Boosting for Multi-Graph Classification

Jia Wu, *Student Member, IEEE*, Shirui Pan, Xingquan Zhu, *Senior Member, IEEE,* and Zhihua Cai

***Abstract*—In this paper, we formulate a novel graph-based learning problem, multi-graph classification (MGC), which aims to learn a classifier from a set of labeled bags each containing a number of graphs inside the bag. A bag is labeled positive, if at least one graph in the bag is positive, and negative otherwise. Such a multi-graph representation can be used for many real-world applications, such as webpage classification, where a webpage can be regarded as a bag with texts and images inside the webpage being represented as graphs. This problem is a generalization of multi-instance learning (MIL) but with vital differences, mainly because instances in MIL share a common feature space whereas no feature is available to represent graphs in a multi-graph bag. To solve the problem, we propose a boosting based multi-graph classification framework (bMGC). Given a set of labeled multi-graph bags, bMGC employs dynamic weight adjustment at both bag- and graph-levels to select one subgraph in each iteration as a weak classifier. In each iteration, bag and graph weights are adjusted such that an incorrectly classified bag will receive a higher weight because its predicted bag label conflicts to the genuine label, whereas an incorrectly classified graph will receive a lower weight value if the graph is in a positive bag (or a higher weight if the graph is in a negative bag). Accordingly, bMGC is able to differentiate graphs in positive and negative bags to derive effective classifiers to form a boosting model for MGC. Experiments and comparisons on real-world multi-graph learning tasks demonstrate the algorithm performance.**

***Index Terms*—Boosting, graph classification, multi-graph, multi-instance learning, subgraph mining.**

I. Introduction

GRAPH classification, in which the object to be classified is a graph, has found many applications in the past decade, such as chemical compounds [1], XML documents [2], program flows [3], and images [4]. Despite its success in a broad spectrum of areas, standard graph classification setting is rather restrictive for many real-world learning problems. One of such problems is multi-graph classification (MGC), in which the object to be classified is a bag of graphs. For example, a webpage may consist of texts and images, where texts can be represented as graphs to preserve contextual information [5] and images can also be represented as graphs to describe structural dependency between image regions [6]. As a result, a webpage can be regarded as a bag containing a number of graphs, each of which represents a certain part of the webpage content. For an information seeker, a webpage is interesting to him/her if one or multiple parts of the webpage (texts and/or images) draws his/her attention—a graph bag is positive if at least one graph in the bag is positive. On the other hand, the webpage is not interesting to the viewer if none of the content attracts the viewer—a graph bag is negative if all graphs inside the bag are negative.

The above multi-graph setting can be found useful in many other domains. For bio-pharmaceutical test, labeling individual molecules (which can be represented as graphs) is expensive and time-consuming. Molecular group activity prediction can be used to investigate the activity of a group (i.e., a bag) of molecules, with the active group (i.e., positive bag), in which at least one molecule is active, being further investigated for individual activity test. Another MGC application is scientific publication classification, where a paper and its references can be represented as a bag of graphs and each graph (i.e., a paper) is formed by using the correlations between keywords in the paper, as shown in Fig. 1. A bag is labeled positive, if the paper or any of its references is relevant to a specific topic. Similarly, for online review based product recommendation, each product receives many customer reviews. For each review composed of detailed text descriptions, we can use a graph to represent the review descriptions. Thus, a product can be represented as a bag of graphs. Assume customers mainly concern about several key properties, such as affordability and durability, of the product. A product (i.e., a bag) can be labeled as positive if it receives very positive review in any of these properties, and negative otherwise. As a result, we can use MGC learning to help recommend products to customers.

Indeed, the MGC problem is a generalization of multi-instance learning (MIL) to graph data, but with significant complications. Existing MIL methods cannot be simply applied to the multi-graph setting because they can only handle bags with all instances being represented in a common vectorial feature space. Unfortunately, in the MGC problem setting, graphs cannot directly provide feature vectors for learning. On the other hand, existing graph classification methods cannot be used to tackle the MGC problem neither, because they require each single graph to be labeled in order to learn a classifier. One simple solution is to represent all graphs in the same feature space, by using some subgraph feature selection methods [7]–[9] to convert graphs as instances, and then

Manuscript received July 14, 2013; revised January 25, 2014; accepted May 13, 2014. Date of publication July 8, 2014; date of current version February 12, 2015. This paper was recommended by Associate Editor M. Last.

J. Wu is with the School of Computer Science, China University of Geosciences, Wuhan 430074, China, and also with the Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering & Information Technology, University of Technology, Sydney, Ultimo, NSW 2007, Australia (e-mail: jia.wu@student.uts.edu.au).

S. Pan is with the Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering & Information Technology, University of Technology, Sydney, Ultimo, NSW 2007, Australia (e-mail: shirui.pan@student.uts.edu.au).

X. Zhu is with the Department of Computer and Electrical Engineering & Computer Science, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: xzhu3@fau.edu).

Z. Cai is with the School of Computer Science, China University of Geosciences, Wuhan 430074, China (e-mail: zhcai@cug.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2014.2327111

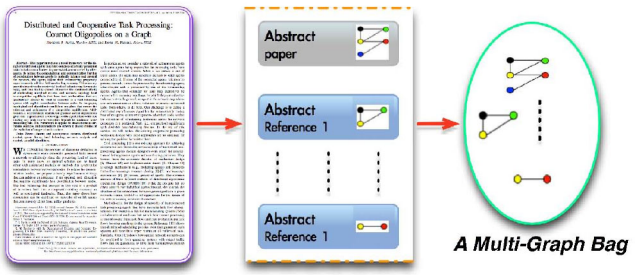

Fig. 1. Example of multi-graph representation for a scientific publication. Each paper is represented as a multi-graph bag, where each graph inside the bag corresponds to the abstract of the paper or the abstract of the reference cited in the paper (a graph is formed by using keywords of the abstract as nodes and their correlations as edges). The graph construction details are reported in Section VII-A.

apply existing MIL methods to the instance bags. However, this simple solution suffers from three inherent disadvantages.

1) *Large Subgraph Feature Space:* The graph substructure feature space increases, with respect to the number of edges and nodes, in an exponential order. It is computationally inefficient, or even infeasible, to enumerate all subgraph features, and then select some subgraph features for classification.
2) *Feature Filtering Inefficiency:* By separating subgraph feature mining and feature selection into two steps, the filtering process of finding salient subgraph patterns will depend on the optimal solution of the subsequent learning algorithm. It is very difficult to theoretically guarantee that the statistical criterion provides good features for the subsequent learning algorithm. This is the problem of all filter methods (as discussed in [10]).
3) *Bag Constraints:* The bag constraints in the multi-graph learning provide important information to differentiate positive and negative graphs, whereas the simple solution directly extracts subgraphs from all graphs without considering multi-graph bag constraints for effective learning.

In summary, the MGC problem for the aforementioned real-world applications needs to address two essential challenges.

1) *Labeling Ambiguity:* Labels are only available at bag level instead of instance level (i.e., a bag is labeled positive if it has at least one positive graph and negative otherwise).
2) *Structured Data Representation:* Instances in a bag are not vectors but graphs, which implies that all instances are not represented in a common feature space for calculating similarities or distances.

Motivated by the above challenges, in this paper, we propose a boosting based multi-graph classification framework (bMGC) for multi-graph classification. In each boosting iteration, bMGC explores the most informative subgraph to construct a single weak classifier, which is used to update the weights of graphs and bags to obtain the next informative subgraph. At the end of the boosting process, the selected weak classifiers are combined to form a strong classifier. A unique characteristic of bMGC is that it combines bag- and graph-level constraints to assess the informativeness score of a subgraph. By adapting the score as a pruning criterion, we combine subgraph mining and informative subgraph exploration to dynamically construct weak classifiers on the fly. As a result, the proposed learning framework not only addresses the labeling ambiguity issue by using a novel two-level (bag and graph) weighting strategy but also addresses the structured data representation issue through a dynamic subgraph selection criterion. The experimental results on real-world data demonstrate that bMGC is effective for MGC.

The remainder of the paper is organized as follows. A brief review of related works is reported in Section II. The problem definition and the overall framework are described in Sections III and IV, respectively. Section V introduces the proposed subgraph selection criterion. The bMGC algorithm is presented in Section VI, followed by the experiments in Section VII. Section VIII discusses the properties of the proposed bMGC, and we conclude the paper in Section IX.

II. Related Work

A. Multi-instance Learning

MGC is a generalization of the MIL problem, which was first proposed by Dietterich *et al.* [11] for drug activity prediction. Since then, it has drawn increasing interest in the machine learning community for many real-world applications, such as image categorization [12], web mining [13], language recognition [14], and computer security [15]. The key assumption of MIL formulation is that the training set is composed of some labeled bags, each of which contains a number of instances. A bag is labeled positive if at least one of its instances is positive and negative otherwise. The goal of MIL is to predict the label of an unknown bag. Several off-the-shelf methods have been developed to solve the MIL problem, which can roughly be divided into two categories.

1) Single-Instance Learner Based MIL: One approach to solve MIL problems is to upgrade generic single-instance learning methods to deal with multi-instance data. For example, lazy learning Citation-KNN and Bayesian-KNN [16] extend the k-nearest neighbor (KNN) algorithm for MIL. Tree learning MITI [17] and MIRI [18] are variations of decision trees for MIL. Rule learning RIPPER-MI adapts the RIPPER algorithm [19] for MIL. Neural network BP-MIP extends standard neural networks [20], and kernel method MISMO adapts the classical support vector machine [21] for MIL. Logistic learning MILR [22] applies the logistic regression to MIL, and ensemble approaches [23], [24] which extend bagging and boosting [25] to MIL.

2) Bag-Based MIL Algorithms: The first specifically designed method for MIL is the axis-parallel rectangle (APR) algorithm [11], which approximates the APRs constructed by the conjunction of features. Based on the idea of APR, a number of algorithms have also been designed for MIL. Examples include diverse density (DD) [26], which searches a point in the feature space by maximizing the DD function that measures a co-occurrence of similar instances from different positive bags; MIEMDD [27], which combines expectation-maximization (EM) algorithm with DD to search the most likely concept; and MIOptimalBall [28], another boosting optimal ball based approach, which uses balls (with respect to

various metrics) as weak hypotheses centered at instances of positive bags.

B. Graph Classification

The MGC problem can also be viewed as a generalization of graph classification where objects are bags of graphs (instead of individual graphs). Existing graph classification methods can be broadly classified into the following two categories.

1) Global Distance Based Approaches: The global distance based methods consider correlations [29] or similarities between two graphs and plug the kernel matrix into a off-the-shelf learner, such as support vector machines, to learn a model for graph classification. Examples include graph kernels [30], [31], graph embedding [32], and graph transformation [33]. One obvious drawback of global distance based approaches is that the distance is calculated based on the similarity of global graph structures, such as random walks or paths, between two graphs. Therefore, it is not clear which substructures (or which parts of the graph) are mostly discriminative for differentiating graphs between different classes.

2) Local Subgraph Feature Based Approaches: For many graph classification tasks, such as chemical compound classification [1], research has shown that graphs within the same class may not have high global similarity but merely share some unique substructures. Accordingly, extracting important subgraph features, using some predefined criteria, to represent a graph in a vectorial space becomes a popular solution for graph classification. The most common subgraph selection criterion is frequency, which intends to select frequently appearing subgraphs by using frequent subgraph mining methods. For example, one of the most popular algorithms for frequent subgraph mining is gSpan [34]. Other methods include AGM [35], FSG [7], MoFa [36], and Gaston [37]. The subgraph feature mining approach seems applicable to the MGC problem as a preprocessing step to transform all graphs into feature vectors. However, one major deficiency of this approach is that it is computationally demanding to enumerate all frequent subgraphs in the target graph set, which inhibits its ability to handle large graph sets.

To overcome this drawback, some supervised subgraph feature extraction approaches have been developed, such as LEAP [38], gPLS [39], and COPK [8], which search directly for discriminative subgraph patterns for classification. Moreover, Jin *et al.* [40] proposes an efficient graph classification method using evolutionary computation for mining discriminative subgraphs for graph classification in large databases. Besides, some graph boosting methods [41]–[44] also exist to use each single subgraph feature as a weak classifier to build boosting algorithm, including some other types of boosting approaches [45], [46] for graph classification.

III. Problem Definition

In this section, we define important notations and concepts, which will be used throughout the paper. We also formally define the MGC problem in this section.

Definition 1 (Connected Graph): A graph is represented as $G = (\mathcal{V}, E, \mathcal{L}, l)$ where $\mathcal{V}$ is a set of vertices, $E \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, and $\mathcal{L}$ is the set of symbols for the vertices and edges. $l: \mathcal{V} \cup E \rightarrow \mathcal{L}$ is the function assigning labels to the vertices and edges. A connected graph is a graph such that there is a path between any pair of vertices.

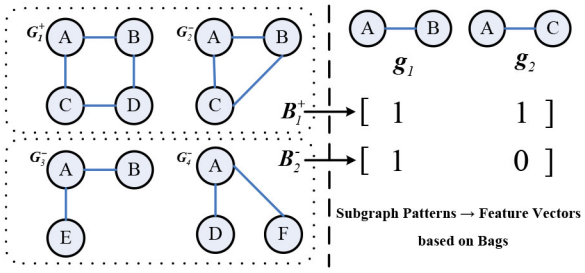

Fig. 2. Example of subgraph feature representation for bags. B_1^+ and B_2^- are positive and negative bags, respectively. G_1^+ is a positive graph and G_2^-, G_3^-, and G_4^- are labeled negative. The feature value of a bag corresponding to each subgraph g_1 or g_2 is set to 1, iff there is a graph in the bag contains the subgraph, and 0 otherwise.

Definition 2 (Bag of Graphs): A graph bag contains a number of graphs, denoted by $B_i = \{G_1^i, \ldots, G_{n_i}^i\}$, where G_j^i and n_i denote the jth graph and the total number of graphs in the ith bag, respectively. For ease of representation, we also use G_j to denote the jth graph in a given bag. A bag B_i's label is denoted by $y_i \in \{-1, +1\}$. A bag is either positive (B_i^+) or negative (B_i^-).

In this paper, we use $\mathcal{B} = \{B_1, \ldots, B_p\}$ to denote a set of bags associated with the weights $\mathbf{w}^B = \{w_1^B, \ldots, w_p^B\}$, where p denotes the number of bags in $\mathcal{B}$. We can also aggregate all graphs in $\mathcal{B}$ as $\mathcal{G} = \{G_1, \ldots, G_q\}$ associated with the weights $\mathbf{w}^G = \{w_1^G, \ldots, w_q^G\}$, where q denotes the number of graphs in $\mathcal{G}$. Similarly, the set of positive bags in $\mathcal{B}$ is denoted by $\mathcal{B}^+$, with $\mathcal{B}^-$ denoting the set of negative bags.

Definition 3 (Subgraph): Let $G = (\mathcal{V}, E, \mathcal{L}, l)$ and $g_k = (\mathcal{V}', E', \mathcal{L}', l')$ each denotes a connected graph. g_k is a subgraph of G, i.e., $g_k \subseteq G$, iff there exists an injective function $\varphi: \mathcal{V}' \rightarrow \mathcal{V}$ s.t.: 1) $\forall v \in \mathcal{V}', l'(v) = l(\varphi(v))$; and 2) $\forall (u, v) \in E', (\varphi(u), \varphi(v)) \in E$ and $l'(u, v) = l(\varphi(u), \varphi(v))$. If g_k is a subgraph of G, then G is a supergraph of g_k.

Definition 4 (Subgraph Feature Representation for Graph): Let $\mathcal{S}_g = \{g_1, \ldots, g_s\}$ denote a set of subgraph patterns discovered from a given set of graphs. For each graph G_i, we use a subgraph feature vector $\mathbf{x}_i^G = [(x_i^{g_1})^G, \ldots, (x_i^{g_s})^G]^\top \in \{0, 1\}^s$ to represent G_i in the feature space, where $(x_i^{g_k})^G = 1$, iff g_k is a subgraph of G_i (i.e., $g_k \subseteq G_i, g_k \in S_g$) and $(x_i^{g_k})^G = 0$ otherwise.

Definition 5: (Subgraph Feature Representation for Bag): Given a set of subgraphs $\mathcal{S}_g = \{g_1, \ldots, g_s\}$, a graph bag B_i can be represented by a feature vector $\mathbf{x}_i^B = [(x_i^{g_1})^B, \ldots, (x_i^{g_s})^B]^\top \in \{0, 1\}^s$, where $(x_i^{g_k})^B = 1$, iff g_k is a subgraph of any graph G_j in bag B_i (i.e., $\exists G_j \in B_i \wedge G_j \supseteq g_k, g_k \in S_g$) and $(x_i^{g_k})^B = 0$ otherwise.

An example of subgraph feature representation for graph bags is illustrated in Fig. 2, where two graph bags (B_1^+ and B_2^- on the left panel) are represented as two 2-D feature vectors (on the right panel) based on two subgraph patterns (g_1 and g_2).

Given a multi-graph set $\mathcal{B}$ with a number of labeled graph bags, where each positive bag contains at least one positive graph and all graphs in each negative bag are negative (i.e., the

bag constraint in MGC), the aim of MGC is to build a prediction model from the training multi-graph bag set $\mathcal{B}$ to predict some previously unseen graph bags with unknown label with maximum bag classification accuracy.

IV. Overall Framework of bMGC

In multi-graph bags, there is no feature available to represent graphs, so existing MIL methods, which require instances to have a vectorized feature representation, cannot be applied to MGC. In addition, due to lack of labeling information for individual graphs inside positive bags, subgraph feature based graph classification cannot be directly applied to MGC neither.

To solve the above issues, in this section we propose a bMGC framework, which applies dynamic weight adjustment at both graph- and bag-levels to select one subgraph in each iteration to construct a single weak classifier. In each iteration, the bag and graph weights are adjusted by the last bag-level and graph-level weak classifiers, respectively. By doing so, bMGC is able to differentiate graphs in positive or negative bags to derive effective learning models by boosting all the single subgraph bag-level weak classifiers. The proposed bMGC framework, as shown in Fig. 3, includes the following four major steps.

1) *Subgraph Candidate Generation:* Generating subgraph candidates is a key step for selecting the most informative subgraph. To find subgraph candidates with diverse structures, we aggregate graphs in multi-graph bags into three graph sets: a) graphs in all bags; b) graphs in all positive bags; and c) graphs in all negative bags. A gSpan [34] based subgraph mining procedure is applied to each graph set, through which a set of diverse subgraph candidate patterns can be discovered for validation.
2) *Bag Constrained Subgraph Exploration:* In the tth iteration, an informative subgraph g_t is selected to form a weak classifier for MGC under the weighted bag- and graph-level constraints. To obtain the $t+1$th informative subgraph, the weights of bags and graphs should be updated. After m iterations, the selected m subgraphs will correspond to m weak classifiers for learning.
3) *Updating Weights of Bags and Graphs:* After we find the tth informative subgraph g_t, a bag-level classifier $\mathcal{H}_t^B$ and a graph-level classifier $\mathcal{H}_t^G$ will be trained, respectively. For graphs, due to our assumption that we apply bag labels to graphs, some graphs in positive bags have been assigned wrong labels. If a graph G_i in positive bag set $\mathcal{B}^+$ is misclassified by $\mathcal{H}_t^G$, in the next iteration we will decrease G_i's weight to reduce its impact on the learning process. If a graph G_i in negative bag set $\mathcal{B}^-$ is misclassified, its weight will be increased, such that G_i in the negative bag set will play a more important role to help the learning algorithm find better subgraphs.
4) *Boosting Classification:* After the subgraphs are selected in all iterations to form the corresponding single weak classifiers, they can be weighted to construct a strong classifier for MGC.

In the following two sections, we first propose our subgraph exploration criterion in Section V and then introduce detailed procedures of bMGC in Section VI.

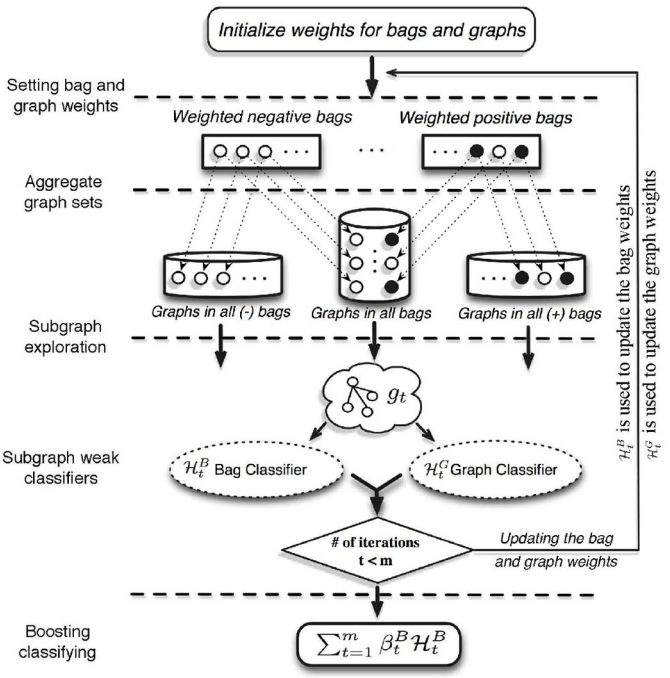

Fig. 3. Overview of the proposed bMGC framework.

V. Subgraph Exploration

Exploring optimal bag constrained subgraphs in each iteration of bMGC is a nontrivial task. This process has two main challenges.

1) How to utilize the information of the labeled graphs in negative bags?
2) How to tackle the problem that the labels of graphs in positive bags are unknown?

Assume that a set of candidate graphs are collected from the bag set $\mathcal{B}$, let $\mathcal{S}_g$ denote the complete set of subgraphs in $\mathcal{B}$, and g_t be the optimal subgraph selected from $\mathcal{S}_g$ in the tth iteration. Our bag constrained subgraph exploration aims to find the most informative subgraph g_t in each iteration with weight updating for both bags and graphs. Let $\mathcal{Z}(g_k)$, the evaluation criterion for a single subgraph $g_k \in \mathcal{S}_g$, be a function to measure the informativeness of g_k as

$$g_t = \arg\max_{g_k \in \mathcal{S}_g} (\mathcal{Z}(g_k)). \tag{1}$$

The objective function in (1) indicates that the optimal bag constrained subgraph g_t should have the maximum discriminative capability for MGC.

A. Evaluation Criterion for Subgraphs

In order to measure the informativeness of a subgraph g_k, i.e., $\mathcal{Z}(g_k)$, such that we can discover the most informative subgraph for bags, we impose constraints to the labeled bags in the multi-graph bag set $\mathcal{B}$, through which the subgraph selection criterion $\mathcal{Z}(g_k)$ can be properly defined. For two bags, B_i and B_j, if they have the same class labels, there is a pairwise must-link constraint between them. If B_i and B_j have different class labels, there is a cannot-link constraint between them.

To further take the data distributions in each bag into consideration, we also add graph-level constraints to ensure that the selected subgraphs can make graphs in each negative bag close to each other and make graphs in each positive bag be maximally separated. In summary, a good subgraph should satisfy the following constraints.

1) *Weighted Bag Must-Link:* If there is a must-link between B_i and B_j, their subgraph feature vectors $\mathbf{x}_i^B$ and $\mathbf{x}_j^B$ should be close to each other. In a MGC scenario, each bag B_i is associated with a weight w_i^B. For each pair of bags with the same class label, the selected subgraph should ensure that bags with similar weights (analogous to importance) have a high similarity.
2) *Weighted Bag Cannot-Link:* If there is a cannot-link between B_i and B_j, the underlying subgraph feature vectors $\mathbf{x}_i^B$ and $\mathbf{x}_j^B$ should be distinct from each other. For each pair of bags in different classes, the smaller the weight difference between the two bags, the more impact the constraint will have for selecting subgraph to represent the distinction between them.
3) *Weighted Graph Must-Link:* If there is a must-link between G_i and G_j, their subgraph feature vectors $\mathbf{x}_i^G$ and $\mathbf{x}_j^G$ should be close to each other. In bMGC, only graphs in negative bags are known to have genuine labels, in which the feature representations of the two weighted graphs should have low diversity.
4) *Weighted Graph Separability:* If genuine labels of graphs G_i and G_j are unknown, the corresponding subgraph feature vectors $\mathbf{x}_i^G$ and $\mathbf{x}_j^G$ should be different. This is similar to the principal component analysis (PCA)'s assumption [47], which aims to find the component with the largest possible variance. This constraint applies to positive bags, because genuine labels of all graphs in each positive bag are unknown. As a result, we apply this constraint to encourage each positive bag to have a large diversity inside the bag. Similar assumption has also been used in [9] to handle unlabeled graphs in a semi-supervised learning setting.

In summary, the bag must-link and bag cannot-link constraints are applied to bags with the same label and different labels, respectively. While the graph must-link and graph separability constraints are only applied to graphs in negative bags and graphs in positive bags, respectively.

By imposing constraints to both bag- and graph- levels, our evaluation criterion intends to capture informative subgraph features for MGC. Based on the above considerations, we derive a criterion $\mathcal{Z}(g_k)$ for measuring the informativeness of a subgraph g_k as follows:

$$\begin{aligned}\mathcal{Z}(g_k) = &\frac{1}{2A}\sum_{y_iy_j=-1}\left(\mathcal{D}_{g_k}w_i^B\mathbf{x}_i^B - \mathcal{D}_{g_k}w_j^B\mathbf{x}_j^B\right)^2\\ &-\frac{1}{2B}\sum_{y_iy_j=1}\left(\mathcal{D}_{g_k}w_i^B\mathbf{x}_i^B - \mathcal{D}_{g_k}w_j^B\mathbf{x}_j^B\right)^2\\ &-\frac{1}{2C}\sum_{\forall G_i,G_j\in\mathcal{B}^-}\left(\mathcal{D}_{g_k}w_i^G\mathbf{x}_i^G - \mathcal{D}_{g_k}w_j^G\mathbf{x}_j^G\right)^2\\ &+\frac{1}{2D}\sum_{\forall G_i,G_j\in\mathcal{B}^+}\left(\mathcal{D}_{g_k}w_i^G\mathbf{x}_i^G - \mathcal{D}_{g_k}w_j^G\mathbf{x}_j^G\right)^2 \quad (2)\end{aligned}$$

where w_i^B, w_j^B, w_i^G, and w_j^G are the weights for B_i, B_j, G_i, and G_j, respectively. $\mathcal{D}_{g_k} = diag(d(g_k))$ is a diagonal matrix indicating which subgraph feature g_k is selected from S_g to represent the bags or graphs, $d(g_k)_i = I(g_i = g_k, g_i \in S_g)$ with $I(\cdot)$ equaling to 1 if the condition inside is true and 0 otherwise. $A = \sum_{y_iy_j=-1} 1$, $B = \sum_{y_iy_j=1} 1$, $C = \sum_{G_i,G_j\in\mathcal{B}^-} 1$, and $D = \sum_{G_i,G_j\in\mathcal{B}^+} 1$ assess the total pairwise sets of constraints in the bag cannot-link, bag must-link, graph must-link and graph separability.

We define two matrices for bag-level and graph-level constraints, denoted by $M_B = [M_{ij}^B]^{p\times p}$ and $M_G = [M_{ij}^G]^{q\times q}$, respectively, where $M_{ij}^B = \{1/A, y_iy_j = -1;\ -1/B, y_iy_j = 1\}$, and $M_{ij}^G = \{-1/C, \forall G_i, G_j \in \mathcal{B}^-; 1/D, \forall G_i, G_j \in \mathcal{B}^+; 0, \text{otherwise}\}$.

As a result, (2) can be rewritten as

$$\begin{aligned}\mathcal{Z}(g_k) &= \mathcal{Z}(g_k)^B + \mathcal{Z}(g_k)^G\\ &= \frac{1}{2}\sum_{y_iy_j}\left(\mathcal{D}_{g_k}w_i^B\mathbf{x}_i^B - \mathcal{D}_{g_k}w_j^B\mathbf{x}_j^B\right)^2 M_{ij}^B\\ &\quad+\frac{1}{2}\sum_{G_iG_j}\left(\mathcal{D}_{g_k}w_i^G\mathbf{x}_i^G - \mathcal{D}_{g_k}w_j^G\mathbf{x}_j^G\right)^2 M_{ij}^G. \quad (3)\end{aligned}$$

For bag-level evaluation $\mathcal{Z}(g_k)^B$, we have

$$\begin{aligned}\mathcal{Z}(g_k)^B &= \frac{1}{2}\sum_{y_iy_j}\left(\mathcal{D}_{g_k}w_i^B\mathbf{x}_i^B - \mathcal{D}_{g_k}w_j^B\mathbf{x}_j^B\right)^2 M_{ij}^B\\ &= tr\left(\mathcal{D}_{g_k}^\top X_B W_B(D_B - M_B)W_B^\top X_B^\top \mathcal{D}_{g_k}\right)\\ &= tr\left(\mathcal{D}_{g_k}^\top X_B W_B L_B W_B^\top X_B^\top \mathcal{D}_{g_k}\right)\\ &= \left(\boldsymbol{f}_{g_k}^B\right)^\top W_B L_B W_B^\top \boldsymbol{f}_{g_k}^B\\ &= \left(\boldsymbol{f}_{g_k}^B\right)^\top Q_B\, \boldsymbol{f}_{g_k}^B \quad (4)\end{aligned}$$

where $L_B = D_B - M_B$ is a Laplacian matrix, where $D_B = diag(d_i^B)$ is a diagonal matrix with $d_i^B = \sum_j M_{ij}^B$. $Q_B = W_B L_B W_B^\top$, where W_B is a also a diagonal matrix, with $W_{ii}^B = w_i^B$ denoting the weight of the ith bag B_i. $X_B = [\mathbf{x}_1^B, \ldots, \mathbf{x}_p^B] = [\boldsymbol{f}_{g_1}^B, \ldots, \boldsymbol{f}_{g_s}^B]^\top \in \{0,1\}^{s\times p}$, where $\boldsymbol{f}_{g_k}^B$ is an indicator vector of subgraph g_k with respect to all the bags in $\mathcal{B}$. Specifically, $\boldsymbol{f}_{g_k}^B = [f_{g_k}^{B_1}, \ldots, f_{g_k}^{B_p}]^\top \in \{0,1\}^p$, where $f_{g_k}^{B_i} = 1$ iff $\exists G \in B_i \wedge G \supseteq g_k$ and $f_{g_k}^{B_i} = 0$ otherwise.

Similarly, the graph-level evaluation $\mathcal{Z}(g_k)^G$ can be rewritten in the form of matrix. Taking both the bag-level and graph-level evaluation functions together, we have

$$\begin{aligned}\mathcal{Z}(g_k) &= \mathcal{Z}(g_k)^B + \mathcal{Z}(g_k)^G\\ &= \left(\boldsymbol{f}_{g_k}^B\right)^\top Q_B \boldsymbol{f}_{g_k}^B + \left(\boldsymbol{f}_{g_k}^G\right)^\top Q_G \boldsymbol{f}_{g_k}^G\\ &= \boldsymbol{f}_{g_k}^\top Q \boldsymbol{f}_{g_k} \quad (5)\end{aligned}$$

where $Q_G = W_G L_G W_G^\top$, with W_G a diagonal matrix, i.e., $W_{ii}^G = w_i^G$, denoting the weight of the ith graph G_i. $L_G = D_G - M_G$ is known as a Laplacian matrix, where $D_G = diag(d_i^G)$ is a diagonal matrix with $d_i^G = \sum_j M_{ij}^G$. Meanwhile, $\boldsymbol{f}_{g_k}^G$ is an indicator vector of subgraph g_k with respect to

all graphs in $\mathcal{G}$, and $\boldsymbol{f}_{g_k}^{G} = [f_{g_k}^{G_1}, \ldots, f_{g_k}^{G_q}]^\top \in \{0,1\}^q$, where $f_{g_k}^{G_i} = 1$ iff $g_k \subseteq G_i$ and $f_{g_k}^{G_i} = 0$ otherwise.

According to (5), we have

$$\boldsymbol{f}_{g_k} = \begin{bmatrix} \boldsymbol{f}_{g_k}^{B} \\ \boldsymbol{f}_{g_k}^{G} \end{bmatrix} \qquad Q = \begin{bmatrix} Q_B & 0 \\ 0 & Q_G \end{bmatrix} \tag{6}$$

where $\boldsymbol{f}_{g_k}$ is an indicator vector of subgraph g_k with respect to the data combined with bag matrix X_B and graph matrix X_G. By denoting the function as $h(g_k, Q) = \boldsymbol{f}_{g_k}^\top Q \boldsymbol{f}_{g_k}$, the problem of maximizing $\mathcal{Z}(g_k)$ in (1) is equivalent to finding a subgraph that can maximize the $h(g_k, Q)$, which can be represented as

$$g_t = \max_{g_k \in \mathcal{S}_g} h(g_k, Q) \tag{7}$$

Definition 6 (bScore): Given two matrices M_B and M_G embedding the label information, respectively, and two corresponding weight matrices W_B and W_G, the informativeness score of a subgraph g_k is defined in (8)

$$r(g_k) = h(g_k, Q) = \boldsymbol{f}_{g_k}^\top Q \boldsymbol{f}_{g_k}. \tag{8}$$

In the above definition, a larger bScore $r(g_k)$ value represents a stronger dependency between this subgraph feature and the corresponding labels. In other words, good subgraph features should have high bScore values. To find the optimal subgraph in each iteration, we can calculate bScore values of all subgraphs in $\mathcal{S}_g$, and then select the topmost subgraph with the highest $r(g_k)$ value.

B. Upper Bound of bScore

Before we introduce detailed algorithm to mine the optimal subgraph in each iteration, we derive a bScore upper bound to help prune the subgraph search space.

Theorem 1: Given two subgraphs $g_k, g_k{}' \in S_g$, g'_k is a supergraph of g_k (i.e., $g'_k \supseteq g_k$). The bScore value g'_k ($r(g'_k)$) is bounded by $\hat{r}(g_k)$, i.e., $r(g'_k) \leq \hat{r}(g_k)$

$$\hat{r}(g_k) = \boldsymbol{f}_{g_k}^\top \hat{Q} \boldsymbol{f}_{g_k} \tag{9}$$

where $\hat{Q} = \begin{bmatrix} \hat{Q}_B & 0 \\ 0 & \hat{Q}_G \end{bmatrix}$, in which $\hat{Q}_B$ and $\hat{Q}_G$ are defined as $\hat{Q}_{ij}^B = max(0, Q_{ij}^B)$ and $\hat{Q}_{ij}^G = max(0, Q_{ij}^G)$.

For any $g'_k \supseteq g_k$, $r(g'_k) \leq \hat{r}(g_k)$. The corresponding proof is given in Appendix.

C. Mining Bag Constrained Subgraph

For subgraph selection, we employ a depth-first search (DFS) based algorithm gSpan [34] to enumerate subgraphs. The key idea of gSpan is that each subgraph has a unique DFS code, which is defined by a lexicographic order of the discovery time during the search process. Two subgraphs are isomorphism iff they have the same minimum DFS code. By employing a depth-first search strategy on the DFS code tree (where each node is a subgraph), gSpan can enumerate all frequent subgraphs efficiently.

Algorithm 1 reports the proposed bag constrained subgraph exploration process, which starts with an empty optimal subgraph set and continuously enumerates subgraphs by recursively visiting the DFS code tree. If a subgraph g_k is not a frequent subgraph, both g_k and its subtree will be pruned (lines 3–5), in which $freq(g_k)$ denotes the percentage of graphs containing the subgraph g_k in graph dataset $\mathcal{G}$; otherwise, we calculate g_k's bScore value $r(g_k)$ (line 6). If $r(g_k)$ is larger than the current optimal score $r(g_t)$ or it is the first step (i.e., the optimal subgraph set is empty), we regard g_k as the current optimal subgraph g_t (lines 7–9). After that, the upper bound pruning module will check if $\hat{r}(g_k)$ is less than $r(g_t)$, if so, it means that the bScore value of any supergraph g'_k of g_k (i.e., $g'_k \supseteq g_k$) will not be greater than $r(g_t)$. Thus, we can safely prune subtrees rooted from g_k in the search space. If $\hat{r}(g_k)$ is indeed greater than the bScore of g_t, we cannot prune this space since there might exist a supergraph $g'_k \supseteq g_k$ with $r(g'_k) \geq r(g_t)$, so the DFS will continue by following the children of g_k (lines 10–12), until the frequent subgraph mining process is completed.

Algorithm 1 BSE: Bag Constrained Subgraph Exploration

Input: $\mathcal{G}$: A graph dataset;
min_sup: The threshold of the frequent subgraph;
Output: g_t: The optimal subgraph;

```
1:  while Recursively visit the DFS Code Tree in gSpan do
2:    g_k ← current visited subgraph in DFS Code Tree of G;
3:    if freq(g_k) < min_sup then
4:      continue;
5:    end if
6:    Compute the bScore r(g_k) for subgraph g_k;
7:    if g_t == NULL or r(g_k) > r(g_t) then
8:      g_t ← g_k;
9:    end if
10:   if r̂(g_k) ≥ r(g_t) then
11:     Depth-first search the subtree rooted from node g_k;
12:   end if
13: end while
14: return g_t;
```

VI. bMGC

The detailed procedures of bMGC are reported in Algorithm 2, which iteratively expands the candidate graph set to exact informative subgraphs, then explores the optimal subgraphs based on bScore. After m iterations, bMGC boosts the m selected weak classifiers to obtain the final classification model.

A. bMGC Algorithm

In Algorithm 2, bMGC differentiates and considers graph in three sets: graphs in positive bags $\mathcal{G}^+$, graphs in negative bags $\mathcal{G}^-$, and graphs in both positive and negative bags $\mathcal{G}$. The benefit of separating graphs into three sets is that the subgraph mining process, which is carried out on each set respectively, will increase the candidate graph set for exploring subgraphs. By doing so, the subgraph space becomes more dense, through which good subgraph features can be discovered.

The while loop in Algorithm 2 represents the boosting process of bMGC. In each iteration, the subgraph mining is

Algorithm 2 bMGC: Boosting for Multi-graph Classification

Input:
$\mathcal{B}$: Multi-graph bag set; $\mathcal{G}$: Graph dataset in $\mathcal{B}$;
m: The number of iterations;
min_sup: The threshold of the frequent subgraph;
1: **Output:** The class label y_k of a testing bag B_k.
2: **Initialize** $\forall w_i^B \in \mathbf{w}^B$:$w_i^B = 1$; $\forall w_i^G \in \mathbf{w}^G$:$w_i^G = 1$; $t = 0$;
// Training Phase:
3: $\{\mathcal{G}^+, \mathcal{G}^-\} \leftarrow$ Graphs in $\mathcal{B}^+$ and $\mathcal{B}^-$, respectively;
4: $\{p, q, q^+\} \leftarrow \#of\ bags\ \in \mathcal{B}, \#\ of\ graphs\ \in \mathcal{G}, \mathcal{G}^+$;
5: **while** $t < m$ **do**
6: $t \leftarrow t + 1$;
7: $\mathbf{w}^B \leftarrow \mathbf{w}^B / \sum_{i=1}^{p} w_i^B$, $\mathbf{w}^G \leftarrow \mathbf{w}^G / \sum_{i=1}^{q} w_i^G$;
8: $g_t^{\mathcal{G}} \leftarrow BSE(\mathcal{G}, min\_sup)$; //Algorithm 1
9: $g_t^{\mathcal{G}^+} \leftarrow BSE(\mathcal{G}^+, min\_sup)$; //Algorithm 1
10: $g_t^{\mathcal{G}^-} \leftarrow BSE(\mathcal{G}^-, min\_sup)$; //Algorithm 1
11: $g_t \leftarrow$ The subgraph with the highest bScore ($g_t^{\mathcal{G}}$, $g_t^{\mathcal{G}^+}$, $g_t^{\mathcal{G}^-}$);
// Error Calculation:
12: $\varepsilon_t^B \leftarrow$ Calculate the error of $\mathcal{H}_t^B$ corresponding to g_t on $\mathcal{B}$;
13: **if** $\varepsilon_t^B > 1/2$ **then**
14: $\mathcal{H}_t^B \leftarrow -\mathcal{H}_t^B$, $\varepsilon_t^B \leftarrow 1 - \varepsilon_t^B$;
15: **end if**
16: $\varepsilon_t^{G^-} \leftarrow$ Calculate the error of $\mathcal{H}_t^G$ corresponding to g_t on $\mathcal{G}^-$;
17: **if** $\varepsilon_t^{G^-} > 1/2$ **then**
18: $\mathcal{H}_t^G \leftarrow -\mathcal{H}_t^G$; $\varepsilon_t^{G^-} \leftarrow 1 - \varepsilon_t^{G^-}$;
19: **end if**
20: $\beta_t^B \leftarrow (1 - \varepsilon_t^B)/\varepsilon_t^B$;
21: $\beta_t^{G^-} \leftarrow \varepsilon_t^{G^-}/(1 - \varepsilon_t^{G^-})$, $\beta^{G^+} \leftarrow 1/(1 + \sqrt{2Inq^+/m})$;
// Increase weight for incorrectly classified bag:
22: $w_i^B \leftarrow w_i^B (\beta_t^B)^{I\left(\mathcal{H}_t^B(B_i) \neq c(B_i)\right)}$, $\forall B_i \in \mathcal{B}$;
// Decrease weight for incorrectly classified graph in $\mathcal{B}^+$:
23: $w_j^{G^+} \leftarrow w_j^{G^+} (\beta^{G^+})^{I\left(\mathcal{H}_t^G(G_j) \neq c(G_j)\right)}$, $\forall G_j \in \mathcal{G}^+$;
// Increase weight for incorrectly classified graph in $\mathcal{B}^-$:
24: $w_k^{G^-} \leftarrow w_k^{G^-} (\beta_t^{G^-})^{-I\left(\mathcal{H}_t^G(G_k) \neq c(G_k)\right)}$, $\forall G_k \in \mathcal{G}^-$;
25: **end while**
// Testing Phase:
26: $y_k \leftarrow sign\left(\sum_{t=1}^{m} \beta_t^B \mathcal{H}_t^B(B_k)\right)$

carried out on three graph sets as shown from lines 7 to 9. The current optimal subgraph g_t is the one with the highest bScore with respect to the subgraph discovered from each individual graph sets (line 10). In bMGC, the subgraph g_t is directly used as a weak bag classifier $\mathcal{H}_t^B$ or a weak graph classifier $\mathcal{H}_t^G$, with $\mathcal{H}_t^B(B_i) = 1$ iff $(x_i^{g_t})^B = 1$, and $\mathcal{H}_t^B(B_i) = -1$, otherwise. The same classification method is also used in graph based subgraph classifier $\mathcal{H}_t^G$. Accordingly, the steps from lines 11 to 20 use the error rates of the weak classifiers to update the parameters of the boosting framework.

1) Updating Bag and Graph Weights: To obtain the $t+1$th optimal subgraph g_{t+1}, we must update the weights of bags and graphs using the tth optimal subgraph g_t. The error ε_t^B (line 11) on a bag set $\mathcal{B}$ can be defined as follows:

$$\varepsilon_t^B = \sum_{i=0}^{p} \frac{w_i^B I\left(\mathcal{H}_t^B(B_i) \neq c(B_i)\right)}{\sum_{i=1}^{p} w_i^B} \tag{10}$$

where $c(B_i)$ returns the label for the ith bag and $I(\cdot)$ is the indicator function. The error $\varepsilon_t^{G^-}$ (line 15) on a negative graph set can also be obtained in a similar way. Note that ε_t^B and $\varepsilon_t^{G^-}$ are required to be smaller than 1/2. If not, the underlying classifier is worse than random hypothesis, and then we should use $-\mathcal{H}_t^B$ and $-\mathcal{H}_t^G$ to replace the current bag- and graph-level classifiers, respectively. As a result, the underlying errors on bag set and negative graph set become $1 - \varepsilon_t^B$ and $1 - \varepsilon_t^{G^-}$, respectively (lines 12–14 and 16–18).

According to the specific characteristics of bags and graphs, we employ two different weighting strategies. Because bags are the target of the classification and their genuine labels are given, if a bag is misclassified by the current subgraph g_t classifier $\mathcal{H}_t^B$, the bag weight is increased by using the weight coefficient factor β_t^B (line 19) in order to find more informative subgraph in the next iteration to deal with incorrectly predicted bags (line 21). This bag-level weighting mechanism is similar to the AdaBoost algorithm [25]. At individual graph level, because we propagate bag labels to graphs at the very beginning of the algorithm, some graphs in positive bags might have been assigned with wrong labels. Therefore, if a graph in positive bags is misclassified (i.e., $\mathcal{H}_t^G(G_j) \neq c(G_j)$), in the next iteration we decrease its weight to reduce its effect through multiplying its weight by $(\beta^{G^+})^{I(\mathcal{H}_t^G(G_j) \neq c(G_j))} \in (0, 1]$, where β^{G^+} is the weight coefficient factor for positive graph (line 20). Thus, the misclassified graphs in positive bags will have reduced impact on the learning process in the next round (line 22). The graphs with large training weights will help the learning algorithm find better subgraphs. For negative bags, the weight updating mechanism is the same for all graphs inside the bag (line 23). This graph-level weighting mechanism is similar to the TrAdaBoost algorithm [48].

In the test phase, the test bag B_k will be tested using a weighted classifier $sign\left(\sum_{t=1}^{m} \beta_t^B \mathcal{H}_t^B(B_k)\right)$ by boosting all the m weak classifiers $\mathcal{H}_t^B$, $t = 1, 2, \ldots, m$ to obtain its class label y_k (line 25).

The key technical advantage of the bMGC process can be summarized as follows.

a) *Bag Constrained Subgraph Mining:* The two-level weight updating mechanism seamlessly integrates the unique bag- and graph-level constraints into a repetitive and progressive mining process. It helps explore informative subgraphs to represent multi-graph bags.
b) *Implicit Feature Representation:* bMGC selects a subgraph to directly form a weak classifier in each iteration. This can efficiently tackle the challenge that no feature vectors are available for MGC.
c) *Generic Boosting Framework for MGC:* The proposed framework solves MGC by exploring informative subgraphs as weak classifiers to form a strong boosting model. The framework can be easily adjusted to accommodate other types of graph or bag classifiers for MGC.

VII. Experiments

A. DataSets

1) DBLP Multi-Graph Dataset: The DBLP dataset consists of bibliography data in computer science. We download a DBLP version called DBLP-Citation-network V5 from Arnetminer (http://arnetminer.org/citation). Each record in

TABLE I
DBLP DATASET USED IN EXPERIMENTS

Categories	Selected Conferences	# of Papers	# of Abstracts
AI	IJCAI,AAAI,NIPS,UAI,COLT, ACL,KR,ICML,ECML,IJCNN	800	4486
CV	ICCV, CVPR, ECCV, ICPR, ICIP ACM Multimedia, ICME	800	4206
DB	SIGMOD, PODS, VLDB, ICDE, CIKM DASFAA, ICDT, SSD, DASFAA,	800	4627

DBLP is associated with a number of attributes including title, abstract, author names, year, venue, and reference names etc. [49]. To build multi-graph bags, we select papers published in artificial intelligence (AI), computer vision (CV), and database (DB) fields to form MGC tasks. The goal is to predict which field a paper belongs to (AI, CV, or DB), by using abstracts of each paper and the abstracts of its references. For each abstract, a fuzzy cognitive map (E-FCM) [50] based approach is used to extract a number of keywords and correlations between keywords. In our experiments, we use keywords as nodes and correlations between two keywords as edge weight values to build a graph. A threshold (0.005) is used to remove edges whose correlation values are less than the threshold. At the last step, the graph is converted into an unweighted graph by setting the weight values of all remaining edges as 1. The similar graph representation was also used in previous works [51]–[54].

A conceptual view of building a multi-graph bag is shown in Fig. 1. Notice that AI, CV, and DB are overlapped in many aspects, such as machine learning, optimization and data mining, which make them challenging MGC tasks. The original DBLP dataset contains a significant number of papers without references. We choose 2400 papers, each of which containing one to ten references, to form two MGC tasks: DBLP (AI versus CV) with positive (AI) and negative (CV) bags, and DBLP (AI versus DB) with positive (AI) and negative (DB) bags. The last two columns in Table I report the number of bags (papers) and graphs (abstracts) in each category.

2) NCI Chemical Compound Multi-Graph Dataset: The NCI cancer screening database is a commonly used graph classification benchmark. We download two NCI datasets with ID 1 and 109 from PubChem (http://pubchem.ncbi.nlm.nih.gov). Each NCI dataset belongs to a bioassay task for anticancer activity prediction, where each chemical compound is represented as a graph, with atoms representing nodes and bonds denoting edges. A chemical compound is positive if it is active against the corresponding cancer, or negative otherwise. The original NCI datasets are highly imbalanced, with about 5% positive graphs, which is used to generate our multi-graph bags. To build multi-graph bags, we randomly select 1 to 4 positive graphs and several negative graphs to form a positive bag, and randomly select a number of negative graphs to form a negative bag. In order to address different targets, we design two NCI multi-graph classification tasks. One is NCI(1), which is generated from NCI dataset with ID 1, and the other is NCI(109), which is generated from NCI dataset with ID 109. The number of graphs in each bag may vary from 1 to 10.

TABLE II
NCI CANCER SCREEN DATASETS: NCI(1) AND NCI(109)

Dataset	ID	# of Bag	# of Positive Graphs	# of Negative Graphs
NCI(1)	1	800	1593	6614
NCI(109)	109	800	1600	6998

Table II summarizes the NCI(1) and NCI(109) datasets used in our experiments, where columns 4–5 show the numbers of positive and negative graphs in all multi-graph bags. In the NCI MGC, a bag of graphs can be regarded as a molecular group. Investigating the activity of a molecular group is meaningful in the bio-pharmaceutical field. Because labeling individual compounds is expensive and time-consuming, it is desirable to design effective methods (bMGC) to label molecular groups (i.e., bags).

B. Baseline Methods

To demonstrate the effectiveness of our MGC framework, we compare the proposed bMGC with both supervised and unsupervised bag constrained subgraph selection methods in the traditional MIL framework. The baseline methods are summarized as follows.

1) *Information Gain Based Approach (IG+MI):* In these methods, a set of frequent subgraphs are mined from graphs in all bags by using gSpan [34]. A supervised feature selection based on information gain (IG) is used to select m subgraphs with the highest IG scores. After obtaining the m subgraphs, IG based multi-instance approach (IG+MI) utilizes the selected subgraphs to represent graphs in bags, so a bag of graphs are converted into a bag of instances, through which the existing MIL methods can be applied for MGC learning.
2) *Top-k Based Approach (Topk+MI):* This is an unsupervised feature selection method which uses frequency as evaluation criterion to select subgraphs discovered by gSpan [34]. The Top-k subgraphs with the highest frequency from graphs in bags are selected. Top-k based multi-instance approach (Topk+MI) transforms each bag of graphs into a bag of instances for learning.

To compare our MGC framework bMGC's performance with MIL, two types of benchmark multi-instance classifiers, including boosting based (MIBoost and MIOptimallBall) and four different kinds of general approaches (CitationKNN, MIRI, MIEMDD, and MISMO), are used in our experiments. In the following, CitationKNN denotes a lazy learning based method, MIRI is an improvement of tree learning based approach, MIEMDD is an improved DD [26], and MISMO is an implementation of support vector machine for MIL. The baseline MIL methods used in our experiments and their abbreviations are listed as follows.

1) Boosting for MI Learning Approaches.

a) MIBoost is an algorithm [24] inspired by AdaBoost that builds a series of weak classifiers (decision stump is used

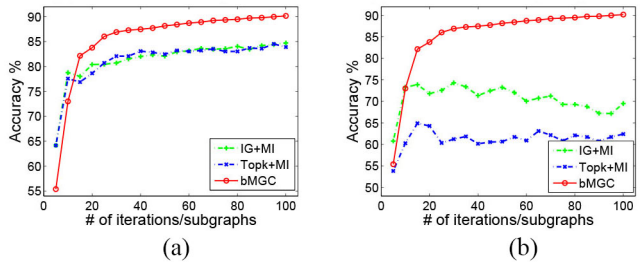

Fig. 4. Accuracy on DBLP(AI versus CV) by using proposed bMGC and boosting based MI learning methods. (a) MIBoost. (b) MIOptimalBall.

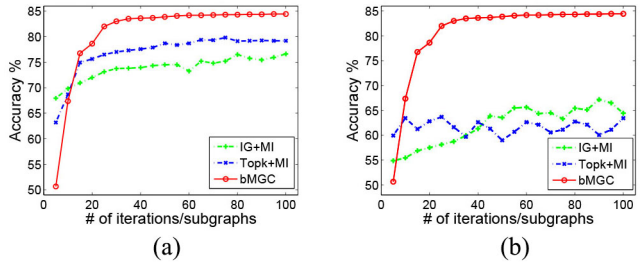

Fig. 5. Accuracy on DBLP(AI versus DB) by using proposed bMGC and boosting based MI learning methods. (a) MIBoost. (b) MIOptimalBall.

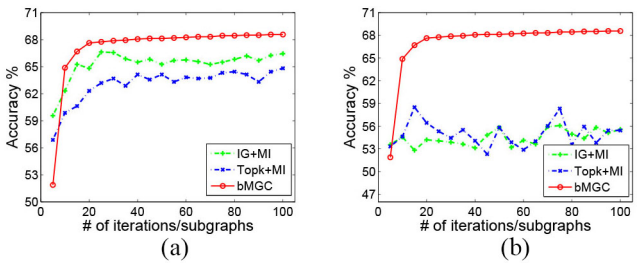

Fig. 6. Accuracy on NCI(1) by using proposed bMGC and boosting based MI learning methods. (a) MIBoost. (b) MIOptimalBall.

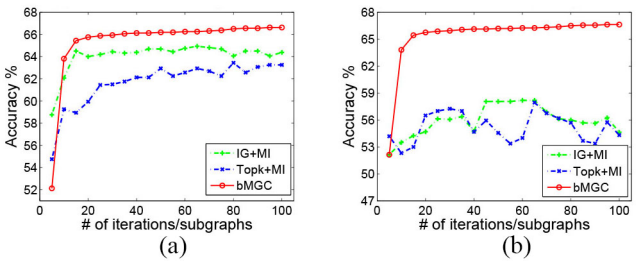

Fig. 7. Accuracy on NCI(109) by using proposed bMGC and boosting based MI learning methods. (a) MIBoost. (b) MIOptimalBall.

in our experiment) using a single instance learner based on appropriately reweighted versions of the input data.

b) MIOptimalBall treats the weak hypotheses for AdaBoost as balls [28] and the classification is based on the distance to a reference point. More specifically, this method attempts to find a ball in the instance space so that all instances of all negative bags are outside the ball and at least one instance of each positive bag is inside the ball.

2) General MI Learning Approaches.

a) CitationKNN, a nearest-neighbor-based approach, measures the distance between bags using Hausdorff distance [16]. The nearest neighbor example to be classified is the one nearest to both references and citers.

b) MIEMDD is the EM version of DD with the most-likely-cause model [26], which is used to find the most likely target points based on the DD model that has been learned [27].

c) MIRI is a multi-instance classifier that utilizes partial MITI trees [17] with a single positive leaf to learn and represent rules. MIRI [18] is a simple modification to MITI to yield a rule learner for MIL.

d) MISMO constructs a support vector machine classifier for multi-instance data [21], where the standard sequential minimization algorithm is used for support vector learning in conjunction with an MI kernel as described in [55].

C. Experiment Settings

In our experiments, all reported results are based on 10 times 10-fold cross-validation with classification accuracy being used as the performance metrics. Unless specified otherwise, the default parameter settings are as follows: minimum support threshold $min\_sup = 4\%$ for DBLP datasets and $min\_sup = 15\%$ for NCI datasets. All the above classifiers for traditional MIL utilize the versions provided in WEKA machine learning workbench [56], with default parameter settings. Besides, all experiments are conducted on a Linux cluster computing node with an Interl(R) Xeon(R) @3.33GHZ CPU and 3GB memory.

D. Accuracy on Multi-Graph Classification

In this section, we report experimental results on DBLP and NCI datasets, by comparing the performance of bMGC with two types of MIL methods, including boosting based and general approaches under the supervised and unsupervised feature selection settings respectively. All methods are compared by using the same number of subgraphs. For our boosting based bMGC, one subgraph is selected in each iteration until the total number reaches m, whereas for baseline methods, a number of m subgraphs are selected in one time. As expected, bMGC clearly outperforms existing traditional MIL methods on both DBLP and NCI multi-graph datasets with different number of subgraphs (varying from 1 to 100).

1) bMGC Versus Boosting for MI Learning Approaches: We compare bMGC to MIBoost and MIOptimalBall, where the two boosting based baselines are two variants of the well known AdaBoost algorithm [25] with the objective of minimizing the exponential loss for bags of instances. Like other boosting schemes, these two algorithms greedily fit an additive model to the training data. In each iteration of the sequential boosting process, a weak learner (a decision stump for MIBoost, and a ball for MIOptimalBall) is applied to generate one component of the underlying additive model.

Results in Figs. 4(a) to 7(a) show that both bMGC and MIBoost can achieve a high accuracy on DBLP (AI versus

TABLE III
PAIRWISE t-TEST RESULT OF BMGC VERSUS BOOSTING BASED MI LEARNING METHODS ON (a) DBLP AND (b) NCI DATASETS. A, B, AND C DENOTE BMGC, IG+MI, AND TOPK+MI, RESPECTIVELY. $\mathcal{H}_1$ AND $\mathcal{H}_2$ DENOTE MIBOOST AND MIOPTIMALBALL, RESPECTIVELY

(a) t-test on DBLP dataset

DBLP(AI vs CV)				DBLP(AI vs DB)			
$\mathcal{H}$	A-B	A-C	B-C	$\mathcal{H}$	A-B	A-C	B-C
$\mathcal{H}_1$	1.65E-04	7.83E-05	**4.35E-01**	$\mathcal{H}_1$	9.97E-05	6.54E-04	6.53E-06
$\mathcal{H}_2$	1.73E-08	1.74E-12	5.21E-13	$\mathcal{H}_2$	2.15E-11	2.39E-09	**6.02E-01**

(b) t-test on NCI dataset

NCI(1)				NCI(109)			
$\mathcal{H}$	A-B	A-C	B-C	$\mathcal{H}$	A-B	A-C	B-C
$\mathcal{H}_1$	1.87E-03	1.03E-07	3.85E-10	$\mathcal{H}_1$	5.79E-03	8.21E-09	1.53E-08
$\mathcal{H}_2$	1.84E-12	8.63E-12	**2.08E-01**	$\mathcal{H}_2$	9.38E-13	1.94E-11	**6.80E-02**

CV, AI versus DB) and NCI (1, and 109) datasets. Meanwhile, bMGC consistently outperforms MIBoost when the number of selected subgraphs is 20 or more. On the other hand, comparing our bMGC with MIOptimallBall, significant performance gain can be observed in Figs. 4(b) to 7(b) on both datasets. The superior performance of bMGC is due to the optimal subgraph mining strategy combined with AdaBoost and TrAdaBoost algorithms. Further more, it seems that MIOptimallBall fails to adapt to the feature space composed of subgraphs.

Our results also show that bMGC has a very low accuracy in early iterations, and its accuracy may be worse than baselines such as MIBoost in some cases. This is mainly because that the boosting model of bMGC relies on weak classifiers to achieve better performance. When the number of weak classifiers is small (normally happens at the early stage of the boosting process), the accuracy of bMGC is noticeably low. In order to show that this situation will not affect the performance of bMGC, we summarize the pairwise t-test results (with confidence level $\alpha = 0.05$) of bMGC and boosting MI learning methods on both datasets in Table III. Each entry (value) denotes the p-value for a t-test between two algorithms, and a p-value less than $\alpha = 0.05$ indicates that the difference is statistically significant. From Table III, bMGC statistically outperforms boosting based MI learning baselines in all cases.

2) bMGC Versus General MI Learning Approaches: We carry out another experimental comparison to demonstrate the performance of bMGC, with other four different types of general MI learning approaches (CitationKNN, MIRI, MIEMDD and MISMO). From the results in Figs. 8(c) to 11(c), MIEMDD shows ineffective performance for MGC, and increasing number of subgraphs cannot result in additional accuracy gain.

Although the performance of CitationKNN, MIRI, and MISMO based methods improve as the number of subgraphs increases, they still cannot reach the best performance achieved by bMGC except for IG+MIRI on NCI(109) dataset as shown in Fig. 11(b). It is worth mentioning that bMGC may achieve comparable performances over other baselines in some cases, such as Topk+CitationKNN [Fig. 9(a)] and MISMO [Figs. 8(d) and 9(d)] on DBLP dataset, IG+MISMO [Figs. 10(d) and 11(d)] on NCI dataset.

To further validate the statistical performance of bMGC, in Table IV, we also report the pairwise t-test to validate the statistical significance between two methods. From Table IV, bMGC statistically outperforms general MI learning baselines in all cases. This is mainly attributed to the effectiveness of the proposed bag constrained subgraph exploration criterion and the specially designed boosting strategy, which weights a set of single weak classifiers under our specially designed weighting mechanism.

E. Effectiveness of Subgraph Candidate Generation in bMGC

As discussed above, one main component of bMGC is the utilization of subgraph candidate generation (as described in Section IV). More specifically, in addition to aggregating graphs in all bags $\mathcal{G}$, we also aggregate: 1) graphs in all positive bags $\mathcal{G}^+$, and 2) graphs in all negative bags $\mathcal{G}^-$. As a result, a set of diverse subgraph candidate patterns can be discovered for validation. In order to further illustrate the effectiveness of the proposed strategy for subgraph candidate generation and validate whether using the two extra graph sets $\mathcal{G}^+$ and $\mathcal{G}^-$ can indeed improve the performance of bMGC, we compare bMGC with an approach which only uses the $\mathcal{G}$ to generate the subgraphs for learning, namely bMGC-G. In Fig. 12(a) and (b), we report the accuracy with respect to different iterations on DBLP (AI versus CV) and NCI(1) datasets, respectively. The results show that the classification accuracy of bMGC using all three graph sets is normally 3%–5% higher than bMGC-G which only uses the $\mathcal{G}$. This is due to the fact that the separation of graphs into $\mathcal{G}^+$ and $\mathcal{G}^-$ can help find some unique subgraph patterns, which do not appear in the whole graph set $\mathcal{G}$. Indeed, because the subgraph exploration essentially relies on a threshold (i.e., the support value) to discover frequent subgraphs. When aggregating all graphs in one set $\mathcal{G}$, it is possible that a good subgraph in $\mathcal{G}^+$ may not be discovered from $\mathcal{G}$, simply because the frequency of the subgraph is below the given threshold in $\mathcal{G}$. The separation of graphs into three sets $\mathcal{G}^+$, $\mathcal{G}^-$, and $\mathcal{G}$ will therefore help discover a rich set of subgraph candidates, through which bMGC can find the ones with the highest informativeness scores.

F. Convergence Study

Fig. 13 reports the error rate curves of bMGC in terms of the number of iterations on four multi-graph datasets. The curves are quite smooth, but converge well, which is consistent with the theoretical analysis and the existing observations from Adaboost [25]. The error rates of bMGC, after the algorithm reaches the convergence, are higher on DBLP datasets than on the NCI datasets. Overall, bMGC on all four datasets receives a fast convergence speed.

For NCI datasets, the convergence is reached within ten iterations, whereas for DBLP datasets, bMGCs convergence

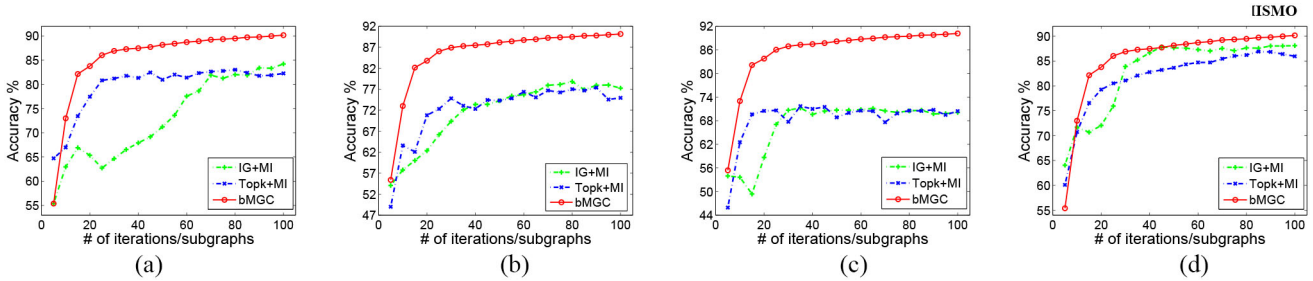

Fig. 8. Accuracy on DBLP(AI versus CV) by using bMGC and generic MI learning methods. (a) CitationKNN. (b) MIRI. (c) MIEMDD. (d) MISMO.

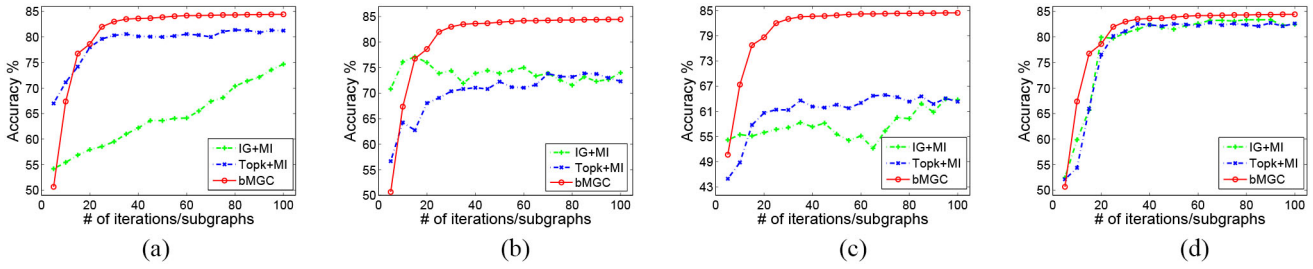

Fig. 9. Accuracy on DBLP(AI versus DB) by using bMGC and generic MI learning methods. (a) CitationKNN. (b) MIRI. (c) MIEMDD. (d) MISMO.

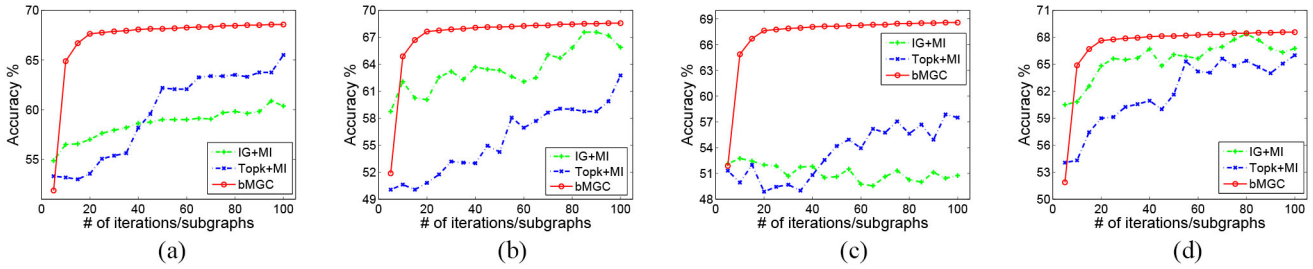

Fig. 10. Accuracy on NCI(1) by using bMGC and generic MI learning methods. (a) CitationKNN. (b) MIRI. (c) MIEMDD. (d) MISMO.

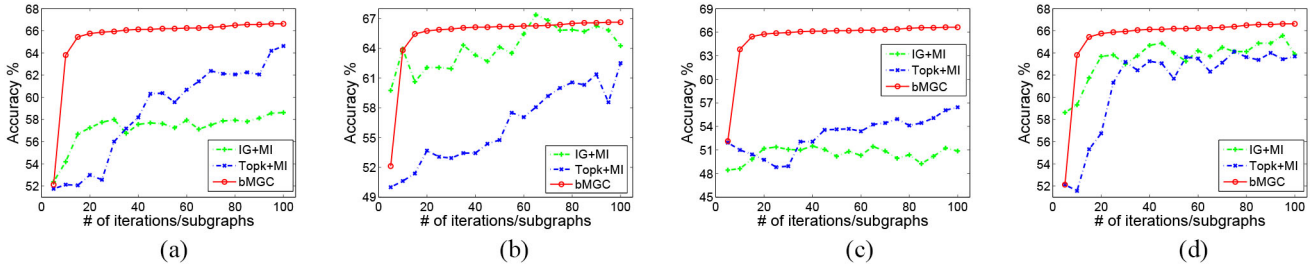

Fig. 11. Accuracy on NCI(109) by using bMGC and generic MI learning methods. (a) CitationKNN. (b) MIRI. (c) MIEMDD. (d) MISMO.

TABLE IV
PAIRWISE t-TEST RESULT OF BMGC VERSUS GENERAL MI LEARNING METHODS ON (a) DBLP AND NCI DATASETS. A, B, AND C DENOTE BMGC, IG+MI, AND TOPK+MI, RESPECTIVELY. $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$, AND $\mathcal{H}_4$ DENOTE CITATIONKNN, MIRI, MIEMDD, AND MISMO, RESPECTIVELY

(a) t-test on DBLP dataset

DBLP(AI vs CV)				DBLP(AI vs DB)			
$\mathcal{H}$	A-B	A-C	B-C	$\mathcal{H}$	A-B	A-C	B-C
$\mathcal{H}_1$	5.73E-08	9.41E-07	2.10E-04	$\mathcal{H}_1$	4.71E-10	6.76E-03	3.31E-11
$\mathcal{H}_2$	1.97E-11	2.53E-15	**4.76E-01**	$\mathcal{H}_2$	1.12E-03	2.24E-09	3.63E-03
$\mathcal{H}_3$	3.95E-12	3.04E-15	**2.66E-01**	$\mathcal{H}_3$	2.97E-11	4.60E-16	5.08E-03
$\mathcal{H}_4$	2.01E-02	3.99E-07	**1.18E-01**	$\mathcal{H}_4$	3.64E-03	2.38E-03	**1.25E-01**

(b) t-test on NCI dataset

NCI(1)				NCI(109)			
$\mathcal{H}$	A-B	A-C	B-C	$\mathcal{H}$	A-B	A-C	B-C
$\mathcal{H}_1$	2.96E-11	1.59E-07	**1.44E-01**	$\mathcal{H}_1$	2.11E-13	2.95E-07	**7.03E-02**
$\mathcal{H}_2$	4.68E-05	3.45E-11	4.99E-12	$\mathcal{H}_2$	3.52E-02	3.90E-10	3.10E-11
$\mathcal{H}_3$	4.42E-13	5.08E-12	1.19E-02	$\mathcal{H}_3$	8.64E-16	1.66E-12	2.06E-04
$\mathcal{H}_4$	1.49E-02	2.79E-07	1.03E-07	$\mathcal{H}_4$	9.54E-04	5.21E-06	7.87E-04

is reached after 20 or more iterations. Notice that each weak classifier in bMGC denotes one subgraph, this indicates that more subgraph features are needed in order to differentiate the object classes in the DBLP dataset. Indeed, because DBLP tasks involve overlapping domains (such as AI versus CV), using more subgraph features (which correspond to keywords

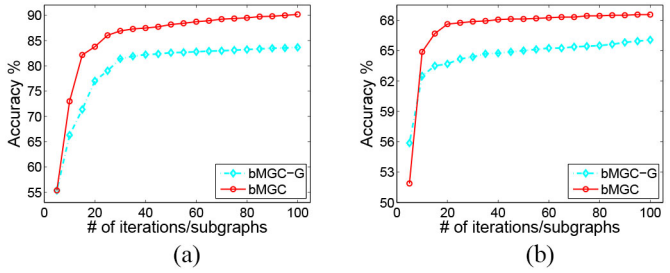

Fig. 12. Accuracy comparisons by using bMGC and bMGC-G on DBLP and NCI datasets, respectively. (a) DBLP (AI versus CV) dataset. (b) NCI (1) dataset.

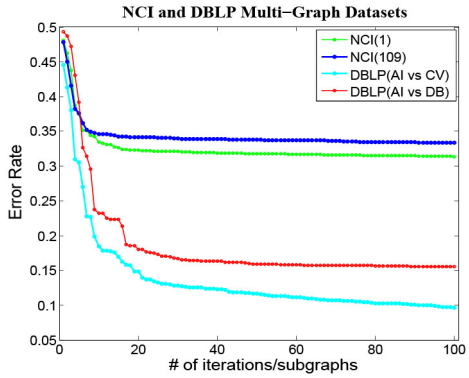

Fig. 13. Error rate curves on DBLP (AI versus CV, AI versus DB) and NCI(1 and 109) multi-graph datasets in terms of the number of iterations.

and their correlations) can constantly help improve the classification accuracy. For NCI graphs, the positive versus negative graphs are mostly separated by some unique subgraph features. So as long as such unique patterns are discovered, the algorithm can quickly converge.

G. Effectiveness Results

To evaluate the effectiveness of the pruning module of bMGC in reducing the search space (as described in Section V-C), we compare bMGC with an approach which does not have pruning module in the subgraph search space (denoted by ubMGC). In our implementation, ubMGC first uses gSpan to find a set of frequent subgraphs, and then selects the optimal subgraph by using the same criteria as bMGC in each iteration. In Fig. 14(a) and (b), we report the average CPU runtime with respect to different minimum support values *min_sup* (the number of selected subgraphs is fixed to 100) on DBLP(AI versus CV) and NCI(1) datasets, respectively. The results show that as the *min_sup* values increase, the runtime of both pruning and unpruning bMGC decrease, this is mainly because a larger *min_sup* value will reduce the number of candidates for validation. Accordingly, by incorporating the proposed pruning strategy, bMGC can improve the runtime performance. The reason is that the bScore upper bound of bMGC can effectively help prune the subgraph search space without decreasing the quality of classification.

VIII. Discussion

In this paper, we focus on using subgraph based boosting framework for MGC. Indeed, the idea of exploiting subgraphs

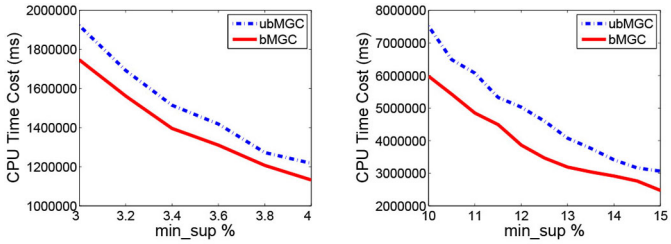

Fig. 14. Average CPU runtime for bMGC versus unpruned ubMGC with different *min_sup* under a fixed number of subgraphs $m = 100$ on DBLP and NCI datasets, respectively.

for graph classification has been studied in a number of existing works, including a recent ensemble based semi-supervised graph stream classification approach [9]. The core of the proposed bMGC approach is to combine two types of boosting strategies: AdaBoost [25] for bag-level boosting and TrAdaBoost [48] for graph-level boosting, to integrate graph- and bag-level learning for MGC. Boosting algorithms for graph classification have already been studied in several previous works. For example, Kudo *et al.* [41] proposes an AdaBoost based graph classification approach, which is the original algorithm among many variants [42]–[44]. Meanwhile, LPBoost [57], namely linear programming Boosting, is another type of boosting algorithm for graph classification. The proposed bMGC follows similar subgraph search approaches as used in these existing works. For bMGC, it uses gSpan algorithm [34] in each boosting iteration, together with the proposed pruning strategy, to explore subgraphs.

The main complication of MGC is that the genuine labels of graphs inside a positive bag are unknown. To tackle uncertainty inside positive bags, bMGC takes the bag constraints into consideration and explores subgraphs to represent graphs with maximum diversity, as defined in (2). This is similar to the way of handling unlabeled graphs in an existing semi-supervised graph stream classification method [9]. In [9], an instance weighting mechanism has also been proposed but is different from the weighting approach in bMGC, where the weights are directly associated to the graphs and bags. In addition, the weight updating strategy in [9] is based on AdaBoost [25], which only considers labeled graphs. In bMGC, we borrow the weighting strategy from TrAdaBoost [48] to update the graph weighs in both labeled and unlabeled graph sets. In summary, the idea in [9] provides inspirations to motivate the proposed MGC design.

We believe that the proposed bMGC opens a new opportunity to expand existing MIL to increasingly popular graph applications. Although bMGC proposes to use subgraph mining to tackle the MGC challenges, the principle of combining graph and bag level constraints can be extended to many other types of approaches to handle MGC problems. For example, for kernel based methods, MGC problem can be solved by two subtasks: 1) add multi-graph constraints to traditional graph kernel, and 2) propose a new multi-graph kernel framework. In addition, one can also impose multi-graph constrains to graph embedding methods (e.g., the one in [32]) to directly calculate the distance between two graphs or between two

graph bags. With the calculated distances between graphs and between bags, standard learning algorithms (including MIL algorithms) can be applied to solve MGC tasks.

IX. Conclusion

In this paper, we investigated a novel MGC problem, in which a number of graphs form a bag, with each bag being labeled as either positive or negative. Multi-graph representation can be used to represent many real-world applications, where label is only available for a bag of objects with dependency structures. To build a learning model for MGC, we proposed a bMGC, which employs dynamic weight adjustment, at both graph- and bag-levels, to select one subgraph in each iteration to form a set of weak graph classifiers. The MGC is achieved by using weighted combination of weak graph classifiers. Experiments on two real-world MGC tasks, including DBLP citation network and NCI chemical compound classification, demonstrate that our method is effective in finding informative subgraph, and its accuracy is significantly better than baseline methods.

Appendix

Proof of the Theorem 1

According to (8), for any $g'_k \supseteq g_k$ we have

$$\begin{aligned} r\left(g'_k\right) &= \boldsymbol{f}_{g'_k}^{\top} \boldsymbol{Q} \boldsymbol{f}_{g'_k} \\ &= \begin{bmatrix} \left(\boldsymbol{f}^B_{g'_k}\right)^{\top} & \left(\boldsymbol{f}^G_{g'_k}\right)^{\top} \end{bmatrix} \begin{bmatrix} Q_B & 0 \\ 0 & Q_G \end{bmatrix} \begin{bmatrix} \boldsymbol{f}^B_{g'_k} \\ \boldsymbol{f}^G_{g'_k} \end{bmatrix} \\ &= \left(\boldsymbol{f}^B_{g'_k}\right)^{\top} Q_B \boldsymbol{f}^B_{g'_k} + \left(\boldsymbol{f}^G_{g'_k}\right)^{\top} Q_G \boldsymbol{f}^G_{g'_k} \\ &= \sum_{i,j:B_i,B_j \in \mathcal{B}(g'_k)} Q^B_{ij} + \sum_{i,j:G_i,G_j \in \mathcal{G}(g'_k)} Q^G_{ij} \end{aligned} \tag{11}$$

where $\mathcal{B}(g'_k) \triangleq \{B_i | g'_k \subseteq G_j \in B_i, 1 \le i \le p, 1 \le j \le q\}$ and $\mathcal{G}(g'_k) \triangleq \{G_j | g'_k \subseteq G_j, 1 \le j \le q\}$. Since g'_k is the supergraph of g_k (i.e., $g'_k \supseteq g_k$), according to the anti-monotonic property, we have $\mathcal{B}(g'_k) \subseteq \mathcal{B}(g_k)$ and $\mathcal{G}(g'_k) \subseteq \mathcal{G}(g_k)$

$$\begin{aligned} r\left(g'_k\right) &= \sum_{i,j:B_i,B_j \in \mathcal{B}(g'_k)} Q^B_{ij} + \sum_{i,j:G_i,G_j \in \mathcal{G}(g'_k)} Q^G_{ij} \\ &\le \sum_{i,j:B_i,B_j \in \mathcal{B}(g'_k)} \hat{Q}^B_{ij} + \sum_{i,j:G_i,G_j \in \mathcal{G}(g'_k)} \hat{Q}^G_{ij} \\ &\le \sum_{i,j:B_i,B_j \in \mathcal{B}(g_k)} \hat{Q}^B_{ij} + \sum_{i,j:G_i,G_j \in \mathcal{G}(g_k)} \hat{Q}^G_{ij} \\ &= \left(\boldsymbol{f}^B_{g_k}\right)^{\top} \hat{Q}_B \boldsymbol{f}^B_{g_k} + \left(\boldsymbol{f}^G_{g_k}\right)^{\top} \hat{Q}_G \boldsymbol{f}^G_{g_k} \\ &= \boldsymbol{f}_{g_k}^{\top} \hat{\boldsymbol{Q}} \boldsymbol{f}_{g_k} = \hat{r}(g_k). \end{aligned} \tag{12}$$

Thus, for any $g'_k \supseteq g_k$, $r(g'_k) \le \hat{r}(g_k)$.

References

[1] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis, "Frequent substructure-based approaches for classifying chemical compounds," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 8, pp. 1036–1050, Aug. 2005.

[2] W. Lian, D.-L. Cheung, N. Mamoulis, and S.-M. Yiu, "An efficient and scalable algorithm for clustering XML documents by structure," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 1, pp. 82–96, Jan. 2004.

[3] C. Chen *et al.*, "Mining graph patterns efficiently via randomized summaries," in *Proc. 35th Int. Conf. VLDB*, Lyon, France, 2009, pp. 742–753.

[4] H. Wang, H. Huang, and C. Ding, "Image categorization using directed graphs," in *Proc. 11th ECCV*, Crete, Greece, 2010, pp. 762–775.

[5] R. Angelova and G. Weikum, "Graph-based text classification: Learn from your neighbors," in *Proc. 29th Annu. Int. ACM SIGIR*, Seattle, WA, USA, 2006, pp. 485–492.

[6] Z. Harchaoui and F. Bach, "Image classification with segmentation graph kernels," in *Proc. 20th IEEE Conf. CVPR*, Minneapolis, MN, USA, 2007, pp. 1–8.

[7] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," in *Proc. 1st ICDM*, 2001, pp. 313–320.

[8] M. Thoma *et al.*, "Near-optimal supervised feature selection among frequent subgraphs," in *Proc. 9th SDM*, 2009, pp. 1075–1086.

[9] S. Pan, X. Zhu, C. Zhang, and P. Yu, "Graph stream classification using labeled and unlabeled graphs," in *Proc. 29th IEEE ICDE*, Brisbane, QLD, USA, 2013, pp. 398–409.

[10] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, nos. 1–2, pp. 273–324, 1997.

[11] T. Dietterich, R. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artif. Intell.*, vol. 89, no. 1–2, pp. 31–71, 1997.

[12] Z. Fu, A. Robles-Kelly, and J. Zhou, "MILIS: Multiple instance learning with instance selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 958–977, May 2011.

[13] Z.-H. Zhou, K. Jiang, and M. Li, "Multi-instance learning based web mining," *Appl. Intell.*, vol. 22, no. 2, pp. 135–147, 2005.

[14] D. Kelly, J. McDonald, and C. Markham, "Weakly supervised training of a sign language recognition system using multiple instance learning density matrices," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 526–541, Apr. 2011.

[15] Z. Zhou, M. Zhang, S. Huang, and Y. Li, "Multi-instance multi-label learning," *Artif. Intell.*, vol. 176, no. 1, pp. 2291–2320, 2012.

[16] J. Wang, "Solving the multiple-instance problem: A lazy learning approach," in *Proc. 17th ICML*, San Francisco, CA, USA, 2000, pp. 1119–1125.

[17] H. Blockeel and A. Srinivasan, "Multi-instance tree learning," in *Proc. 22th ICML*, Bonn, Germany, 2005, pp. 57–64.

[18] L. Bjerring and E. Frank, "Beyond trees: Adopting MITI to learn rules and ensemble classifiers for multi-instance data," in *Proc. 24th Int. Conf. Adv. AI*, Berlin, Heidelberg, 2011, pp. 41–50.

[19] Y. Chevaleyre and J. Zucker, "A framework for learning rules from multiple instance data," in *Proc. 12th ECML*, Freiburg, Germany, 2001, pp. 49–60.

[20] M. Zhang and Z. Zhou, "Improve multi-instance neural networks through feature selection," *Neural Process. Lett.*, vol. 19, no. 1, pp. 1–10, 2004.

[21] X. Qi and Y. Han, "Incorporating multiple SVMs for automatic image annotation," *Pattern Recogn.*, vol. 40, no. 2, pp. 728–741, 2007.

[22] S. Ray and M. Craven, "Supervised versus multiple instance learning: An empirical comparison," in *Proc. 22nd ICML*, New York, NY, USA, 2005, pp. 697–704.

[23] H. Yuan, M. Fang, and X. Zhu, "Hierarchical sampling for multi-instance ensemble learning," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 12, pp. 2900–2905, Dec. 2013.

[24] X. Xu and E. Frank, "Logistic regression and boosting for labeled bags of instances," in *Proc. 8th PAKDD*, 2004, pp. 272–281.

[25] M. Telgarsky, "A primal-dual convergence analysis of boosting," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 561–606, 2012.

[26] O. Maron and T. Lozano-Pérez, "A framework for multiple-instance learning," in *Proc. 12th Annu. Conf. NIPS*, Cambridge, MA, USA, 1998, pp. 570–576.

[27] Q. Zhang and S. Goldman, "EM-DD: An improved multiple-instance learning technique," in *Proc. 15th Annu. Conf. NIPS*, 2001, pp. 1073–1080.

[28] P. Auer and R. Ortner, "A boosting approach to multiple instance learning," in *Proc. 15th ECML*, Pisa, Italy, 2004, pp. 63–74.

[29] J. Wu, X. Zhu, C. Zhang, and Z. Cai, "Multi-instance multi-graph dual embedding learning," in *Proc. 13th ICDM*, Dallas, TX, USA, 2013, pp. 827–836.

[30] S. V. N. Vishwanathan, K. M. Borgwardt, R. I. Kondor, and N. N. Schraudolph, "Graph kernels," *J. Mach. Learn. Res.*, vol. 11, pp. 1201–1242, Apr. 2008.

[31] P. Mahe, N. Ueda, T. Akutsu, J. Pettet, and J. Vert, "Extensions of marginalized graph kernels," in *Proc. 21st ICML*, New York, NY, USA, 2004, pp. 552–559.
[32] K. Riesen and H. Bunke, "Graph classification by means of Lipschitz embedding," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1472–1483, Dec. 2009.
[33] K. Riesen and H. Bunke, *Graph Classification and Clustering Based on Vector Space Embedding*. Singapore: World Scientific, 2010.
[34] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *Proc. 2nd ICDM*, Washington, DC, USA, 2002, pp. 721–724.
[35] A. Inokuchi, T. Washio, and H. Motoda, "An apriori-based algorithm for mining frequent substructures from graph data," in *Proc. 4th Eur. Conf. PKDD*, Lyon, France, 2000, pp. 13–23.
[36] C. Borgelt and M. Berthold, "Mining molecular fragments: Finding relevant substructures of molecules," in *Proc. 2nd ICDM*, 2002, pp. 51–58.
[37] S. Nijssen and J. Kok, "A quickstart in frequent structure mining can make a difference," in *Proc. 10th ACM SIGKDD*, Seattle, WA, USA, 2004, pp. 647–652.
[38] X. Yan, H. Cheng, J. Han, and P. S. Yu, "Mining significant graph patterns by leap search," in *Proc. 27th ACM SIGMOD*, Vancouver, BC, Canada, 2008, pp. 433–444.
[39] H. Saigo, N. Krämer, and K. Tsuda, "Partial least squares regression for graph mining," in *Proc. 14th ACM SIGKDD*, Las Vegas, NV, USA, 2008, pp. 578–586.
[40] N. Jin, C. Young, and W. Wang, "GAIA: Graph classification using evolutionary computation," in *Proc. 29th ACM SIGMOD*, Indianapolis, IN, USA, 2010, pp. 879–890.
[41] T. Kudo, E. Maeda, and Y. Matsumoto, "An application of boosting to graph classification," in *Proc. 18th Annu. Conf. NIPS*, 2004, pp. 729–736.
[42] S. Nowozin, K. Tsuda, T. Uno, T. Kudo, and G. Bakir, "Weighted substructure mining for image analysis," in *Proc. 20th IEEE Conf. CVPR*, Minneapolis, MN, USA, 2007, pp. 1–8.
[43] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda, "gBoost: A mathematical programming approach to graph classification and regression," *Mach. Learn.*, vol. 75, no. 1, pp. 69–89, 2009.
[44] S. Pan and X. Zhu, "Graph classification with imbalanced class distributions and noise," in *Proc. 23rd IJCAI*, 2013, pp. 1586–1592.
[45] H. Fei and J. Huan, "Boosting with structure information in the functional space: An application to graph classification," in *Proc. 16th ACM SIGKDD*, Washington, DC, USA, 2010, pp. 643–652.
[46] B. Zhang *et al.*, "Multi-class graph boosting with subgraph sharing for object recognition," in *Proc. 20th ICPR*, Istanbul, Turkey, 2010, pp. 1541–1544.
[47] M. Grbovic, C. Dance, and S. Vucetic, "Sparse principal component analysis with constraints," in *Proc. 26th Conf. AAAI*, 2012, pp. 935–941.
[48] W. Dai, Q. Yang, G. Xue, and Y. Yu, "Boosting for transfer learning," in *Proc. 24th ICML*, Corvallis, OR, USA, 2007, pp. 193–200.
[49] J. Tang *et al.*, "ArnetMiner: Extraction and mining of academic social networks," in *Proc. 14th ACM SIGKDD*, Las Vegas, NV, USA, 2008, pp. 990–998.
[50] K. Perusich and M. McNeese, "Using fuzzy cognitive maps for knowledge management in a conflict environment," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 36, no. 6, pp. 810–821, Nov. 2006.
[51] X. L. Q. Hu, W. Xu, and Z. Yu, "Discovery of textual knowledge flow based on the management of knowledge maps," *Concurr. Comput. Pract. Exp.*, vol. 20, no. 15, pp. 1791–1806, 2008.
[52] X. Luo, Z. Xu, J. Yu, and X. Chen, "Building association link network for semantic link on web resources," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 3, pp. 482–494, Jul. 2011.
[53] J. Wu *et al.*, "Multi-graph learning with positive and unlabeled bags," in *Proc. 14th SIAM Int. Conf. Data Mining*, 2014, pp. 217–225.
[54] J. Wu, X. Zhu, C. Zhang, and P. Yu, "Bag constrained structure pattern mining for multi-graph classification," *IEEE Trans. Knowl. Data Eng.*, to be published.
[55] T. Gartner, P. A. Flach, A. Kowalczyk, and A. J. Smola, "Multi-instance kernels," in *Proc. 19th ICML*, 2002, pp. 179–186.
[56] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Amsterdam, The Netherlands: Morgan Kaufmann, 2005.
[57] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *Mach. Learn.*, vol. 46, no. 1–3, pp. 225–254, 2002.

Jia Wu (S'14) received the bachelor's degree in computer science from the China University of Geosciences, Wuhan, China, in 2009, where he is currently pursuing the Ph.D. degree in computer science. He is also pursuing the Ph.D. degree from the Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney, Ultimo, NSW, Australia.

His current research interests include data mining and machine learning.

Shirui Pan received the master's degree in computer science from Northwest A&F University, Yangling, China, in 2011. He is currently pursuing the Ph.D. degree from the Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney, Ultimo, NSW, Australia.

His current research interests include data mining and machine learning.

Xingquan Zhu (SM'12) received the Ph.D. degree in computer science from Fudan University, Shanghai, China.

He is an Associate Professor with the Department of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL, USA. Prior to that, he was with the Centre for Quantum Computation and Intelligent Systems, University of Technology, Sydney, Ultimo, NSW, Australia. His current research interests include data mining, machine learning, and multimedia systems. Since 2000, he has published over 170 refereed journal and conference papers in these areas.

Dr. Zhu is an Associate Editor of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING during (2014 - date), and is currently serving on the Editorial Board of the international journal of *Social Network Analysis and Mining*, since 2010, and *Network Modeling Analysis in Health Informatics* and *Bioinformatics* journal, since 2014. He served or is serving as a Program Committee Co-Chair for the 14th IEEE International Conference on Bioinformatics and BioEngineering (BIBE-2014), the IEEE International Conference on Granular Computing (GRC-2013), the 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI-2011), and the 9th International Conference on Machine Learning and Applications (ICMLA-2010). He also served as a Conference Co-Chair for ICMLA-2012. He was a recipient of two Best Paper Awards and one Best Student Paper Award.

Zhihua Cai received the B.Sc. degree from Wuhan University, Wuhan, China, in 1986, the M.Sc. degree from the Beijing University of Technology, Beijing, China, in 1992, and the Ph.D. degree from the China University of Geosciences, Wuhan, in 2003.

He is currently a faculty member with the School of Computer Science, China University of Geosciences. He has published over 50 research papers in journals and international conferences, such as IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON CYBERNETICS, *Applied Soft Computing*, *Information Sciences*, *Knowledge-Based Systems*, and *Knowledge and Information Systems*. His current research interests include data mining, machine learning, evolutionary computation, and their applications.