Joint Structure Feature Exploration and Regularization for Multi-Task Graph Classification

Shirui Pan, Jia Wu, Xingquan Zhu, *Senior Member, IEEE*,
Chengqi Zhang, *Senior Member, IEEE*, and Philip S. Yu, *Fellow, IEEE*

Abstract—Graph classification aims to learn models to classify structure data. To date, all existing graph classification methods are designed to target one single learning task and require a large number of labeled samples for learning good classification models. In reality, each real-world task may only have a limited number of labeled samples, yet multiple similar learning tasks can provide useful knowledge to benefit all tasks as a whole. In this paper, we formulate a new multi-task graph classification (MTG) problem, where multiple graph classification tasks are jointly regularized to find discriminative subgraphs shared by all tasks for learning. The niche of MTG stems from the fact that with a limited number of training samples, subgraph features selected for one single graph classification task tend to overfit the training data. By using additional tasks as evaluation sets, MTG can jointly regularize multiple tasks to explore high quality subgraph features for graph classification. To achieve this goal, we formulate an objective function which combines multiple graph classification tasks to evaluate the informativeness score of a subgraph feature. An iterative subgraph feature exploration and multi-task learning process is further proposed to incrementally select subgraph features for graph classification. Experiments on real-world multi-task graph classification datasets demonstrate significant performance gain.

Index Terms—Graph classification, subgraph features, regularization, multi-task learning, supervised learning

1 Introduction

Recent years have witnessed a wide range of applications that involve learning and classifying objects with structure dependence and complex relationships, where each object is represented as a graph with node-edge representation. Typical graph applications include predicting biological activity of molecules [1], identifying errors in computer programs [2], and categorizing scientific publications [3]. Unlike traditional vector data, graphs are only characterized by node-edge representation and no features are readily available for training prediction models. This challenge has motivated numerous studies on graph classification [4], [5], [6], [7], where existing methods either try to learn global similarities between two graphs [6], or select local discriminative subgraphs [1], [7] as features to transfer graphs into feature-vector format so that traditional machine learning algorithms can be applied.

- *S. Pan, J. Wu, and C. Zhang are with the Centre for Quantum Computation and Intelligent Systems, FEIT, University of Technology Sydney, N.S.W. 2007, Australia. E-mail: {shirui.pan, jia.wu, chengqi.zhang}@uts.edu.au.*
- *X. Zhu is with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431, and with the School of Computer Science, Fudan University, Shanghai, China. E-mail: xzhu3@fau.edu.*
- *P.S. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, and with the Institute for Data Science, Tsinghua University, Beijing, China. E-mail: psyu@cs.uic.edu.*

Manuscript received 28 Dec. 2014; revised 13 Sept. 2015; accepted 28 Sept. 2015. Date of publication 26 Oct. 2015; date of current version 2 Feb. 2016. Recommended for acceptance by L. Khan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2015.2492567

Although existing methods have advanced the graph classification from learning efficiency and classification accuracy perspectives, they typically share similar deficiencies in their designs: (1) in order to explore graph structures for training good classification models, they require a large number of training graphs; and (2) they can only work on a single learning task. In reality, due to the inherent complexity of the graph data and the costs involved in the labeling process, collecting a large number of labeled graphs for a specific task is difficult. However, it is quite common that multiple similar graph classification tasks, each having a limited number of training samples, may co-exist and need to be handled.

The structure of molecules in molecular medicine [8], for example, plays a crucial role in determining functions of the molecules, so graphs are commonly used to represent models in order to preserve the structure information. Labeling molecules, i.e., graphs, requires time, effort, and expensive resources [9] to test whether a chemical compound is active or inactive in relation to a cancer type, which makes it difficult to obtain labeled graphs. However similar bioassay tasks, such as anti-cancer tests for Melanoma and Prostate, are usually available. Instead of treating each task as a single-task graph classification (STG) problem, multi-task graph classification (MTG) studied in this paper can simultaneously handle multiple relevant graph classification tasks with improved performance gain.

When solving MTG problems, one simple approach is to treat each task independently and train an STG algorithm (e.g., gBoost [4]) for each task. The result from this approach is, however, far from optimal. This is because (1) the insufficient number of labeled graphs for each task makes it difficult for learning algorithms to comprehend graph

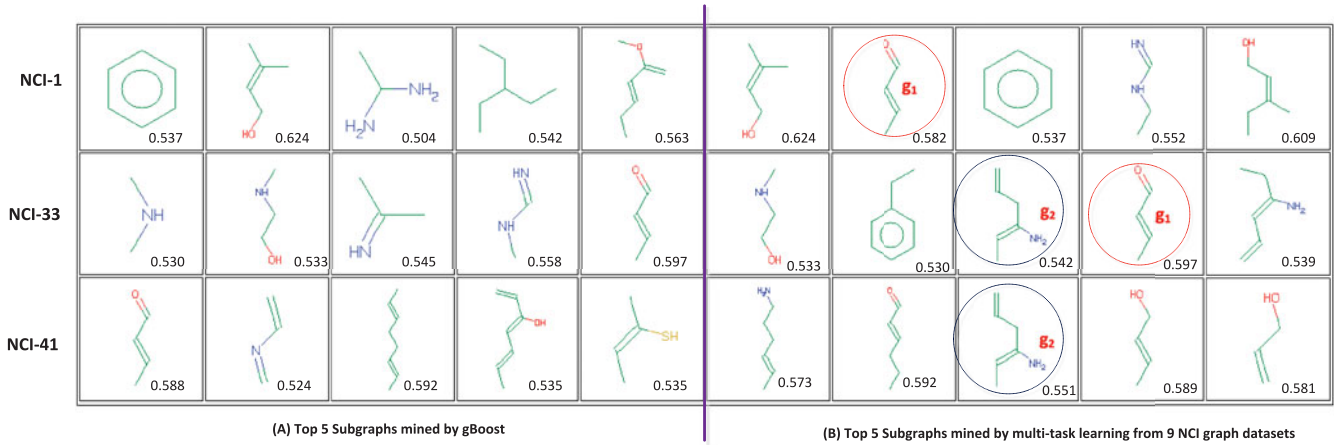

Fig. 1. Comparisons of the top five most discriminative subgraphs for each graph classification task, mined by (A) gBoost [4], or (B) multi-task learning (using 50 training graphs for each task). The numeric value next to each subgraph indicates the classification accuracy on test graphs using this single subgraph as a feature (i.e., an indicator of the classification quality of this subgraph). Multi-task learning in (B) favors subgraphs which also have high discriminative powers across all tasks. For instance, the circled g_1 is ranked at second place for NCI-1 on the training data because it also has a high score on NCI-33. g_1's score 0.582 in NCI-1 outperforms four out of the top five features selected by gBoost, but it was not discovered by gBoost to be one of the top five useful subgraphs (thus, its importance is under-evaluated when learning with the NCI-1 task alone).

structures and find effective subgraphs to train classification models. From a machine learning perspective, a small number of labeled graphs are biased samples obtained from sampling a large population. As a result, subgraph features discovered from these graph samples may be also biased, and are ineffective to classify test graphs; and (2) a learning model trained from a small number of labeled graphs tends to overfit the training samples and result in poor performance on the test data.

A second approach to solving MTG problems is to first mine frequent subgraphs [10] as features to transfer graphs into feature-vector format, and then apply state-of-the-art multi-task learning (MTL) algorithms [11], [12] to the vectors. This method is still suboptimal, mainly because subgraph feature exploration process is not tied to the learning tasks (given that there are multiple learning tasks). With suboptimal features, it is hard, if not impossible, to achieve good classification performance.

Instead of treating MTG as a group of multiple independent learning tasks, we advocate multi-task driven subgraph (MTDS) mining in this paper to explore high dimensional discriminative subgraph features and simultaneously train classification models for all tasks. By integrating MTDS-based feature selection into our multi-task graph classification objective function, we enable knowledge sharing across all tasks for better subgraph validation and model regularization. The niche of our multi-task subgraph feature exploration and multi-task graph classification stems from the following key observations.

Multi-task shared subgraphs. Because multiple graph classification tasks are relevant to each other, some common discriminative subgraph features may exist across different tasks. A significant subgraph on one task may also have a high discriminative score on other tasks. For instance, in Fig. 1, g_1 is a common subgraph of tasks NCI-1 and NCI-33. However, when performing subgraph selection on NCI-1 task only, g_1 will be missed by an STG algorithm (e.g., gBoost [4]). In this context, combining NCI-1 and NCI-33 as a multi-task problem clearly helps the NCI-1 task find a better discriminative subgraph for classification.

Implicit evaluation set and better regularization. To avoid overfitting incurred by insufficient training samples, machine learning algorithms usually validate their models on some evaluation sets before testing or incorporating regularization terms for model learning. With a small number of training graphs, any subgraph explored from a single task has a high risk of overfitting the training data. Taking the NCI graph classification task as an example, STG algorithms (i.e., gBoost) can easily fit the training graphs very well (achieved 100% or near 100% classification accuracy using 50 or 400 training graphs) but their performance on test graphs is much worse (about 60% or 65% as shown in Fig. 2). By unifying multiple tasks as one objective function, other tasks can be used as implicit evaluation sets for each task. An MTG objective function can thus help prune subgraph features which are only useful in the biased training data of an individual task but are not promising for other tasks.

As illustrated in Fig. 2, in the case of 400 training graphs, MTG algorithms can essentially reduce the risk of overfitting and achieve better graph classification, because relevant graph samples from other tasks are considered to be an implicit evaluation set to help validate the sub-graph mining process for better regularization.

Motivated by the above observations, we propose a multi-task graph classification algorithm which iteratively selects the most discriminative subgraphs to minimize regularized loss for all tasks. The multi-task graph classification is achieved by combining subgraph selection and model

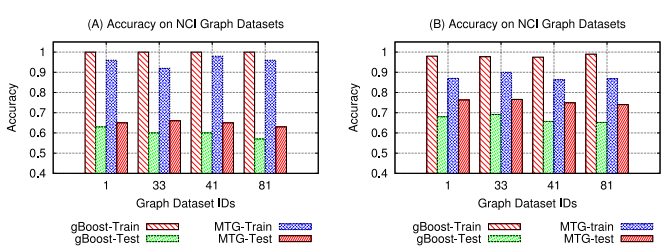

Fig. 2. Accuracy comparisons on training and test graphs. (A) 50 training graphs, and (B) 400 training graphs for each task. The MTG algorithm can release or prevent overfitting.

learning into an iterative process, which mutually benefits subgraph exploration and multi-task learning. For subgraph selection, we emphasize on low dimensional subgraphs shared among all tasks by employing the MTDS selection scheme; and for multi-task learning, all tasks are jointly regularized to achieve an optimization goal.

The main contributions of this paper can be summarized as follows:

- To the best of our knowledge, this is the first work handling multi-task learning for graph data. We propose an algorithm with theoretically proved convergence to jointly regularize multiple tasks to exploit discriminative subgraphs for multi-task graph classification.
- We generalize the *column generation technique* [4] to the multi-task graph classification setting. Any differentiable loss function, such as least squares, exponential, and logistic loss functions can be used in our algorithm.
- We propose to integrate two sparsity-inducing regularization norms, ℓ_1-norm and $\ell_{2,1}$-norm, for multi-task learning for graph data.
- We derive two branch-and-bound rules to prune the search space for multi-task driven subgraph mining.

The remainder of the paper is structured as follows. We review related work in Section 2. Problem definitions and preliminary are described in Section 3. Section 4 reports the proposed algorithm for multi-task graph classification. Experimental results are presented in Section 5, and we conclude the paper in Section 6.

2 Related Work

Our work is closely related to graph classification, multi-task learning, and infinite feature selection.

Graph classification. Existing methods for graph classification [4], [5], [6], [7], [13], [14], [15], [16] can be roughly distinguished into two groups: similarity-based methods and subgraph feature-based methods.

Similarity-based approaches aim to directly learn global similarities between graphs by using graph kernels [6] or graph embeddings [17]. The obtained global similarities are then fed into learning algorithms, such as support vector machines (SVM), for learning. One obvious drawback of global similarity-based approaches is that their similarity is directly calculated based on global graph structures, such as random walks or embedding space. Therefore, important substructures useful for differentiating graphs between different classes remain unknown.

In many graph classification domains, such as molecule classification, graphs from a specific class may have low global similarities but actually share some unique substructures. Therefore, using discriminative substructures as features and transferring graphs into vector space becomes a popular solution. For subgraph-based methods, a key issue is to define a measurement to assess the utility of each subgraph. Yan et al. proposed [18] a LEAP algorithm to exploit correlations between structure similarity and significance similarity, so that a branch-and-bound rule could be derived to effectively prune unpromising searching space efficiently. Ranu and Singh [19] proposed a scalable GraphSig algorithm, which is able to mine significant subgraphs with low frequencies. Thoma et al. [20] proposed a CORK algorithm, in combination with a frequent subgraph mining algorithm such as gSpan [10], to find subgraph features. Instead of carrying out explicit subgraph feature mining, a recent work [13] proposes to find a class-conditional "signal-subgraph" which is defined as the connection of edges that are probabilistically different between classes.

Recently, researchers have also studied complicated graph classification tasks, such as semi-supervised classification [7], multi-label classification [21], multi-view-graph learning [22], [23], and multi-graph classification [24], [25], [26]. In multi-view-graph learning, an object consists of multiple views and each view is represented as a graph structure. In multi-graph classification, the objective is to classify a bag which consists of multiple graphs.

Boosting methods [4], [5], [14], [27] are also popular for graph classification. In [5], the authors proposed to boost subgraph decision stumps from frequent subgraphs. In [4], the authors proposed a gBoost algorithm which iteratively selects subgraphs from the whole subgraph space (instead of using frequent subgraphs), which has demonstrated better performance for graph classification than using frequent subgraph filter-based methods. We have recently extended the gBoost algorithm to imbalanced graph classification settings [14], [28] and cost-sensitive learning [29].

The aforementioned graph classification methods, regardless of similarity-based methods or subgraph-based approaches, only consider a single task. Therefore, they are ineffective for multi-task settings where several graph classification tasks are related to each other and need to be learned in order to achieve maximum classification accuracy for all tasks.

Multi-task learning. State-of-the-art algorithms on multi-task learning [11], [12], [30], [31], [32], [33], [34] can also be roughly divided into two categories: (1) Regularized multi-task feature learning methods [11], [12], [31], [35], which assume that all tasks are homogeneous and that the purpose of the learning is to discover common feature representation across all tasks without exploring task relationships. (2) Task relationship exploration methods [30], [32], [33], [36], which either exploit task relationships via trace norm regularization to achieve some similar parameters among similar tasks [36], or try to learn a task covariance matrix from data if the task relationship is unknown in advance [30], [33].

Note that multi-task learning is closely related to transfer learning [37], but the difference is fundamental. Transfer learning aims to improve the learning on a single target task by using data from other tasks as auxiliary information. For multi-task learning, all tasks are equally important and should be learned simultaneously. A recent work [38] addresses transfer learning for graph databases, but its scope and objective are different from the proposed MTG.

Another study [39] exploits multi-task metric learning for networked data, where each network consists of a set of nodes with attribute contents. This problem setting still belongs to traditional multi-task learning, except that the instances share some dependency relationships. Our study differs from [39] in that we have many graphs, and each graph has a class label indicating the property of the graph (such as the activity of a chemical compound). A graph is

essentially one instance with structure information (such as a chemical compound). Our learning objective is to build classification models from labeled graphs, in order to accurately determine the label of an unlabeled graph.

Infinite feature selection. In considering the use of subgraph features for multi-task graph classification, our work is also closely related to feature (subgraph) selection from a large (possibly infinite) feature space. Saigo et al. [4] proposed using the column generation technique to progressively select subgraphs when dealing with infinite [40], [41] or streaming features [42]. In their methods, the objective function needs to be formulated into a linear programming problem with a hinge loss function, which restricts its applicability to general graph classification problems. A regularized loss minimization algorithm RLMD [16] was proposed recently to incrementally select subgraphs for graph classification, and can deal with any differentiable loss $\mathcal{L}$ together with both ℓ_1 and ℓ_2-norm regularizers. The grafting method [41] provides another solution for online feature selection. However all methods in [4], [16] and [41] are typically single task learning algorithms, so they are ineffective for MTL problems. A recent research [40] proposed dealing with a mixed norm regularized framework in a boosting framework for MTL problems by iteratively generating features to reduce the empirical loss. However, this method only considers traditional vector data, rather than graph data which is the focus of this paper.

3 Definition and Preliminaries

3.1 Problem Definition

Definition 1 (Connected Graph). *A graph is denoted by $G = (\mathcal{V}, E, L)$, where $V = \{v_1, \ldots, v_{n_v}\}$ is a set of vertices, $E \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, and L is a labeling function assigning labels to a node or an edge. A connected graph is a graph such that there is a path between any pair of vertices.*

In this paper, we focus on connected graphs and assume that each graph G has a class label y, $y \in \mathcal{Y} = \{-1, +1\}$ indicating the overall property of the graph, such as the active/inactive response of a chemical compound [1].

Definition 2 (Subgraph). *Given two graphs $G = (\mathcal{V}, E, L)$ and $g_k = (\mathcal{V}', E', L')$, g_k is a subgraph of G (*i.e *$g_k \subseteq G$) if there is an injective function f: $\mathcal{V}' \rightarrow \mathcal{V}$, such that $\forall (a, b) \in E'$, we have $(f(a), f(b)) \in E$, $L'(a) = L(f(a))$, $L'(b) = L(f(b))$, $L'(a, b) = L(f(a), f(b))$. If g_k is a subgraph of G ($g_k \subseteq G$), G is a supergraph of g_k ($G \supseteq g_k$).*

Multi-task graph classification. Given a set of graph classification tasks, where each task $t \in \{1, 2, \ldots, T\}$ has a set of labeled graphs $\{(G_{t,1}, y_{t,1}), \ldots, (G_{t,n_t}, y_{t,n_t})\}$, we use $G_{t,i} \in \mathcal{G}$ ($\mathcal{G}$ is the graph space) to denote the i^{th} graph in task t, and $G_{t,i}$'s class label is $y_{t,i} \in \mathcal{Y} = \{+1, -1\}$. Multi-task graph classification *aims* to learn T functions (classification models) $f_t : \mathcal{G} \rightarrow \mathcal{Y}, t \in [1, T]$, which have best classification performance on unseen graphs for *all* tasks.

3.2 Preliminaries

Single task graph classification. To support graph classification, state-of-the-art algorithms [4], [5] use a set of subgraphs discovered from the training graphs as features. Each subgraph g_k can then map a given graph $G_{t,i}$ to the class label space $\mathcal{Y} = \{+1, -1\}$:

$$\hbar_{g_k}(G_{t,i}) = 2I(g_k \subseteq G_{t,i}) - 1; \tag{1}$$

Here $I(a) = 1$ if a holds, and 0 otherwise.

Let $\mathcal{F} = \{g_1, \ldots, g_m\}$ be the full set of subgraphs in $\mathcal{G}$. We can use $\mathcal{F}$ as features to represent each graph $G_{t,i}$ in a vector space as $\boldsymbol{x}_{t,i} = \{\hbar_{g_1}(G_{t,i}), \ldots, \hbar_{g_m}(G_{t,i})\}$, with $\boldsymbol{x}^k_{t,i} = \hbar_{g_k}(G_{t,i})$. In the following subsection, $G_{t,i}$ and $\boldsymbol{x}_{t,i}$ are used interchangeably as they both refer to the same graph (i.e., the i-th graph in task t). Given the full subgraphs $\mathcal{F}$, the prediction function for task t is a linear classifier:

$$f_t(\boldsymbol{x}_{t,i}) = \boldsymbol{x_{t,i}} \cdot \boldsymbol{w_t} + b_t = \sum_{g_k \in \mathcal{F}} w_{t,k} \hbar_{g_k}(G_{t,i}) + b_t, \tag{2}$$

where $\boldsymbol{w}_t = [\boldsymbol{w}_{t,1}, \ldots, \boldsymbol{w}_{t,m}]'$ is the weight vector of all features for task t, and b_t is the bias of the model. The predicted class of $\boldsymbol{x}_{t,i}$ is +1 if $f_t(\boldsymbol{x}_{t,i}) > 0$ or -1 otherwise.

For single task graph classification, the state-of-the-art algorithm gBoost [4] formulates its objective function as a linear programming problem, then integrates the discriminative subgraph mining into the model leaning process via column generation techniques.

4 Multi-Task Graph Classification

In this section, we describe proposed multi-task graph classification algorithm.

4.1 Regularized Multi-Task Graph Classification Formulation

To achieve multi-task graph classification, our theme is to use multi-task to guide an iterative subgraph exploration process in order to achieve the lowest regularized empirical risks for all tasks. This can be formulated in the following objective function:

$$\mathcal{J} = \min_{\boldsymbol{W}, \boldsymbol{b}} \underbrace{\sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} \mathcal{L}(y_{t,i}, f_t(\boldsymbol{x}_{t,i}))}_{\mathcal{C}} + \gamma R(\boldsymbol{W}) \tag{3}$$

Here $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_T]$ is a weight matrix indicating weights of each subgraph *w.r.t.* different tasks, $\boldsymbol{b} = [b_1, \ldots, b_T]$ are the bias parameters for each function f_t, and n_t is the number of training graphs in task t. The first term $\mathcal{C}$ measures the loss on the training graphs for all tasks, where $\mathcal{L}(y_{t,i}, f(\boldsymbol{x}_{t,i}))$ is a loss function measuring misclassification penalty of a graph $G_{t,i}$. The second part is a regularization term to enforce sparse solutions, and a parameter γ is used to trade-off between these two parts. In this paper, we mainly consider Logistic loss function

$$\mathcal{L}(y_{t,i}, f_t(x_{t,i})) = \log\left(1 + \exp\{-y_{t,i} f_t(\boldsymbol{x}_{t,i})\}\right). \tag{4}$$

Note that any other differentiable loss function, such as least square loss $\mathcal{L}(y, f_t) = \frac{1}{2}(y - f_t)^2$ or exponential loss $\mathcal{L}(y, f_t) = \exp\{-y f_t\}$, can be used in our algorithm. As for the second term $R(\boldsymbol{W})$, our main objective is to obtain a sparse solution on $\boldsymbol{W}$, i.e., a finite set of subgraph features shared by all tasks. We consider the following regularizers:

ℓ_1-norm Lasso regularization $R(\boldsymbol{W}) = \sum_{k,t} |\boldsymbol{W}_{k,t}|$.

The rationale is that the ℓ_1-norm regularizer can produce solutions with many coefficients being 0, which is known as Lasso [43] and has been widely applied for variable selections. A simplification of Lasso in MTG is to use a parameter γ to control the regularization of all tasks, assuming that different tasks share the same sparsity parameter.

$\ell_{2,1}$-norm regularization. Because the total subgraph space is infinitely large and we want to select only a subset of most important subgraphs among all possible subgraphs, we propose to use a mixed-norm regularizer $\ell_{2,1}$ norm

$$\|\boldsymbol{W}\|_{2,1} = \sum_{k=1}^{m} \sqrt{\sum_{t=1}^{T} |\boldsymbol{W}_{k,t}|^2} = \sum_{k=1}^{m} \|\boldsymbol{W}_{k,\cdot}\|_2,$$

where $\boldsymbol{W}_{k,\cdot}$ is the k-th row of $\boldsymbol{W}$. The $\ell_{2,1}$ regularizer first computes the ℓ_2-norm (across the tasks) of each row in $\boldsymbol{W}$ and then calculates the ℓ_1-norm of the vector $d(\boldsymbol{W}) = (\|\boldsymbol{W}_{1,\cdot}\|_2, \ldots, \|\boldsymbol{W}_{m,\cdot}\|_2)$. This is a special case of group Lasso [44] for group variable selection and was previously applied in [12] for multi-task learning on vector data. This norm ensures that common features will be selected across all tasks. Using this regularizer can produce some rows of $\boldsymbol{W}$ as 0. If a row $\boldsymbol{W}_{k,\cdot} = \boldsymbol{0}$[1], the subgraph (feature) g_k will not be used in all tasks. In the following, we propose our gradient/subgradient-based algorithms for multi-task graph classification. Because ℓ_1 and $\ell_{2,1}$ norm regularizations result in different subgradients, we will handle each case separately.

4.2 Multi-Task Graph Classification: Challenges and Solution Sketch

*Challenges.*When the whole feature set $\mathcal{F} = \{g_1, \ldots, g_m\}$ is small and available for learning, the objective function in Eq. (3) can be effectively solved by using an existing toolbox [31] for either ℓ_1 or $\ell_{2,1}$ norm regularization. For graph data, however, the challenge is twofold: (1) the whole feature set $\mathcal{F}$ is implicit and unavailable, and enumerating subgraph features is NP-complete; and (2) the number of subgraphs is huge and possibly infinite ($m \to +\infty$).

Solution sketch. To solve the aforementioned challenges, we propose to iteratively include features/subgraphs into our objective function. In other words, multi-task subgraph selection and model learning are integrated into one objective function for mutual benefits. More specifically, we carry out subgraph selection based on the subgradient of the objective function $\mathcal{J}$, so the empirical loss can always be reduced when selecting and adding the most discriminative subgraph to the existing subgraph feature set. After a new subgraph is incorporated, we re-solve the new *restricted master problem*[2] of Eq. (3), which is defined as follows:

$$\mathcal{J}_1 = \min_{\boldsymbol{W}^{(s)}, \boldsymbol{b}^{(s)}} \underbrace{\sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} \mathcal{L}(y_{t,i}, f_t(\boldsymbol{x}_{t,i}^{(s)}))}_{\mathcal{C}} + \gamma R(\boldsymbol{W}^{(s)}), \quad (5)$$

where $\boldsymbol{W}^{(s)}$ and $\boldsymbol{b}^{(s)}$ are the solutions based on the selected features in the s-th iteration, and $\boldsymbol{x}_{t,i}^{(s)}$ is feature representation of $\boldsymbol{x}_{t,i}$ *w.r.t.* the selected features.

1. $\boldsymbol{0}$ or $\boldsymbol{1}$ indicates T dimensional vectors with all 0 or 1 values.
2. A reduced problem based on the selected features only.

The aforementioned feature selection and model learning procedure continues until the algorithm converges. To handle the huge subgraph space, we derive two branch-and-bound pruning rules to reduce the search space. The above algorithm design enjoys two unique advantages: (1) the discriminative subgraph selection is driven by the well defined multi-task learning objective function for model learning; and (2) the model learning will be further enhanced by the inclusion of newly selected discriminative subgraph features.

Our method is based on the gradient/subgradient functional space of the objective function Eq. (3). Let us define the gradient of the loss term $\mathcal{C}$ in Eq. (3) on the subgraph feature g_k with respect to the t-th task as $\nabla \mathcal{C}_{k,\boldsymbol{w}_t}$.

$$\begin{aligned} \nabla \mathcal{C}_{k,\boldsymbol{w}_t} &= \frac{\partial \mathcal{C}}{\partial \boldsymbol{w}_{k,t}} = \frac{1}{n_t} \sum_{i=1}^{n_t} \frac{\partial \mathcal{L}(y_{t,i}, f_t(\boldsymbol{x}_{t,i}))}{\partial f_t(\boldsymbol{x}_{t,i})} \frac{\partial f_t(\boldsymbol{x}_{t,i})}{\partial \boldsymbol{w}_{k,t}} \\ &= -\frac{1}{n_t} \sum_{i=1}^{n_t} \frac{y_{t,i} \boldsymbol{x}_{t,i}^k}{1 + e^{y_{t,i} f_t(\boldsymbol{x}_{t,i})}} = \sum_{i=1}^{n_t} y_{t,i} \alpha_{t,i} \boldsymbol{x}_{t,i}^k. \end{aligned} \quad (6)$$

Here, $\alpha_{t,i} = -\frac{1}{n_t(1+e^{y_{t,i} f_t(\boldsymbol{x}_{t,i})})}$ is a constant, given the feature represented graph sample $\boldsymbol{x}_{t,i}$. Later on, we will regard it as a weight associated to graph $G_{t,i}$ for the subgraph mining process.

Then the gradient vector of feature g_k over all T tasks is defined as

$$\nabla C_{k,\cdot} = [\nabla \mathcal{C}_{k,\boldsymbol{w}_1}, \ldots, \nabla \mathcal{C}_{k,\boldsymbol{w}_T}]. \quad (7)$$

4.3 Optimal Subgraph Candidate Exploration

Because we assume that some subgraphs/features g_k will not be used for learning the classification models, i.e., $\boldsymbol{W}_{k,\cdot} = \boldsymbol{0}$, it makes sense to partition all subgraph features $\mathcal{F}$ into two disjoint subsets $\mathcal{F}_1$ and $\mathcal{F}_2$. $\mathcal{F}_1$ stores active features which are used to learn the classification model and this set is frequently updated as desired. $\mathcal{F}_2$ includes unselected graphs with $\boldsymbol{0}$ weights (i.e., for $g_k \in \mathcal{F}_2, \boldsymbol{W}_{k,\cdot} = \boldsymbol{0}$). We can then iteratively select the best features from $\mathcal{F}_2$ to $\mathcal{F}_1$.

Stopping conditions and conditional score. According to the optimal conditions, after reaching the optimum, the first derivative of Eq. (3) should be 0:

$$\nabla C_{k,\boldsymbol{w}_t} + \gamma \boldsymbol{o}_{k,t} = 0, \quad (8)$$

where $\boldsymbol{o}_{k,t}$ is the subgradient of the ℓ_1 or $\ell_{2,1}$ norm of $\boldsymbol{W}_{k,t}$.

Let $\boldsymbol{o}_k = [\boldsymbol{o}_{k,1}, \ldots, \boldsymbol{o}_{k,T}]$ be the subgradient vector over all tasks. For the ℓ_1-norm of $\boldsymbol{W}_{k,\cdot}$ (i.e., $|\boldsymbol{W}_{k,\cdot}|$), each dimension of $\boldsymbol{o}_k$ is as follows [45], [46]:

$$\boldsymbol{o}_{k,t} \in \begin{cases} [-1, 1], & \boldsymbol{W}_{k,t} = 0, \\ \mathbf{sign}(\boldsymbol{W}_{k,t}), & \boldsymbol{W}_{k,t} \neq 0. \end{cases} \quad (9)$$

Now we can state the optimal condition for ℓ_1 norm regularization. According to Eq. (8) and Eq. (9), a vector $\hat{\boldsymbol{W}} = [\hat{\boldsymbol{w}}_1, \ldots, \hat{\boldsymbol{w}}_t]$ is the optimal solution of our objective function Eq. (3) if and only if:

$$\|\nabla \mathcal{C}_{k,\cdot}\|_\infty \leq \gamma, \quad \text{if} \quad \hat{\boldsymbol{W}}_{k,\cdot} = \boldsymbol{0}, \quad (10)$$

$$\nabla \mathcal{C}_{k,\cdot} + \gamma\, \mathbf{sign}(\hat{\boldsymbol{W}}_{k,\cdot}) = \boldsymbol{0}, \quad \text{if} \quad \hat{\boldsymbol{W}}_{k,\cdot} \neq \boldsymbol{0}, \quad (11)$$

where $\|\nabla \mathcal{C}_{k,\cdot}\|_\infty = \max_{t=1}^{T} |\nabla \mathcal{C}_{k,\boldsymbol{w}_t}|$. Eq. (10) ensures that $\forall t, |\nabla \mathcal{C}_{k,\boldsymbol{w}_t}| \leq \gamma$.

Similarly, for the $\ell_{2,1}$-norm, $\boldsymbol{o}_k$ for $\|\boldsymbol{W}_{k,\cdot}\|_2$ are [46]:

$$\boldsymbol{o}_k \in \begin{cases} z \in \mathbb{R}^T, & \|z\|_2 \leq 1: \quad \boldsymbol{W}_{k,\cdot} = \boldsymbol{0}, \\ & \frac{\boldsymbol{W}_{k,\cdot}}{\|\boldsymbol{W}_{k,\cdot}\|_2}: \quad \boldsymbol{W}_{k,\cdot} \neq \boldsymbol{0}. \end{cases} \tag{12}$$

Therefore, according to Eq. (8) and Eq. (12), a vector $\hat{\boldsymbol{W}} = [\hat{\boldsymbol{w}}_1, \ldots, \hat{\boldsymbol{w}}_t]$ is the optimal solution to our objective function Eq. (3) if and only if:

$$\|\nabla \mathcal{C}_{k,\cdot}\|_2 \leq \gamma, \quad \text{if} \quad \hat{\boldsymbol{W}}_{k,\cdot} = \boldsymbol{0}, \tag{13}$$

$$\nabla \mathcal{C}_{k,\cdot} + \gamma \|\hat{\boldsymbol{W}}_{k,\cdot}\|_2^{-1} \hat{\boldsymbol{W}}_{k,\cdot} = \boldsymbol{0}, \quad \text{if} \quad \hat{\boldsymbol{W}}_{k,\cdot} \neq \boldsymbol{0}. \tag{14}$$

To reduce the objective value of $\mathcal{J}$ in Eq. (3), we propose to select subgraphs in $\mathcal{F}_2$ whose weight violates Eq. (10) for ℓ_1-norm regularizer or Eq. (13) for $\ell_{2,1}$-norm regularizer, and update the selected active set $\mathcal{F}_1$ with the newly-selected features and re-optimize Eq. (3) with the current features. This process will repeat until no candidate violates either Eq. (10) or Eq. (13). In other words, these two equations can naturally induce the stopping criterion for our process. Let us define the conditional score of a subgraph as follows:

Definition 3 (Conditional Score). *For a subgraph pattern g_k, its conditional score over all T tasks is defined as*

$$\Upsilon(g_k) = \|\nabla \mathcal{C}_{k,\cdot}\|_q, q \in \{\infty, 2\}, \tag{15}$$

where $q = \infty$ for ℓ_1 regularization and $q = 2$ for $\ell_{2,1}$ regularization; $\nabla \mathcal{C}_{k,\cdot}$ is defined in Eq. (7).

As a result, all potential subgraphs which violate Eq. (10) or Eq. (13) can be defined as

$$\mathcal{F}_3 = \{g_k | g_k \in \mathcal{F}_2, \Upsilon(g_k) > \gamma\}. \tag{16}$$

$\mathcal{F}_3$ defines all candidate subgraphs which can be selected and added to $\mathcal{F}_1$.

Optimal multi-task subgraph selection: Intuitively, any subgraph in $\mathcal{F}_3$ can be selected and added to $\mathcal{F}_1$ in each iteration. To ensure quick convergence, we will select the one with the most significant impact in reducing the function value of $\mathcal{J}$ in Eq. (3). From Eq. (3) and Eq. (8), the gradients for subgraph g_k over T tasks are defined as

$$\Gamma = \sum_{t=1}^{T} \nabla C_{k,\boldsymbol{w}_t} + \gamma \sum_{t=1}^{T} \boldsymbol{o}_{k,t} = \nabla C_{k,\cdot} \cdot \boldsymbol{1} + \gamma \boldsymbol{o}_k \cdot \boldsymbol{1}. \tag{17}$$

From Eq. (9) and Eq. (12), we know that $\boldsymbol{0}$ is a feasible subgradient for both ℓ_1 and $\ell_{2,1}$ norm regularizers. Therefore, we can set $\boldsymbol{o}_k = \boldsymbol{0}$, in such case, $\Gamma = \nabla \mathcal{C}_{k,\cdot} \cdot \boldsymbol{1}$. Then we can compute the absolute value $|\nabla \mathcal{C}_{k,\cdot} \cdot \boldsymbol{1}|$, and choose the subgraph with the largest value each time (because it will possibly have the most significant impact in reducing $\mathcal{J}$ in Eq. (3)).

4.4 Multi-Task Graph Classification Algorithm

Before explaining our multi-task graph classification algorithm details, we formally define a multi-task score for a subgraph to quantify its utility value for MTG as follows:

Definition 4 (MTG Discriminative Score). *For a subgraph pattern g_k, its discriminative score over all T tasks is defined as follows:*

$$\Theta(g_k) = |\nabla \mathcal{C}_{k,\cdot} \cdot \boldsymbol{1}| = |\sum_{t=1}^{T} \nabla \mathcal{C}_{k,\boldsymbol{w}_t}|, \tag{18}$$

where $\nabla \mathcal{C}_{k,\cdot}$ and $\nabla \mathcal{C}_{k,\boldsymbol{w}_t}$ are defined in Eq. (7) and Eq. (6).

Algorithm 1 illustrates the detailed steps of our iterative subgraph feature learning for multi-task graph classification. Initially, the weights for all training graphs in each task are set equally as $1/n_t$ (n_t is the number of labeled subgraphs in task t), and the active set $\mathcal{F}_1$ is initialized to be empty.

On the next step, the algorithm mines a set of subgraphs $\mathcal{P}$ from $\mathcal{F}_3$ which have the highest MTG discriminative scores defined by Eq. (18). This step involves a multi-task driven subgraph mining procedure, which will be addressed in the next subsection. To reduce the number of iterations for subgraph mining, top K subgraphs are used in each iteration (instead of the best subgraph). The impact of the K values on the algorithm performance is studied in Section 5.2.3.

If the current graph set $\mathcal{P}$ is empty on steps 4-5, it means that no more subgraphs are violating the optimal condition of Eq. (10) or Eq. (13), so the algorithm will stop. On step 6, we add newly selected subgraphs $\mathcal{P}$ to the existing subgraph set $\mathcal{F}_1$, and re-solve the restricted objective function Eq. (5) on step 7. To solve the restricted objective function, we use the MALSAR toolbox[3] in our experiments.

On the last step, the algorithm updates the weight $\alpha_{t,i}$ for each graph $G_{t,i}$. This will help compute the gradient vector of $\nabla \mathcal{C}_{k,\cdot}$ for the purpose of computing the MTG discriminative score of each subgraph in preparation for subgraph mining in the next round.

Algorithm 1. Multi-task Graph Classification Algorithm

Input:
$\{(G_{t,1}, y_{t,1}), \ldots, (G_{t,n}, y_{t,n})\}, t \in \{1, 2, \ldots, T\}$: Graph Datasets from T tasks;
S_{max}: Maximum number of iterations;
K: Number of optimal subgraphs used in each iteration;
Output:
$\boldsymbol{W}^{(s)}, \boldsymbol{b}^{(s)}$: Parameters for multi-task models
1: $\alpha_{ti} = 1/n_t$; $\mathcal{F}_1 \leftarrow \emptyset$; $s \leftarrow 1$;
2: **while** $s \leq S_{max}$ **do**
3: Mine top-K subgraph features $\mathcal{P} = \{g_i\}_{i=1,\ldots,k}$ from $\mathcal{F}_3$ with maximum discriminative score defined by Eq. 18; //Algorithm 2;
4: **if** $\mathcal{P} = \emptyset$ **then**
5: **break;**
6: $\mathcal{F}_1 \leftarrow \mathcal{F}_1 \bigcup \mathcal{P}$;
7: Solve Eq. (5) based on $\mathcal{F}_1$ to get new weights matrix $\boldsymbol{W}^{(s)}, \boldsymbol{b}^{(s)}$;
8: Update the graph weights on each training graph $\alpha_{t,i} = -\frac{1}{n_t(1+e^{y_{t,i} f_t(\boldsymbol{x}_{t,i})})}$
9: $s \leftarrow s + 1$;
10: **return** $\boldsymbol{W}^{(s)}, \boldsymbol{b}^{(s)}$;

Theorem 1 (Convergence Properties). *Algorithm 1 guarantees that the restricted objective function Eq. (5) will monotonically decrease.*

3. http://www.MALSAR.org

Proof. Without loss of generality, we assume in each iteration that, a subgraph is selected and added to $\mathcal{F}_1$, i.e., we set $K = 1$ in Algorithm 1. Let the optimal objective value based on current s features (*i.e.*, $|\mathcal{F}_1| = s$) with respect to Eq. (5) be obtained at $(\hat{\boldsymbol{W}}^{(s)}, \hat{\boldsymbol{b}}^{(s)})$, i.e.,

$$\mathcal{J}_1(\hat{\boldsymbol{W}}^{(s)}, \hat{\boldsymbol{b}}^{(s)}) = \underbrace{\sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} \mathcal{L}(y_{t,i}, f_t(\boldsymbol{x}_{t,i}^{(s)}))}_{\mathcal{C}} + \gamma R(\hat{\boldsymbol{W}}^{(s)}).$$

Then in the $s+1$-th iteration, the optimal objective value of Eq. (5) is

$$\begin{aligned} \min \mathcal{J}_1(\boldsymbol{W}^{(s+1)}, \boldsymbol{b}^{(s+1)}) &= \min(\mathcal{C} + \gamma R) | \forall (\boldsymbol{W}^{(s+1)}, \boldsymbol{b}^{(s+1)}) \\ &\leq (\mathcal{C} + \gamma R) | ([\hat{\boldsymbol{W}}^{(s)}; \boldsymbol{0}], \hat{\boldsymbol{b}}^{(s)}). \end{aligned}$$

Thus the objective value of the restricted problem Eq. (5) based on the currently selected features $\mathcal{F}_1$ will always monotonically decrease in two successive iterations. Because the objective function value is non-negative (bounded), we can ensure that it will finally converge as iteration continues. The proof is complete.

According to optimal conditions (Eq. (10) or Eq. (13)), if the algorithm has reached the optimal solution, $\forall g_k, g_k \notin \mathcal{F}_1$, we will have $\boldsymbol{W}_{k,\cdot} = \boldsymbol{0}$, thus its conditional score $\Upsilon(g_k) = \|\nabla \mathcal{C}_{k,\cdot}\|_q < \gamma, q \in \{\infty, 2\}$. In this case, no more subgraphs will be obtained in $\mathcal{P}$ from $\mathcal{F}_3$, i.e., $\mathcal{P} = \emptyset$. Thus our stopping condition (steps 4-5) guarantees the optimal solution of our algorithm. □

4.5 Multi-Task Driven Subgraph Mining

To obtain a set of discriminative features $\mathcal{P}$ in $\mathcal{F}_3$ from the T tasks of training graphs, we need to perform a subgraph enumeration procedure. The mining of the top-K subgraphs on step 3 of Algorithm 1 also needs to enumerate the entire set of subgraph patterns from the training graphs of all tasks. In our MTG algorithm, we employ a frequent subgraph mining-based algorithm, gSpan [10]. 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. 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.

During the subgraph mining process, an effective pruning scheme is essential because the search space is exponentially large/infinite. In this subsection, we will first derive the upper-bound of the MTG discriminative score, and then provide an upper-bound of conditional score. Both scores will help prune the search space and speed up the subgraph mining.

Theorem 2 (Discriminative Score Upper-bound). *Let g and g' be two subgraph patterns, and $g \subseteq g'$, for the subgraph g, we define*

$$A_1(g) = 2 \sum_{t=1}^{T} \sum_{\{i | y_{t,i} = +1, g \in G_{t,i}\}} \alpha_{t,i},$$

$$A_2(g) = 2 \sum_{t=1}^{T} \sum_{\{i | y_{t,i} = -1, g \in G_{t,i}\}} \alpha_{t,i},$$

$$A_3 = \sum_{t=1}^{T} \sum_{i=1}^{n_t} \alpha_{t,i} y_{t,i},$$

$$\hat{\Theta}(g) = \begin{cases} \max\{|A_1(g) - A_3|, |A_2(g)|\} : & A_3 \geq 0, \\ \max\{|A_2(g) + A_3|, |A_1(g)|\} : & A_3 < 0, \end{cases}$$

then $\Theta(g') \leq \hat{\Theta}(g)$, where $\Theta(g')$ is defined in Eq. (18).

Proof. We start with the definition of $\Theta(g')$:

$$\begin{aligned} \Theta(g') &= |\sum_{t=1}^{T} \sum_{i=1}^{n_t} y_{t,i} \alpha_{t,i} f_t(\boldsymbol{x}_{t,i})| \\ &= |\sum_{t=1}^{T} \sum_{i=1}^{n_t} y_{t,i} \alpha_{t,i} \cdot [2I(g' \subseteq G_{t,i}) - 1]| \\ &= |2 \sum_{t=1}^{T} \sum_{g' \subseteq G_t} y_{t,i} \alpha_{t,i} - \sum_{t=1}^{T} \sum_{i=1}^{n_t} \alpha_{t,i} y_{t,i}| \\ &= |A_1(g') - A_2(g') - A_3| \\ &\leq \begin{cases} \max\{|A_1(g') - A_3|, |A_2(g')|\} : & A_3 \geq 0, \\ \max\{|A_2(g') + A_3|, |A_1(g')|\} : & A_3 < 0, \end{cases} \\ &\leq \begin{cases} \max\{|A_1(g) - A_3|, |A_2(g)|\} : & A_3 \geq 0, \\ \max\{|A_2(g) + A_3|, |A_1(g)|\} : & A_3 < 0, \end{cases} \\ &= \hat{\Theta}(g). \end{aligned}$$

The first inequality holds because for $\alpha_{t,i} < 0$, $A_1(g') \leq 0$ and $A_2(g') \leq 0$, so the upper-bound depends on A_3. If $A_3 \geq 0$, $A_1(g')$ and A_3 will have different signs, then the upper-bound is the maximum of $\{|A_1(g') - A_3|, |A_2(g')|\}$. The case is similar for $A_3 < 0$. The second inequality holds because $|A_1(g')| \leq |A_1(g)|$ and $|A_2(g')| \leq |A_2(g)|$ for $g \subseteq g'$. □

Theorem 2 states that for any super graph of a subgraph g, its MTG discriminative score, over T tasks, is upper-bounded by $\hat{\Theta}(g)$.

Single discriminativeness bound. The above discriminative score upper-bound can also be applied to each single task separately. If only task t is considered, the single discriminativeness upper-bound is defined as $\hat{\Theta}(g, t)$, which requires that $A_1(g), A_2(g)$ and A_3 in Theorem 2 are computed over task t only.

Theorem 3 (Conditional Score Upper-bound). *Given two subgraph features g and g_k ($g \subseteq g_k$), and a set of upper-bounds of single discriminativeness bound:*

$$\hat{\Upsilon}_{(g,\cdot)} = [\hat{\Theta}(g, 1), \ldots, \hat{\Theta}(g, T)]$$

let

$$\hat{\Upsilon}(g) = \|\hat{\Upsilon}_{(g,\cdot)}\|_q, q \in \{\infty, 2\},$$

then $\Upsilon(g_k) \leq \hat{\Upsilon}(g)$, where $\Upsilon(g_k)$ is defined in Eq. (15).

Proof. The conditional score for g_k on the task t is $|\nabla \mathcal{C}_{k,\boldsymbol{w}_t}|$, which is upper-bounded by $\hat{\Theta}(g, t)$, i.e., $\hat{\Theta}(g, t) \geq |\nabla \mathcal{C}_{k,\boldsymbol{w}_t}|$. Because every entry in $\Upsilon(g_k)$ is smaller than that in $\hat{\Upsilon}_{(g,\cdot)}$, the ℓ_∞ or ℓ_2 norm on the vector also holds, i.e., $\hat{\Upsilon}(g) \geq \Upsilon(g_k)$. □

According to Theorem 3, once a subgraph g is generated, the conditional scores for all its super-graphs are

upper-bounded by $\hat{\Upsilon}(g)$. Therefore, we use this rule to prune unpromising candidates.

Algorithm 2. Multi-Task Driven Subgraph Mining

Input:
$\{(G_{t,1}, y_{t,1}), \ldots, (G_{t,n}, y_{t,n})\}, t \in \{1, 2, \ldots, T\}$: Graph Datasets from T tasks;
γ : Predefined regularization parameter;
α_{ti} : Weight for each graph example;
K: Number of optimal subgraph patterns;
$\mathcal{F}_1$: Already selected subgraph set;
Output:
$\mathcal{P} = \{g_k\}_{k=1,\ldots,K}$: The top-$K$ subgraphs;
1: $\eta = 0$, $P \leftarrow \emptyset$;
2: **while** Recursively visit the DFS Code Tree in gSpan **do**
3: $g_p \leftarrow$ current visited subgraph in DFS Code Tree;
4: **if** g_p has been examined **then**
5: **continue**;
6: Compute scores $\Theta(g_p)$ and $\Upsilon(g_k)$ for subgraph g_p according Eq. (18) and Eq. (15);
7: **if** $g_p \in \mathcal{F}'_3$ & $\Theta(g_p) > \eta$ **then**
8: $\mathcal{P} \leftarrow \mathcal{P} \bigcup g_p$;
9: **if** $|\mathcal{P}| > K$ **then**
10: $g^{\bigstar} \leftarrow \arg\min_{g_k \in \mathcal{P}} \Theta(g_k)$;
11: $\mathcal{P} \leftarrow \mathcal{P}/g^{\bigstar}$;
12: $\eta \leftarrow \min_{g_k \in \mathcal{P}} \Theta(g_k)$;
13: **if** $\hat{\Theta}(g_p) > \eta$ & $\hat{\Upsilon}(g_p) > \gamma$ **then**
14: Depth-first search the subtree rooted from node g_p;
15: **return** $\mathcal{P} = \{g_k\}_{k=1,\cdots,K}$;

Multi-task driven subgraph mining algorithm. Our multi-task driven subgraph mining algorithm is listed in Algorithm 2. The minimum value η in optimal set $\mathcal{P}$ is initialized on step 1. Duplicated subgraph features are pruned on steps 4-5, and the discriminative score $\Theta(g_p)$ and conditional score $\Upsilon(g_p)$ for g_p are calculated on step 6. If g_p is included in the current candidate set $\mathcal{F}'_3 = \{g_k | g_k \in \mathcal{F}_2, \Upsilon(g_k) > \gamma + \varepsilon\}$ and $\Theta(g_p)$ is larger than η, we add g_p to the feature set $\mathcal{P}$ (steps 7-8). Here, we have relaxed $\mathcal{F}_3$ from Eq. (16) to a ε-tolerance set, *i.e.*, $\mathcal{F}'_3$, because $\Upsilon(g_k)$ only changes subtly in the last few iterations (ε=0.005 in our experiments).

When the size of $\mathcal{P}$ exceeds the predefined size K, the subgraph with the minimum discriminative score is removed (steps 9-11). The algorithm then updates the minimum optimal value η on step 12, and uses two branch-and-bound pruning rules, Theorems 2 and 3, to prune the search space on step 13. The two rules will reduce unpromising candidates by using discriminative scores and conditional scores, respectively. Lastly, the optimal set $\mathcal{P}$ is obtained on step 15.

The above pruning process is a key feature of our algorithm, because it does not require any support threshold for subgraph mining, whereas all other subgraph mining methods will require users to predefine a minimum threshold value to find frequent subgraphs.

4.6 Relation to gBoost Algorithm

The gBoost algorithm [4] can be considered as a special case of Eq. (3) with constraint $\boldsymbol{w} \geq 0$ and a single task only. Interested readers can refer to the supplementary material section for detailed analysis.

TABLE 1
Datasets Used in Experiments

Collections	ID	#Pos	#Total	Learning tasks
NCI Tasks	1	1,793	37,349	Non-Small Cell Lung
	33	1,467	37,022	Melanoma
	41	1,350	25,336	Prostate
	47	1,735	37,298	Central Nerv Sys
	81	2,081	37,549	Colon
	83	1,959	25,550	Breast
	109	1,773	37,518	Ovarian
	123	2,715	36,903	Leukemia
	145	1,641	37,043	Renal
PTC Tasks	Sub_{MR}	32	87	Male Rat (MR)
	Sub_{FR}	35	85	Female Rat (FR)
	Sub_{MM}	29	85	Male Mouse (MM)
	Sub_{FM}	35	88	Female Mouse (FM)

5 Experiment

5.1 Experimental Settings

Benchmark data. We validate the performance of the proposed algorithm on two types of multi-task graph classification domains.

Anti-cancer activity prediction (NCI): The NCI graph collection[4] is a benchmark for predicting biological activities of small molecules for different types of cancers. Each molecule is represented as a graph, with atoms representing nodes and bonds denoting edges. A molecule is positive if it is active against a certain type of cancer, or negative otherwise. Table 1 summarizes nine NCI graph classification tasks used in our experiments, where columns 2-5 show the bioassay ID, number of positive graphs, total number of graphs in the original dataset, and the description of learning task, respectively. In our experiments, we randomly select #Pos number of negative graphs from each original graph set to create a multi-task graph classification problem with relatively balanced training graphs for each task. This allows us to concentrate on the performance of general graph classification, without any complication of severely imbalanced class distributions. For highly imbalanced graph data, special designs are needed to prevent all graph samples from being classified into a single class, by considering minority and majority classes or reweighing the misclassification cost of different graphs, as reported in our recent studies [28], [29]. Note that although each of the nine tasks focuses on the prediction of different type of cancers, all these tasks are relevant in cancer prediction and some common discriminative substructures may exist for all cancer types (as shown in Fig. 1). This makes NCI an ideal benchmark for multi-task graph classification.

Predictive Toxicology Challenge Dataset (PTC): The PTC challenge includes a number of carcinogenicity tasks for toxicology prediction of chemical compounds[5]. The dataset we selected contains 417 compounds with four types of test animal: MM (male mouse), FM (female mouse), MR (male rat), and FR (female rat). Each compound has one label selected from {CE, SE, P, E, EE, IS, NE, N}, which stands for Clear

4. http://pubchem.ncbi.nlm.nih.gov
5. http://www.predictive-toxicology.org/ptc/

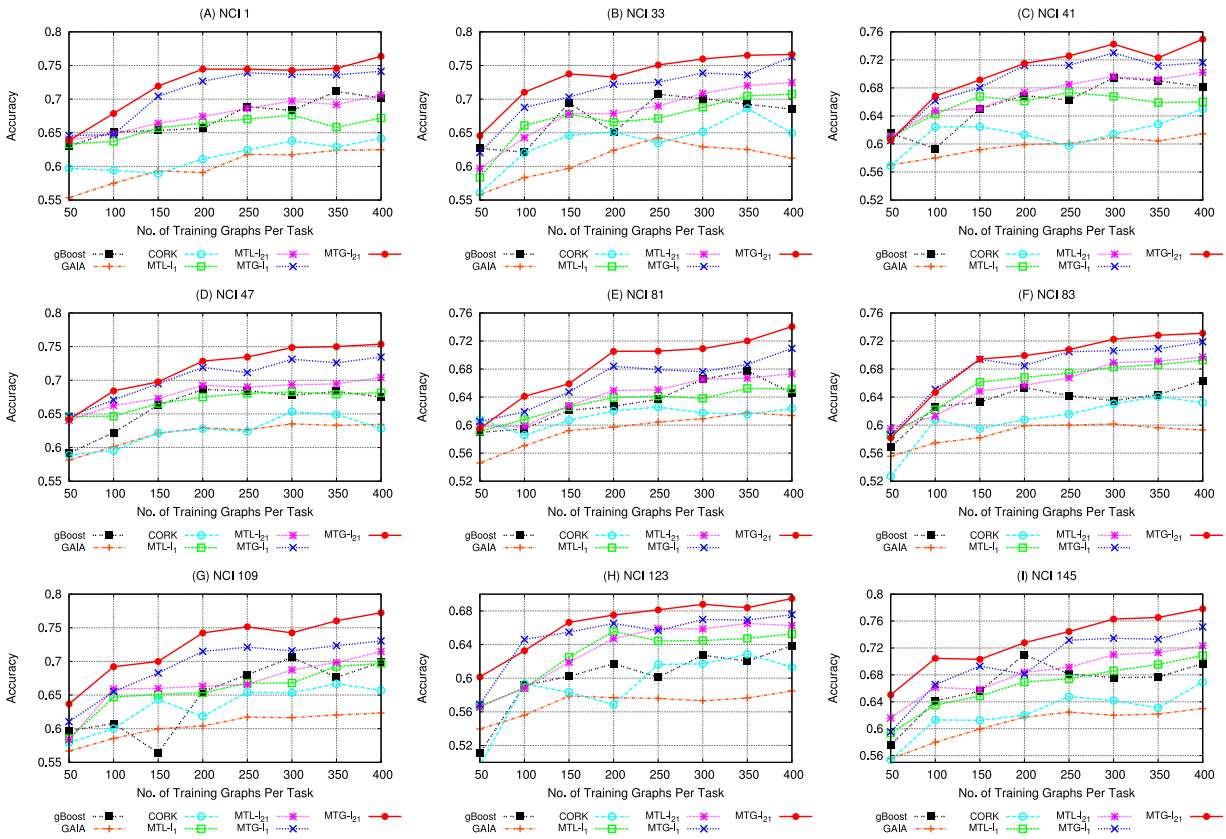

Fig. 3. The classification accuracy of each single task *w.r.t.* the number of training graphs in each task.

Evidence of Carcinogenic Activity (CE), Some Evidence of Carcinogenic Activity (SE), Positive (P), Equivocal (E), Equivocal Evidence of Carcinogenic Activity (EE), Inadequate Study of Carcinogenic Activity (IS), No Evidence of Carcinogenic Activity (NE), and Negative (N). Similar to [27], we set {CE, SE, P} as positive labels, and {NE, N} as negative labels. To formulate an MTG dataset, we randomly split 417 compounds into four equal and disjointed subsets. For each subset, we only consider one type of carcinogenicity test as its learning task. The subset information is also listed in Table 1.

Comparing methods. In our experiments, we consider two types of baseline method from graph classification and multi-task learning perspectives, as follows:

- *gBoost* simply applies gBoost algorithm [4] to each graph classification task separately, without considering graph samples from other tasks.
- *GAIA* algorithm [15] selects discriminative subgraphs as features using evolutionary computation. An SVM classifier is trained using selected subgraphs.
- *CORK* algorithm [20] mines informative subgraphs by optimizing a submodular quality criterion so that greedy feature selection leads to a near-optimal solution. Again, an SVM model is trained using selected features.
- *MTL-ℓ_1 and MTL-ℓ_{21}* first mine a set of frequent subgraphs from all training graphs (we set the minimum support as 0.1, which results in over 2,500 subgraph features on each NCI dataset), and then use those features to transfer each graph dataset into vector format. We then apply traditional Multi-task Learning algorithms to the transferred vector datasets. For *MTL-ℓ_1*, ℓ_1 regularization is used. And for *MTL-ℓ_{21}*, $\ell_{2,1}$ regularization is employed, as in [12]. Both methods are implemented with Logistic loss function and are available in the MALSAR toolbox [31].
- *MTG-ℓ_1 and MTG-ℓ_{21}* are our proposed methods, with *MTG-ℓ_1* being regularized by ℓ_1 norm, and *MTG-ℓ_{21}* being regularized by $\ell_{2,1}$ norm.

Note that GAIA, CORK, and gBoost are single task graph classification algorithms. Specifically, GAIA and CORK are filter-based methods while gBoost is an embedded algorithm. All these algorithms can automatically determine the number of discriminative subgraph features.

Unless otherwise specified, the parameters for MTG are set as follows: $K = 15$, and $S_{max} = 15$. $\gamma = 0.01$ is set for both MTL-ℓ_1 and MTG-ℓ_1, $\gamma = 0.02$ is set for both MTL-ℓ_{21} and MTG-ℓ_{21}. Detailed studies of parameters K and γ are reported in Section 5.2.3. For gBoost algorithm, the parameter v is set to 0.2, as it usually achieves good results on both NCI and PTC datasets.

5.2 Experimental Results

5.2.1 Results on NCI Tasks

For NCI multi-task learning, we randomly label a small set of graphs as training graphs for each task, with remaining graphs being used for test. The number of training graphs in each task varies from 50 to 400. We conduct each group of experiment 10 times and report the average accuracies for each task in 10 trials of experiments in Fig. 3. The

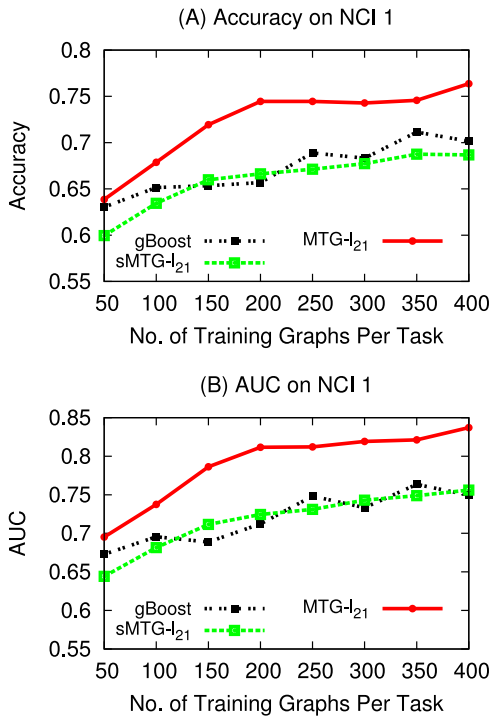

Fig. 4. Comparison of gBoost, MTG-ℓ_{21}, and sMTG-ℓ_{21}, where sMTG-ℓ_{21} only learn with a single task, i.e., NCI 1.

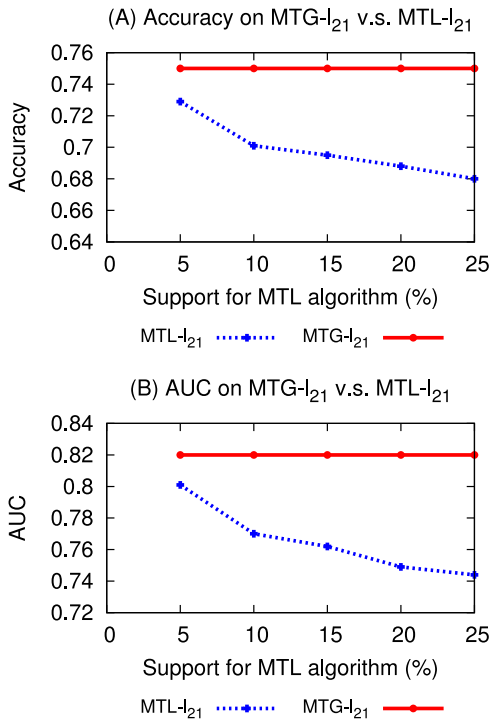

Fig. 5. Comparison of MTL-ℓ_{21} and MTG-ℓ_{21} with respect to different support values.

average AUC values for each task are reported in the supplementary materials.

The results in Fig. 3 show that with the increase of training data for each task, all algorithms achieve performance gains in accuracy values. Over all graph classification tasks, MTG algorithms, including MTG-ℓ_1 and MTG-ℓ_{21}, significantly outperform both STG algorithms (gBoost, GAIA, and CORK) and MTL algorithms for vector data. Among these algorithms, GAIA and CORK are inferior to other algorithms. This may be because: (1) they are filter-based algorithms while others are embedding algorithms. For filter-based algorithms, the selected features may not fit the learning model (SVM in our case) very well. This is a common drawback of filter-based algorithms [47]. (2) the numbers of subgraphs obtained by GAIA (about 50) and CORK (about 20), which are automatically determined by the algorithms, are much smaller than other algorithms (over 200 subgraphs are used in gBoost, MTL, and MTG algorithms). With a small number of features, their models may under-fit the data, resulting in deteriorated performance.

Comparison of gBoost and MTG. As for gBoost, although it is an embedded algorithm, its performance is significantly worse than MTG-ℓ_1 and MTG-ℓ_{21}, mainly because it ignores relevant graphs from similar tasks. To validate the benefit brought by multi-task learning, we further compare gBoost with sMTG-ℓ_{21} algorithm, which is the same as MTG-ℓ_{21} algorithm except that it only considers a single task and ignores relevant tasks, and report the results in Fig. 4. It is clear in Fig. 4 that gBoost is very closed to sMTG-ℓ_{21}. Both gBoost and sMTG-ℓ_{21} are inferior to MTG–ℓ_{21}. When considering multiple tasks, the classification performance of MTG-ℓ_{21}, in terms of accuracy and AUC values, can be significantly improved. This experiment demonstrates the benefits of multi-task graph classification.

TABLE 2
Comparison of Subgraphs Selected by MTL-ℓ_{21} and MTG-ℓ_{21} with 400 Training Samples

Support (%)	# Frequent	# MTL-ℓ_{21}	# MTG-ℓ_{21}	Overlap
25	347	168	216	45
20	518	206	216	52
15	990	411	216	85
10	2,648	855	216	126
5	15,121	2,598	216	178

Comparison of MTG and MTL. Our experimental results in Fig. 3 also show a clear performance gain over MTL algorithms. This is because regardless of whether ℓ_1 or $\ell_{2,1}$ regularization is used, MTL methods will first mine a set of frequent subgraphs as features and then employ multi-task learning techniques for learning and classification. Although these methods can enjoy the benefits of MTL by jointly optimizing related learning tasks, their subgraph mining process is not driven by the multi-task learning objective, and these methods will miss some genuine discriminative subgraphs at the first step. To validate this hypothesis, we report overlap subgraphs selected by MTG-ℓ_{21} and MTL-ℓ_{21} in Table 2 and the classification performance in Fig. 5. Note that MTG and MTL solve the same objective function (Eq. (3)) with different selected subgraph features. A better subgraph set will result in better graph classification results.

In Table 2, we vary the support threshold from 5 to 25. Columns 2-5 indicate the number of frequent subgraphs obtained by the corresponding support value, the number of selected subgraphs learned by MTL-ℓ_{21} and MTG-ℓ_{21}[6],

6. The subgraphs with non-zero weights are selected.

TABLE 3
Accuracies on Nine NCI Graph Classification Tasks *w.r.t* Different Numbers of Training Graphs in Each Task

#	gBoost	GAIA	CORK	MTL-ℓ_1	MTL-ℓ_{21}	MTG-ℓ_1	MTG-ℓ_{21}
50	0.590	0.559	0.564	0.600	0.605	0.609	**0.622**
100	0.617	0.578	0.604	0.632	0.636	0.656	**0.673**
150	0.638	0.595	0.614	0.653	0.653	0.684	**0.697**
200	0.658	0.604	0.615	0.661	0.669	0.701	**0.719**
250	0.665	0.613	0.627	0.666	0.676	0.709	**0.727**
300	0.674	0.611	0.635	0.671	0.690	0.715	**0.735**
400	0.675	0.613	0.642	0.675	0.693	0.715	**0.738**
400	0.676	0.615	0.640	0.680	0.701	0.727	**0.750**

and the overlapping subgraphs selected by both MTL-ℓ_{21} and MTG-ℓ_{21}. Note that the maximum number of subgraphs selected by MTG-ℓ_{21} is $15 \times 15 = 225$ (S_{max} =15 and $K = 15$). Because some weight values are 0, we have 216 valid subgraphs in total.

The result shows that when decreasing support values, the number of frequent subgraphs increases exponentially and the number of overlapping subgraphs also rises. When support value is 10 percent, there are only 126 common subgraphs between MTL-ℓ_{21} and MTG-ℓ_{21}, which means that a large number of useful subgraphs $(216 - 126 = 90)$ are under 10 percent support but they are not selected by MTL-ℓ_{21}. Theoretically, unless we set the absolute support to 1, which is impractical for the exponentially large (or infinite) subgraph space, the MTL algorithm may be subject to risk of missing discriminative subgraphs. The reason is that some subgraph features may be very informative for classification but are infrequent *w.r.t.* the support threshold value, so cannot be discovered by MTL algorithms. In contrast, MTG can overcome this issue because it does not require any support threshold for subgraph mining. As shown in Fig. 5, MTG outperforms MTL because it can explore all potentially discriminative subgraph features for graph classification.

Another interesting finding observed from Fig. 3 is that MTG-ℓ_{21} outperforms MTG-ℓ_1 on most tasks. This is because $\ell_{2,1}$ regularization considers group effect, which is a special group lasso [44] and usually has better performance for group variable selection.

The average results, in terms of accuracy and AUC values, with respect to various training graphs over all tasks are reported in Tables 3 and 4. The results demonstrate that MTG-$\ell_{2,1}$ can achieve significant improvement over gBoost and MTL methods. For instance, it outperforms gBoost and MTL-ℓ_1 with 9.3 and 7.7 percent gain, respectively, in terms of AUC value (400 samples each task).

TABLE 4
AUC Values on Nine NCI Graph Classification Tasks *w.r.t* Different Numbers of Training Graphs in Each Task

#	gBoost	GAIA	CORK	MTL-ℓ_1	MTL -ℓ_{21}	MTG-ℓ_1	MTG-ℓ_{21}
50	0.619	0.559	0.564	0.630	0.645	0.651	**0.667**
100	0.656	0.578	0.604	0.679	0.683	0.713	**0.731**
150	0.682	0.595	0.614	0.707	0.711	0.745	**0.761**
200	0.713	0.604	0.615	0.719	0.730	0.763	**0.785**
250	0.716	0.613	0.627	0.727	0.738	0.773	**0.792**
300	0.727	0.611	0.635	0.734	0.752	0.781	**0.804**
400	0.730	0.613	0.642	0.737	0.758	0.784	**0.812**
400	0.727	0.615	0.640	0.743	0.770	0.795	**0.820**

TABLE 5
Accuracies on PTC Tasks

	gBoost	GAIA	CORK	MTL-ℓ_1	MTL-ℓ_{21}	MTG-ℓ_1	MTG-ℓ_{21}
MR	0.573	0.628	0.634	0.664	0.643	0.594	0.655
FR	0.561	0.615	0.581	0.522	0.547	0.541	0.607
MM	0.640	0.641	0.663	0.648	0.623	0.658	0.677
FM	0.680	0.582	0.716	0.602	0.626	0.671	0.682
Avg.	0.613	0.617	0.649	0.609	0.610	0.616	**0.655**

5.2.2 Results on PTC Tasks

The number of training graphs for each PTC graph classification task is very limited, so instead of varying the training samples for each task (such as for NCI tasks), we conduct 10-fold cross-validation on PTC tasks. In this way, we can reduce the bias of each method caused by having a small number of training samples. The accuracies and AUC values are reported in Tables 5 and 6.

The results in Tables 5 and 6 show that MTG methods achieve considerable performance gains over gBoost and MTL methods for almost all tasks. MTG-ℓ_{21} outperforms other methods for all tasks in terms of AUC. Note that for PTC tasks, AUC values are more important because all tasks have imbalanced class distributions.

5.2.3 Convergence Study and Parameter Analysis

In this subsection, we study the impact of parameters K and γ on algorithm performance.

Impact of K values. To study the role of the K value, which denotes the number of subgraphs selected in each iteration, in algorithm performance, we report the convergence and runtime performance of MTG *w.r.t.* different K values in Table 7. The results show that small K values (e.g., $K = 1$) require a large number of iterations and more system runtime. When K values continuously increase, the number of iterations and runtime drop dramatically because more subgraphs are discovered and included in the feature set in each iteration. For large K values, there is no significant difference in terms of algorithm runtime.

Interestingly, Table 7 shows that although different K values will result in different number of subgraphs ultimately being selected in $\mathcal{F}_1$, the algorithm will always converge to an ε-tolerance optimal solution (ε is used in Algorithm 2) via solving objective function Eq. (5), as shown

TABLE 6
AUC Values on PTC Tasks

	gBoost	GAIA	CORK	MTL-ℓ_1	MTL-ℓ_{21}	MTG-ℓ_1	MTG-ℓ_{21}
MR	0.574	0.502	0.500	0.574	0.586	0.631	**0.656**
FR	0.522	0.526	0.500	0.516	0.517	0.505	**0.591**
MM	0.600	0.500	0.500	0.563	0.597	0.624	**0.671**
FM	0.686	0.530	0.663	0.601	0.623	0.696	**0.702**
Avg.	0.596	0.515	0.541	0.563	0.581	0.614	**0.655**

TABLE 7
Running Statistics *w.r.t* Different K Values for MTG-ℓ_{21}
(50 Training Graphs for Each Task, $S_{max} = 150$)

K	Obj($\mathcal{J}_1$)	#Iter	$\lvert\mathcal{F}_1\rvert$	Accuracy	AUC	Time(s)
1	2.0754	122	122	0.621	0.670	1032
5	2.0814	29	138	0.620	0.667	283
10	2.0861	18	164	0.620	0.669	203
15	2.0779	15	172	0.622	0.667	186
20	2.0771	14	209	0.622	0.669	181

in column 2 of Table 7. As a result, their accuracy and AUC values are very close to each other, regardless of different K values being used in the experiments. This result actually demonstrates the convergence of our algorithms, and assures that K will mainly affect the algorithm runtime performance.

The number of iterations in Table 7 shows that our algorithm has a fast convergence speed. When $K = 15$, it will take 15 iterations to reach convergence. In practice, we found that there is no need to wait until the algorithm reaches convergence for optimal results, so we set the maximum number of iterations $S_{max} = 15$ in our experiments.

Impact of γ values. We vary the regularization parameter γ from 0.005 to 0.5, and report the results in Table 8, where the sparsity denotes the percentage of zero elements in the final weight matrix $\boldsymbol{W}$. The results show that increasing γ values will result in increased sparsity, because ℓ_1 norm regularizes more elements to be 0. For small γ values (from 0.005 to 0.05), the accuracy and AUC values have minor differences, but for very large γ values ($\gamma = 0.5$), the regularization term dominates the objective function Eq. (3), with no subgraph being used for classification, and results in poor AUC values. Similar results are also observed in the MTG-ℓ_{21} algorithm.

5.2.4 Runtime Efficiency Study

In this subsection, we investigate the pruning efficiency of MTG in reducing the search space [Theorems 2 and 3 in Section 4.5] for subgraph feature exploration. Because the whole subgraph search space is exponentially large (or infinite), it is challenging to assess the pruning effectiveness of MTG. Accordingly, we introduce a threshold value min_sup, which denotes the minimum frequency of each qualified subgraph feature in the training graph datasets, to bound the number of subgraphs in the search space. In this way, we know the total number of subgraph candidates, as a result of which we can assess the pruning efficiency by checking the percentage of candidates pruned by the pruning process.

TABLE 8
Results *w.r.t.* Different γ Values for MTG-ℓ_1
(50 Training Graphs for Each Task, $S_{max} = 15$)

γ	$\lvert\mathcal{F}_1\rvert$	Sparsity	Accuracy	AUC
0.005	178	0.720	0.613	0.654
0.01	225	0.785	0.609	0.651
0.05	219	0.905	0.606	0.641
0.1	115	0.922	0.588	0.619
0.5	8	1	0.5	0

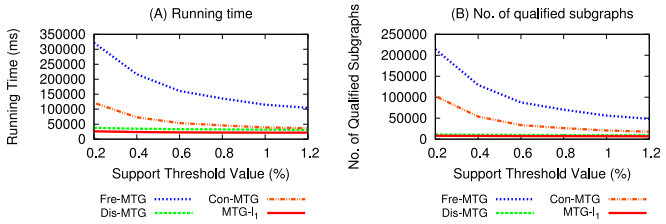

Fig. 6. Pruning effectiveness with different pruning modules on NCI tasks for subgraph mining. (A) Running time. (B) Number of enumerated subgraphs.

In our experiments, the min_sup threshold value, together with Theorems 2 and 3, are used for pruning the search space on step 13 of Algorithm 2. Then our MTG is compared with the following baselines:

- *Fre-MTG*. this method only uses the support threshold min_sup to prune the search space [in step 13 of Algorithm 2], with Theorems 2 and 3 being discarded. In other words, the multi-task driven subgraph mining procedure is reduced to a classical frequent subgraph mining problem.
- *Dis-MTG*. this method uses the support threshold min_sup and the discriminative score bound (Theorem 2) to prune the search space.
- *Con-MTG*. this method uses support threshold min_sup and the conditional score bound (Theorem 3) to prune the search space.

The experimental results in Fig. 6A show that with the increase of the support threshold value min_sup, all methods experience reduced running time. This is because that a large support value will result in a small number of subgraph features (Fig. 6B). Among all compared methods, Fre-MTG consumes much more time than other methods because there is no pruning process to help reduce the search space.

By sequentially including the upper-bounds of the discriminative score (Dis-MTG) and conditional score (Con-MTG) in the pruning process, algorithm runtime is reduced significantly. For instance, when using a small threshold 0.2 for NCI tasks, it only takes about 28,000 ms for MTG to mine the optimal subgraphs, whereas Fre-MTG requires about 330,000 ms. MTG algorithm is an order of magnitude faster than Fre-MTG, which shows the significant pruning efficiency of MTG.

The results in Fig. 6 also reveal that discriminative score bound is more effective than conditional score bound. Note that these two bounds mainly affect runtime efficiency, but will not influence the accuracy of the algorithm. This is because these two bounds can safely prune unpromising subgraphs, as proved in Theorems 2 and 3. If these two pruning bounds are removed, the algorithm will fail because without a support threshold value, the number of subgraph candidates will grow exponentially.

Our runtime efficiency study suggests that MTG is not only efficient in pruning the subgraph feature space to find high quality subgraph features, it can also carry out subgraph feature exploration without requiring a minimum support threshold value min_sup. As a result, it is efficient and effective in finding discriminative subgraph features for multi-task graph classification.

6 Conclusion

In this paper, we formulated a unique multi-task graph classification problem. Our goal is to combine multiple graph classification tasks as one learning objective for all tasks to achieve maximum classification accuracy. We argued that due to the inherent complexity of the graph data and the costs involved in the labeling process, many graph classification tasks have a very limited number of training samples. By unifying multiple tasks to guide the subgraph feature exploration and the succeeding learning process, multi-task graph classification has clear advantages in finding good subgraph features and avoiding overfitting, compared to models learned from single tasks alone. In this paper, an MTG algorithm is proposed which combines all tasks as a jointly regularized function. The joint regularization ensures that the inclusion of subgraph features can result in minimized regularization loss, which in turn leads to optimal learning models. Two Branch-and-bound pruning rules are also proposed to prune the search space. Experiments and comparisons on real-world data confirm the performance of our algorithms.

Acknowledgments

The authors thank anonymous reviewers for their constructive comments. This work was supported in part by ARC through DP140102206, DP140100545 and LP120100566, by the program for professor of special appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning, and by NSF through grants III-1526499, CNS-1115234, and OISE-1129076, and Google Research Award.

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] H. Cheng, D. Lo, Y. Zhou, X. Wang, and X. Yan, "Identifying bug signatures using discriminative graph mining," in *Proc. 18th Int. Symp. Softw. Testing Anal.*, 2009, pp. 141–152.

[3] S. Pan, X. Zhu, C. Zhang, and P. S. Yu, "Graph stream classification using labeled and unlabeled graphs," in *Proc. 28th Int. Conf. Data Eng.*, 2013, pp. 398–409.

[4] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda, "gboost: A mathematical programming approach to graph classification and regression," *Mach. Learning*, vol. 75, pp. 69–89, 2009.

[5] H. Fei and J. Huan, "Boosting with structure information in the functional space: An application to graph classification," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Washington DC, USA, 2010, pp. 643–652.

[6] H. Kashima, K. Tsuda, and A. Inokuchi, "Kernels for graphs," in *Kernel Methods in Computational Biology*, B. Schölkopf, K. Tsuda, and J. P. Vert, Eds. Cambridge MA, USA: MIT Press, 2004.

[7] X. Kong and P. Yu, "Semi-supervised feature selection for graph classification," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 793–802.

[8] O. Ivanciuc, "Chemical graphs, molecular matrices and topological indices in chemoinformatics and quantitative structure-activity relationships," *Current Comput. Aided Drug Designs*, vol. 9, no. 2, pp. 153–163, 2013.

[9] P. Anchuri, M. Zaki, O. Barkol, S. Golan, and M. Shamy, "Approximate graph mining with label costs," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 518–526.

[10] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *Proc. IEEE Int. Conf. Data Mining*, 2002, p. 721.

[11] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 109–117.

[12] A. Evgeniou and M. Pontil, "Multi-task feature learning," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 19, 2007, pp. 41–48.

[13] J. Vogelstein, W. Roncal, R. Vogelstein, and C. Priebe, "Graph classification using signal-subgraphs: Applications in statistical connectomics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 7, pp. 1539–1551, Jul. 2013.

[14] S. Pan and X. Zhu, "Graph classification with imbalanced class distributions and noise," in *Proc. 23rd Int. Joint Conf. Artif. Intell.*, 2013, pp. 1586–1592.

[15] N. Jin, C. Young, and W. Wang, "GAIA: Graph classification using evolutionary computation," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2010, pp. 879–890.

[16] S. Pan, J. Wu, X. Zhu, G. Long, and C. Zhang, "Finding the best not the most: Regularized loss minimization subgraph selection for graph classification," *Pattern Recog.*, vol. 48, no. 11, pp. 3783–3796, 2015.

[17] 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.

[18] X. Yan, H. Cheng, J. Han, and P. S. Yu, "Mining significant graph patterns by leap search," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2008, pp. 433–444.

[19] S. Ranu and A. Singh, "Graphsig: A scalable approach to mining significant subgraphs in large graph databases," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2009, pp. 844–855.

[20] M. Thoma, H. Cheng, A. Gretton, J. Han, H. Kriegel, A. Smola, L. Song, P. Yu, X. Yan, and K. Borgwardt, "Near-optimal supervised feature selection among frequent subgraphs," in *Proc. SIAM Int. Conf. Data Mining*, 2009, pp. 1076–1087.

[21] X. Kong and P. S. Yu, "Multi-label feature selection for graph classification," in *Proc. 10th Int. Conf. Data Mining*, 2010, pp. 274–283.

[22] J. Wu, Z. Hong, S. Pan, X. Zhu, and C. Zhang, "Multi-graph-view learning for graph classification," in *Proc. Int. Conf. Data Mining*, 2014, pp. 590–599.

[23] J. Wu, S. Pan, X. Zhu, Z. Cai, and C. Zhang, "Multi-graph-view learning for complicated object classification," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 3953–3959.

[24] J. Wu, S. Pan, X. Zhu, and Z. Cai, "Boosting for multi-graph classification," *IEEE Trans. Cybern.*, vol. 45, no. 3, pp. 430–443, Mar. 2015.

[25] J. Wu, X. Zhu, C. Zhang, and P. Yu, "Bag constrained structure pattern mining for multi-graph classification," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2382–2396, Oct. 2014.

[26] J. Wu, Z. Hong, S. Pan, X. Zhu, C. Zhang, and Z. Cai, "Multi-graph learning with positive and unlabeled bags," in *Proc. SIAM Int. Conf. Data Mining*, 2014, pp. 1586–1592.

[27] T. Kudo, E. Maeda, and Y. Matsumoto, "An application of boosting to graph classification," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2004, pp. 729–736.

[28] S. Pan, J. Wu, X. Zhu, and C. Zhang, "Graph ensemble boosting for imbalanced noisy graph stream classification," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 940–954, May 2015.

[29] S. Pan, J. Wu, and X. Zhu, "Cogboost: Boosting for fast cost-sensitive graph classification," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 11, pp. 2933–2946, Nov. 2015.

[30] Y. Zhang and D.-Y. Yeung, "Multi-task boosting by exploiting task relationships," in *Proc. ECML/PKDD*, 2012, pp. 697–710.

[31] J. Zhou, J. Chen, and J. Ye, *MALSAR: Multi-Task Learning via Structural Regularization*. Tempe, AZ, USA: Arizona State Univ., 2012.

[32] H. Fei and J. Huan, "Structured feature selection and task relationship inference for multi-task learning," *Knowl. Inf. Syst.*, vol. 35, no. 2, pp. 345–364, 2013.

[33] Y. Zhang and D.-Y. Yeung, "A convex formulation for learning task relationships in multi-task learning," in *Proc. Conf. Uncertainty Artif. Intell.*, 2010, pp. 733–442.

[34] Z. Hong, X. Mei, D. Prokhorov, and D. Tao, "Tracking via robust multi-task multi-view joint sparse representation," in *Proc. IEEE Int. Conf. Comput. Vision*, 2013, pp. 649–656.

[35] J. Liu, S. Ji, and J. Ye, "Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization," in *Proc. Conf. Uncertainty Artif. Intell.*, 2009, pp. 339–348.

[36] B. Bakker and T. Heskes, "Task clustering and gating for bayesian multitask learning," *J. Mach. Learning Res.*, vol. 4, pp. 83–99, 2003.

[37] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[38] X. Shi, X. Kong, and P. S. Yu, "Transfer significant subgraphs across graph databases," in *Proc. SIAM Int. Conf. Data Mining*, 2012, pp. 552–563.

[39] C. Fang and D. Rockmore, "Multi-task metric learning on network data," in *Proc. 19th Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining*, 2015, pp. 317–329.
[40] A. Rakotomamonjy, R. Flamary, and F. Yger, "Learning with infinitely many features," *Mach. Learning*, vol. 91, no. 1, pp. 43–66, 2013.
[41] S. Perkins, K. Lacker, and J. Theiler, "Grafting: Fast, incremental feature selection by gradient descent in function space," *J. Mach. Learning Res.*, vol. 3, pp. 1333–1356, 2003.
[42] X. Wu, K. Yu, W. Ding, and X. Zhu, "Online feature selection with streaming features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 5, pp. 1178–1192, May 2013.
[43] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Royal Statist. Soc. Ser. B*, vol. 73, pp. 273–282, 1996.
[44] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. Royal Statist. Soc., Ser. B*, vol. 68, pp. 49–67, 2006.
[45] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, "Optimization with sparsity-inducing penalties," *Foundations TrendsMach. Learning*, vol. 4, no. 1, pp. 1–106, 2012.
[46] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. New York, NY, USA: Springer, 2004.
[47] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learning Res.*, vol. 3, pp. 1157–1182, 2003.

Shirui Pan received the master's degree in computer science from Northwest A&F University, Yangling, Shaanxi, China, in 2011. Since September 2011, he has been working toward the PhD degree at the Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney. His research interests include data mining and machine learning.

Jia Wu received the bachelor's degree in computer science from the China University of Geosciences (CUG), Wuhan, China, in 2009. Since September 2009, he has been working toward the PhD degree under the Master-Doctor combined program in computer science at CUG. He is also working toward the PhD degree in Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney. His research interests include data mining and machine learning.

Xingquan Zhu (SM'12) received the PhD degree in computer science from Fudan University, Shanghai, China. He is currently an associate professor in the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, and a distinguished visiting professor (eastern scholar) at the Shanghai Institutions of Higher Learning. His research interests mainly include data mining, machine learning, and multimedia systems. Since 2000, he has published more than 200 refereed journal and conference papers in these areas, including two Best Paper Awards and one Best Student Paper Award. He is a senior member of the IEEE.

Chengqi Zhang (SM'95) received the PhD degree from the University of Queensland, Brisbane, Australia, in 1991, and the DSc degree (higher doctorate) from Deakin University, Geelong, Australia, in 2002. Since December 2001, he has been a professor of information technology with the University of Technology, Sydney (UTS), Sydney, Australia, and he has been the Director of the UTS Priority Investment Research Centre for Quantum Computation and Intelligent Systems since April 2008. His research interests mainly focus on data mining and its applications. He is a general co-chair of KDD 2015 in Sydney, the local arrangements chair of IJCAI-2017 in Melbourne, a fellow of the Australian Computer Society, and a senior member of the IEEE.

Philip S. Yu (F'93) is a distinguished professor in computer science at the University of Illinois at Chicago and also holds the Wexler Chair in information technology. He spent most of his career at IBM, where he was a manager of the Software Tools and Techniques group at the Watson Research Center. His research interests include big data, including data mining, data stream, database, and privacy. He has published more than 780 papers in refereed journals and conferences. He holds or has applied for more than 250 US patents. He is a fellow of the ACM and the IEEE. He is the editor-in-chief of the *ACM Transactions on Knowledge Discovery from Data*. He is on the steering committee of the IEEE Conference on Data Mining and ACM Conference on Information and Knowledge Management and was a member of the IEEE Data Engineering steering committee. He was the editor-in-chief of the *IEEE Transactions on Knowledge and Data Engineering* from 2001 to 2004.