Semi-supervised Clustering of Graph Objects: A Subgraph Mining Approach

Xin Huang[1], Hong Cheng[1], Jiong Yang[2], Jeffery Xu Yu[1], Hongliang Fei[3], and Jun Huan[3]

[1] The Chinese University of Hong Kong
[2] Case Western Reserve University
[3] University of Kansas
{xhuang,hcheng,yu}@se.cuhk.edu.hk, jxy55@case.edu, {hfei,jhuan}@ittc.ku.edu

Abstract. Semi-supervised clustering has recently received a lot of attention in the literature, which aims to improve the clustering performance with limited supervision. Most existing semi-supervised clustering studies assume that the data is represented in a vector space, e.g., text and relational data. When the data objects have complex structures, e.g., proteins and chemical compounds, those semi-supervised clustering methods are not directly applicable to clustering such graph objects.

In this paper, we study the problem of semi-supervised clustering of data objects which are represented as graphs. The supervision information is in the form of pairwise constraints of must-links and cannot-links. As there is no predefined feature set for the graph objects, we propose to use discriminative subgraph patterns as the features. We design an objective function which incorporates the constraints to guide the subgraph feature mining and selection process. We derive an upper bound of the objective function based on which, a branch-and-bound algorithm is proposed to speedup subgraph mining. We also introduce a redundancy measure into the feature selection process in order to reduce the redundancy in the feature set. When the graph objects are represented in the vector space of the discriminative subgraph features, we use semi-supervised kernel K-means to cluster all graph objects. Experimental results on real-world protein datasets demonstrate that the constraint information can effectively guide the feature selection and clustering process and achieve satisfactory clustering performance.

Keywords: Semi-supervised clustering, frequent subgraph mining.

1 Introduction

Complex structures in many scientific applications can be represented as graphs, e.g., protein structures, chemical compounds, program flows and XML documents. In many applications, it would be very useful if we can automatically partition a set of data objects which are represented as graphs into disjoint clusters. For example, in bioinformatics, graph clustering can distinguish different families of proteins based on their structural similarity. In practice, we may also have some prior information about the graph data objects, e.g., some proteins are similar (or dissimilar) based on the similarities of their amino acid sequences and three-dimensional structure, or some proteins share a common evolutionary origin. If we can effectively incorporate the prior information into clustering, the clustering performance could be significantly boosted.

Semi-supervised clustering has recently received a lot of attention in the literature. Traditional clustering approaches fall into the category of unsupervised learning, as only unlabeled data is used for clustering. When a small amount of supervision information is available, it can be incorporated into the clustering process to improve the clustering performance. There has been research focusing on constraint-based [1] or distance-based [2], [3], [4], [5] semi-supervised clustering. However, most existing semi-supervised clustering methods assume that the input data is in a feature vector space, e.g., text and relational data. When the data objects have complex structures but no predefined feature space, such as proteins and chemical compounds, these methods are not directly applicable to cluster the data objects.

In this paper, we study the problem of clustering graph objects with a limited amount of supervision information. Supervision in the form of pairwise constraints is usually more realistic than requiring class labels in many applications. Thus we consider supervision information including *must-links* and *cannot-links*, indicating respectively whether two graph objects should belong to the same cluster or not. Such pairwise constraints occur naturally in many domains.

The first challenge in clustering graph objects is the lack of feature vector representation of the graph objects. As an effective solution adopted in recent graph classification methods [6], [7], [8], [9], we use subgraphs as features to represent a graph object in a binary vector. But different from graph classification as a supervised learning problem, we do not have class labels in our clustering problem to supervise the feature selection process. In order to evaluate the usefulness of the subgraph features, we propose a semi-supervised feature mining and selection algorithm – an objective function for subgraph feature selection is designed which incorporates the pairwise constraints, with the aim to satisfy as many constraints as possible. In order to avoid exhaustive enumeration of all subgraph features, we integrate the objective function into the subgraph mining process and push it deep for pruning the search space. Given any subgraph g, an upper bound of the objective function for g's supergraphs can be derived, based on which, we develop a branch-and-bound algorithm to efficiently search for optimal subgraph features by pruning the subgraph search space. In addition, considering the high redundancy between subgraph patterns, we design a redundancy control mechanism in order to generate a redundancy-aware feature set. Based upon the subgraphs features, all graph objects can be represented in a feature space. Then we perform the semi-supervised kernel K-means algorithm [10] to cluster the graph objects. Experimental results on real protein graphs demonstrate that our semi-supervised feature selection and graph object clustering algorithms can accurately generate clusters which are very close to the underlying family labels of the protein data. The branch-and-bound algorithm expedites the subgraph mining process and prunes a lot of low-quality subgraph features by considering both the constraint and unconstraint graph objects.

The rest of the paper is organized as follows. Section 2 discusses related work on semi-supervised clustering, graph clustering and graph mining methods. We define the semi-supervised graph clustering problem in Section 3. In Section 4 we formulate the subgraph feature mining problem as an optimization problem and develop a branch-and-bound algorithm for the feature mining. We discuss two clustering algorithms we have implemented in Section 5. Experimental results are presented in Section 6. Finally we conclude our paper in Section 7.

2 Related Work

Semi-supervised clustering algorithms aim to improve clustering results using limited supervision. The supervision is generally given as pairwise constraints. [2] proposed an algorithm that, given examples of similar or dissimilar pairs of points in $\mathbb{R}^n$, learns a distance metric over $\mathbb{R}^n$ that respects these relationships. [4] studied the problem of learning distance metrics using side-information in the form of equivalence relations, which provide small groups of data points that are known to be similar or dissimilar. [5] proposed a probabilistic model for semi-supervised clustering based on Markov Random Fields that provides a principled framework for incorporating supervision into prototype-based clustering. Most semi-supervised clustering methods in the literature assume that the input is in a vector space [1], [2], [3], [4], [5]. [10] proposed a semi-supervised clustering algorithm SS-Kernel-Kmeans, which uses a kernel approach to cluster a large graph into k disjoint components.

Most existing studies on graph clustering aim to find a k-way disjoint partitioning of a large graph to minimize a certain objective function, such as ratio cut and normalized cut [11]. Other graph clustering criteria include modularity [12], density [13], and stochastic flows [14]. To the best of our knowledge, this paper is the first work on semi-supervised feature selection and clustering of a set of graph objects.

Extracting subgraph patterns from graph data has been studied a lot. Frequent subgraph mining methods can be categorized into two major approaches: an Apriori-based approach [15], [16] and a pattern-growth approach [17], [18], [19]. Recently, graph classification [6], [7], [8] has received a lot of attention. Kong and Yu studied the semi-supervised feature selection for graph classification and proposed a solution called gSSC [9]. A common property of the above methods is to use discriminative subgraphs as the feature space for graph classification. The feature evaluation function, e.g., information gain, is integrated into the subgraph mining process. To expedite the search process, these mining algorithms may not strictly follow the traditional depth-first or breadth-first traversal order to find discriminative subgraphs, for example, [6] uses leap search to prune sibling branches, [8] follows an evolutionary computation strategy to enumerate subgraphs, and gSSC [9] uses the branch-and-bound search strategy.

3 Problem Formulation

In this section, we formulate the problem of semi-supervised clustering of graph objects based on subgraph features.

A graph object is denoted as $G = (V, E, l)$, where V is the vertex set, $E \subseteq V \times V$ is the edge set, and l is the label function mapping a vertex or an edge to a label. The size of a graph is defined as the number of edges. Given a set of graph objects

$\mathcal{D} = \{G_1, G_2, \ldots, G_n\}$ and pairwise constraints in the form of must-links and cannot-links, the goal of semi-supervised clustering is to partition the graph objects in $\mathcal{D}$ into k disjoint clusters $\{\pi_c\}_{c=1}^{k}$, where π_c represents the c-th cluster, such that the total distance between the graph objects and the corresponding cluster centroids is minimized and a minimum number of constraints are violated. The must-link constraint indicates that two graph objects should belong to the same cluster, and the cannot-link constraint indicates that two graphs should belong to different clusters. Moreover, we call the graph objects occurring in the pairwise constraints as *constraint graphs*, otherwise as *unconstraint graphs*. Thus, we can divide $\mathcal{D}$ into a constraint subset $\mathcal{D}_c$ and an unconstraint subset $\mathcal{D}_u$. $\mathcal{D} = \mathcal{D}_c \cup \mathcal{D}_u$.

Different from traditional clustering problems which assume the input is represented in a feature space, graph objects have complex topological structures, but no predefined feature space. Thus we follow the idea of subgraph-based representation, where a set of subgraphs is used as the feature set for representing the graph objects in a feature space. A graph g is a subgraph of another graph G, if there exists a subgraph isomorphism from g to G, denoted as $g \subseteq G$. G is called a supergraph of g. The definitions of subgraph isomorphism and subgraph frequency are given as follows.

Definition 1 (Subgraph Isomorphism). *For two labeled graphs g and G, a subgraph isomorphism is an injective function $f : V(g) \to V(G)$, s.t., (1) $\forall v \in V(g), l(v) = l'(f(v))$; and (2) $\forall (u, v) \in E(g), (f(u), f(v)) \in E(G)$ and $l(u, v) = l'(f(u), f(v))$, where l and l' are the labeling functions of g and G, respectively.*

Definition 2 (Frequency). *Given a graph dataset $\mathcal{D} = \{G_1, \ldots, G_n\}$ and a subgraph g, the supporting graph set of g is $\mathcal{D}_g = \{G_i | g \subseteq G_i, G_i \in \mathcal{D}\}$. The frequency of g is $\frac{|\mathcal{D}_g|}{|\mathcal{D}|}$, denoted as $freq(g)$.*

Given a set of subgraph features $\{g_1, \ldots, g_m\}$, a graph G_i can be represented as a binary vector $\mathbf{x}_i = [x_i^1, \ldots, x_i^m]^\mathsf{T}$, where the k-th component x_i^k in $\mathbf{x}_i$ denotes whether g_k is a subgraph of G_i. $x_i^k = 1$ iff $g_k \subseteq G_i$, $x_i^k = 0$ otherwise. Due to the expressiveness of subgraph features, we adopt the subgraph-based feature representation in our clustering framework. In the paper we use the following notations.

- $\mathcal{S} = \{g_1, g_2, \ldots, g_m\}$: the full set of subgraph features that can be enumerated from the graph objects in $\mathcal{D}$. Only a subset of subgraph features $T \subseteq \mathcal{S}$ is selected for graph object clustering.
- X: the matrix representation of the graph objects $\mathcal{D} = \{G_1, \ldots, G_n\}$ in the feature space of $\mathcal{S}$. $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_m]^\mathsf{T} \in \{0, 1\}^{m \times n}$, where $X = [X_{ij}]^{m \times n}$. $X_{ij} = 1$ iff $g_i \subseteq G_j$, $X_{ij} = 0$ otherwise.
- $\mathcal{M}_0$ and $\mathcal{C}_0$: $\mathcal{M}_0 = \{(G_i, G_j) | \pi(G_i) = \pi(G_j)\}$ denotes the given set of *must-link* constraints where a must-link indicates that two graphs should belong to the same cluster. Here $\pi(G_i)$ denotes the cluster label of graph G_i. $\mathcal{C}_0 = \{(G_i, G_j) | \pi(G_i) \neq \pi(G_j)\}$ denotes the given set of *cannot-link* constraints where a cannot-link indicates that two graphs should belong to different clusters.

Example 1. In Fig.1, we show a set of graph objects $\mathcal{D} = \{G_1, G_2, G_3\}$ and a set of subgraph features $T = \{g_1, g_2, g_3, g_4\}$. There is a must-link between (G_1, G_2) and a

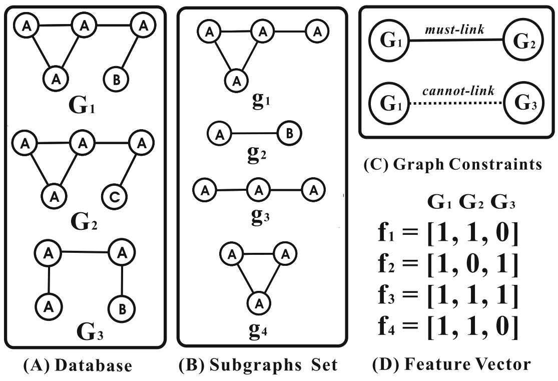


Fig. 1. A Running Example

cannot-link between (G_1, G_3). The feature vectors are also shown. For example, g_1 is a subgraph of G_1 and G_2, but not a subgraph of G_3. So the corresponding feature vector is $f_1 = [1, 1, 0]$.

4 Semi-supervised Subgraph Mining

The first and perhaps the biggest challenge in our graph object clustering problem is how to mine discriminative subgraph features based on both constraint and unconstraint graphs. It is infeasible and unnecessary to enumerate all subgraph patterns from $\mathcal{D}$ for the clustering purpose, as the number of subgraphs is exponential to the graph size. In graph classification [6], [7], [8], [9] where the class label information is available, an evaluation measure such as information gain can be used to select discriminative subgraph features. However, in our clustering problem, the limited supervision is in the form of pairwise constraints, rather than class labels. It is non-trivial to design an objective function for subgraph feature selection, with the aim to satisfy as many constraints as possible. In addition, we need a strategy to integrate the objective function into the subgraph mining process, in order to discover the set of optimal subgraph features wrt. the objective function in a timely fashion, and effectively prune the search space composed of low-quality features.

As our goal is to find a set of high-quality subgraphs for clustering wrt. the constraints, we first formulate the subgraph feature mining problem as an optimization problem, given the semi-supervised information:

$$T^* = \arg\max_{T \subseteq \mathcal{S}} \Psi(T) \ s.t. \ |T| \leq t, \tag{1}$$

where $\Psi(T)$ is an objective function to evaluate the usefulness of a subgraph feature subset T, T^* is the optimal set of subgraph features, $|T|$ represents the size of the subgraph feature set T, and t is the maximum feature number we use.

4.1 Objective Function

We consider both constraint and unconstraint graph objects in defining the objective function Ψ. To fully utilize the supervision information, in the preprocessing step, we try to infer additional constraints from the given constraint sets $\mathcal{M}_0$ and $\mathcal{C}_0$ by assuming consistency of the constraints. For the must-links in $\mathcal{M}_0$, we compute the transitive closure of the must-links to derive connected components consisting of graph objects connected by must-links. Let there be κ connected components, which are used to create κ initial clusters $\{\Re_p\}_{p=1}^{\kappa}$. We use $\mathcal{M}_{inf}$ to denote the must-link constraints inferred from the transitive closure that were not in the initial set, and use $\mathcal{M} = \mathcal{M}_0 \cup \mathcal{M}_{inf}$ to denote the augmented must-link set. For each pair of initial clusters $\Re_p$ and $\Re_q$ that have at least one cannot-link between them, we add cannot-link constraints between every pair of graphs in $\Re_p$ and $\Re_q$, and denote the inferred cannot-links as $\mathcal{C}_{inf}$. The augmented cannot-link set is denoted as $\mathcal{C} = \mathcal{C}_0 \cup \mathcal{C}_{inf}$. This augmentation step can infer as many additional constraints as possible from the given constraint sets. Considering both constraint and unconstraint graphs, the objective function on subgraph features should satisfy the following aspects:

- *must-link*: each pair of graph objects $(G_i, G_j) \in \mathcal{M}_0$ should be close to each other;
- *cannot-link*: each pair of graph objects $(G_i, G_j) \in \mathcal{C}_0$ should be far away from each other;
- *separability*: unconstraint graph objects should be separated from each other. Subgraph features that are too frequent or too rare are not useful, as graph objects represented in such feature space cannot be separated from each other;
- *inner-cluster distance*: graph objects in the same initial cluster $\Re_p$ should be close to each other;
- *inter-cluster distance*: graph objects in different initial clusters $\Re_p$ and $\Re_q$ should be far away from each other.

Based on the above properties, we define an objective function $\Psi(T)$ which we want to maximize on a subgraph feature set T as follows:

$$\begin{aligned}\Psi(T) = {} & \frac{\alpha}{|\mathcal{C}_0|} \sum_{(G_i,G_j)\in\mathcal{C}_0} (D_T\mathbf{x}_i - D_T\mathbf{x}_j)^2 - \frac{\beta}{|\mathcal{M}_0|} \sum_{(G_i,G_j)\in\mathcal{M}_0} (D_T\mathbf{x}_i - D_T\mathbf{x}_j)^2 \\ & + \frac{\gamma}{|\mathcal{D}_u|^2} \sum_{G_i,G_j\in\mathcal{D}_u} (D_T\mathbf{x}_i - D_T\mathbf{x}_j)^2 - \frac{\delta}{|\mathcal{M}|} \sum_{p=1}^{\kappa} \sum_{\substack{G_i,G_j\in\Re_p,\\(G_i,G_j)\in\mathcal{M}}} (D_T\mathbf{x}_i - D_T\mathbf{x}_j)^2 \\ & + \frac{\eta}{|\mathcal{C}|} \sum_{\substack{p,q=1,\\p\neq q}}^{\kappa} \sum_{\substack{G_i\in\Re_p,G_j\in\Re_q,\\(G_i,G_j)\in\mathcal{C}}} (D_T\mathbf{x}_i - D_T\mathbf{x}_j)^2 \qquad (2)\end{aligned}$$

where $D_T = diag(d(T))$ is a diagonal matrix indicating which features are selected into the feature set T from $\mathcal{S}$, $d(T)_i = I(g_i \in T)$ is an indicator function. $\alpha, \beta, \gamma, \delta, \eta$ are five parameters, which adjust the weights of the five types of constraints.

For two graphs $G_i, G_j \in \mathcal{D}$, we define a symmetric matrix $W = [W_{ij}]^{n \times n}$ as:

$$W_{ij} = \begin{cases} -\frac{2\beta}{|\mathcal{M}_0|} - \frac{2\delta}{|\mathcal{M}|} & \text{if } (G_i, G_j) \in \mathcal{M}_0 \\ -\frac{2\delta}{|\mathcal{M}|} & \text{if } (G_i, G_j) \in \mathcal{M}_{inf} \\ \frac{2\alpha}{|\mathcal{C}_0|} + \frac{2\eta}{|\mathcal{C}|} & \text{if } (G_i, G_j) \in \mathcal{C}_0 \\ \frac{2\eta}{|\mathcal{C}|} & \text{if } (G_i, G_j) \in \mathcal{C}_{inf} \\ \frac{2\gamma}{|\mathcal{D}_u|^2} & \text{if } G_i, G_j \in \mathcal{D}_u \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

We give a higher weight W_{ij} to the given must-links and cannot-links in $\mathcal{M}_0$ and $\mathcal{C}_0$, and a lower weight to those inferred constraints in $\mathcal{M}_{inf}$ and $\mathcal{C}_{inf}$, as we assume the provided constraints are stronger than the inferred ones. Then we can rewrite the objective function $\Psi(T)$ in Eq.(2) as follows:

$$\begin{aligned} \Psi(T) &= \frac{1}{2} \sum_{i,j} (D_T \mathbf{x}_i - D_T \mathbf{x}_j)^2 W_{ij} = trace(D_T^{\mathsf{T}} X (D - W) X^{\mathsf{T}} D_T) \\ &= trace(D_T^{\mathsf{T}} X L X^{\mathsf{T}} D_T) = \sum_{g_k \in T} (\mathbf{f}_k^{\mathsf{T}} L \mathbf{f}_k) \end{aligned}$$

where D is a diagonal matrix whose entries are column sums of W, i.e., $D_{ii} = \sum_j W_{ij}$. We denote the matrix $D - W$ as L.

When we use a feature evaluation measure q to denote $q(g_k) = \mathbf{f}_k^{\mathsf{T}} L \mathbf{f}_k$, the optimization problem in Eq.(1) can be rewritten as

$$\max_{T \subseteq \mathcal{S}} \sum_{g_k \in T} q(g_k) \ s.t. \ |T| \leq t \quad (4)$$

Suppose the values for all subgraphs are denoted as $q(g_1) \geq q(g_2) \geq \ldots \geq q(g_m)$ in the descending order. The optimal solution to the optimization problem is: $T^* = \{g_i | i \leq t\}$.

Example 2. Continue our example in Fig.1. After we propagate the two given constraints, we generate two initial clusters $\Re_1 = \{G_1, G_2\}$, $\Re_2 = \{G_3\}$. Suppose all five parameters $\alpha, \beta, \gamma, \delta, \eta$ are set to be 1, we can compute $W = \begin{pmatrix} 0 & -2 & \frac{3}{2} \\ -2 & 0 & \frac{1}{2} \\ \frac{3}{2} & \frac{1}{2} & 0 \end{pmatrix}$, and the corresponding matrix $L = \begin{pmatrix} -\frac{1}{2} & 2 & -\frac{3}{2} \\ 2 & -\frac{3}{2} & -\frac{1}{2} \\ -\frac{3}{2} & -\frac{1}{2} & 2 \end{pmatrix}$. With L, all subgraph features' scores can be calculated according to our objective function as: $q(g_1) = \mathbf{f}_1^{\mathsf{T}} L \mathbf{f}_1 = 2$, $q(g_2) = -\frac{3}{2}$, $q(g_3) = 0$, and $q(g_4) = 2$. g_2 has the lowest score, as it violates the must-link (G_1, G_2) and the cannot-link (G_1, G_3). g_1 and g_4 are the best features.

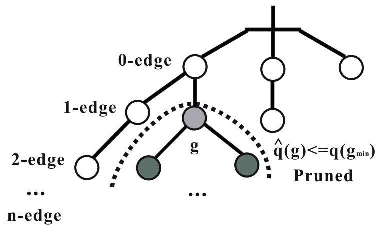


Fig. 2. Subgraph Pattern Searching Tree

4.2 Subgraph Mining with Branch-and-Bound Pruning

To select the optimal feature set T^*, we need to find t subgraphs $g_1, \dots, g_t$ from $\mathcal{D}$ with the highest scores $q(\cdot)$. A straightforward solution is to enumerate the full set of subgraphs $\mathcal{S}$ first, and then calculate the q scores and return the top-t subgraphs. Obviously, this two-step process is not scalable, as the number of subgraphs in $\mathcal{S}$ is exponential to the size of graph objects in $\mathcal{D}$, and could be extremely large. Thus the exhaustive enumeration approach is too expensive to be practical.

Our subgraph feature mining is built based on the gSpan algorithm by Yan and Han [17]. gSpan is an efficient depth-first search algorithm to enumerate subgraphs in their minimum DFS code order. Given a minimum support threshold $min\_sup \in [0, 1]$, gSpan outputs all subgraphs whose frequency is no less than the minimum support.

To further improve the mining efficiency, we can integrate the feature evaluation function $q(\cdot)$ into gSpan and push it deep for search space pruning. If we can derive a tight upper bound of the feature evaluation function q, we can follow a branch-and-bound search strategy to quickly identify the top subgraphs and prune low-quality subgraph features. Theorem 1 gives an upper bound of the q function. The similar principle has been used in some related studies on graph mining and classification [6], [9].

Theorem 1. *(Upper Bound of q Function): Given two subgraphs $g, g' \in \mathcal{S}$, g' is a supergraph of g, i.e., $g \subseteq g'$. The q value of g', $q(g')$, is upper bounded by $\hat{q}(g)$, which is defined as : $\hat{q}(g) = \mathbf{f}_g^\top \hat{L} \mathbf{f}_g$, where the matrix $\hat{L}$ is defined as $\hat{L}_{ij} = \max(0, L_{ij})$.*

Proof. *We compute $q(g') = \mathbf{f}_{g'}^\top L \mathbf{f}_{g'} = \sum_{G_i, G_j \in \mathcal{D}_{g'}} L_{ij}$ where $\mathcal{D}_{g'} = \{G_i | g' \subseteq G_i, G_i \in \mathcal{D}\}$. Since g' is the supergraph of g, we have $\mathcal{D}_{g'} \subseteq \mathcal{D}_g$ according to the Apriori property. Moreover, $\hat{L}_{ij} = \max(0, L_{ij})$, we have $\hat{L}_{ij} \geq L_{ij}$ and $\hat{L}_{ij} \geq 0$. Therefore, we have*

$$q(g') = \sum_{G_i, G_j \in \mathcal{D}_{g'}} L_{ij} \leq \sum_{G_i, G_j \in \mathcal{D}_{g'}} \hat{L}_{ij} \leq \sum_{G_i, G_j \in \mathcal{D}_g} \hat{L}_{ij} = \hat{q}(g)$$

For now, we complete the proof.

With the derived upper bound $\hat{q}$, we can develop a branch-and-bound subgraph mining algorithm on top of gSpan for mining the optimal subgraph feature set T^*. During the depth-first search of the DFS code tree, we maintain the current top-t subgraphs according to the q function. Let g be the currently visited subgraph in the DFS code tree. If there are less than t subgraphs in T, we directly add g into T and recursively perform mining in the DFS code tree; if there are t subgraphs in T, then we will check whether the q value of g, $q(g)$ is higher than the current minimum q value in T, i.e., $q(g) > \min_{g' \in T} q(g')$. If yes, we will replace the lowest-ranked subgraph in T, i.e., $\arg\min_{g' \in T} q(g')$, with g. Before we recursively search the subtree rooted at g, we will first estimate the upper bound of any supergraph g' of g by $\hat{q}(g) = \mathbf{f}_g^\top \hat{L} \mathbf{f}_g$. If $\hat{q}(g)$ is less than the minimum q value in T we have so far, we can safely prune the DSF code subtree rooted at g, as all supergraphs of g cannot have a higher q value than the current top-t subgraphs in T. Fig. 2 illustrates the idea of branch-and-bound search in the DFS code tree. Algorithm 1 shows the branch-and-bound algorithm.

Algorithm 1. Branch-and-Bound Subgraph Mining

Input: Graph objects $\mathcal{D} = \{G_1, \ldots, G_n\}$, must-links $\mathcal{M}$ and cannot-links $\mathcal{C}$, minimum support threshold min_sup, maximum number of features selected t

Output: A set of optimal subgraph features T^*

```
1:  formulate the subgraph feature evaluation function $q$ from $\mathcal{M}$ and $\mathcal{C}$;
2:  $T \leftarrow \emptyset$;
3:  recursively DFS traverse the DFS Code Tree in gSpan:
4:      $g \leftarrow$ currently visited subgraph in DFS Code Tree;
5:      if $|T| < t$
6:          $T \leftarrow T \cup \{g\}$;
7:          recursively DFS traverse the subtree rooted at $g$;
8:      else if $q(g) > \min_{g' \in T} q(g')$
9:          $g_{min} = \arg\min_{g' \in T} q(g')$ and $T = T - \{g_{min}\}$;
10:         $T = T \cup \{g\}$ and update $g_{min} = \arg\min_{g' \in T} q(g')$;
11:     if $\hat{q}(g) > q(g_{min})$ and $freq(g) \geq min\_sup$
12:         recursively DFS traverse the subtree rooted at $g$;
13: return $T^* = T$
```

4.3 Redundancy-Aware Subgraph Features

Based on the feature evaluation function q, we aim to find t subgraph features with the highest q function scores. However, there is a potential issue due to the high redundancy between subgraph patterns: a graph pattern g often occurs in a similar set of graph objects in the database with its supergraph or subgraph patterns. If a graph has a very high q function score, it is likely that its supergraphs or subgraphs have high scores as well. But such supergraphs and subgraphs are redundant to each other. If we return t subgraph features with high redundancy, the useful information contained in the t features is not maximized. To avoid this case, we introduce another function R to measure the redundancy between two subgraphs by the overlap of their supporting graph sets. Given two subgraphs g_i and g_j, the redundancy is defined as $R(g_i, g_j) = \frac{|\mathcal{D}_{g_i} \cap \mathcal{D}_{g_j}|}{|\mathcal{D}_{g_i} \cup \mathcal{D}_{g_j}|}$, where $\mathcal{D}_g$ is the set of graph objects containing a subgraph g. $R \in [0, 1]$. It measures the co-occurrences of two subgraphs in the graph database. The higher $R(g_i, g_j)$ is, the more redundant g_i and g_j are.

Taking the redundancy between graphs into consideration, our goal is to find t subgraphs which have high q function scores and low mutual redundancy. Formally we set a redundancy threshold $\delta \in [0, 1]$. For any two graphs g_i, g_j in the answer set T, we require $R(g_i, g_j) \leq \delta$. With the redundancy requirement, our branch-and-bound subgraph mining algorithm (Algorithm 1) can be revised as follows. Let g be the currently visited subgraph pattern. If $q(g) > \min_{g' \in T} q(g')$, we further check the redundancy between g and every graph $g' \in T$. If $\forall g' \in T$ where $q(g') \geq q(g)$, we have $R(g, g') \leq \delta$ hold, then g is added to T. Otherwise, g is not added to T and we proceed with the recursive subgraph mining process. If g is added to T, then we further check for every $g' \in T$ where $q(g') < q(g)$. If $R(g, g') > \delta$, we will remove g' from T, to make sure every pair of subgraphs in T satisfy the redundancy requirement. Our revised algorithm makes a tradeoff between the feature optimality (wrt. the q function) and the redundancy. For a high-quality subgraph feature with a high q value, if it is redundant to another good feature in the answer set, then the former one will not be considered. With such redundancy control, we will select a high-quality feature set with diversity.

Example 3. In Fig. 1, g_1 is a supergraph of g_4. As both g_1 and g_4 are subgraphs of G_1 and G_2, we have $\mathcal{D}_{g_1} = \mathcal{D}_{g_4} = \{G_1, G_2\}$. The redundancy $R(g_1, g_4) = \frac{|\{G_1,G_2\} \cap \{G_1,G_2\}|}{|\{G_1,G_2\} \cup \{G_1,G_2\}|} = 1$. With the redundancy control mechanism, we choose only one of them.

5 Semi-supervised Graph Object Clustering

Based on the optimal subgraph feature set T^*, we can represent each graph object $G_i \in \mathcal{D}$ as a vector $\mathbf{x}_i$. Then we can use traditional clustering approaches to cluster the vector representation of the graphs objects. In this section, we describe two clustering algorithms we have tested. One is the widely used clustering algorithm K-means, and the other is the kernel-based semi-supervised clustering algorithm SS-Kernel-Kmeans [10]. We use squared Euclidean distance as the unified clustering distortion measure.

5.1 K-Means

In K-means, we first choose k random points as initial centroids. Then each point is assigned to the closest centroid. After that, the centroid of each cluster is updated by taking the average of the vectors of all points in that cluster. We repeat the point assignment and centroid update steps until no point changes its cluster assignment, or we reach the user-specified maximum iterations. The goal of K-means clustering is to find k clusters $\{\pi_c\}_{c=1}^k$ which minimize the sum of the squared distance of each point to its closest centroid. The objective function we aim to minimize can be expressed as:

$$\mathcal{J}\left(\{\pi_c\}_{c=1}^k\right) = \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \|\mathbf{x}_i - \mathbf{m}_c\|^2, \text{ where } \mathbf{m}_c = \frac{\sum_{\mathbf{x}_i \in \pi_c} \mathbf{x}_i}{|\pi_c|} \quad (5)$$

Note that in K-means, we do not utilize the must-links $\mathcal{M}$ and cannot-links $\mathcal{C}$.

5.2 Semi-supervised Kernel-Kmeans

In this part, we discuss the semi-supervised kernel K-means algorithm [10] which considers must-link and cannot-link constraints during the clustering process. Assume two points $\mathbf{x}_i$, $\mathbf{x}_j$ belong to clusters π_p, π_q respectively. The objective function we aim to minimize can be formulated as:

$$\mathcal{J}\left(\{\pi_c\}_{c=1}^{k}\right) = \sum_{c=1}^{k}\sum_{\mathbf{x}_i\in\pi_c}\|\mathbf{x}_i-\mathbf{m}_c\|^2 - \sum_{\substack{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}\\ \pi_p=\pi_q}}\frac{\hat{w}_{ij}}{|\pi_p|} + \sum_{\substack{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}\\ \pi_p=\pi_q}}\frac{\hat{w}_{ij}}{|\pi_p|} \quad (6)$$

The first term in Eq.(6) is the standard K-means objective function, the second term is a cluster-size weighted reward function for must-link constraint satisfaction, and the third is a cluster-size weighted penalty function for cannot-link constraint violation.

6 Experimental Study

In this section, we report our experimental results to demonstrate the effectiveness and mining efficiency of our semi-supervised feature selection and clustering methods. Our algorithm is implemented in C++ and compiled with g++ 2.95.3. The experiments are preformed on a machine with 2.66GHz CPU.

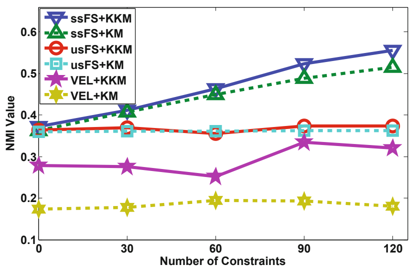
6.1 Datasets

We use protein datasets in our experiments. The protein datasets consist of protein structures from Protein Data Bank (http://www.rcsb.org/pdb/) classified by SCOP (Structural Classification of Proteins). A protein can be represented as a graph object, where a node represents an amino acid and is labeled with the amino acid type. An edge exists between two nodes if the distance between the two alpha carbons in the amino acids is less than 11.5 angstroms and the edge is labeled based on the distance between the alpha carbons. The protein families we use and their sizes are listed in Table 1. The average node number is 217 and the average edge number is 2141 in a protein graph. In the first group of experiments, the graph dataset is created by selecting three protein families with SCOP id 52592, 56251, and 56437. We set the cluster number $k = 3$ in this experiment. In the second group of experiments, the dataset consists of all six protein families listed in Table 1. We set $k = 6$ in this experiment.

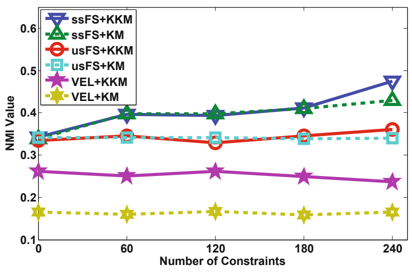
Table 1. List of SCOP Families

SCOP ID	Family Name	Number of Proteins
47617	Glutathione S-transferase (GST)	36
50514	Eukaryotic proteases	44
52592	G proteins	33
56251	Proteasome subunits	35
56437	C-type lection domains	38
88634	Picornaviridae-like VP	39

To generate the pairwise constraints, we randomly select pairs of graph objects (G_i, G_j) from the protein dataset. If G_i and G_j belong to the same family, we create a must-link between G_i and G_j; otherwise, we create a cannot-link between them.



(a) Cluster Number $k = 3$



(b) Cluster Number $k = 6$

Fig. 3. Clustering Performance of Different Methods

6.2 Clustering Methodology and Evaluation Measure

We perform 2-fold cross validation in our experiments: 50% of the graph dataset is used as the training set, from which the must-link and cannot-link constraints are sampled. The other 50% of the dataset is used as the test set. The subgraph mining and clustering steps are run on the whole dataset, while the clustering evaluation is done on the test set only. All results are averaged over 10 runs of the 2-fold cross validation. In the feature set objective function Ψ in Eq.(2), all five parameters $\alpha, \beta, \gamma, \delta, \eta$ are set to be 1.

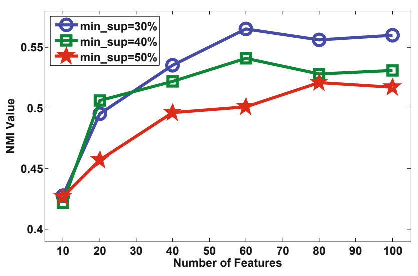
We use *normalized mutual information* (NMI) to evaluate our clustering results. It estimates how closely the clustering algorithm could reconstruct the underlying label distribution in the data. If X is the random variable denoting the cluster labels of the graph objects and Y is the random variable denoting the underlying class labels of the graphs, then NMI is defined as $NMI = \frac{I(X;Y)}{(H(X)+H(Y))/2}$, where $I(X;Y) = H(X) - H(X|Y)$ is the mutual information between the random variables X and Y, $H(X)$ is the Shannon entropy of X, and $H(X|Y)$ is the conditional entropy of X given Y.

We test the following feature selection methods, where the first two use subgraph features and the third uses simple vertex and edge based features.

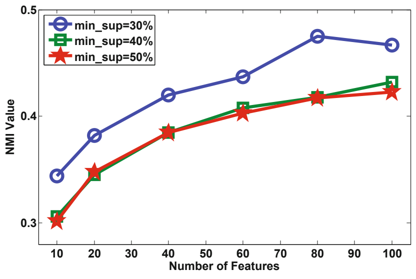
– *Semi-supervised subgraph feature selection*. We select the optimal subgraph feature set T^* wrt. the objective function $\Psi(T)$ which incorporates the supervision information. This method is denoted as **ssFS**.
– *Unsupervised subgraph feature selection*. We select a feature set T wrt. the objective function $\Psi(T)$, but the constraints are not used. Thus we only consider the *separability* of the subgraph features on the unconstraint graph objects. This method is denoted as **usFS**.
– *Using vertex and edge labels as features*. As a baseline method, we use vertex and edge labels as features to represent a graph object as a binary vector. This method is denoted as **VEL**.

We also test the following two clustering methods.

– *Semi-supervised clustering*. The semi-supervised kernel based clustering algorithm SS-Kernel-Kmeans [10] is used for clustering. This method is denoted as **KKM**.

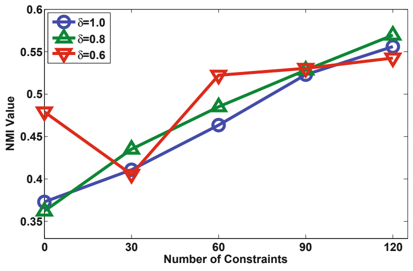


(a) Cluster Number $k = 3$

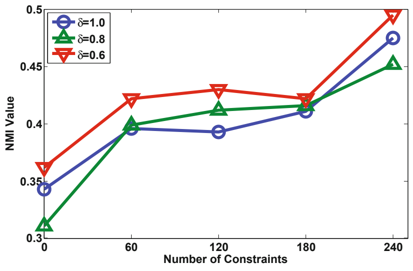


(b) Cluster Number $k = 6$

Fig. 4. Clustering Performance with Different Number of Features and Minimum Support



(a) Cluster Number $k = 3$



(b) Cluster Number $k = 6$

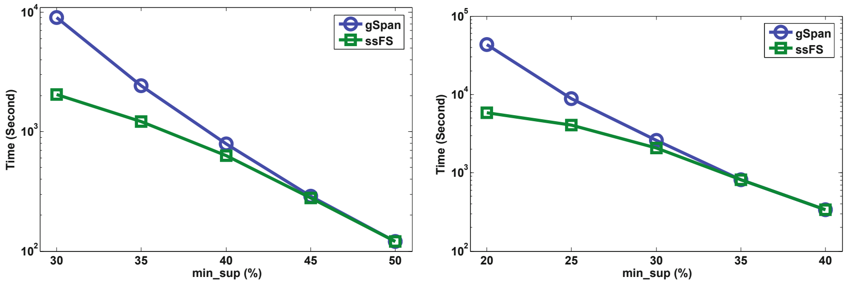
Fig. 5. Clustering Performance with Different Redundancy Thresholds

– *Unsupervised clustering*. Traditional K-means is used for clustering. This method is denoted as **KM**.

We combine different feature selection mechanisms with the two clustering methods, and compare their clustering performance.

6.3 Performance on Graph Clustering

In the first experiment, we compare the clustering performance of different methods listed above. Figure 3 shows the NMI value of different clustering methods when we increase the number of constraints on the two protein graph datasets we created ($k = 3$ and $k = 6$). We set $min\_sup = 30\%$, the number of features $t = 80$, and the redundancy threshold $\delta = 1.0$. A general trend we observe is an increasing NMI value with the increasing number of constraints. In addition, ssFS+KKM achieves the highest NMI value as it utilizes the supervision information in both feature selection and clustering. ssFS+KM comes next, which also shows the usefulness of the optimal feature set when considering constraints for feature selection. The performance of usFS+KKM and usFS+KM is much worse, as the feature selection step is unsupervised. We can see the NMI value of usFS+KM remains unchanged when we increase the constraint number, as it does not utilize the supervision information at all. Finally, the performance of



(a) Cluster Number $k = 3$ (b) Cluster Number $k = 6$

Fig. 6. Semi-supervised Subgraph Feature Mining Time with Branch-and-Bound Search

VEL is the worst, which shows that subgraph features are more effective than simple label features. This experiment demonstrates that our feature selection objective function $\Psi(T)$ is very effective in selecting discriminative subgraph features for clustering. When considering supervision information, the semi-supervised clustering algorithm can also improve the clustering performance.

In the second experiment, we test the clustering performance by varying the parameters min_sup and the feature number t. We fix the redundancy threshold $\delta = 1.0$ and use the semi-supervised method ssFS+KKM. We use 120 constraints for the 3-cluster dataset ($k = 3$) and 240 constraints for the 6-cluster dataset ($k = 6$). As we can see from Figure 4, the NMI value increases in general with the increasing number of features. The NMI value also increases when min_sup decreases, as a lower min_sup implies a larger set of subgraph candidates for feature selection.

In the third experiment, we test the clustering performance by varying the redundancy threshold δ. We set $min\_sup = 30\%$ and the number of features $t = 80$. We run the semi-supervised method ssFS+KKM. As we can see from Figure 5, in general the clustering performance improves as the redundancy threshold decreases, i.e., with a lower redundancy tolerance. In most cases, the NMI value is the highest when $\delta = 0.6$ and it is the lowest when $\delta = 1.0$, i.e., with no redundancy control. For all three redundancy thresholds, the NMI values increase with the number of available constraints.

6.4 Subgraph Mining Efficiency

In this part, we study the subgraph mining efficiency of the branch-and-bound search algorithm (Algorithm 1). In this experiment we set the number of features $t = 100$ and the redundancy threshold $\delta = 1.0$. For semi-supervised feature selection, we use 120 constraints for the 3-cluster dataset ($k = 3$) and 240 constraints for the 6-cluster dataset ($k = 6$). Figure 6 shows the subgraph feature mining time in a logarithmic scale by our feature selection method (ssFS) and the original gSpan mining algorithm, when we vary the min_sup threshold. When the min_sup is low, ssFS is about an order of magnitude faster than gSpan, which shows the effectiveness of the branch-and-bound based pruning. When the min_sup is high, the difference becomes smaller, as a lot of subgraph candidates can be pruned in gSpan as well simply based on min_sup.

7 Conclusions

In this paper, we study the problem of semi-supervised graph object clustering, where pairwise constraints as must-links and cannot-links are used to guide the feature selection and clustering steps. As graph objects are not represented in a vector space, we propose to use subgraphs as features to represent the graph objects in a feature space. An objective function for feature selection is designed, which incorporates the pairwise constraints. We integrate the objective function into gSpan for mining the optimal feature set. An upper bound of the objective function enables us to prune the subgraph search space effectively in a branch-and-bound manner. A semi-supervised kernel based clustering algorithm is used to cluster the graph objects. Our experiments demonstrate that the semi-supervised subgraph feature selection and clustering approach is very effective in boosting the clustering performance.

Acknowledgment. The work described in this paper was partially supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No.: CUHK 419008, 411310 and 411211). Hongliang Fei and Jun Huan are partially supported by the US NSF grant IIS 0845951.

References

1. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: ICML, Williamstown, MA, pp. 577–584 (June 2001)
2. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. In: NIPS, Vancouver, BC, pp. 505–512 (December 2002)
3. Klein, D., Kamvar, S., Manning, C.: From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In: ICML, Sydney, Australia, pp. 307–314 (July 2002)
4. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning distance function using equivalence relations. In: ICML, Washington, DC, pp. 11–18 (August 2003)
5. Basu, S., Bilenko, M., Mooney, R.J.: A probabilistic framework for semi-supervised clustering. In: KDD, Seattle, WA, pp. 59–68 (August 2004)
6. Yan, X., Cheng, H., Han, J., Yu, P.S.: Mining significant graph patterns by scalable leap search. In: SIGMOD, Vancouver, Canada, pp. 433–444 (June 2008)
7. Ranu, S., Singh, A.K.: GraphSig: A scalable approach to mining significant subgraphs in large graph databases. In: ICDE, Shanghai, China, pp. 844–855 (March 2009)
8. Jin, N., Young, C., Wang, W.: GAIA: graph classification using evolutionary computation. In: SIGMOD, Indianapolis, IN, pp. 879–890 (June 2010)
9. Kong, X., Yu, P.S.: Semi-supervised feature selection for graph classification. In: KDD, Washington, DC, pp. 793–802 (July 2010)
10. Kulis, B., Basu, S., Dhillon, I., Mooney, R.: Semi-supervised graph clustering: A kernel approach. In: ICML, Bonn, Germany, pp. 457–464 (August 2005)
11. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
12. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E 69, 026113 (2004)
13. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.J.: SCAN: A structural clustering algorithm for networks. In: KDD, San Jose, CA, pp. 824–833 (August 2007)

14. Satuluri, V., Parthasarathy, S.: Scalable graph clustering using stochastic flows: Applications to community discovery. In: KDD, Paris, France, pp. 737–746 (June 2009)
15. Inokuchi, A., Washio, T., Motoda, H.: An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. In: Żytkow, J.M. (ed.) PKDD 1998. LNCS, vol. 1510, pp. 13–23. Springer, Heidelberg (1998)
16. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: ICDM, San Jose, CA, pp. 313–320 (November 2001)
17. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. In: ICDM, Maebashi, Japan, pp. 721–724 (December 2002)
18. Huan, J., Wang, W., Prins, J.: Efficient mining of frequent subgraph in the presence of isomorphism. In: ICDM, Melbourne, FL, pp. 549–552 (November 2003)
19. Nijssen, S., Kok, J.: A quickstart in frequent structure mining can make a difference. In: KDD, Seattle, WA, pp. 647–652 (August 2004)