GraphPFM : Pyramidal Feature Matching for Graph Classification

Krishna Kant Singh*
IIT Hyderabad

Ayushi Patel*
IIT Hyderabad
Hyderabad, India

Ramakrishna RM*
IIT Hyderabad
Hyderabad, India

Mausam Jain
IIT Hyderabad
Hyderabad, India

Vineeth N Balasubramanian
IIT Hyderabad
Hyderabad, India

Naveen Sivadasan
Tata Consultancy Services - Innovation Labs
Hyderabad, India

[1]

Abstract

We present a novel method to learn graph representations that can be used for graph-level classification tasks. Our method first computes a node embedding based on its topological context. Three different methods are then proposed to obtain a graph embedding using a multi-level aggregation of node embeddings based on: (a) feature resolution; (b) spatial context of nodes; and (c) the natural hierarchy within the graph. Each of these multi-level aggregation methods generate pyramids of histograms where the histograms are level-wise 'bag of nodes' representation of the graph. We use pyramidal matching on these multi-level features to perform classification tasks, and outperform existing methods including some deep learning methods on benchmark graph classification datasets, despite being a purely topological method and not using node attributes or node labels.

Introduction

Graphs are one of the most ubiquitous data structures in today's digital world, and are used to represent a wide range of information such as electronic circuit models, social media networks, protein structures, etc. While node-level (vector) representations are useful for tasks such as node classification and link prediction, graph-level representations find use in tasks such as graph classification and community representation. Existing graph classification methods in literature can be broadly categorized as kernel-based methods or neural network-based methods. Kernel-based methods represent the graph as a vector of atomic substructures such as graphlets, shortest-path and WL-subtrees. By deploying a kernel-based learning algorithm like SVM, the graphs are classified using the kernel over the substructures (Kriege and Mutzel 2012; Vishwanathan et al. 2010; Shervashidze et al. 2011; Shervashidze et al. 2009) On the other hand, most neural network-based methods attempt to generalize the convolution operation for graphs. Some of them work by recursively diffusing /convolving the representation over a local spatial patch around a node, and further aggregating the node representations either by averaging over them, performing 1-D convolutions or stacking a dense layer and classifying them with a softmax layer (Atwood and Towsley 2016; Simonovsky and Komodakis 2017). Alternatively, graph convolutional networks (Bruna et al. 2013; Defferrard, Bresson, and Vandergheynst 2016) have been proposed to use parameterized spectral filters to perform convolution in the spectral domain.

In this work, we propose a fresh approach towards graph classification by first computing the local topological representation of each node of the graph and then classifying the graph by performing feature-matching, which is performed either at different levels of topological resolution, spatial context or the natural hierarchy of nodes with a graph. Unlike kernel methods where spatial information is lost by simply counting substructures or neural network methods that sum over node embeddings and as a result lose global topology information, the proposed method, GraphPFM, captures the graph topology at varying scales of granularity. We believe that our approach thus strikes a balance between local and global topology in comparison to other representation approaches, as validated by our results in this paper.

A node in a graph can be described by the topological structure of the graph around it, apart from any properties or features provided with the node itself. Representing each node based purely on the topological structure can find importance in tasks where graphs contain unlabeled, unattributed nodes, and we hence propose a topological representation of the node in this work, where each node is represented by its surrounding topological context or the subgraph denoting the local neighborhood of the node. (We also compare the performance of our methodology against methods that use node label information in our results.) We then propose three different ways of combining the node representations to obtain graph representations in order to also capture the global topology of the graph (which is otherwise lost in methods that average over the node representations): (i) the varying levels of similarity between nodes within a graph viewed at different levels of resolution; (ii) the spatial context of nodes; and (iii) the graph at different levels of hierarchy representing the local and global features of the graph. All three methods adopt a *'bag of nodes'* approach, which is novel for graph understanding to the best of our knowledge, and follow the intuition that a graph can be identified by its various substructures at different levels of resolution of the graph. For example, a social network

[1] *Equal Contribution

graph can be viewed as a graph of communities, with each community a further collection of individuals. A complex molecule such as a protein or an enzyme can be viewed as a combination of simpler compounds which in turn are combinations of atoms. We hypothesize that preserving this nature while representing a graph using classical kernel-based (non-neural network-based) methods can significantly improve the performance of the embeddings.

Our key contributions in this work can be summarized as follows:

- We adopt a classical kernel-based (non neural network-based) approach for graph classification, where a local topological representation for a node is combined in multiple new ways to capture the global topological characteristics of a graph
- In particular, we propose three variants of a 'bag-of-nodes' approach to capture a graph's global topology, which allows us to understand graphs at varying levels of granularity
- We show that the proposed method, GraphPFM, outperforms other contemporary methods consistently on standard graph datasets
- We also show that even when node label information is included for our competing methods, the proposed GraphPFM shows comparable and sometimes better results in comparison to state-of-the-art methods despite not using node label or node attribute information.

Related Work

Graph classification methods in literature can be broadly divided under two categories: kernel-based approaches and neural network-based approaches. Classical kernel-based approaches are discussed in great detail in (Vishwanathan et al. 2010); in particular, this work formulates a kernel function based on R-convolution (Haussler 1999), thus providing a unified framework for constructing a kernel for discrete objects after decomposing it into fundamental substructures based of graphlets (Shervashidze et al. 2009), shortest paths (Borgwardt and Kriegel 2005), random walks (Vishwanathan et al. 2010), subtrees (Shervashidze et al. 2011), etc. Graphlets or motifs have been shown to be highly characteristic of graphs with regard to classifying them (Alon 2007), and so we use them in our method to generate node embeddings. We however do not sum over all the graphlet counts as done in traditional graphlet kernel methods (Shervashidze et al. 2009), since this ignores global topology. (Morris, Kersting, and Mutzel 2017) highlights the importance of comparing graphs globally to improve accuracy in classification. Deep Graph Kernels (DGK) (Yanardag and Vishwanathan 2015) improves upon Graph Kernels by accounting for correlation between the atomized substructures leading to sub-optimal performance of kernel. This is corrected by introducing a matrix that encodes the relationships between different substructures which is evaluated by learning latent representations of substructures by treating it as a neural language model. This generalized framework realized performance gains in all the graph-kernel methods they evaluated on. Subgraph2vec (Narayanan et al. 2016) took inspiration from DGK, but used rooted subgraphs around every node generated using the Weisfeiler-Lehman relabeling process. They learned the embeddings of these rooted subgraphs using a Radial Skip-gram model they introduced in the paper. Their idea of aggregating rooted subgraphs is somewhat close to our method, as we count graphlets in a radius around each node and aggregate it, except we perform aggregation using feature matching in a way that captures the global topology of the graph.

More recently, neural network-based methods have become popular for graph classification, with the success of deep learning in other domains. Graph Neural Network (Scarselli et al. 2009) used a recurrent neural network over walks on the graph to propagate the node representation information over the graph. These representations could then be used for classification. (Bruna et al. 2013) pioneered the idea of extending Convolution Neural Networks (CNNs) to a graph setting in the spectral domain, i.e., filters are applied over the graph's frequency nodes computed by a graph Fourier transform. This work is followed by (Defferrard, Bresson, and Vandergheynst 2016) who used spectral filter approximations to reduce cost of evaluating the graph Fourier transform. More recently, graph classification in the spatial domain have outperformed other methods. Diffusion-CNN (DCNN) (Atwood and Towsley 2016) modeled the convolution operation as a diffusion process around each node given by the probability of jumping from one node to another. Their method generalizes to node, edge and graph classification. Dynamic Edge-Conditioned Filters in CNNs (Simonovsky and Komodakis 2017) alternatively formulated convolution where filters are conditioned on edge-labels which improved over DCNN. However, both these methods average over the node embeddings to obtain the graph embedding. This results in loss of global topology information. Patchy-SAN (Niepert, Ahmed, and Kutzkov 2016), who call their method PSCN, addressed the problem of ordering by first determining the order of nodes, forming a "receptive field" of a fixed size around each node and mapping it to a space of fixed linear order. The so formed normalized neighborhood graphs act as the CNN's receptive field analogous to a patch in image-based CNNs. More recently, DGCNN (Zhang et al. 2018) proposed a novel SortPooling layer in addition to the convolutional architecture. The SortPooling layer imposes a consistent ordering on the graph vertices. There exist few methods for generating whole graph embeddings which are very useful in graph classification and provide an efficient solution to computing graph similarities. (Mousavi et al. 2017) is a method for embedding image-based graphs, which like our method, is inspired by multi-resolution pyramids in computer vision. However this method is only used for and tested on object detection datasets from computer vision. Another effort by (Zheng et al. 2016) jointly optimizes the node and community embedding to detect and embed communities within a graph, but is very different from our approach. We point the interested reader to (Goyal and Ferrara 2017) for a survey of all other efforts.

Methodology

We present our Graph-Pyramidal Feature Matching (Graph-PFM) method to compute graph embedding using only the graph topology. These embeddings can either be used to compute similarity between two graphs as a Gram matrix for kernel Support Vector Machine (SVM), or can be used as representations for graph classification. We use Graph-PFM to denote our approach of treating graphs as a bag of node representations and performing feature matching for classification. It is pyramidal in the sense that our model captures different levels of granularity in space, hierarchy and resolution. We now first describe our methodology for obtaining the node embeddings, then describe our 'bag-of-nodes' representation model, followed by our methods to aggregate these representations at multiple resolutions/levels of a graph.

Computing Node Embeddings: Given a graph G, for each node $v_i \in V$, we perform a breadth first traversal starting at v_i up to some depth p. In this way, we consider neighbourhoods of each node up to depth p. Let the induced subgraph of the visited nodes be denoted as G_i. Let f_i denote the graphlet frequency vector of all 29 graphlets of sizes 3, 4 and 5 (shown in Fig 1) in G_i. We use f_i as the node embedding of v_i at this stage. The topological neighborhood of a node is thus used in computing its embedding. Each node in the neighborhood of v_i contributes towards the frequency of graphlets of size $3, 4$ and 5. The nodes that are closer to the boundary of G_i have 'reduced' contribution as they belong to fewer number of graphlets (especially graphlets of larger size) in comparison to the nodes that are closer to v_i. This allows a weighted contribution of neighborhood nodes to the embedding based on their proximity to v_i. This is in contrast to approaches such as DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) where the contribution depends on walks that contain the neighborhood node.

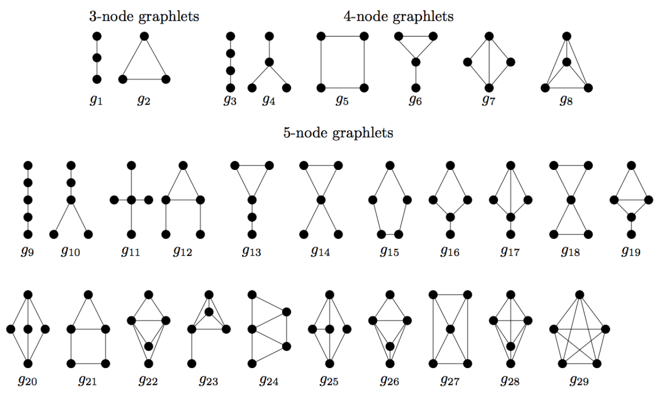

Figure 1: All graphlets of size 3, 4 and 5

A potential limitation of a graphlet counting-based representation approach is the computational overhead in frequency counting. However, exact and approximate algorithms for efficient graphlet counting in large networks have been extensively researched in the past, in particular for small graphlets. Efficient algorithms and implementations for exact counting (Shervashidze et al. 2009; Hočevar and Demšar 2014; Ahmed et al. 2015; Melckenbeeck et al. 2017) are available, especially for graphlets of size up to 5. Approximate counting algorithms (Han and Sethu 2016; Chen et al. 2016; Bressan et al. 2017) make use of different sampling techniques and can provide high accuracy estimates for the distribution of small graphlets in a matter of seconds even for graphs with thousands of nodes.

Bag-of-nodes Representation Model: The primary issue while dealing with a set of features (such as those in a graph) is that the features themselves are unordered, and a brute force comparison between feature sets is computationally expensive. In document classification, a bag of words model is used to represent a document as a histogram of words present in it. Similarly, in image classification, one of the methods used to represent an image is a bag of (visual) words or features. In (Csurka et al. 2004), co-occurrences are captured based on the proximity of such visual words to the k centroids of a k-means clustering over the node features. All elements that fall into the same cluster/bin are taken to be similar and the frequency distribution on the k centroids is then used as the image representation.

We extend this bag-of-words approach to graphs, and our *'bag-of-nodes'* model representations are subsequently used as components in pyramidal feature matching. In particular, we first obtain the node embeddings as described above for all the graphs in the dataset. We then cluster the node embeddings of all the graphs and obtain k cluster centers. We then bin the nodes belonging to each graph into these k bins based on the closest k-center, thus computing graph-specific histograms. We note that a bag-of-nodes approach trivially addresses the challenge of handling varying number of nodes in an input graph (which is common in graph datasets).

When we treat a graph as a set of node-level embeddings, we are sacrificing, to an extent, certain topological information in the graph, especially at the global scale. However, our node embeddings themselves capture the local topological information and as a consequence, the topological neighborhood overlap of co-located nodes significantly influences their proximity in the embedded space in our approach. To further address this issue, we capture the graph topology at varying scales of granularity, by applying the bag-of-nodes approach to the graph at multiple scales of resolution/hierarchy. Our methodology thus captures both local and global topologies of a graph in its final representation.

Aggregation of Node Embeddings: We propose three different methods for aggregation of the embeddings discussed so far. These approaches attempts to capture the topological context at varying scales from local to global at different levels of resolution. Visual illustrations of each of these approaches are included in the supplementary section (Figures 3, 4, and 5).

(i) Pyramid Match Kernel (PMK) for Graph Classification: The Pyramid Match Kernel (Grauman and Darrell 2005) has been proposed earlier for image classification in order to take into account image features at multiple resolutions. This method can be generalized to the graph domain as well, as we show. When matching any two sets of embeddings, some embeddings may not match one another at a higher resolution, but at a lower resolution, their differences can be ignored and they may fall into the same bin. The intersection of histogram pyramids at different levels of resolution allows multi-scale matching between the two sets of features rather than just clustering and binning them in the naive bag-of-nodes approach.

Let $\{G_1, \ldots, G_n\}$ be the set of graphs in the dataset. Each node in a graph is represented as a feature point in a d-dimensional feature space. A graph G is then represented by a set of such feature points $\mathcal{G} = \{x_1, .., x_m\}$. A histogram pyramid ϕ for each graph G is then constructed as: $\phi(G) = [H_0(\mathcal{G}), \ldots, H_{L-1}(\mathcal{G})]$, where $H_i(\mathcal{G})$s are defined below.

For a graph G_k and a resolution level i, a histogram $H_i(G_k)$ is formed by first clustering the feature points from all graphs in $\{G_1, .., G_n\}$ by taking a fixed number of cluster centres equal to 2^{i+1}. Each cluster center at clustering level i represents one bin of the histogram. To construct the histogram for graph G_k at level i, each feature point from graph G_k is assigned to one of the 2^{i+1} bins based on the cluster where the point belongs to. This produces the bag-of-nodes representation at level i. Figure 3 (in the Appendix) demonstrates the histogram construction as part of the PMK algorithm.

The feature set of each graph is mapped to multi-resolution histograms where the distinct characteristics of individual features are retained at the highest resolution and the similarity between features are retained at lower levels. At the lowest level of resolution, i.e., level 0, only 2 bins are created, which are wide bins. At higher levels of resolution, the number of bins increases and the bins are narrower. Depending on the degree of similarity of two nodes when looked at from varying scales of resolution, they fall into the same or separate bins at a certain resolution level. The histogram pyramids are then compared using a weighted histogram intersection; given two histograms A and B of r bins each, their intersection is given by:

$$M(A, B) = \sum_{j=1}^{r} \min(A_j, B_j) \tag{1}$$

For any two graphs G_r and G_s, their Pyramid Match Kernel is defined as:

$$K(G_r, G_s) = \sum_{i=0}^{L} w_i N_i \tag{2}$$

where L is the highest resolution level, $w_i = \frac{1}{2^{L-i}}$ and N_i represents the new matches found at level i, i.e:

$$N_i = M[i] - M[i+1] \tag{3}$$

where $M[i] = M(H_i(G_r), H_i(G_s))$. (If we want to compute the match between two graphs taking number of levels up to 2, the PMK match, K, is given by $M[2] + \frac{1}{2}(M[1] - M[2]) + \frac{1}{4}(M[0] - M[1])$. $K(G_r, G_s)$ is subsequently used in the Gram matrix for classifying the graph using the kernel SVM. The PMK algorithm is summarized in Algorithm 1.

Algorithm 1 Pyramid Match Kernel for Graph Classification

```
Input: $G_1, G_2, .., G_n$
Output: predicted labels : $y_0, y_1 .... y_n$
 1: for $G_i$ in $G_1, \ldots, G_n$ do
 2:     $N(G_i) \leftarrow$ NODEEMBEDDING($G_i$)
 3:     $nodes = nodes \cup N(G_i)$
 4: end for
 5: for $j$ in $0, 1...L$ do
 6:     $clustering \leftarrow$ CLUSTER($nodes, 2^{j+1}$)
 7:     for $G_i$ in $G_1, \ldots, G_n$ do
 8:         $H_j(G_i) \leftarrow$ BAG-OF-NODES($G_i, clustering$)
 9:     end for
10: end for
11: for $G_r, G_s$ from $G_1, \ldots, G_n$ do
12:     for $j$ in $0, \ldots, L$ do
13:         $A = H_j(G_r); B = H_j(G_s)$
14:         $M[j] = \sum_{b=1}^{2^{j+1}} \min(A_b, B_b)$
15:     end for
16:     $K(G_r, G_s) = \sum_{j=0}^{L} \frac{1}{2^{L-i}}(M[j] - M[j+1])$
17: end for
18: return $y_1, y_2 ... y_n \leftarrow$ SVM($K$)
```

(ii) Spatial Pyramid Matching (SPM) for Graph Representation: Spatial Pyramid Matching has been proposed for image classification in (Lazebnik, Schmid, and Ponce 2006), where the spatial context of the features is captured by partitioning the image into grids of various sizes at different levels. At the ith level, the image is partitioned into 2^i grid cells. Each partition is then represented by a bag-of-words histogram, with each histogram hence being representative of a spatial part of the image (unlike PMK). The spatial cells of various sizes allow us to view the image as a whole, as well as composed of parts and their spatial contexts.

Similarly, in graphs, nodes have proximity to certain other nodes more than the others. In this method, apart from global features, feature locality or local connectivity of nodes also plays an important role. We define a multi-level graph split that captures node proximities. The graph at each level is further partitioned into two halves. To do this, Breadth First Search (BFS) procedures are performed starting from the vertices with the least degree till half of the vertices are visited. If more than half the vertices are visited, the additional ones from the last BFS leg are randomly dropped. This procedure splits the graph into two halves. This method of partitioning the graph maintains the local topology in each part.

We first compute the node embeddings of all the nodes in the original graph using the method described in the earlier section. A graph G is represented by a pyramid of histograms where each histogram represents a spatial part of the original graph (unlike PMK). The pyramid is constructed by repeatedly splitting the graph into partitions such that at each level i there exist 2^i partitions. For all partitions at each level in this hierarchy a histogram representation is computed. For graph G, a pyramid of spatial histograms $\phi(G)$ is constructed where each level i of the pyramid contains 2^i histograms, each representing one partition of the graph. For instance, $H_0(G)$ will have 1 histogram representing the original graph, $H_1(G)$ will have 2 histograms each representing a half and $H_2(G)$ will have 4 histograms, each representing a quarter of the graph and so on. Thus,

$$\phi(G) = \{H_0(G), H_1(G), ...H_m(G)\} \tag{4}$$

where $H_i(G)$ is formed by first splitting the graph into 2^i grid cells and computing the bag-of-nodes representation for each grid cell. The bag-of-nodes representation is computed by clustering all the nodes from all the graphs $\{G_1, \ldots, G_n\}$ with a fixed number of cluster centres z. Each cluster center at clustering level i represents one bin of the histogram at level i. Then to construct the histograms for graph G_k at level i, nodes belonging to each partition are assigned to one of the z bins based on the cluster to which the node belongs to. Unlike the PMK, the number of bins across all levels is kept constant, while the graph being represented is taken at different splits at each level. Histograms obtained from the grid cells are concatenated to form the feature vector at that grid level, i.e., $H_i(G)$ is formed by concatenating the 2^i partitions at level i. Once we have the histograms at all the levels, the graph embedding is constructed by concatenation $H_0(G) \cdot H_1(G) \cdots H_m(G)$ of the histogram pyramids $H_0(G), \ldots, H_m(G)$ (Algorithm 2).

Algorithm 2 Spatial Pyramids for Graph Representation

```
Input: $G_1, G_2, .., G_n$
Output: $\phi(G_1), \phi(G_2), .., \phi(G_n)$
 1: for $G_i$ in $G_1, G_2 ... G_n$ do
 2:     $N(G_i) \leftarrow$ NODEEMBEDDING$(G_i)$
 3:     $nodes = nodes + N(G_i)$
 4: end for
 5: $clustering \leftarrow$ CLUSTER$(nodes, z)$
 6: for $G_i$ in $G_1, G_2 ... G_n$ do
 7:     for $j$ in $0, 1 ... k$ do
 8:         $partitions \leftarrow$ CREATEPARTITIONS$(2^j)$
 9:         for $p$ in $partitions$ do
10:             $H(G_i^{j,p}) \leftarrow$ BAG-OF-NODES$(G_i^{j,p}, clustering)$
11:             $H(G_i^j) = H(G_i^j) \cdot H(G_i^{j,p})$
12:         end for
13:         $\phi(G_i) = \phi(G_i) \cdot H(G_i^j)$
14:     end for
15: end for
16: return $\phi(G_1), \phi(G_2), ... \phi(G_n)$
```

The SPM embedding can be viewed as a special case of a hybrid embedding of G that combines: (a) a clustering (binning) of the vertices based on their node level embedding which captures their local topology; and (b) a hierarchical clustering of the vertices in each bin based on their pairwise distances in the graph, which captures their connectivity at the global level. The BFS split is a methodology to provide the hierarchical clustering. We provide a more formal treatment of this observation in the supplementary section.

(iii) Hierarchical Pyramid Matching (HPM) for Graph Representation: While graphlet counts are representative

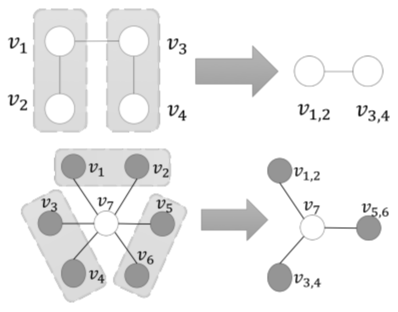

Figure 2: Illustration of Edge Collapsing (top) and Star Collapsing (below), which form the two alternating steps in graph coarsening *(Source: (Chen et al. 2017))*

of the local neighborhood around a node, in order to capture high-level graph features, we also propose the use of graph coarsening to recursively extract features for nodes in coarser graphs. We decompose the graphs into levels or a hierarchy, from the finest graph (original graph) to the coarsest graph. It recursively coalesces the nodes in the original graph to get a series of successively smaller graphs. The finer graphs provide a view of the local topology while the coarser graphs provide views of the global topology (at different scales).

Given a graph G, a hierarchy of graphs $G_0, \ldots, G_k$ is created by using graph coarsening algorithms such as in (Chen et al. 2017). To generate this hierarchy of graphs, we use the Hybrid Coarsening Scheme which combines edge collapsing and star collapsing (Fig 2) used in (Chen et al. 2017). In each coarsening step, the hybrid coarsening scheme first decomposes the input graph with star collapsing, then adopts the edge collapsing scheme to generate the coalesced graph. This is repeated for 3 levels of coarsening. The coarsened graphs do not contain the same nodes as the original graphs as these nodes have been created by merging nodes from the finer level graphs. Therefore, the node embeddings must be computed for all the coarsened levels of the graph.

A graph G is represented by a hierarchy of coarsened graphs $G^0, G^1, \ldots, G^k$. For each G^i, we compute the node embeddings $\{x_1, \ldots, x_m\}$. Then for graph G, a histogram pyramid $\phi(G)$ of the coarsened graphs is constructed as $\phi(G) = \{H_0(G^0), H_1(G^1) ... H_k(G^k)\}$ is constructed, where $H_i(G^i)$ is formed by first clustering the feature points at level i from all graphs $\{G_1^i, .., G_n^i\}$ in the dataset taking a fixed number of cluster centres z. Each cluster center at clustering level i represents one bin of the histogram at level i. Then to construct the histogram for graph G_k at level i or coarsened graph G_k^i, each feature point belonging to coarsened graph G_k^i is assigned to one of the z bins based on the cluster to which the point belongs to. Unlike PMK, the number of bins accross all levels is constant, but the coarsened graph at each level is different. Once we have the histograms of coarsened graphs, the graph embedding is constructed by concatenating the levels of the histogram pyramid $H_0(G^0) \cdot H_1(G^1) \cdots H_k(G^k)$. The algorithm is summarized in Algorithm 3.

Algorithm 3 Hierarchical Pyramids for Graph Representation

```
Input: $G_1, G_2, .., G_n$
Output: $\phi(G_1), \phi(G_2), .., \phi(G_n)$
  for $G_i$ in $G_1, G_2...G_n$ do
      $G_i^0, G_i^1...G_i^k \leftarrow$ GRAPHCOARSENING($G_i$)
      for $j$ in $0, 1...k$ do
          $N(G_i^j) \leftarrow$ NODEEMBEDDING($G_i^j$)
      end for
  end for
  for $j$ in $0, 1...k$ do
      for $G_i$ in $G_1, \ldots, G_n$ do
          $level\_nodes = level\_nodes \cup N(G_i^j)$
      end for
      $clustering \leftarrow$ CLUSTER($level\_nodes, z$)
      for $G_i$ in $G_1, \ldots, G_n$ do
          $H(G_i^j) \leftarrow$ BAG-OF-NODES($G_i^j, clustering$)
      end for
  end for
  for $G_i$ in $G_1, \ldots, G_n$ do
      for $j$ in $0, \ldots, k$ do
          $\phi(G_i) = \phi(G_i) \cdot H(G_i^j)$
      end for
  end for
  return $\phi(G_1), \phi(G_1), \ldots, \phi(G_n)$
```

Results and Discussion

Dataset	# graphs	# classes	# nodes (avg, min, max)	# edges (avg)
ENZYMES	600	6	32.63±15.28, 2, 126	62.14
MUTAG	188	2	17.93±4.58, 10, 28	19.79
PROTEINS	1113	2	39.06±49.76, 4, 620	72.82
PTC	344	2	14.29±16.25, 2, 109	14.69
NCI1	4110	2	29.87±13.56, 3, 111	32.30
NCI109	4127	2	29.68±13.47, 4, 111	32.13

Table 1: Datasets

Dataset	DGK	S2V'	BoN	GraphPFM		
				PMK	SPM	HPM
MUTAG	82.66	84.1	86.84	87.47	**88.37**	87.95
ENZYMES	27.09	38.33	23.3	**64.1**	24.1	27.83
PROTEINS	71.68	72.3	72.60	**75.53**	73.13	73.22
PTC	57.35	57.14	60.94	**62.91**	58.1	61.04
NCI1	62.48	65.61	65.64	64.18	67	**72.5**
NCI109	62.69	65.45	65.64	64.2	65.85	**71.03**

Table 2: Classification Results (Accuracy) *(Note: S2V' = Subgraph2Vec w/o node labels)*

Datasets: We validated our methodology on six benchmark datasets used to study graph classification methods: MUTAG, PTC, ENZYMES, PROTEINS, NCI1, and NCI109. Table 1 presents the summary statistics of these datasets. A detailed description of the datasets is presented in the supplementary material.

Dataset	S2V	DCNN	DGCNN	PSCN	GraphPFM
MUTAG	87.17	80.13	85.83	**88.95**	**88.37**
ENZYMES	52.2	18.1	-	-	**64.1**
PROTEINS	73.38	70.45	**75.54**	75.00	**75.53**
PTC	60.11	56.6	58.58	62.29	**62.91**
NCI1	**78.39**	62.61	74.44	76.34	72.5
NCI109	**78.05**	62.86	-	-	71.03

Table 3: Classification Results (Accuracy) - against methods with node information *(Note: S2V = Subgraph2Vec with node labels)*

Baselines and Experimental Setup: We compare our method against classical kernel-based methods (DGK (Yanardag and Vishwanathan 2015), Subgraph2vec (Narayanan et al. 2016)) as well as recent neural network-based methods (DCNN (Atwood and Towsley 2016), DGCNN (Zhang et al. 2018), PSCN (Niepert, Ahmed, and Kutzkov 2016)). Due to the large literature in this domain, we are unable to compare with every method, but to widely used and most successful kernel-based and recent neural network-based methods. We note that both Subgraph2Vec as well as DCNN, DGCNN and PSCN use either node labels or node features or both (note that the proposed GraphPFM uses only the topology, and does not use any node-level information). Considering Subgraph2Vec has an implementation without additional node information, we compare against both Subgraph2Vec with and without node labels.

Following the conventional setting, we evaluated by performing a 10 times 10-fold cross validation with a fixed random seed (thus, amounting to 100 runs per dataset). We used an SVM classifier with C=1 to classify the graph embeddings generated by our methods. In case of Pyramid Match Kernel, the kernel generated by the method was used by the SVM. In other cases we used a RBF kernel. We tabulated the best results of GraphPFM-HPM and GraphPFM-PMK for up to four levels, after hyper-parameter tuning for number of clusters (GraphPFM-HPM) and number of branches (GraphPFM-PMK) respectively. Spatial pyramid matching is done at only 2 levels, since all the datasets have at least one graph with only 3 or 4 nodes. We considered the best performing hyperparameters for the competing methods on the respective datasets.

Results and Discussion: The results for the aforementioned comparisons are presented in Tables 2 and 3 (in Table 3, we present the results of the best variant of GraphPFM). Table 2 presents the comparisons against other methods that use only topology. GraphPFM clearly outperforms the other methods, including DGK and Subgraph2Vec (without node labels). We note that in Table 3, GraphPFM, which uses no node information, performs on par with (and in some cases, better than) state-of-the-art methods, all of which use node labels/attributes. (We highlight results in which we are better or comparable to the best in the table.)

Our performance with just the topology on graphs indicates that node labels and attributes may not be absolutely essential in all graph classification problems. However, in certain cases, label information makes a significant differ-

Dataset	DeepWalk	Node2vec	GraphPFM-SPM
MUTAG	83.06	81.05	**88.37**
ENZYMES	**25.66**	24.5	24.1
PROTEINS	72.15	72.17	**73.13**
PTC	57.22	**58.45**	**58.1**
NCI1	62.77	61.92	**67**
NCI109	61.57	61.84	**65.85**

Table 4: Comparison with other node embeddings for SPM

ence (please see S2V's performance on NCI datasets in Tables 2 and 3), and it is mostly on such datasets that our GraphPFM method loses out to methods that use node information. On other datasets, GraphPFM performs comparably or better than state-of-the-art methods that use node information, just by using the topology alone.

Among the GraphPFM variants, PMK performs the best on the PROTEINS, ENZYMES and PTC datasets, as it is inherently a better method of feature matching when compared to options such as clustering and binning. SPM outperforms other GraphPFM variants on the MUTAG dataset. We believe this is because MUTAG is a dataset with smaller graphs but the number of nodes in each graph remains more or less same. By splitting a graph, we are able to compare the global topology more accurately. HPM outperforms other methods in case of NCI1 and NCI109 because these datasets contain significantly larger graphs and the graph-coarsening based approach lets us capture the higher-order features.

Other Node Embeddings: The node embeddings computed by GraphPFM work better with the proposed bag-of-nodes aggregating methods as compared to the node embeddings computed by other methods such as DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) or Node2vec (Grover and Leskovec 2016). The comparison between the performances of different node embedding methods is summarized in Table 4 (We outperform or perform comparably w.r.t. other node embeddings on almost all datasets). We hypothesize that this is because a random walk looks at the path neighborhood of a node and not its full neighborhood context. Multiple random walks do not guarantee to cover the entire neighborhood of a node. Alternately, our method is based on the counts of graphlets within the depth context of a node which we believe captures the topology around the node better. Our approach also gives the immediate neighborhood of a node more importance than nodes further away. This is because, the immediate neighborhood is included more often in the graphlet vector because of the larger graphlets as it is less likely to contain larger graphlets further away from the node due to the neighborhood boundary.

Impact of Number of Levels: We also studied the performance of the proposed methods with varying number of levels (while aggregating over the graph), and report our results on the PMK and HPM methods in Tables 5 and 6 respectively. In case of GraphPFM-HPM, the NCI1 and NCI109 datasets achieve their best results at the second level. We hypothesize that this is because most graphs in these datasets coarsen only up to two levels after which they reduce to the most elementary graphs. The PROTEINS dataset has a

Dataset	1-level	2-levels	3-levels	4-levels
MUTAG	81.09	82.2	87.47	86.7
ENZYMES	64.1	57.06	54.24	60.67
PROTEINS	63.92	64.52	72.84	75.53
PTC	47.61	56.87	62.06	62.91
NCI1	57.93	63.71	64.18	-
NCI109	56.02	62.9	64.2	-

Table 5: Impact of number of levels in GraphPFM-PMK

Dataset	1-level	2-levels	3-levels	4-levels
MUTAG	86.84	86.81	86.34	87.95
ENZYMES	27.83	24.8	25.33	26.33
PROTEINS	72.05	73.02	73.22	72.41
PTC	60.94	59.33	60.44	61.04
NCI1	64.6	72.51	70.7	69.8
NCI109	65.54	71.03	67.78	68.51

Table 6: Impact of number of levels in GraphPFM-HPM

larger number of nodes per graph (see Table 1) and performs best at level 3 for similar reasons. In Graph-PMK, the number of levels indicates the degree of fineness while performing feature matching. A noisy dataset such as ENZYMES performed better with fewer levels, while other datasets generally showed monotonic improvement with increase in number of levels.

With regard to runtime for computing our embedding, as discussed earlier, there are several efficient algorithms and implementations available for computation of graphlet distribution, especially for graphlet sizes up to 5 and small graphs. In our case, we perform the graphlet computation only on a local neighborhood of each node for graphlet sizes up to 5. Hence, the existing counting methods can be readily applied in our context. The graphlet computation is done only once for a graph in PMK and SPM. In HPM, the computation is repeated at each level but with reduced number of nodes in the coarsened graph. Our method took under two hours to train even for very large datasets such as NCI109, while neural network-based models are known to take many hours of training for large datasets.

Conclusions and Future Work

We have described a novel method GraphPFM to aggregate node-level embeddings to obtain graph level embedding taking into consideration different levels of feature, spatial and topological resolution. Our method is generic and can be extended to include other types of node-level features, with the suitable clustering of the resultant node embeddings. It is worthwhile to explore other types of node level features, which could also possibly incorporate node level attributes along with its surrounding topology. Furthermore, multiple features like the node-embeddings generated by GCNs can be concatenated with the intrinsic node features by defining appropriate aggregation functions. It will also be interesting to come up with a comprehensive method that can simultaneously aggregate at different levels of resolutions, space and hierarchy of topological features. We plan to explore these ideas in the future.

References

[Ahmed et al. 2015] Ahmed, N. K.; Neville, J.; Rossi, R. A.; and Duffield, N. 2015. Efficient graphlet counting for large networks. In *ICDM*, 1–10. IEEE.

[Alon 2007] Alon, U. 2007. Network motifs: theory and experimental approaches. *Nature Reviews Genetics* 8(6):450.

[Atwood and Towsley 2016] Atwood, J., and Towsley, D. 2016. Diffusion-convolutional neural networks. In *NIPS*, 1993–2001.

[Borgwardt and Kriegel 2005] Borgwardt, K. M., and Kriegel, H.-P. 2005. Shortest-path kernels on graphs. In *ICDM*, 8–pp. IEEE.

[Bressan et al. 2017] Bressan, M.; Chierichetti, F.; Kumar, R.; Leucci, S.; and Panconesi, A. 2017. Counting graphlets: Space vs time. In *ICWSDM*, 557–566. ACM.

[Bruna et al. 2013] Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

[Chen et al. 2016] Chen, X.; Li, Y.; Wang, P.; and Lui, J. 2016. A general framework for estimating graphlet statistics via random walk. *Proc of the VLDB Endowment* 10(3):253–264.

[Chen et al. 2017] Chen, H.; Perozzi, B.; Hu, Y.; and Skiena, S. 2017. Harp: Hierarchical representation learning for networks. *arXiv preprint arXiv:1706.07845*.

[Csurka et al. 2004] Csurka, G.; Dance, C.; Fan, L.; Willamowski, J.; and Bray, C. 2004. Visual categorization with bags of keypoints. In *ECCV Workshop on statistical learning in Computer Vision*, volume 1, 1–2. Prague.

[Defferrard, Bresson, and Vandergheynst 2016] Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 3844–3852.

[Goyal and Ferrara 2017] Goyal, P., and Ferrara, E. 2017. Graph embedding techniques, applications, and performance: A survey. *arXiv preprint arXiv:1705.02801*.

[Grauman and Darrell 2005] Grauman, K., and Darrell, T. 2005. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, volume 2, 1458–1465. IEEE.

[Grover and Leskovec 2016] Grover, A., and Leskovec, J. 2016. Node2vec: Scalable feature learning for networks. In *KDD*, 855–864. ACM.

[Han and Sethu 2016] Han, G., and Sethu, H. 2016. Waddling random walk: Fast and accurate mining of motif statistics in large graphs. In *ICDM*, 181–190. IEEE.

[Haussler 1999] Haussler, D. 1999. Convolution kernels on discrete structures. Technical report, Department of Computer Science, University of California at Santa Cruz.

[Hočevar and Demšar 2014] Hočevar, T., and Demšar, J. 2014. A combinatorial approach to graphlet counting. *Bioinformatics* 30(4):559–565.

[Kriege and Mutzel 2012] Kriege, N., and Mutzel, P. 2012. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483*.

[Lazebnik, Schmid, and Ponce 2006] Lazebnik, S.; Schmid, C.; and Ponce, J. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, volume 2, 2169–2178. IEEE.

[Melckenbeeck et al. 2017] Melckenbeeck, I.; Audenaert, P.; Colle, D.; and Pickavet, M. 2017. Efficiently counting all orbits of graphlets of any order in a graph using autogenerated equations. *Bioinformatics* 34(8):1372–1380.

[Morris, Kersting, and Mutzel 2017] Morris, C.; Kersting, K.; and Mutzel, P. 2017. Global weisfeiler-lehman graph kernels. *arXiv preprint arXiv:1703.02379*.

[Mousavi et al. 2017] Mousavi, S. F.; Safayani, M.; Mirzaei, A.; and Bahonar, H. 2017. Hierarchical graph embedding in vector space by graph pyramid. *Pattern Recognition* 61:245–254.

[Narayanan et al. 2016] Narayanan, A.; Chandramohan, M.; Chen, L.; Liu, Y.; and Saminathan, S. 2016. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *arXiv preprint arXiv:1606.08928*.

[Niepert, Ahmed, and Kutzkov 2016] Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *ICML*, 2014–2023.

[Perozzi, Al-Rfou, and Skiena 2014] Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*, 701–710. ACM.

[Scarselli et al. 2009] Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The graph neural network model. *IEEE Trans on Neural Networks* 20(1):61–80.

[Shervashidze et al. 2009] Shervashidze, N.; Vishwanathan, S.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. 2009. Efficient graphlet kernels for large graph comparison. In *AISTATS*, 488–495.

[Shervashidze et al. 2011] Shervashidze, N.; Schweitzer, P.; Leeuwen, E. J. v.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. *JMLR* 12(Sep):2539–2561.

[Simonovsky and Komodakis 2017] Simonovsky, M., and Komodakis, N. 2017. Dynamic edgeconditioned filters in convolutional neural networks on graphs. In *CVPR*.

[Vishwanathan et al. 2010] Vishwanathan, S. V. N.; Schraudolph, N. N.; Kondor, R.; and Borgwardt, K. M. 2010. Graph kernels. *JMLR* 11(Apr):1201–1242.

[Yanardag and Vishwanathan 2015] Yanardag, P., and Vishwanathan, S. 2015. Deep graph kernels. In *KDD*, 1365–1374. ACM.

[Zhang et al. 2018] Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An end-to-end deep learning architecture for graph classification. In *AAAI*.

[Zheng et al. 2016] Zheng, V. W.; Cavallari, S.; Cai, H.; Chang, K. C.-C.; and Cambria, E. 2016. From node embedding to community embedding. *arXiv preprint arXiv:1610.09950*.

Appendix: Interpretation of Spatial Pyramid Matching (SPM) for Graphs

Theorem 1. *Let the vertices of G be paritioned into $B_1, \ldots, B_z$ based on their node embedding based clustering. Let binary tree T_i correspond to an m level hierarchical clustering of the vertices in B_i based on their shortest path distance in G. SPM is an instance of the embedding $\sum_{v \in V} \psi_m(v)$, where $\psi_m : V \rightarrow \{0,1\}^{z(2^{m+1}-1)}$ is defined based on the given $T_1, \ldots, T_z$.*

Proof. Let $V = \{1, \ldots, n\}$ be the nodes of graph G and let f_i denote the embedding of node i. We define two types of distance functions between every pair of vertices. Let $t(i,j)$ denote the distance between f_i and f_j for vertices i and j, which is used in the clustering in SPM. Let $d(i,j)$ be the shortest path (edge hop count) distance between vertices i and j in G. In SPM, we cluster the node embeddings of all candidate graphs and identify z cluster centers. The number of clusters and the cluster centers remain the same at each level of the SPM hierarchy.

We now define a hybrid embedding of G based on the distances $t()$ and $d()$. We will later show that the SPM embedding is a specific instance of this hybrid embedding. With respect to the z cluster centers, let $(B_1, \ldots, B_z)$ denote the binning of the vertices of G into z bins where B_i consists of the vertices assigned to the ith cluster center. Consider an m level hierarchical clustering of all vertices from a bin say B_i, based on the distance function d between the vertices in B_i. The parent is split into two children at each level of the hierarchy. Consequently, the vertices of G can be viewed as binned into z bins $B_1, \ldots, B_z$ based on distance $t()$ and the vertices in each bin B_i are further clustered hierarchically using m levels based on distance $d()$.

Let T_i denote the binary tree corresponding to the hierarchical clustering of B_i where each vertex in B_i is present in all the nodes on a path from the root to some leaf node of T_i. We define a vertex embedding $\psi_m(u)$ for $u \in V$ with respect to the partition $T_1, \ldots, T_z$ of V. Observing that there are at most $2^{m+1} - 1$ vertices in any T_i and any root to leaf path of length $m+1$ can be encoded using a $2^{m+1} - 1$ binary string where only $m+1$ bits are turned ON, we define $\psi_m : V \rightarrow \{0,1\}^{z(2^{m+1}-1)}$, where $2^{m+1} - 1$ bits are reserved for each tree T_i. In $\psi_m(u)$, $m+1$ bits are turned ON corresponding to the root to leaf path of u in its associated tree T_i. Define $\psi_m(G) = \sum_{u \in V} \psi_m(u)$.

Embedding $\psi_m(G)$ attempts to strike a balance between the local topology and the global interconnections. Two vertices u and v can fall into the same bin B_i due to their local similarity. But, their relative localizations with respect to other vertices in B_i could contribute differently to the final $\psi_m(u)$ and $\psi_m(v)$ because of the difference in the paths corresponding to u and v in tree T_i.

It is straightforward to see that the SPM embedding $\phi(G)$ is a special case of $\psi_m(G)$ where the specific hierarchical clustering is given by the BFS based graph split. Clearly, for $m = 0$, $\phi(G) = H_0(G) = \psi_0(G)$. We recall that for $m > 0$, each vertex $u \in V$ contributes a '1' to $m+1$ coordinates of $\phi(G)$ where these coordinates map to the hierarchy of bins to which u belongs to as a result of the m level BFS based splitting of V. Moreover, recalling that the z cluster centers for vertex binning remain the same at each of the $m+1$ levels, it is straightforward to see that all vertices contributing to a level i coordinate is split (due to BFS) across two new coordinates created for level $i+1$. Thus, it follows that $\phi(G)$ is identical to $\psi_m(G)$ under a permutation of the coordinates. □

Appendix: Visual Illustrations of PMK, SPM and HPM Methods

In support of our methodologies presented for aggregation of node embeddings in Section , we present visual illustrations of each of our aggregation methodologies: PMK, SPM and HPM in Figures 3, 4, and 5 respectively, for convenience of understanding.

Appendix: Dataset Description

In continuation to the summary statistics of the datasets presented in Table 1, we describe each of the datasets here. MUTAG is a dataset of 188 mutagenic or otherwise aromatic and heteroaromatic nitro compounds. PTC is a dataset of 344 chemical compounds that reports the carcinogenicity for male and female rats. NCI1 and NCI109 datasets, made publicly available by the National Cancer Institute (NCI), are two subsets of balanced datasets of chemical compounds screened for ability to suppress or inhibit the growth of a panel of human tumor cell lines. ENZYMES is a balanced dataset of 600 protein tertiary structures obtained from the BRENDA database. It consists of 100 proteins from each of the Enzyme Commission top level enzyme classes. PROTEINS is a two-class dataset indicating if a protein is a enzyme or otherwise, where nodes are secondary structure elements (SSEs) and there is an edge between two nodes if they are neighbors in the amino-acid sequence or in 3D space.

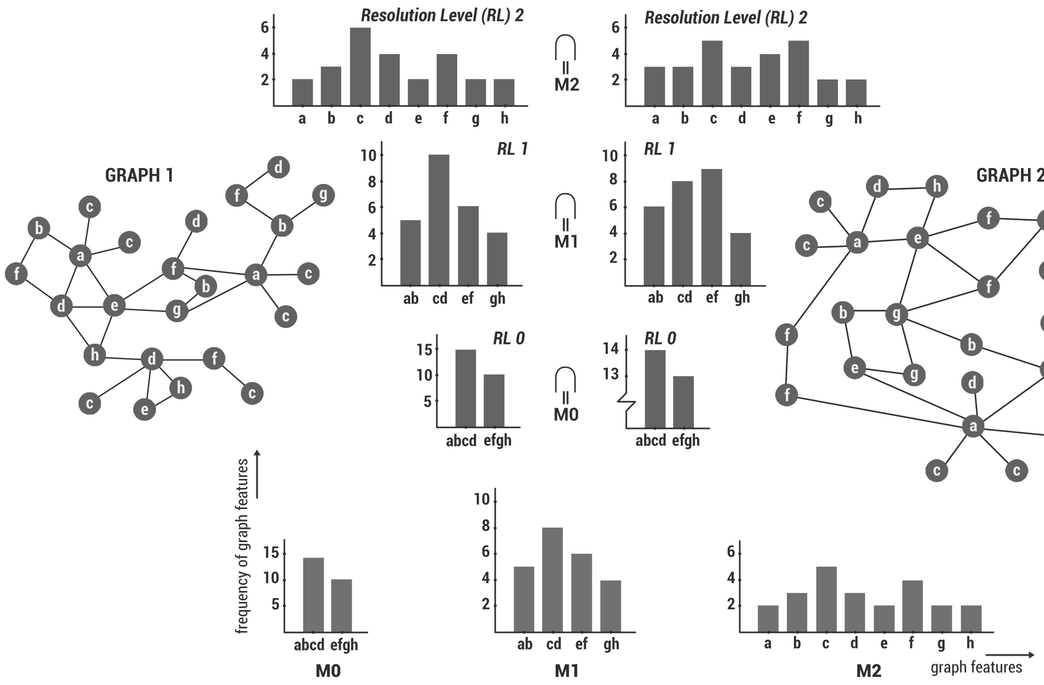

Figure 3: Pyramid match kernel for graph classification: Eight clusters with centres denoted as ‘a’ to ‘h’. ‘ab’, ‘cd’, ‘ef’ and ‘gh’ are cluster centres obtained for 4 clusters at the next level of resolution. All nodes labelled ‘a’ or ‘b’ fall in the cluster ‘ab’ and so on. ‘abcd’ and ‘efgh’ are cluster centres obtained for 2 clusters. All nodes labelled ‘a’, ‘b’, ‘c’ or ‘d’ fall in the cluster ‘abcd’ and so on.

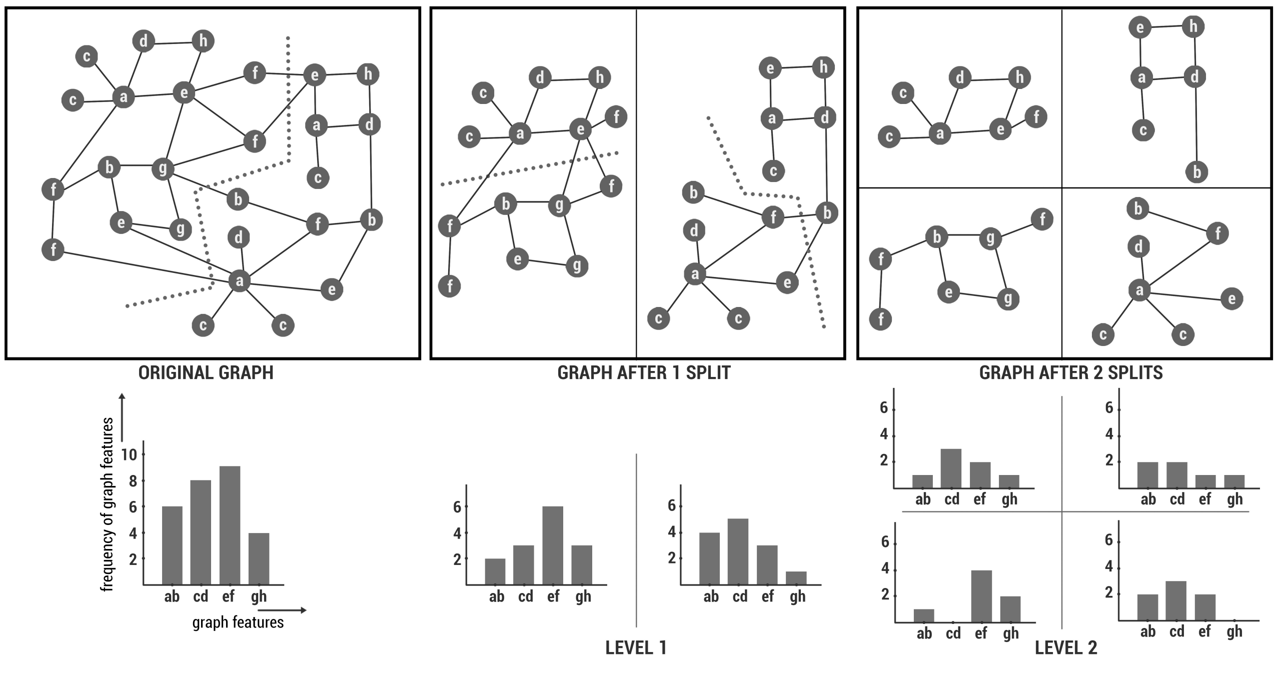

Figure 4: Spatial pyramids for graph representation. Here, it is assumed that ‘ab’, ‘cd’, ‘ef’, ‘gh’ are cluster centers closest to the nodes a & b, c & d, e & f, g & h respectively.

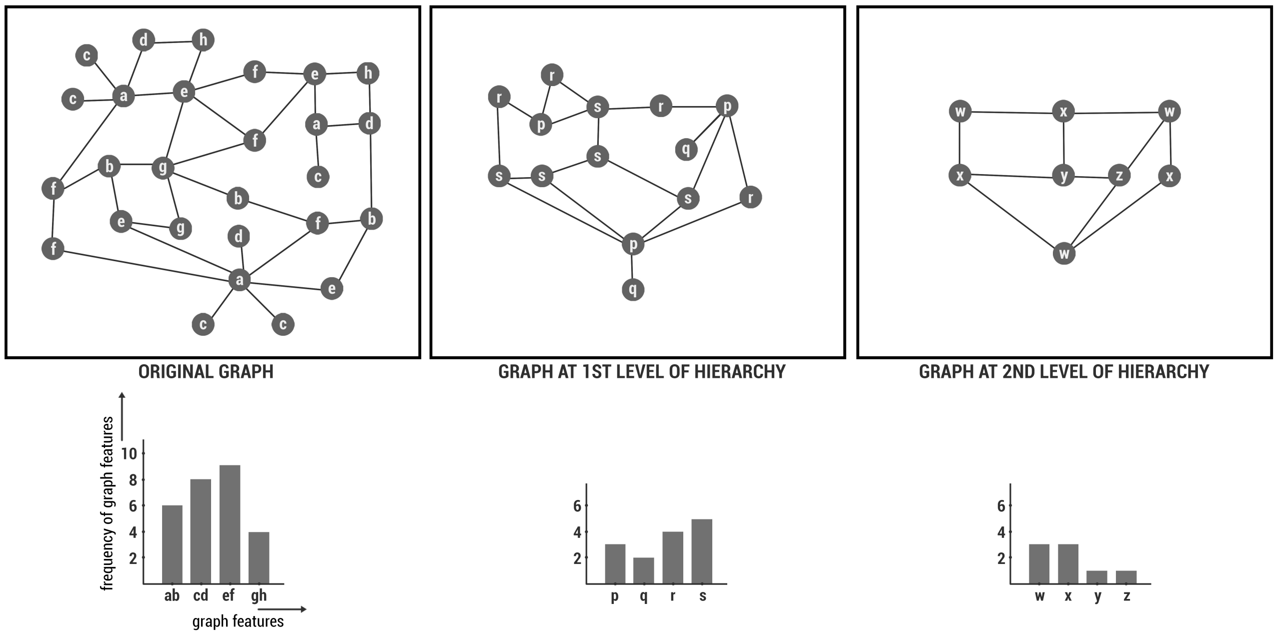
ORIGINAL GRAPH
GRAPH AT 1ST LEVEL OF HIERARCHY
GRAPH AT 2ND LEVEL OF HIERARCHY
frequency of graph features
graph features
ab
cd
ef
gh
p
q
r
s
w
x
y
z

Figure 5: Hierarchical pyramids for graph representation For the 3 levels when clustering is carried out with 4 centres, the cluster centres obtained are as shown above. The nodes at each level are different as they are formed by collapsing nodes from the previous level.