Neural Computing and Applications manuscript No.
(will be inserted by the editor)

Hierarchical Stochastic Graphlet Embedding for Graph-based Pattern Recognition

Anjan Dutta · Pau Riba · Josep Lladós · Alicia Fornés

Received: 1 August 2019 / Accepted: 22 November 2019

Abstract Despite being very successful within the pattern recognition and machine learning community, graph-based methods are often unusable because of the lack of mathematical operations defined in graph domain. Graph embedding, which maps graphs to a vectorial space, has been proposed as a way to tackle these difficulties enabling the use of standard machine learning techniques. However, it is well known that graph embedding functions usually suffer from the loss of structural information. In this paper, we consider the hierarchical structure of a graph as a way to mitigate this loss of information. The hierarchical structure is constructed by topologically clustering the graph nodes, and considering each cluster as a node in the upper hierarchical level. Once this hierarchical structure is constructed, we consider several configurations to define the mapping into a vector space given a classical graph embedding, in particular, we propose to make use of the Stochastic Graphlet Embedding (SGE). Broadly speaking, SGE produces a distribution of uniformly sampled low to high order graphlets as a way to embed graphs into the vector space. In what follows, the coarse-to-fine structure of a graph hierarchy and the statistics fetched by the SGE complements each other and includes important structural information with varied contexts. Altogether, these two techniques substantially cope with the usual information loss involved in graph embedding techniques, obtaining a more robust graph representation. This fact has been corroborated through a detailed experimental evaluation on various benchmark graph datasets, where we outperform the state-of-the-art methods.

Keywords Graph embedding · Hierarchical graph · Stochastic graphlets · Graph hashing · Graph classification

1 Introduction

Graph-based methods have been very successful for pattern recognition, computer vision and machine learning tasks [16,25,77]. However, due to their symbolic and relational nature, graphs have some limitations if we compare them with the traditional statistical (vector-based) representations. Some trivial mathematical operations do not have an equivalence in the graph domain. For example, computing pairwise sums or products (which are elementary operations in many classification and clustering algorithms) is not defined in a standard way in the graph domain. In the literature, a possible way this problem has been addressed is by means of embedding functions. Given a graph space $\mathbb{G}$, an *explicit embedding* function is defined as $\varphi : \mathbb{G} \to \mathbb{R}^n$ which maps a given graph to a vector representation [12,29,47,65,68] whereas an *implicit embedding* function is defined as $\varphi : \mathbb{G} \to \mathcal{H}$ which maps a given graph to a high dimensional Hilbert space $\mathcal{H}$ where a dot product defines the similarity between two

Anjan Dutta and Pau Riba have contributed equally to this work.

Anjan Dutta
Department of Computer Science, University of Exeter, Innovation Centre, Streatham Campus, Exeter, EX4 4RN, UK
Tel.: +44 1392 724067
E-mail: A.Dutta@exeter.ac.uk

Pau Riba · Josep Lladós · Alicia Fornés
Computer Vision Center, Autonomous University of Barcelona, Edifici O, Campus UAB, Bellaterra, 08193 Barcelona, Spain
Tel.: +34 93 581 18 28
Fax: +34 93 581 16 70
E-mail: {priba,josep,afornes}@cvc.uab.es

graphs $K(G, G') = \langle \varphi(G), \varphi(G') \rangle$, $G, G' \in \mathbb{G}$ [18,27, 32,35]. In the graph domain, the process of implicitly embedding graph is termed as *graph kernel* which basically defines a way to compute the similarity between two graphs. However, defining such embedding functions is extremely challenging, when the constraints on time efficiency and preserving the underlying structural information is concerned. The problem becomes even more difficult with the growing size of graphs, as the structural complexity increases the possibility of noise and distortion in structure, and raises risk of loosing information. Hierarchical representation is often used as a way to deal with noise and distortion [50,76], which provides a stable delineation for an underlying object. Hierarchical representations allow to incrementally contract the graph, in a space-scale representation, so the salient features (relevant subgraphs) remain in the hierarchy. Thus, top levels become a compact and stable summarization.

Processing information using a multiscale representation is successfully employed in computer vision and image processing algorithms, which is mostly inspired by its resemblance with human visual perception [1]. It is observed that a naturalistic visual interpretation always demands a data structure able to represent scattered local information as well as summarized global facts [33]. Hierarchical representation is often used as a paradigm to efficiently extract the global information from the local features. Apart from that, hierarchical models are also believed to provide time and space efficient solutions [76]. Motivated by the above mentioned intuition and the existing works in the related fields, many authors have come up with different hierarchical graph structures for solving various problems [22,23, 48,76]. In this sense, it is worth to mention the work of Mousavi *et al.* [50], who presented a hierarchical framework for graph embedding, although they did not explore the complex encoding of the hierarchy.

In this paper, motivated by the successes of the hierarchical models and the efficiency of graph embedding theory, we propose a general hierarchical graph embedding formulation that first creates a hierarchical structure from a given graph, and then utilizes the multiscale structure to explicitly embed a graph in a real vector space by means of local graphlets. First, we make use of the graph clustering algorithm proposed in [31] to obtain a hierarchical graph representation of a given input graph. Here, each cluster of nodes in a level i is depicted as a single node in the upper hierarchical level $i + 1$, whereas the edges in a level are connected depending on the original topology of the base graph, and the hierarchical edges are created by joining a node representing a cluster to all the nodes in the lower level. Thus we propose a richer encoding than Mousavi [50], because our hierarchy not only contains different graph abstractions but also encodes useful hierarchical contractions through the hierarchical edges.

Once the hierarchical structure of a graph is created, we propose a novel use of the *Stochastic Graphlet Embedding* (SGE) [21] to exploit this hierarchical information. On the one hand, we can exploit the local configuration in form of graphlets thanks to the SGE design, because graphlets provide information at different neighborhood sizes. On the other hand, the hierarchical connections allow to encode more abstract information and hence to deal with noise present in the data. As a result, the Hierarchical Stochastic Graphlet Embedding (HSGE) encodes a global and compact representation of the graph that is embedded in a vector space. The consideration of the entire graph hierarchy for the embedding instead of only the base graph empowers the representation ability and handles the loss of information that usually occurs in graph embedding methods. Moreover, the statistics obtained from the uniformly sampled graphlets of increasing size model the complex interactions among different object parts represented as graph nodes. Here, the hierarchical graph structure and the statistics of increasing sized graphlets fetch important structural information of varied contexts.

As a result, our approach produces robust representations that can benefit from the advantages of the two above mentioned strategies: we first take advantage of the embedding ability for mapping symbolic relational representations to n-dimensional spaces, so machine learning approaches can be used; and second, the ability of hierarchical structures to reduce noise and distortion inherently involved in graph representations of real data, keeping the more stable and relevant substructures in a compact way.

In conclusion, the main contribution of our work is the exploitation of the hierarchical structure of a given graph, rather than only studying the base graph for graph embedding purposes. Assessing the hierarchical information of a graph pyramid allows to extend the representation power of the embedded graph and tolerate the instability caused due to noise and distortion. Our proposal is robust because, on the one hand, it organizes the structural information in the hierarchical abstraction, and on the other hand, it considers the relation between object parts and their complex interactions with the help of uniformly sampled graphlets of unbounded size. Additionally, the proposed method is generic and can adapt any other graph embedding algorithm in the framework. In this sense, we extensively validated our proposed algorithm on many different benchmark graph datasets coming from different application domains.

The rest of this paper is organized as follows: Sect. 2 describes the related works in the literature. In Sect. 3, we introduce some definitions and notations related to the work. Our generic hierarchical graph representation is presented in Sect. 4. Sect. 5 introduces the Stochastic Graphlet Embedding as the base embedding we will use. Afterward, Sect. 7 reports our experimental validation and compares the proposed method with available state-of-the-art algorithms. Finally, in Sect. 8 we draw the conclusions and describe the future direction of the present work.

2 Related work

In what follows, we review the related works respectively on explicit and implicit graph embedding techniques, different hierarchical models and graph summarization methods, which we believed to be relevant to the main focus of the present paper.

2.1 Graph embedding

Graph embedding methods are mainly divided into two different categories: (1) explicit graph embedding, (2) implicit graph embedding or graph kernel.

2.1.1 Explicit graph embedding

Explicit graph embedding refers to those techniques that aim to explicitly map graphs to vector spaces. The methods belonging to this category can be further divided into four different classes. The first one, known as *graph probing* [47], needs measuring the frequency of specific substructures (that capture content and topology) into graphs. Based on different graph substructures (*e.g.*, node, edge, subgraph etc.) considered, different embedding techniques have been proposed. For example, Shervashidze *et al.* [68] studied the non-isomorphic graphlets, albeit, node label and edge relation statistics are considered by Gibert *et al.* [29]. Saund in [65], introduced a bottom up graph lattice in order to efficiently extract the subgraph features in preprocessed administrative documents, while Dutta and Sahbi [21] proposed a distribution of stochastic graphlets for embedding graphs into a vector space. The second class of graph embedding techniques is based on *spectral graph theory* [13,82,64,37,39,34], which aims to analyze the structural properties of graphs in terms of the eigenvectors/eigenvalues of the adjacency or Laplacian matrices of a graph [82]. Recently, Verma and Zhang [78] proposed a family of graph spectral distances for robust graph feature representation. Despite of their relative successes, spectral methods are quite prone to structural noise and distortions. The third class of methods is inspired by *dissimilarity measurements* proposed in [56]; in this context, Bunke and Riesen have presented several works on the vectorial description of a given graph by its distances to a number of pre-selected prototype graphs [63,62,12,9]. Motivated by the recent advancements of deep learning and neural networks, many researchers have proposed to utilize neural network for obtaining a vectorial representation of graphs [4,17,36,55,30], which results in the fourth category of methods, called *geometric deep learning.*

2.1.2 Implicit graph embedding

Implicit graph embedding or graph kernel methods is primarily another way to embed graphs into a vector space. They are also popular for the ability to efficiently extend the existing machine learning algorithms to non-linear data, such as, graphs, strings etc. Graph kernel methods can be roughly divided into three different categories. The first one, known as *diffusion kernel*, is based on the similarity measures among the subparts of two graphs, and propagating them on the entire structure to obtain global similarity measure for two graphs [72,43]. The second class of methods, called as *convolution kernel*, aims to measure the similarity of composite objects (modeled with graph) from the similarity of their parts (*i.e.* nodes) [80]. This type of graph kernel derives the similarity between two graphs G, G' from the sum, over all decompositions, of the similarity products of the subparts of G and G' [52]. Recently, Kondor and Pan [38] proposed multiscale Laplacian graph kernel having the property of lifting a base kernel defined on the vertices of two graphs to a kernel between graphs. The third class of methods is based on the analysis of the common substructures that belong to both graphs, and is termed as *substructure kernel.* This family includes the graph kernel methods that consider random walks [27,79], backtrackless walks [5], shortest paths [8], subtrees [68], graphlets [70] as the substructure. Different from the above three categories, Shervashidze *et al.* [69] proposed a family of efficient graph kernels on the Weisfeiler-Lehman test of graph isomorphism, which maps the original graph to a sequence of graphs. More recently, inspired by the successes of deep learning, Yanardag and Viswanathan [83] presented a unified framework to learn latent representations of substructures for graphs. They claimed that given a pre-computed kernel of graphs, their proposed technique produces an improved representation that leverages hidden representations of sub-structures.

2.2 Hierarchical graph representation

In general, hierarchical models have been successfully employed in many different domains within the computer vision and image processing field, such as, image segmentation [48, 22], scene categorization [23], action recognition [54], shape classification [18], graphic recognition [10], 3D object recognition [76] etc. These approaches usually exploit some kind of pyramidal structure containing information at various resolutions. Usually, at the finest level of the pyramid, the captured information is related to local features, whereas, at coarser levels, global aspects of the underlying data are represented. This way of representation helps to interpret knowledge in a naturalistic way [33].

Inspired by the above intuition, hierarchical structures are often employed to extract coarse-to-fine information from a graph representation. Pelillo *et al.* [57] proposed to match two hierarchical structures as a clique detection problem on their association graph, which was solved with a dynamic programming approach. In [71], Shokoufandeh *et al.* presented a spectral characterization based framework for indexing hierarchical structures that embed the topological information of a directed acyclic graph. Hierarchical representation of objects and an elastic matching procedure are also proposed from deformable shape matching in [24]. In [46], Liu *et al.* utilized hierarchical graph representation and a stochastic sampling strategy for layered shape matching and registration problem. A graph kernel based on hierarchical bag-of-paths where each path is associated to a hierarchy encoding successive simplifications is presented in [18]. Ahuja and Todorovic [2] used a hierarchical graph of segmented regions for object recognition. Motivated by them, Broelemann *et al.* [10, 11] proposed two closely related approaches based on hierarchical graph for error tolerant matching of graphical symbols. Mousavi *et al.* [50] proposed a graph embedding strategy based on hierarchical graph representation, which considers different levels of a graph pyramid. They claimed that the proposed framework is generic enough to incorporate any kind of graph embedding technique. However, the authors did not take advantage of the complex and rich encoding of hierarchy.

From the literature review we can conclude that although there are some works in the graph domain exploiting the hierarchical graph structure, most of them are focused on some kind of error tolerance or elastic matching. Utilization of this type of multiscale representation of graph for vector space embedding is quite rare and has not been properly explored yet. This fact has worked as our motivation to work on a graph hierarchical structure for explicit graph embedding task.

3 Definitions and notations

In this section, we introduce some definitions and notations, which are relevant to the proposed work.

Definition 1 (Attributed graph) An *attributed graph* is a $4-$tuple $G = (V, E, L_V, L_E)$ comprising a set V of *vertices* together with a set $E \subseteq V \times V$ of *edges* and two *mappings* $L_V : V \to \mathbb{R}^m$ and $L_E : E \to \mathbb{R}^n$ which respectively assign attributes to the nodes and edges.

Attributed graphs have been widely used for all sort of real world problems. The most common methodologies are error tolerant graph matching [67, 51], graph kernels and embedding techniques [41].

Definition 2 (Subgraph) Given an attributed graph $G = (V, E, L_V, L_E)$, another attributed graph $G' = (V', E', L'_V, L'_E)$ is said to be a *subgraph* of G and is denoted by $G' \subseteq G$ iff,

- $V' \subseteq V$
- $E' = E \cap V' \times V'$
- $L'_V(u) = L_V(u)$, $\forall u \in V'$
- $L'_E(e) = L_E(e)$, $\forall e \in E'$

A *graphlet* g of G is nothing but a subgraph which inherits the topology and the attributes of G. In the literature, subgraphs are often used for error tolerant matching [75, 66, 73, 7, 19] and frequent pattern discovery problems [42, 6, 2].

Definition 3 (Hierarchical graph) A *hierarchical graph* H is defined as a 6-tuple $H = (V, E_N, E_H, L_V, L_{E_N}, L_{E_H})$ where V is the set of nodes; $E_N \subseteq V \times V$ are the neighborhood edges; $E_H \subseteq V \times V$ are the hierarchical edges; $\mathrm{L_V}$, $\mathrm{L_{E_N}}$ and $\mathrm{L_{E_H}}$ are three labeling functions defined as $\mathrm{L_V} : V \to \Sigma_V \times A_V^k$, $\mathrm{L_{E_N}} : E_N \to \Sigma_{E_N} \times A_{E_N}^l$ and $\mathrm{L_{E_H}} : E_H \to \Sigma_{E_H} \times A_{E_H}^m$, where Σ_V, Σ_{E_N} and Σ_{E_H} are three sets of symbolic labels for vertices and edges, A_V, A_{E_N} and A_{E_H} are three sets of attributes for vertices and edges, respectively, and $k, l, m \in \mathbb{N}$.

Prior works used hierarchical structures for allowing a reasonable tolerance in the representation paradigm [24, 18, 11] and also for bringing robustness in the feature representation [46].

4 Hierarchical embedding

In the literature, only few embedding approaches exploit the idea of multiscale or abstraction information [38]. This section is devoted to provide a framework able to include this information given a graph embedding. Some works that have been proposed to exploit the mentioned multiscale information in the literature [50, 59, 20], discard the hierarchical information provided by the hierarchical edges and focus on abstractions of the original graph.

4.1 Graph clustering

Graph clustering has been widely used in several fields such as social and biological networks [31], recommendation systems [28,44] etc. It can be roughly described as the task of grouping graph nodes into clusters depending on the graph structure. Ideally, the grouping should be performed in such a way that intra-cluster nodes are densely connected whereas the connections among inter-cluster nodes are sparse. For example, Girvan and Newman [31] propose a graph clustering algorithm to detect a community structures for studying social and biological networks. Li *et al.* [45,40,44,28] have proposed several graph clustering techniques for recommendation systems based on different strategies: context awareness [28], inclusion of frequency property [44], distributed clustering confidence [40], etc. Here we do not further review on graph clustering algorithms since it is not within the main scope of this paper. However, we would like to remark that one of the most important aspects of graph clustering is the evaluation of cluster quality, which is crucial not only to measure the effectiveness of clustering algorithms, but also to give insights on the dynamics of relationships in a given graph. For a detailed overview on effective graph clustering metrics, the interested readers are referred to [3].

Even though any graph clustering algorithm can be used, we use the standard divisive-based *Girvan-Newman* algorithm [31] algorithm for our purpose, because it provides structurally meaningful clusters of a given graph. The *Girvan-Newman* algorithm is an intuitive and well-known algorithm used for community detection in complex systems. It is a global divisive algorithm which removes the appropriate edge iteratively until all the edges are deleted. At each iteration, new clusters can emerge by means of connected components. The idea is that the edges with higher centrality are the candidates to be connecting two clusters. Therefore, *betweenness centrality* measure of the edges [26] is used to decide which edge is being removed. *Betweenness centrality* on an edge $e \in E$ is defined as the number of shortest walks between any pair of nodes that cross e. The output of this algorithm is a dendrogram codifying a hierarchical clustering of nodes. This algorithm consists of 4 steps:

1. Calculate the betweenness centrality for all edges in the network.
2. Remove the edge with highest betweenness and generate a cluster for each connected component.
3. Recalculate betweennesses for all edges affected by the removal.
4. Repeat from step 2 until no edges remain.

In this work, *Girvan-Newman* algorithm is early stopped given a reduction ratio $r \in \mathbb{R}$. Therefore, the number of clusters is forced to be $\lfloor r \cdot |V| \rfloor$.

4.2 Hierarchical construction

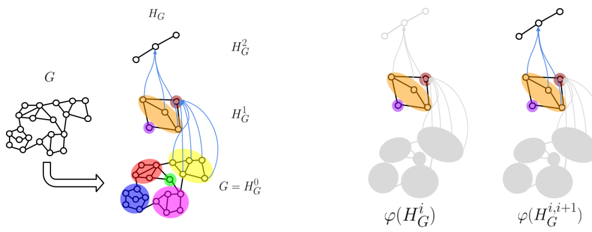
Given a graph G and a clustering $C = \{C_1, \ldots, C_k\}$, each cluster is summarized into a new node with a representative label (see line 5). Let us consider that this label can be defined as the result of an embedding function applied to the subgraph defined by the clustered nodes and their edges. Moreover, edges between the new nodes are created depending on a connection ratio between clusters. That means that an edge is only created if there are enough connections between the set of nodes defined by both clusters (see line 7). Finally, hierarchical edges are created connecting the new node v_{C_i} with all the nodes belonging to the summarized cluster C_i (see line 12). The proposed hierarchical construction is similar to the one proposed by Mousavi *et al.* [50] but including explicitly the summarization generated by the clustering algorithm by means of the hierarchical edges. Thus, the proposed hierarchical construction obtains a representation which encodes abstract information by means of the clusters while keeping the relation with the original graph.

Let us introduce some notations that will be used in the following sections. Given a graph G and a number of levels L, H_G denotes their corresponding hierarchical graph computed from G with L levels. H_G^l, where $l = \{0, \ldots, L\}$ is a graph without hierarchical edges corresponding to the l level of summarization, therefore, $H_G^0 = G$. Moreover, $H_G^{l_1,l_2}$ where $l_i = \{0, \ldots, L\}$ and $l_1 \leq l_2$, corresponds to the hierarchical graph compressed between levels l_1 and l_2. Hence, $H_G = H_G^{0,L}$ and $H_G^l = H_G^{l,l}$. Finally, $H_G^{l_1} \cup H_G^{l_2}$ corresponds to the union of two graphs without hierarchical edges.

Fig. 1a shows the construction of the hierarchy given a graph G. Each level shows an abstraction of the input graph where the nodes have been reduced.

Algorithm 1 Pyramidal Graph Construction(G)

```
Require: $G = (V, E)$, $L$, $\varepsilon$, $\delta$
Ensure: $H = (V, E_N, E_H)$
$H \leftarrow G$; $G_c \leftarrow G$
for $i = 1$ to $L$ do
    $K = \lfloor \varepsilon \cdot |G_c.V| \rfloor$
    $\{C_1, \dots, C_K\} \leftarrow$ ClusterGraph$(G_c, K)$
    $G_n.V = \{$ConsiderAsVertex$(C_j) : j = \{1, \dots, K\}\}$
    for $(u, v) \in G_n.V \times G_n.V$ do
        if ConnectionRatio$(u, v) \geq \delta$ then
            $G_n.E \leftarrow G_n.E \cup (u, v)$
        end if
    end for
    for $j = 1$ to $K$ do
        $E_H \leftarrow E_H \cup \{(u, w) \in G_c.V \times G_n.V : \forall u \in C_j$ and $w = G_n.V_j\}$
    end for
    $G_c \leftarrow G_n$
    $H \leftarrow$ IncludeToHierarchy$(H, G_c, E_H)$
end for
```



(a) (b)

Fig. 1 (a) Hierarchical graph construction is proposed in Algorithm 1. The input graph G is processed to generate a hierarchical graph H_G where each level H_G^i encodes a new abstraction of the original graph. Moreover, hierarchical edges provide the insights of the performed contraction. In this figure, not all the hierarchical edges have been drawn to make it easy to understand, and the node clustering is drawn in color. (b) Following the hierarchical graph construction in Fig. 1a, the graphs taken into consideration in order to construct the hierarchical embedding are shown. $\varphi(H_G^i)$ takes into account one abstraction level whereas $\varphi(H_G^{i,i+1})$ takes into consideration two of these levels and the hierarchical edges involved. (Best viewed in color)

4.3 Hierarchical embedding

This section introduces a novel way to encode hierarchical information of a graph into an embedding. Moreover, the proposed technique is generic in the sense that can be used by any graph embedding function.

Given a graph G which should be mapped into a vectorial space and an embedding function $\varphi : \mathbb{G} \rightarrow \mathbb{R}^n$, we first proceed to obtain hierarchical representation H_G following the proposed methodology in Sect. 4.2. Therefore, H_G has enriched the original graph with abstract information considering L levels. Finally, we propose to make use of the hierarchical information to construct a hierarchical embedding. The general form of the proposed embedding takes into account graphs at multiple scales and hierarchical relations. Thus, the embedding function does not only compactly encode the contextual information of nodes at different abstraction levels, but also it encodes the hierarchy contraction. The embedding function is defined as follows:

$$\begin{aligned} \Phi(H_G) = [&\varphi(H_G^0), \dots, \varphi(H_G^K), \\ &\phi_1^1(H_G), \dots, \phi_1^{k_1}(H_G), \\ &\phi_2^1(H_G), \dots, \phi_2^{k_2}(H_G)] \end{aligned} \tag{1}$$

where,

$$\phi_1^k(H_G) = [\varphi(H_G^{0,k}), \dots, \varphi(H_G^{K-k,K})] \tag{2}$$

$$\phi_2^k(H_G) = [\varphi(H_G^0 \cup \dots \cup H_G^k), \dots, \varphi(H_G^{K-k} \cup \dots \cup H_G^K)] \tag{3}$$

where $K \leq L$ are the hierarchical levels taken into account and $k_1, k_2 \leq K$ indicate the number of levels taken into account at the same time. Note that $K = L$, $k_1 = K$ and $k_2 = K$ will take into account the whole hierarchy and possible combinations. From this general representation of the proposed embedding, we have evaluated some particular cases (the reader is referred to Sect. 7 for more details on the experimental evaluation).

Baseline embedding: This embedding is the one used as a baseline. In this scenario $K = 0$, $k_1 = 0$ and $k_2 = 0$, therefore $\Phi(H_G) = \varphi(H_G^0)$. No abstract information is taken into consideration, hence, $\Phi(H_G) = \varphi(G)$.

Pyramidal embedding: This embedding has been previously proposed in the literature [50, 20]. It combines information of the abstract levels of the graph *i.e.* H_G^i not taking into account hierarchical information. Therefore, the hierarchical edges are discarded and no relation between levels is considered, $K \geq 1$, $k_1 = 0$ and $k_2 = 0$. We define $\Phi_{\text{pyr}}(H_G) = [\varphi(H_G^0), \dots, \varphi(H_G^K)]$. Note that each element corresponds to independent levels of the hierarchy without hierarchical edges.

Generalized pyramidal embedding: Following the previous idea, the information of the abstract levels of the graph *i.e.* H_G^i is combined. Now, hierarchical information is taken into account by embedding unions of levels *i.e.* $H_G^{i_1} \cup H_G^{i_2}$ but discarding hierarchical edges (no clustering information is taken into account). In this scenario $K \geq 1$, $k_1 = 0$ and $k_2 \geq 1$, therefore, we define $\Phi_{\text{gen\_pyr}}(H_G) = [\varphi(H_G^0), \dots, \varphi(H_G^K), \varphi(H_G^0 \cup H_G^1), \dots, \varphi(H_G^{K-1} \cup H_G^K), \dots, \varphi(H_G^0 \cup \dots \cup H_G^{k_2}), \dots, \varphi(H_G^{K-k_2} \cup \dots \cup H_G^K)]$.

Hierarchical embedding: This embedding is computed mixing different levels considering them as a single graph through the hierarchical edges, $K \geq 1$, $k_1 \geq 1$ and $k_2 = 0$. The idea is to create an embedding able to codify both, graph and clustering information. Depending on the embedding, hierarchical edges can make use of special label to treat them differently. The Hierarchy embedding is defined as $\Phi_{\text{hier}}(H_G) = [\varphi(H_G^0), \ldots, \varphi(H_G^K), \varphi(H_G^{0,1}), \ldots, \varphi(H_G^{K-1,K}), \ldots, \varphi(H_G^{0,k_1}), \ldots, \varphi(H_G^{K-k_1,K})]$. Note that each element corresponds to the subhierarchy compressed between the specified levels.

Exhaustive embedding: Finally, in order to take into consideration the whole hierarchy, we can make use of the whole embedding Φ as defined in eqn. (1) where $K \geq 1$, $k_1, k_2 \geq 1$.

Fig. 1b shows the graphs taken into consideration when the hierarchical embeddings are computed.

5 Stochastic graphlet embedding

The *Stochastic Graphlet Embedding* (SGE) can be defined as a function $\varphi : \mathbb{G} \rightarrow \mathbb{R}^n$ that explicitly embeds a graph $G \in \mathbb{G}$ to a high dimensional vector space $\mathbb{R}^n$ [21]. The entire procedure of SGE can be described in two stages (see Fig. 2), where in the first step, the method samples graphlets from G in a stochastic manner and in the second step, it counts the frequency of each isomorphic graphlet from the extracted ones in an approximated but near accurate manner. The entire procedure fetches a precise distribution of connected graphlets with increasing number of edges in G with a controlled complexity, which fetches the relation among information represented as nodes and their complex interaction.

5.1 Stochastic graphlets sampling

Considering a graph $G = (V, E, L_V, L_E)$, the goal of the graphlet extraction procedure is to obtain statistics of stochastic graphlets with increasing number of edges in G. The way of extracting graphlets is stochastic and it uniformly samples graphlets with boundlessly increasing number of edges without constraining their topology or structural properties such as maximum degree, maximum number of nodes, etc. Our graphlet sampling procedure, outlined in Algorithm 2, is recurrent and the number of recurrences is controlled by a parameter M that indicates the number of distinct graphlets to be sampled (see line 2 of Algorithm 2). Also, each of these M recurrent processes is regulated by another parameter T that denotes the maximum number of iterations a single recurrent process should have (see line 5). Since each of these iterations adds an edge to the presently constructing graphlet, T indirectly specifies the maximum number of distinct edges each graphlet should contain. Considering U_t and A_t respectively as the aggregated sets of visited nodes and edges till iteration t, they are initialized at the beginning of each recurrent step as $A_0 = \emptyset$ and $U_0 = \{u\}$ with a randomly selected node u which is uniformly sampled from V (see line 4). Thereafter, at t^{th} iteration (with $t \geq 1$), the sampling procedure randomly selects an edge $(u, v) \in E \backslash A_{t-1}$ that is connected from any node $u \in U_{t-1}$ (see line 7). Accordingly, the process updates $U_t \leftarrow U_{t-1} \cup \{v\}$ and $A_t \leftarrow A_{t-1} \cup \{(u, v)\}$ (see line 8). All these processes within a recurrent step are repeated T times to sample a graphlet with maximum T edges. M is set to relatively large values in order to make the graphlet generation statistically meaningful. Theoretically, the values of M are guided by the theorem of *sample complexity* [81], which is widely studied and used in the Bioinformatics domain [58, 70]. However, the discussion and proof of that is out of scope of the current paper. Intuitively, the graphlet sampling procedure explained in this section follows a random walk process with restart that efficiently parses G and extracts the desired number of connected graphlets with an increasing number of edges. This algorithm allows to sample connected graphlets from a given graph but avoids expensive way of extracting them in an exact manner. Here the hypothesis is that if a sufficient number of graphlets are sampled, then the empirical distribution will be close to the actual distribution of graphlets in the graph. Furthermore, it is important to note that from the above process, one can extract, in total, $M \times T$ graphlets each with number of edges varying from 1 to T.

Algorithm 2 STOCHASTIC-GRAPHLET-PARSING(G): Create a set of graphlets $\mathbb{S}$ by traversing G.

```
Require: $G = (V, E)$, $M$, $T$
Ensure: $\mathbb{S}$
 1: $\mathbb{S} \leftarrow \emptyset$
 2: for $i = 1$ to $M$ do
 3:     $u \leftarrow$ SELECTRANDOMNODE($V$)
 4:     $U_0 \leftarrow u$, $A_0 \leftarrow \emptyset$
 5:     for $t = 1$ to $T$ do
 6:         $u \leftarrow$ SELECTRANDOMNODE($U_{t-1}$)
 7:         $v \leftarrow$ SELECTRANDOMNODE($V$) : $(u, v) \in E \setminus A_{t-1}$
 8:         $U_t \leftarrow U_{t-1} \cup \{v\}$ ,    $A_t \leftarrow A_{t-1} \cup \{(u, v)\}$
 9:         $\mathbb{S} \leftarrow \mathbb{S} \cup \{(U_t, A_t)\}$
10:     end for
11: end for
```

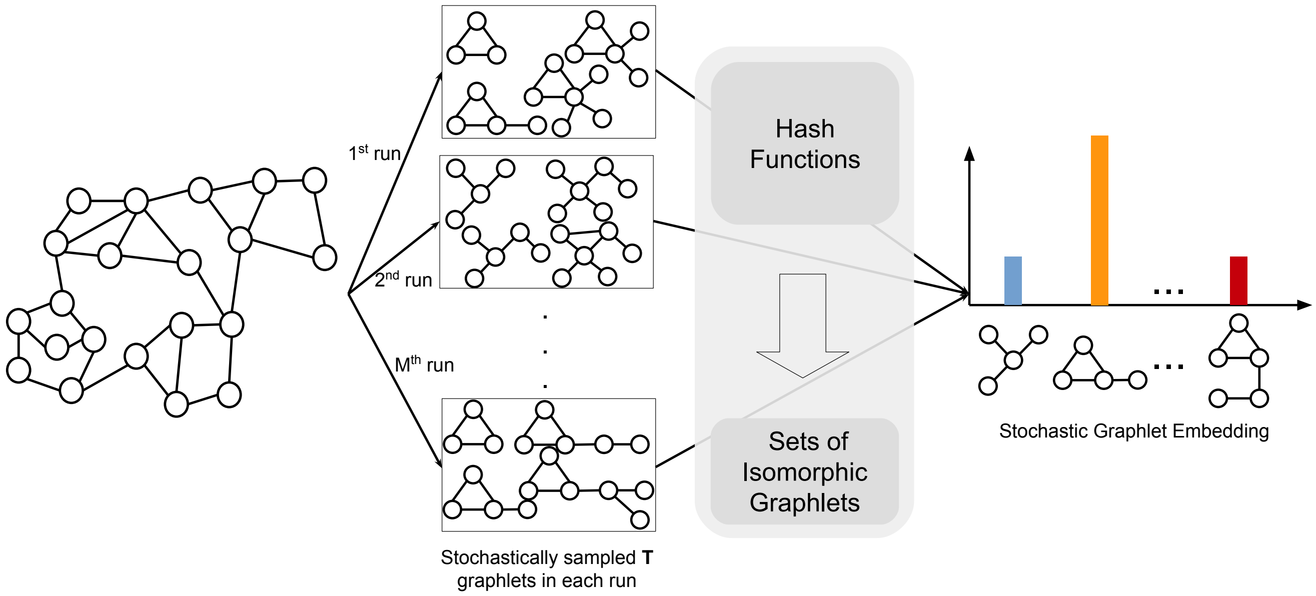

1st run
2nd run
Mth run
Hash Functions
Sets of Isomorphic Graphlets
Stochastically sampled T graphlets in each run
Stochastic Graphlet Embedding

Fig. 2 Overview of stochastic graphlet embedding (SGE). Given a graph G, the stochastic parsing algorithm is able to uniformly sample graphlets of increasing size. Controlled by two parameters M (number of graphlets to be sampled) and T (maximum size of graphlets in terms of number of edges), the method extracts in total $M \times T$ graphlets. These graphlets are encoded and partitioned into isomorphic graphlets using the set of hash functions with a low probability of collision. A distribution of different graphlets is obtained by counting the number of graphlets in each of these partitions. This procedure results in a vector space representation of the graph G referred to as stochastic graphlet embedding.

5.2 Hashed graphlets distribution

For obtaining a distribution of the extracted graphlets from G, it is needed to identify sets of isomorphic graphlets from the sampled ones and then count cardinality of each isomorphic set. A trivial way of doing that certainly involves checking the graph isomorphism for all possible pairs of graphlets for detecting possible partitions that might exist among them. Nevertheless, graph isomorphism is a GI-complete problem [49] for general graphs, so the previously mentioned scheme is extremely costly as the method samples huge number of graphlets with many edges. An alternative, efficient and approximate way of partitioning isomorphic graphlets is *graph hashing*. A graph hash function can be defined as a mapping $h : \mathbb{G} \to \mathbb{R}^m$ that maps a graph into a hash code (a sequence of real numbers) based on the local as well as holistic topological characteristic of graphs. An ideal graph hash function should map two isomorphic graphs to the same hash code as well as two non-isomorphic graphs to two different hash codes. While it is easy to design hash functions satisfying the condition that two isomorphic graphs should have the same hash code, it is extremely difficult to find hash function that ensures different hash codes for every pair of non-isomorphic graphs. An alternative is to design graph hash functions with low *collision probability*, *i.e.*, mapping any two non-isomorphic graphs to the same hash code with a very low probability. For obtaining a distribution of graphlets, the main aim of graph hashing is to assign extracted graphlets from G to corresponding subsets of isomorphic graphlets (a.k.a. partition index or histogram bins) in order to count and quantify their distributions. The proposed mechanism for obtaining the distribution of uniformly sampled graphlets, outlined in Algorithm 3, maintains a global hash table $\mathbf{H}$, whose single entry corresponds to a hash code of a graphlet g produced by the graph hash function. $\mathbf{H}$ grows incrementally as the algorithm confronts new graph hash codes and maintains all the unique hash codes encountered by the system. It is to be noted that the position of each unique hash code is kept fixed, because each position corresponds to a partition index or histogram bin. Now to allocate a given graphlet g to its corresponding histogram bin, its hash code $h(g)$ is mapped to the index of the hash table $\mathbf{H}$, whose corresponding graph hash code gives a hit with $h(g)$ (see line 8). If $h(g)$ does not exist in $\mathbf{H}$ at some instance, it is considered as a new hash code (and hence g as a new graphlet) encountered by the system and appended $h(g)$ at the end of $\mathbf{H}$ (see line 6).

Designing hash functions that yield identical hash codes for two isomorphic graphlets is quite simple, whereas, prototyping those providing two distinct hash codes for two non-isomorphic graphs is very challenging. The chance of mapping two non-isomorphic subgraphs to

Algorithm 3 HASHED-GRAPHLETS-STATISTICS(G): Create a histogram $\mathbf{h}$ of graphlet distribution for a graph G.

```
Require: $G$, $\mathbf{H}$
Ensure: $\mathbf{h}$
$\mathbb{S} \leftarrow$ STOCHASTIC-GRAPHLET-PARSING($G$)
$\mathbf{h}_i \leftarrow 0,\ i = 1, \ldots, |\mathbb{S}|$
for all $g \in \mathbb{S}$ do
    $h(g) \leftarrow$ HASHFUNCTION($g$)
    if $h(g) \notin \mathbf{H}$ then
        $\mathbf{H} \leftarrow \mathbf{H} \cup \{h(g)\}$
    end if
    $i \leftarrow$ GETINDEX-IN-HASHTABLE($h(g)$)
    $\mathbf{h}_i \leftarrow \mathbf{h}_i + 1$
end for
```

the same hash code is termed as *probability of collision.* Indicating H_0 as the set of all pairs of non-isomorphic graphs, the probability of collision can be expressed as the following energy function:

$$E(f) = P((g, g') \in H_0 \quad | \quad h(g) = h(g')) \tag{4}$$

So, in terms of collision probability, the hash functions that produce comparatively lower $E(f)$ values in eqn. (4) are considered to be more reliable for checking the graph isomorphism. It has been studied that sorted *degree of nodes* has 0 collision probability for all graphs with number of edges less or equal to 4 [21]. Moreover, it is also a well known fact that two graphs with the same *betweenness centrality* (sorted) would indeed be isomorphic with high probability [15,53]. For example, sorted *betweenness centrality* has collision probabilities equal to $3.2e^{-4}$, $1.9e^{-4}$, $1.1e^{-4}$ respectively for graphlets with 7, 8 and 9 edges. Interested readers are requested to see [21] for further discussions and analysis on various graph hash functions and corresponding elaboration on probability of collision. Considering the above facts, in this work, we consider sorted *degree of nodes* for graphlets with $t \leq 4$ and the *betweenness centrality* for graphlets with $t \geq 5$.

$$\text{Hash function} = \begin{cases} \text{degree of nodes,} & \text{if } t \leq 4 \\ \text{betweenness centrality,} & \text{otherwise} \end{cases} \tag{5}$$

It should be observed that the distribution of sampled graphlets obtained the way mentioned until now, only considers the topological structure of a graph, and ignores the node and edge attributes. However, it is worth mentioning that the stochastic graphlet embedding permits to consider a small set of nodes and edge attributes by creating respective signatures and then appending it to the hash code encoding the topology of the graphlet. In this work, if needed, we first discretize the existing continuous attributes using a combination of clustering algorithm such as *k-means* and pooling technique. Later, the sorted discrete node and edge labels are used as the attribute signatures and combined with the hash code.

5.3 Hierarchical stochastic graphlet embedding

In this work, we propose to combine the properties of the proposed *Stochastic Graphlet Embedding* with the *Hierarchical Embedding* introduced in the previous Section.

On one hand, SGE provides statistical information about local structures varying the number of edges involved. Therefore, it provides fine-grained insights of the graph which cannot deal with too noisy data. The use of abstractions provided by the graph hierarchy increases the receptive field of each graphlet moving to coarser information that is able to provide insights of the global graph information. Moreover, the use of hierarchical edges during the computation allows to combine information at some levels, *i.e.* combining different levels of detail (see eqn. (1)). For now on, we will denote this embedding as *Hierarchical Stochastic Graphlet Embedding.*

6 Computational complexity

This Section is devoted to study the computational complexity of the proposed approach given a graph $G = (V, E, L_V, L_E)$ where $|V| = n$ and $|E| = m$.

6.1 Hierarchical embedding complexity

Graph clustering algorithms are usually high computational complexity techniques. As it has been stated in Sect. 4.3, the *Girvan-Newman* algorithm has been chosen as a graph clustering technique. The Girvan-Newman algorithm is based on the betweenness centrality of the edges which has a time complexity of $\mathcal{O}(n \cdot m)$ for unweighted graphs and $\mathcal{O}(n \cdot m + n \cdot (n+m)\log(n))$ for weighted graphs. Hence, the *Girvan-Newman* algorithm, which has to remove all the edges, can be computed in $\mathcal{O}(n \cdot m^2)$ for unweighted graphs and $\mathcal{O}(n \cdot m^2 + n \cdot m \cdot (n+m)\log(n))$ for weighted graphs.

Assuming an embedding function φ which has a complexity of $\mathcal{O}(N)$ and assuming that the hierarchical graph construction has a complexity of C_1, then, if we assume L levels, the proposed configurations would become a complexity $\mathcal{O}(C_1 + L \cdot N)$ in the case of the pyramid and $\mathcal{O}(C_1 + L^2 \cdot N)$ for the hierarchy and the exhaustive embeddings.

6.2 Stochastic graphlet embedding complexity

The computational complexity of Algorithm 2 is $\mathcal{O}(M \cdot T)$ where M is the number of graphlets to be sampled and T is the maximum size of graphlets in terms of the number of edges. Assuming a hash function with a complexity of $\mathcal{O}(C_2)$, Algorithm 3 has a time complexity of $\mathcal{O}(M \cdot T \cdot C_2)$ for computing the stochastic graphlet embedding. Here it is worth mentioning that "degree of nodes" and "betweeness centrality", respectively have the time complexity of $\mathcal{O}(n)$ and $\mathcal{O}(n \cdot m)$. From the above explanation, it is clear that the complexity of these two algorithms do not depend on the size of the input graph G, but only on the parameters M, T and the hash functions used.

7 Experimental validation

This section presents the experimental results obtained by our proposed Hierarchical Stochastic Graphlet Embedding method. The main aim of this experimental study is to validate the proposed graph embedding technique for the graph classification task, which demands robust embedding technique for mapping a graph into a vector space. For experimentation, we have considered many different widely used graph datasets with varied characteristics. All these graphs come from real data generated in the fields of Biology, Chemistry, Graphics and Handwriting recognition. The MATLAB code of our experiment is available at `https://github.com/priba/hierarchicalSGE`.

7.1 Experiments on molecular graph datasets

The first set of experiments is conducted on various benchmarks of molecular graphs. Below, we provide a brief description of them followed by the experimental setup, results and discussions.

7.1.1 Dataset description

Several bioinformatics datasets have been used: *MUTAG*, *PTC*, *PROTEINS*, *NCI1*, *NCI109*, *D&D* and *MAO*. These datasets have been widely used as benchmark in the literature. The *MUTAG* dataset contains graph representations of 188 chemical compounds which are either mutagenic aromatic or heteroromatic nitro compounds where nodes can have 7 discrete labels. The *PTC* or Predictive Toxicology Challenge dataset consists of 344 chemical compounds known to cause or not cause cancer in rats and mice. It has 19 discrete node labels. The *PROTEINS* dataset contains relations between secondary structure elements (SSEs) represented by nodes and neighborhood in the amino-acid sequence or in 3D space by edges. It has 3 discrete labels *viz. helix*, *sheet* or *turn*. The *NCI1* and *NCI109* come from the National Cancer Institute (NCI) and are two balanced subsets of chemical compounds screened for their ability to suppress or inhibit the growth of a panel of human tumor cell lines, having 37 and 38 discrete node labels respectively. The *D&D* dataset consists of enzymes and non-enzymes proteins structures, in which their nodes are amino acids. The *MAO* database, taken from GREYC Chemistry graph dataset collection, is composed of 68 graphs representing molecules that either inhibit or not the monoamine oxidase, which is an antidepressant drug. Some more details on the proposed bioinformatics datasets are provided in Table 1.

Table 1 Details of the molecular graph datasets.

Datasets	**# Graphs**	**# Classes**	**Avg. $\lvert V\rvert$**	**Avg. $\lvert E\rvert$**	**Node labels**
MUTAG	188	2 (125 vs. 63)	17.9	39.6	7
PTC	344	2 (192 vs. 152)	25.6	51.9	19
PROTEINS	1113	2 (663 vs. 450)	39.1	145.63	3
NCI1	4110	2 (2057 vs. 2053)	29.9	64.6	37
NCI109	4127	2 (2079 vs. 2048)	29.7	64.3	38
D&D	1178	2 (691 vs 487)	284.3	1431.3	82
MAO	68	2 (30 vs. 38)	18.4	19.6	3

7.1.2 Experimental setup

We have performed two different experiments: the first one does not use the attribute information encoded in the nodes and edges of the graphs, whereas the second experiment does use the available node and edge features. For evaluating the performance of the proposed embedding technique, we have used a C-SVM solver [14] as a classifier. Since the datasets considered in this set of experiments do not contain predefined train and test sets, we have used a 10-fold cross validation scheme to obtain accuracies and have reported the mean accuracies respectively in Table 2 and Table 3 for unlabeled and labeled datasets. We follow a classical graph classification pipeline where, in the first stage, graph embedding is computed by our proposed scheme, whereas in the second step, embedded graphs are classified using a previously trained classifier.

7.1.3 Results and discussion

In Table 2, we present the experimental results obtained by our proposed hierarchical embedding techniques together with other existing works on the unlabeled datasets. The previously mentioned three configurations of our hierarchical embedding are respectively denoted as: pyramidal, hierarchical and exhaustive. For unlabeled datasets, we have considered 10 different state-of-the-art methods: (1) random walk kernel (RW) [27], (2) shortest path kernel (SP) [8], (3) graphlet kernel (GK) [70], (4) Weisfeiler-Lehman kernel (WL) [69], (5) deep graph kernel (DGK) [83], (6) multiscale Laplacian graph kernel (MLK) [38], (7) diffusion CNNs (DCNN) [4], (8) strong graph spectrums (SGS) [37], (9) family of graph spectral distances (F_GSD) [78], and (10) stochastic graphlet embedding (SGE) [21].

From the quantitative results shown in Table 2, it should be observed that for most datasets, the highest accuracy is achieved by one of the hierarchical configurations proposed by us, which sets a new state-of-the-art results on all the datasets considered. Particularly, the best accuracies are obtained either by the pyramidal or the exhaustive configurations, which indicates the importance of considering hierarchical information for the graph embedding problem. As expected, the proposed hierarchical embeddings have achieved better performance than the SGE which is regarded as the baseline of our proposal. It should be observed that with this experimental setting, particularly the hierarchical configuration has performed quite poorly compared to the other two configurations. This fact might suggest that only hierarchical edges together with the connecting levels do not contain sufficient information for a robust graph representation. Information captured in the multiscale graphs thought to play a vital role for graph embedding, which is proved by the excellent performance obtained with the pyramidal and exhaustive configurations.

In Table 3, we demonstrate the results acquired by three different configurations of our proposed hierarchical embedding on the labeled graph datasets. For comparing with other state-of-the-art methods, we have considered two additional techniques: (1) PATCHY-SAN (PSCN) [55] and (2) graphlet spectrum (GS) [39]. Some of the previously considered state-of-the-art techniques do not work with labeled graphs, so they have not been evaluated in this experimentation.

The results presented in Table 3 show that, except on the MUTAG dataset, our proposed hierarchical embedding techniques have achieved the best performances on all the other datasets. This demonstrates the usefulness of considering the hierarchical information for embedding graphs to a vector space. Contrary to the previous experiments on unlabeled datasets, in this case, the hierarchical configuration has performed reasonably better. This fact shows that on labeled graphs, the hierarchical edges together with the connecting levels might provide important structural information. Also, it is important to note that the level information also performed consistently on all the datasets.

7.2 Experiments on AIDS, GREC, COIL-DEL and HistoGraph datasets

While the datasets considered in the previous set of experiments were mostly molecular in nature, the set of experiments to be discussed in this section consider graphs from various fields, such as, Biology, Computer Vision, Graphics Recognition and Handwriting Recognition. Underneath, we give a brief description of the datasets considered followed by the experimental setup, results and discussions.

7.2.1 Dataset description

In this experiment, we consider four different datasets; three of them *viz. AIDS*, *GREC* and *COIL-DEL* are taken from the IAM graph database repository[1] [60]. The first one, *viz.*, the AIDS database consists of 2000 graphs representing molecular compounds which are constructed from the AIDS Antiviral Screen Database of Active Compounds[2]. This dataset consists of two classes, *viz.*, active (400 elements) and inactive (1600 elements), which respectively represent molecules with possible activity against HIV. The GREC dataset consists of 1100 graphs representing 22 different classes (characterizing architectural and electronic symbols) with 50 instances per class; these instances have different noise levels. The COIL-DEL database includes 3900 graphs belonging to 100 different classes with 39 instances per class; each instance has a different rotation angle. The HistoGraph dataset[3][74] consists of graphs representing words from the communicating letters written by the first US president, George Washington. It consists of 293 graphs generated from 30 distinct words. Therefore, given a word, the task of the classifier is to predict its class which should be among the 30 words. Nodes are only labeled with their position in the image. Furthermore, this dataset used 6 different graph representation paradigms for delineating a single word into a graph, which results in 6 different subsets of graphs. The entire dataset is divided into 90, 60 and 143 graphs respectively for train, validation and test purposes.

[1] Available at `http://www.fki.inf.unibe.ch/databases/iam-graph-database`

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

[3] Available at `http://www.histograph.ch`

Table 2 Classification accuracies on *unlabeled* molecular graph datasets. In the table, RW corresponds to the random walk kernel [27], SP stands for the shortest path kernel [8], GK denotes the graphlet kernel [70], WL indicates the Weisfeiler-Lehman kernel [69], DGK corresponds to the deep graph kernel [83], MLK stands for the multiscale Laplacian graph kernel [38], DCNN indicates the diffusion CNNs [4], SGS denotes the strong graph spectrums [37], F_GSD stands for the family of graph spectral distances [78], SGE corresponds to the stochastic graphlet embedding [21], and HSGE indicates the hierarchical graph embeddings proposed by us. The best results obtained on a dataset is indicated by bold face.

Methods	MUTAG	PTC	PROTEINS	NCI1	NCI109	D&D	MAO
RW [27]	83.50	55.52	68.46	44.84	59.80	–	83.52
SP [8]	87.23	58.72	72.14	68.15	68.30	–	90.35
GK [70]	84.04	60.17	71.78	62.07	62.04	75.05	80.88
WL [69]	87.28	55.61	70.06	77.23	78.43	73.76	89.79
DGK [83]	86.17	59.88	71.69	64.40	67.14	72.75	87.76
MLK [38]	87.23	62.20	71.35	77.57	75.91	77.02	91.17
DCNN [4]	66.51	55.79	65.22	63.10	60.67	OMR	76.10
SGS [37]	88.61	–	–	62.72	62.62	–	–
F_GSD [78]	92.12	62.80	73.42	79.80	78.84	77.10	95.58
SGE [21]	91.11	63.53	71.89	83.23	82.92	74.92	95.71
HSGE (pyr.)	91.11	65.29	75.32	**85.24**	**83.24**	78.73	**100.00**
HSGE (gen. pyr.)	92.22	67.94	75.50	83.36	81.73	**79.32**	**100.00**
HSGE (hier.)	**93.33**	67.06	**76.31**	82.85	81.33	72.03	**100.00**
HSGE (exhaus.)	92.22	**70.88**	76.58	83.84	82.12	73.90	**100.00**

Table 3 Classification accuracy on *labeled* molecular graph datasets. In the table, MLK stands for the Multiscale Laplacian Graph kernel [38], DCNN indicates the Diffusion CNNs [4], PSCN corresponds to the PATCHY-SAN [55], GS denotes the Graphlet Spectrum (GS) [39], SGE corresponds to the Stochastic Graphlet Embedding (SGE) [21], and HSGE indicates the hierarchical graph embeddings proposed by us. The best results obtained on a dataset is specified by bold face.

Methods	MUTAG	PTC	PROTEINS	NCI1	NCI109	D&D	MAO
MLK [38]	87.94	63.26	–	81.75	–	78.18	88.29
DCNN [4]	66.98	56.60	–	62.61	–	OMR	75.14
PSCN [55]	92.63	62.90	–	78.59	–	77.12	–
GS [39]	88.11	–	–	65.00	–	–	–
SGE [21]	88.33	57.94	74.05	83.44	81.89	77.37	94.29
HSGE (pyr.)	91.11	62.06	75.68	**84.79**	82.03	77.46	94.29
HSGE (gen. pyr.)	**92.78**	65.59	**76.58**	81.31	80.24	79.66	**97.14**
HSGE (hier.)	91.11	**67.35**	75.77	82.50	82.88	79.32	94.29
HSGE (exhaust.)	91.67	66.18	76.04	84.42	**84.42**	**80.25**	**97.14**

Table 4 Details of the AIDS, GREC, COIL-DEL and HistoGraph datasets.

Datasets	Subsets	# Graphs	# Classes	Avg. $\lvert V \rvert$	Avg. $\lvert E \rvert$	Node labels
AIDS	–	2000 (250, 250, 1500)	2	15.7	16.2	Chemical symbol
GREC	–	1100 (286, 286, 528)	22 (50 each)	11.5	11.9	Type, (x,y) position
COIL-DEL	–	3900 (2400, 500, 1000)	100	21.5	54.2	(x,y) position
HistoGraph	**Keypoint**	293 (90, 60, 143)	30	101.8	94.8	(x,y) position
	Grid-NNA			56.4	81.4	
	Grid-MST			66.1	64.4	
	Grid-DEL			74.1	205.1	
	Projection			63.1	58.8	
	Split			73.3	69.8	

7.2.2 Experimental setup

In this case as well, we have employed a C-SVM solver [14] as a classifier. Since the datasets used in this set of experiments contain well defined train and test sets, we have reported the obtained accuracies on the test set of the respective datasets in Table 5.

7.2.3 Results and discussion

Similar to the experimental results obtained in the previous section, in this set of experiments as well, our proposed hierarchical embeddings have achieved the best results on most datasets. In this set of experiments, the leading scores are mostly obtained by the exhaustive configuration, which shows the effectiveness of combining multiscale structural information together with the hierarchical connections. For some datasets, our hierarchical embedding does not achieve the best results, but it has performed very competitively. This also proves the robustness of the hierarchical graph representation.

7.3 Discussion on the parameters involved in the algorithm

Our algorithm is mainly controlled by three different parameters: (1) the *number of levels* L of the graph pyramid, (2) the *reduction ratio* R and (3) the maximum *number of edges* T of a graphlet. For illustrating how these three parameters control the performance of the system, first we plot the classification accuracy by varying the levels of the graph pyramid (see Fig. 3), reduction ratio (see Fig. 4) and T (see Fig. 5). Here it is worth mentioning that for the sake of simplicity, for each level we just consider the maximum accuracy obtained by any configuration mentioned in Sect. 4.3. From Fig. 3, we can observe that for all the datasets, considering a second level together with the base graph increases the classification accuracy. However, the successive inclusion of hierarchical levels does not always increase the performance. It has been observed that for smaller graphs (with less number nodes and edges; *e.g.*, the graphs from MUTAG), the further inclusion of hierarchical abstraction decreases the performance of the system; this means that for smaller graphs a higher level abstraction can introduce noise or distortion. The reduction ratio R directly decides the number of clusters in a given level, and hence the number of nodes in the next higher level of the hierarchy. For example, $R = 1$ indicates that the number of clusters should remain the same with the number of nodes, while $R = 2$ indicates that the number of clusters should be half the number of nodes in that level. Fig. 4 shows the behaviour of our method with different values of R while we have fixed $L = 2$. From these plots, one must observe that R is completely dependant on the datasets irrespective of the size of graphs they contain. For PTC, PROTEINS, and MAO datasets, the performance mostly increases with the increase of R, while for MUTAG, it improves until $R = 2$, and then it decreases for all hierarchical configurations. For MAO dataset, all the hierarchical configurations behave exactly in the same way with the increase of R, which might be because the smaller sized graphs on which the contribution of different hierarchical configuration is indistinguishable.

In Fig. 5, we show the performance trend on six datasets (i.e. MUTAG, PTC, PROTEINS, NCI1, and NCI109) only with the SGE algorithm, which is the baseline graph embedding technique that we considered. The hierarchical configurations are not considered in this case because they have different graphlet size in different hierarchical levels, so understanding their behaviour would have been complicated. From Fig. 5, it is clear that increasing T mostly improves the performance of the system on all the datasets. Albeit, there are some exceptions (*e.g.*, for PTC dataset, $T = 6$), which suggests that graphlets with T edges are less informative for that particular graph dataset.

7.4 Discussion on the stochasticity of the algorithm

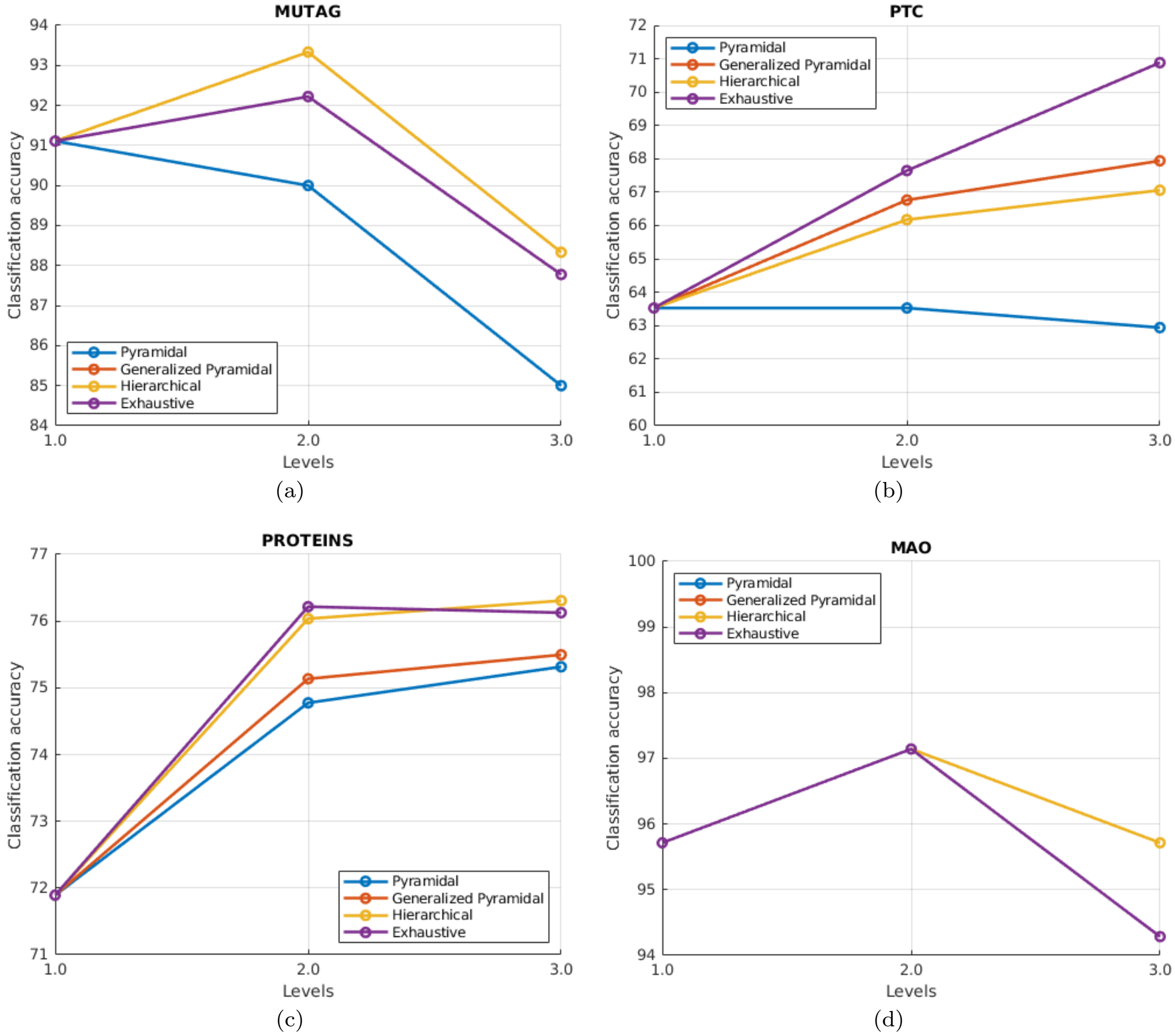
It is important to note that our proposed algorithm is stochastic in nature because of the involvement of the stochastic graphlet sampling and the subsequent graph embedding procedure. The graphlet sampling engaged here uniformly samples graphlets from a given population of graphs, and by the law of large numbers, this sampling guarantees that the empirical distribution of graphlets is asymptotically close to the actual distribution [58]. For demonstrating the fact that the stochastic behaviour of our algorithm does not heavily impact on the experimental results, we repeated the last experiment on all the datasets considered for 10 iterations, and in each iteration, we randomly seeded the sampling algorithm. The mean and standard deviation of the classification accuracy obtained for each dataset is reported in Table 6. The mean accuracies reported in the table are quite close to the ones reported in Table 5, and the standard deviations are comparatively low (all of them are less than 1.0). This suggests that the proposed graph embedding technique, although employed a stochastic process, is consistent in terms of performance.

8 Conclusions

In this paper we have proposed to enhance the information encoded in graph embeddings by means of hierarchical representations. We have experimentally validated that the abstract information is able to improve the graph classification performance. The embedding function is based on a stochastic sampling of graphlets to obtain the graphlet distribution within the graph. Graphlets of different sizes are considered to allow a change on the node context. Moreover, the hashing functions are used to identify graphlets in an efficient way. Event though considering different size graphlets provides robustness in terms of graph distortions, they still provide local information when we consider larger graphs. Therefore, building a graph hierarchy allows to increase the graphlet context without increasing the time needed for identifying the graphlet. In this work,

Table 5 Results obtained on the AIDS, GREC, COIL-DEL and HistoGraph datasets. In the table, RW corresponds to the Random Walk kernel [27], DE stands for the dissimilarity embedding [12], NAS indicates the node attribute statistics [29], GED denotes to the approximated graph edit distance [61], SGE corresponds to the Stochastic Graphlet Embedding (SGE) [21], and HSGE indicates our hierarchical graph embeddings. The best results obtained on a dataset is indicated by bold face.

Methods	**AIDS**	**GREC**	**COIL-DEL**	**HistoGraph**					
				Keypoint	**Grid-NNA**	**Grid-MST**	**Grid-DEL**	**Projection**	**Split**
RW [27]	98.50	96.20	94.20	–	–	–	–	–	–
DE [12]	98.10	95.10	96.80	–	–	–	–	–	–
NAS [29]	98.30	99.20	98.10	–	–	–	–	–	–
GED [61]	–	–	–	77.62	65.03	74.13	62.94	**81.82**	80.42
SGE [21]	98.67	**99.62**	98.14	79.02	**72.73**	77.62	74.83	79.72	81.12
HSGE (pyr.)	98.87	99.43	98.79	79.02	**72.73**	77.62	74.83	79.72	81.12
HSGE (gen. pyr.)	98.35	99.43	98.37	77.62	72.03	77.62	74.13	79.72	81.45
HSGE (hier.)	98.33	99.05	**98.99**	79.02	70.63	76.22	**75.52**	80.42	80.42
HSGE (exhaust.)	**99.00**	99.43	98.86	**79.72**	72.03	**78.32**	74.83	**81.82**	**81.82**



(a)

(b)

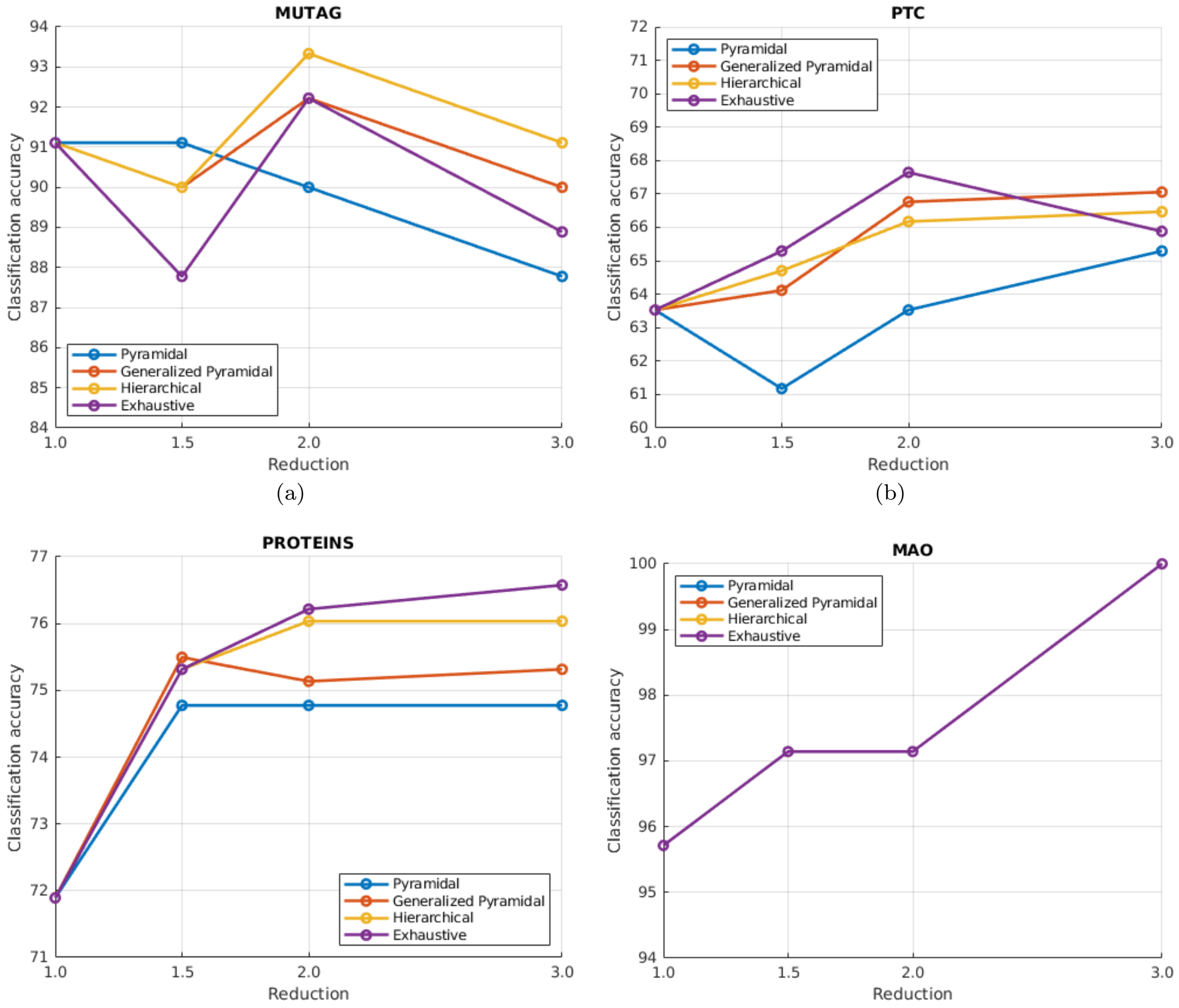
(c)

(d)

Fig. 3 Plots showing classification accuracies by varying the levels of pyramidal graph construction on different datasets.

we have carefully validated the performance of our approach in different application scenarios, showing that we outperform the state-of-the-art approaches in the graph classification task using an SVM as a classifier.

Further research will focus on improving the hierarchical graph construction. Even though the Girvan-Newman algorithm is able to exploit the desired properties of the graph, creating clusterings that allow to



(a) (b)

(c) (d)

Fig. 4 Plots showing classification accuracies by varying the reduction ratio of pyramidal graph construction on different datasets.

Table 6 Mean and standard deviation of the accuracies obtained by repeating the classification task on the AIDS, GREC, COIL-DEL and HistoGraph datasets for 10 iterations. Here the mean accuracies consistent with the ones in Table 5 and the low standard deviations show that the proposed graph embedding is not sensitive to the stochasticity involved in the algorithm. The best results obtained on a dataset is specified by bold face.

Methods	AIDS	GREC	COIL-DEL	HistoGraph					
				Keypoint	Grid-NNA	Grid-MST	Grid-DEL	Projection	Split
HSGE (pyr.)	**98.74** (±0.13)	99.36 (±0.19)	98.74 (±0.21)	78.98 (±0.32)	**72.71** (±0.10)	77.57 (±0.43)	74.79 (±0.62)	79.72 (±0.99)	81.04 (±0.84)
HSGE (gen. pyr.)	98.12 (±0.27)	99.58 (±0.23)	98.49 (±0.49)	**79.31** (±0.52)	71.28 (±0.58)	78.05 (±0.47)	74.96 (±0.71)	79.94 (±0.18)	80.24 (±0.74)
HSGE (hier.)	98.24 (±0.36)	99.04 (±0.16)	**98.98** (±0.60)	79.03 (±0.20)	70.51 (±0.55)	76.20 (±0.40)	**75.47** (±0.86)	80.39 (±0.17)	80.38 (±0.21)
HSGE (exhaust.)	**98.74** (±0.21)	**99.64** (±0.80)	98.84 (±0.17)	79.01 (±0.70)	71.96 (±0.10)	**78.28** (±0.97)	74.79 (±0.01)	**80.82** (±0.46)	**81.53** (±0.94)

create good abstractions, their time complexity is a drawback that should be studied when considering large graphs.

Acknowledgments

This work has been partially supported by the European Union's research and innovation program under the Marie Skłodowska-Curie grant agreement No. 665919 (H2020-MSCA-COFUND-2014:665919), the Spanish projects RTI2018-102285-A-I00 and RTI2018-095645-B-C21, the FPU fellowship FPU15 / 06264 from the Spanish Ministerio de Educación, Cultura y Deporte, the Ramon y Cajal Fellowship RYC-2014-16831, and the CERCA Program / Generalitat de Catalunya. Anjan Dutta was a Marie-Curie Fellow (under the P-SPHERE project) at the Computer Vision Center of Barcelona, where most of the work was done and the paper was written.

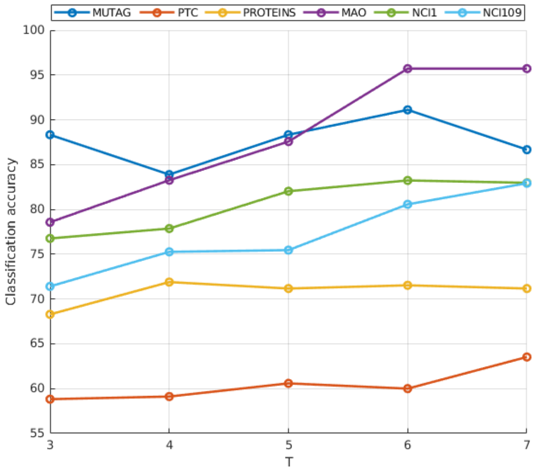


Fig. 5 Plot showing the classification accuracy obtained by SGE by varying the maximum number of edges from 3 to 7 on different datasets: MUTAG, PTC, PROTEINS, MAO, NCI1, NCI109.

References

1. Adelson, E.H., Anderson, C.H., Bergen, J.R., Burt, P.J., Ogden, J.M.: Pyramid methods in image processing. RCA Engineer **29**(6), 33–41 (1984)
2. Ahuja, N., Todorovic, S.: From region based image representation to object discovery and recognition. In: S+SSPR, vol. 6218, pp. 1–19 (2010)
3. Almeida, H., Guedes, D., Meira, W., Zaki, M.J.: Is there a best quality metric for graph clusters? In: MLKDD, pp. 44–59 (2011)
4. Atwood, J., Towsley, D.: Diffusion-convolutional neural networks. In: NIPS, pp. 1993–2001 (2016)
5. Aziz, F., Wilson, R., Hancock, E.: Backtrackless walks on a graph. IEEE TNNLS **24**(6), 977–989 (2013)
6. Barbu, E., Héroux, P., Adam, S., Trupin, E.: Frequent graph discovery: Application to line drawing document images. ELCVIA **5**(2), 47–54 (2005)
7. Bodic, P.L., Héroux, P., Adam, S., Lecourtier, Y.: An integer linear program for substitution-tolerant subgraph isomorphism and its use for symbol spotting in technical drawings. PR **45**(12), 4214–4224 (2012)
8. Borgwardt, K., Kriegel, H.P.: Shortest-path kernels on graphs. In: ICDM, pp. 74–81 (2005)
9. Borzeshi, E.Z., Piccardi, M., Riesen, K., Bunke, H.: Discriminative prototype selection methods for graph embedding. PR **46**(6), 1648–1657 (2013)
10. Broelemann, K., Dutta, A., Jiang, X., Lladós, J.: Hierarchical graph representation for symbol spotting in graphical document images. In: S+SSPR, vol. 7626, pp. 529–538. Springer Berlin Heidelberg (2012)
11. Broelemann, K., Dutta, A., Jiang, X., Lladós, J.: Hierarchical plausibility-graphs for symbol spotting in graphical documents. In: GREC, pp. 13–18 (2013)
12. Bunke, H., Riesen, K.: Improving vector space embedding of graphs through feature selection algorithms. PR **44**(9), 1928–1940 (2010)
13. Caelli, T., Kosinov, S.: An eigenspace projection clustering method for inexact graph matching. IEEE TPAMI **26**(4), 515–519 (2004)
14. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST) **2**(3), 27 (2011)
15. Comellas, F., Paz-Sánchez, J.: Reconstruction of networks from their betweenness centrality. In: AEC, pp. 31–37. Springer Berlin Heidelberg (2008)
16. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. IJPRAI **18**(3), 265–298 (2004)
17. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: NIPS, pp. 1–14 (2016)
18. Dupé, F.X., Brun, L.: Hierarchical bag of paths for kernel based shape classification. In: S+SSPR, pp. 227–236 (2010)
19. Dutta, A., Lladós, J., Bunke, H., Pal, U.: Product graph-based higher order contextual similarities for inexact subgraph matching. PR **76**, 596–611 (2017)
20. Dutta, A., Riba, P., Lladós, J., Fornés, A.: Pyramidal stochastic graphlet embedding for document pattern classification. In: ICDAR, pp. 33–38 (2017)
21. Dutta, A., Sahbi, H.: Stochastic graphlet embedding. IEEE TNNLS **30**(8), 2369–2382 (2019)
22. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. IEEE TPAMI **35**(8), 1915–1929 (2013)
23. Fei-Fei, L., Perona, P.: A bayesian hierarchical model for learning natural scene categories. In: CVPR, pp. 524–531 (2005)
24. Felzenszwalb, P., Schwartz, J.: Hierarchical matching of deformable shapes. In: CVPR, pp. 1–8 (2007)
25. Foggia, P., Percannella, G., Vento, M.: Graph matching and learning in pattern recognition in the last 10 years. IJPRAI **28**(1), 1–40 (2014)
26. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry pp. 35–41 (1977)
27. Gärtner, T.: A survey of kernels for structured data. ACM SIGKDD **5**(1), 49–58 (2003)
28. Gentile, C., Li, S., Kar, P., Karatzoglou, A., Zappella, G., Etrue, E.: On context-dependent clustering of bandits. In: ICML, pp. 1253–1262. JMLR. org (2017)
29. Gibert, J., Valveny, E., Bunke, H.: Graph embedding in vector spaces by node attribute statistics. PR **45**(9), 3072–3083 (2012)
30. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: ICML, pp. 1263–1272 (2017)
31. Girvan, M., Newman, M.: Community structure in social and biological networks. NAS **99**(12), 7821–7826 (2002)
32. Horváth, T., Gärtner, T., Wrobel, S.: Cyclic pattern kernels for predictive graph mining. In: KDD, pp. 158–167 (2004)
33. Jolion, J.M., Rosenfeld, A.: A Pyramid Framework for Early Vision: Multiresolutional Computer Vision. Kluwer Academic Publishers, Norwell, MA, USA (1994)
34. Jouili, S., Tabbone, S.: Graph embedding using constant shift embedding. In: ICPR, pp. 83–92 (2010)
35. Kashima, H., Tsuda, K., Inokuchi, A.: Kernels for graphs. KMCB **39**(1), 101–113 (2004)
36. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR, pp. 1–10 (2017)
37. Kondor, R., Borgwardt, K.M.: The skew spectrum of graphs. In: ICML, pp. 496–503 (2008)
38. Kondor, R., Pan, H.: The multiscale laplacian graph kernel. In: NIPS, pp. 2982–2990 (2016)
39. Kondor, R., Shervashidze, N., Borgwardt, K.M.: The graphlet spectrum. In: ICML, pp. 529–536 (2009)
40. Korda, N., Szörényi, B., Li, S.: Distributed clustering of linear bandits in peer to peer networks. In: ICML (2016)

41. Kriege, N., Mutzel, P.: Subgraph matching kernels for attributed graphs. In: ICML, pp. 1015–1022 (2012)
42. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: IEEE (2001)
43. Lafferty, J., Lebanon, G.: Diffusion kernels on statistical manifolds. JMLR **6**, 129–163 (2005)
44. Li, S., Chen, W., Li, S., Leung, K.: Improved algorithm on online clustering of bandits. In: IJCAI (2019)
45. Li, S., Karatzoglou, A., Gentile, C.: Collaborative filtering bandits. In: SIGIR (2016)
46. Liu, X., Lin, L., Li, H., Jin, H., Tao, W.: Layered shape matching and registration: Stochastic sampling with hierarchical graph representation. In: ICPR, pp. 1–4 (2008)
47. Luqman, M.M., Ramel, J.Y., Lladós, J., Brouard, T.: Fuzzy multilevel graph embedding. PR **46**(2), 551–565 (2013)
48. Marfil, R., Molina-Tanco, L., Bandera, A., Sandoval, F.: The construction of bounded irregular pyramids with a union-find decimation process. In: GbRPR, pp. 307–318 (2007)
49. Mehlhorn, K.: Graph algorithms and NP-completeness. Springer-Verlag New York, Inc. (1984)
50. Mousavi, S.F., Safayani, M., Mirzaei, A., Bahonar, H.: Hierarchical graph embedding in vector space by graph pyramid. PR **61**, 245–254 (2017)
51. Neuhaus, M., Bunke, H.: An error-tolerant approximate matching algorithm for attributed planar graphs and its application to fingerprint classification. In: S+SSPR, pp. 180–189 (2004)
52. Neuhaus, M., Bunke, H.: Bridging the Gap Between Graph Edit Distance and Kernel Machines. World Scientific (2007)
53. Newman, M.J.: A measure of betweenness centrality based on random walks. SN **27**(1), 39–54 (2005)
54. Niebles, J., Fei-Fei, L.: A hierarchical model of shape and appearance for human action classification. In: CVPR, pp. 1–8 (2007)
55. Niepert, M., Ahmed, M., Kutzkov, K.: Learning convolutional neural networks for graphs. In: ICML, pp. 2014–2023 (2016)
56. Pekalska, E., Duin, R.P.W.: The Dissimilarity Representation for Pattern Recognition: Foundations And Applications. World Scientific, USA (2005)
57. Pelillo, M., Siddiqi, K., Zucker, S.W.: Matching hierarchical structures using association graphs. IEEE TPAMI **21**(11), 1105–1120 (1999)
58. Prulj, N.: Biological network comparison using graphlet degree distribution. Bioinformatics **23**(2), e177 (2007)
59. Riba, P., Lladós, J., Fornés, A.: Error-tolerant coarse-to-fine matching model for hierarchical graphs. In: International Workshop on Graph-Based Representations in Pattern Recognition, pp. 107–117. Springer (2017)
60. Riesen, K., Bunke, H.: Iam graph database repository for graph based pattern recognition and machine learning. In: S+SSPR, pp. 287–297 (2008)
61. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. IVC **27**(7), 950–959 (2009)
62. Riesen, K., Bunke, H.: Graph classification by means of lipschitz embedding. IEEE TSMCB **39**(6), 1472–1483 (2009)
63. Riesen, K., Neuhaus, M., Bunke, H.: Bipartite graph matching for computing the edit distance of graphs. In: F. Escolano, M. Vento (eds.) GbRPR, *LNCS*, vol. 4538, pp. 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
64. Robles-Kelly, A., Hancock, E.R.: A riemannian approach to graph embedding. PR **40**(3), 1042–1056 (2007)
65. Saund, E.: A graph lattice approach to maintaining and learning dense collections of subgraphs as image features. IEEE TPAMI **35**(10), 2323–2339 (2013)
66. Schellewald, C., Schnörr, C.: Probabilistic subgraph matching based on convex relaxation. In: EMMCVPR, pp. 171–186 (2005)
67. Serratosa, F., Alquézar, R., Sanfeliu, A.: Efficient algorithms for matching attributed graphs and function-described graphs. In: ICPR, vol. 2, pp. 867–872 (2000)
68. Shervashidze, N., Borgwardt, K.M.: Fast subtree kernels on graphs. In: NIPS, pp. 1660–1668 (2009)
69. Shervashidze, N., Schweitzer, P., van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. JMLR **12**, 2539–2561 (2011)
70. Shervashidze, N., Vishwanathan, S.V.N., Petri, T., Mehlhorn, K., Borgwardt, K.: Efficient graphlet kernels for large graph comparison. In: AISTATS, pp. 488–495 (2009)
71. Shokoufandeh, A., Macrini, D., Dickinson, S., Siddiqi, K., Zucker, S.: Indexing hierarchical structures using graph spectra. IEEE TPAMI **27**(7), 1125–1140 (2005)
72. Smola, A.J., Kondor, R.: Kernels and regularization on graphs. In: COLT, pp. 144–158 (2003)
73. Solnon, C.: Alldifferent-based filtering for subgraph isomorphism. AI **174**(12-13), 850–864 (2010)
74. Stauffer, M., Fischer, A., Riesen, K.: A novel graph database for handwritten word images. In: S+SSPR, pp. 553–563 (2016)
75. Suh, Y., Adamczewski, K., Mu Lee, K.: Subgraph matching using compactness prior for robust feature correspondence. In: CVPR (2015)
76. Ulrich, M., Wiedemann, C., Steger, C.: Combining scale-space and similarity-based aspect graphs for fast 3d object recognition. IEEE TPAMI **34**(10), 1902–1914 (2012)
77. Vento, M.: A long trip in the charming world of graphs for pattern recognition. PR **48**(2), 291–301 (2015)
78. Verma, S., Zhang, Z.L.: Hunt for the unique, stable, sparse and fast feature learning on graphs. In: NIPS, pp. 87–97 (2017)
79. Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R., Borgwardt, K.M.: Graph kernels. JMLR **11**, 1201–1242 (2010)
80. Watkins, C.: Kernels from matching operations. Tech. rep., University of London, Computer Science Department (1999)
81. Weissman, T., Ordentlich, E., Seroussi, G., Verdu, S., Weinberger, M.J.: Inequalities for the l1 deviation of the empirical distribution. Tech. rep., HP Labs, Palo Alto (2003)
82. Wilson, R., Hancock, E., Luo, B.: Pattern vectors from algebraic graph theory. IEEE TPAMI **27**(7), 1112–1124 (2005)
83. Yanardag, P., Vishwanathan, S.: Deep graph kernels. In: KDD, pp. 1365–1374 (2015)