. We sort classes based on average number of nodes in descending order and select top 15 classes. New dataset is named as COIL-15. TABLE I compares characteristics of both datasets. Other dataset is synthetic one which comprises graphs of different sizes i.e. 50,100, …, 500. We have generated the graphs making use of Delaunay triangulations of randomly generated point sets.

TABLE I. Comparison between COIL-15 and COIL-100 datasets

Characteristics	Dataset	
	COIL-100	***COIL-15***
Size (tr, va, te)	(2400, 500, 1000)	(360, 75, 150)
#Classes	100	15
Nodes (min, max, avg)	(3, 79, 21.5)	(18, 79, 42.6)
Edges (min, max, avg)	(3, 228, 54.2)	(45, 228, 116.1)
Node labels	x,y coordinates	x,y coordinates
Edge labels	none	none

B. Experimental setup and results

COIL-15 consists of training, validation and test sets. The training set is used to construct and train the classifier. In the *k*-NN classifier, the graphs of training set are used as the labeled samples to find the nearest neighbors. Meta parameters that cannot obtain from training procedure, are optimized on validation set. Finally the best configuration of them is used for evaluating performance on test set.

Selecting the appropriate cost function is the first step in dissimilarity graph embedding. For COIL-15 dataset, node labels are real vectors and edges are unlabeled. Here, the substitution cost of pair of nodes labels is given by a distance measure (Euclidian distance), and edge substitutions can be carried out for free. So only three parameters have to be validated. They are node deletion/insertion cost τ_{node} , edge deletion/insertion cost τ_{edge} , and weighting parameter $\alpha \in [0,1]$ that controls which one is more important, edit operation cost on nodes or edit operation cost on edges.

After defining new graph domains according to definition, best configuration of $(\tau_{node}, \tau_{edge}, \alpha)$ for each domain is obtained. This is performed by a grid search in the three dimensional parameter space with respect to the *k*-NN classifier accuracy ($k = \{1,3,5\}$) on the validation set. For the task of graph embedding in real vector space, one additional meta parameter has to be validated too, that is the number of prototypes in each domain.

We are going to be flexible and allow validation set to select arbitrary number of prototypes for each domain (g_j) with varying this value over a certain interval. Assume the prototype selection method in each level is SPS. Based on absence or presence of each level, seven different states is considerable. Accuracy related to each of these states for different *k* values has been reported on test set. As you can see in the TABLE II, pyramid embedding framework increase accuracy of classifications resulted from basic method by enriching its embedding.

Our experiments show that adding abstract levels, in addition to increase accuracy, reduce the prototype selection from original level and so reduce the runtime. For example Fig. 5 shows classification accuracy of basic method and our generic framework on validation set along adjusting number of original level prototypes ($\mathcal{P}_J$). Difference between the two curves indicates that adding abstract level to basic method, makes that resulted vector getting more powerful, and so classification accuracy increase

TABLE II. Accuracy of graph pyramid embedding framework for k-NN classifier. SPS is used for each graph domain.

Graph Domains $\mathcal{G}_j$			Accuracy k-NN (%)		
$\mathcal{G}_4$ *(original graphs level)*	$\mathcal{G}_3$ *(1st abstract graphs level)*	$\mathcal{G}_2$ *(2st abstract graphs level)*	*K=1*	*K=3*	*K=5*
✓			95.33	92.67	80.67
	✓		92	88	**84.67**
		✓	81.33	81.33	78
	✓	✓	91.33	85.33	**86**
✓		✓	**98.67**	**95.33**	**88**
✓	✓		**98.67**	**96.67**	**90**
✓	✓	✓	**98**	**94**	**89.33**

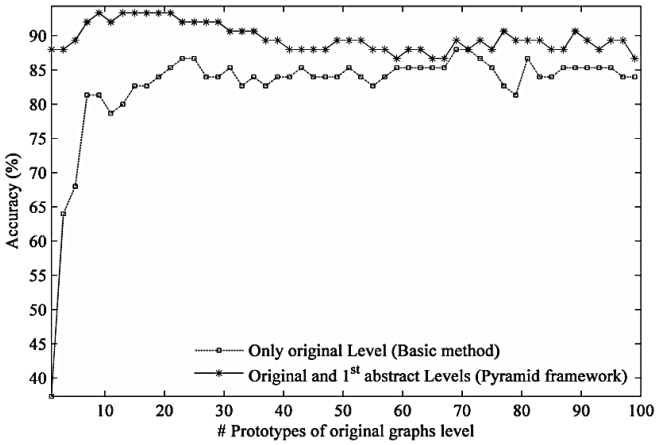


Figure 5. Comparison between achieved accuracy of basic embedding method (only original graphs ($\mathcal{G}_4$)) and graph pyramid embedding (original and first abstract level graphs ($\mathcal{G}_4$, $\mathcal{G}_3$)). For each number of prototypes of original domain, best number of prototypes from abstract graphs domain is considered, in term of accuracy.

significantly. Also while in basic method, maximum accuracy calculate with 70 prototypes from original graphs domain, through this framework we can reach to better accuracy with 20 prototypes from same domain and some prototypes from abstract graphs domain.

Note that the number of selected summarized graphs as prototypes has little effect in embedding time. Since the time complexity of calculating edit distance for original graphs is 8 and 64 times bigger than time complexity for calculating corresponding first and second abstract level graphs respectively. To better understand, on the synthetic data, we have selected 10 graphs as prototypes randomly and then computed average embedding time for each level from five repeated experiments (Fig. 6). Summarization time for underlying graph also is added to embedding time for abstract levels. This experiment is ran on an Intel Core I5 3GHz machine running CentOS 6.3.

I. CONCLUSION

Graph embedding by transforming a graph into a feature vector has an important contribution towards bridging the gap between structural and the statistical pattern recognition. In this paper we provide a generic framework embedding based on graph pyramid. Graph pyramid is a conveyed concept from multiresolution theory in the field of image processing, which is defined through proposed summarization algorithm. The framework tries to enhance an existing embedding algorithm by applying it in several levels of graph pyramid. The experiments show that extracted features from the several levels improve resulted vector from embedding. So it increases recognition accuracy and reduces time of embedding. Important directions of future works include improving other embedding methods by proposed generic framework, applying more algorithms such as SVM on embedded graphs, application of feature selection methods and so on.

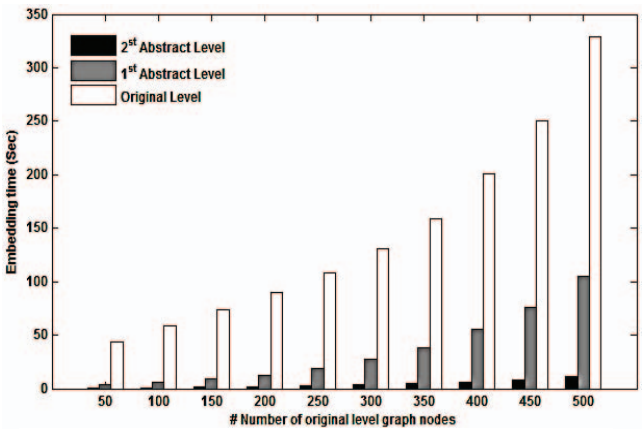


Figure 6. Embedding time for each graph level

REFERENCES

[1] H. Bunke and K. Riesen, "Recent advances in graph-based pattern recognition with applications in document analysis," Pattern Recognition, vol. 44, pp. 1057-1067, 2011.

[2] H. Bunke and K. Riesen, "Towards the unification of structural and statistical pattern recognition," Pattern Recognition Letters, vol. 33, pp. 811-825, 2012.

[3] K. Riesen and H. Bunke, Graph Classification and Clustering Based on Vector Space Embedding vol. 77, 2010

[4] J. Gibert, E. Valveny, and H. Bunke, "Graph embedding in vector spaces by node attribute statistics," Pattern Recognition, vol. 45, pp. 3072-3083, 2012.

[5] M. M. Luqman, J.-Y. Ramel, J. Lladós, and T. Brouard, "Fuzzy multilevel graph embedding," Pattern Recognition, vol. 46, pp. 551-565, 2013.

[6] B. Luo, R. C. Wilson, and E. R. Hancock, "Spectral embedding of graphs," Pattern Recognition, vol. 36, pp. 2213-2230, 2003.

[7] R. C. Wilson, E. R. Hancock, and L. Bin, "Pattern vectors from algebraic graph theory," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 27, pp. 1112-1124, 2005.

[8] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," RCA Engineer, vol. 29, pp. 33--41, 1984.

[9] R. C. Gonzalez and R. E. Woods, Digital Image Processing: Addison-Wesley Longman Publishing Co., Inc., 2001.

[10] Y. Tian, R. A. Hankins, and J. M. Patel, "Efficient aggregation for graph summarization," presented at the Proceedings of the 2008 ACM SIGMOD international conference on Management of data, Vancouver, Canada, 2008.

[11] R. Marfil, L. Molina-Tanco, A. Bandera, J. A. Rodríguez, and F. Sandoval, "Pyramid segmentation algorithms revisited," Pattern Recognition, vol. 39, pp. 1430-1451, 2006.

[12] Y. Haxhimusa, W. Kropatsch, Z. Pizlo, A. Ion, and A. Lehrbaum, "Approximating TSP Solution by MST Based Graph Pyramid," in Graph-Based Representations in Pattern Recognition, vol. 4538, F. Escolano and M. Vento, Eds., ed: Springer Berlin Heidelberg, 2007, pp. 295-306.

[13] "Matching Hierarchical Graphs," in A Graph-Theoretic Approach to Enterprise Network Dynamics. vol. 24, ed: Birkhäuser Boston, 2007, pp. 199-210.

[14] X. Huang and W. Lai, "Clustering graphs for visualization via node similarities," J. Vis. Lang. Comput., vol. 17, pp. 225-253, 2006.

[15] H. B. Michel Neuhaus, Bridging the Gap Between Graph Edit Distance and Kernel Machines vol. 68, 2007.

[16] K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," Image and Vision Computing, vol. 27, pp. 950-959, 2009.

[17] I. S. Dhillon, Y. Guan, and B. Kulis, "A Unified View of Kernel k-means, Spectral Clustering and Graph Cuts," 2005.

[18] K. Riesen and H. Bunke, "IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning", in SSPR/SPR, 2008.