Incremental Subgraph Feature Selection for Graph Classification

Haishuai Wang, Peng Zhang, Xingquan Zhu, *Senior Member, IEEE*, Ivor Wai-Hung Tsang, Ling Chen, Chengqi Zhang, *Senior Member, IEEE*, and Xindong Wu, *Fellow, IEEE*

Abstract—Graph classification is an important tool for analyzing data with structure dependency, where subgraphs are often used as features for learning. In reality, the dimension of the subgraphs crucially depends on the threshold setting of the frequency support parameter, and the number may become extremely large. As a result, subgraphs may be incrementally discovered to form a feature stream and require the underlying graph classifier to effectively discover representative subgraph features from the subgraph feature stream. In this paper, we propose a *primal-dual incremental subgraph feature selection* algorithm (*ISF*) based on a max-margin graph classifier. The ISF algorithm constructs a sequence of solutions that are both primal and dual feasible. Each primal-dual pair shrinks the dual gap and renders a better solution for the optimal subgraph feature set. To avoid bias of ISF algorithm on short-pattern subgraph features, we present a new *incremental subgraph join feature selection* algorithm (*ISJF*) by forcing graph classifiers to join short-pattern subgraphs and generate long-pattern subgraph features. We evaluate the performance of the proposed models on both synthetic networks and real-world social network data sets. Experimental results demonstrate the effectiveness of the proposed methods.

Index Terms—Graph classification, incremental subgraphs, feature selection

1 Introduction

Graph classification is an important tool for social network and biological data analysis, where the objective is to learn and classify objective represented in graph structure. For example, chemical compounds can be represented in graph formats, predicting chemical compound activities in bioassay tests is a known graph classification problem. The main challenge in classifying graph data, compared to classifying data with feature-vector representation, is that graphs do not have readily available features, so existing classification models are inapplicable. Accordingly, many research exists to mine frequent subgraphs [3], [4] as features and convert a graph into a feature vector by examining the occurrence of selected subgraphs.

In reality, the number of subgraphs crucially depends on the setting of the frequent pattern mining threshold. With a very small threshold value, the dimension of the subgraphs may be extremely large. For example, in *cascade outbreak prediction in social networks* [36], each cascade data record can be regarded as an acyclic graph that describes the path of information propagation in a social network. In Fig. 1, two directed networks (with green and yellow nodes, respectively) show two cascades. Although both cascades start from a seed node, the cascade with green nodes propagates to a large number of nodes (i.e., graph labeled as outbreak), whereas the cascade with yellow nodes remains steady or may die off (i.e., graph labeled as non-outbreak). Graph classification can then be used for cascade outbreak prediction to identify outbreak cascades from non-outbreak cascades. The general idea is to extract subgraphs as features by using frequent subgraph pattern mining.

When mining subgraph as features, the number of frequent subgraphs is sensitive to the setting of the support parameter. As shown in Fig. 2, the dimension of the subgraph features increases exponentially as the frequent pattern threshold parameter *Supp* decreases. For example, when the parameter is 50, the number of discovered subgraph features is more than $8 * 10^5$!

The reality of high dimensional or potentially *infinite* subgraph features motivates the need to select a small number of representative subgraph features. In machine learning, high dimensional features are a common challenge and a large number of feature selection models [9], [28], [29] have been proposed to reduce data dimensionality. However, these models are incapable of handling infinite subgraph features.

Subgraph feature selection [2], [5] has been proposed in the literature to combine substructure mining and feature learning in graph data. For example, frequent substructure mining methods, such as AGM [3] and gSpan [4], are used to enumerate frequently appearing subgraph patterns, with each subgraph corresponding to a feature, so existing machine learning methods (e.g., SVM) can be applied to the converted feature space. These works, however, can only handle low dimension subgraph features extracted from

- *H. Wang, P. Zhang, I.W.-H. Tsang, L. Chen, and C. Zhang are with the Centre for Quantum Computing and Intelligent Systems, University of Technology Sydney, NSW 2007, Australia. E-mail: haishuai.wang@student.uts.edu.au, {peng.zhang, ivor.tsang, ling.chen, 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 the School of Computer Science, Fudan University, Shanghai 200433, China. E-mail: xzhu3@fau.edu.*
- *X. Wu is with the School of Computing and Informatics, University of Louisiana at Lafayette, 301 East Lewis Street, Lafayette, LA 70503. E-mail: xzhu3@fau.edu.*

Manuscript received 12 Nov. 2015; revised 30 June 2016; accepted 23 Sept. 2016. Date of publication 11 Oct. 2016; date of current version 5 Dec. 2016. Recommended for acceptance by F. Bonchi.
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.2016.2616305

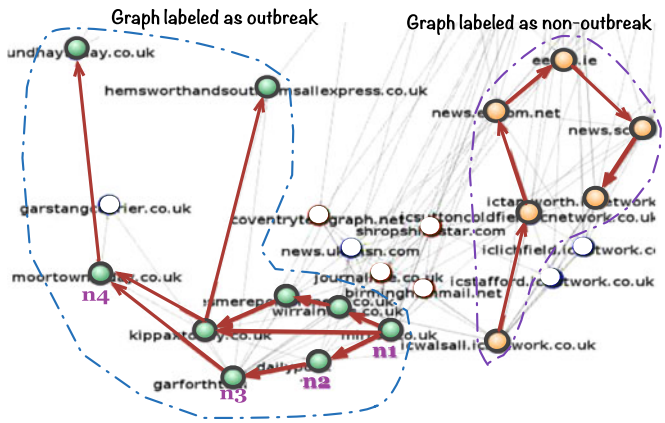


Fig. 1. An example of an information diffusion network. The information propagation cascade can be regarded as a graph. The cascade on the left (the green nodes with bold edges) quickly grows and propagates to an increasing number of nodes (i.e., outbreak), whereas the cascade on the right (the yellow nodes with bold edges) remains steady and is therefore a non-outbreak cascade. Cascade outbreak prediction aims to build a graph classification model to accurately distinguish outbreak cascades from non-outbreak cascades.

small graphs. For instance, a recent work [5] used only 4,337 subgraph features in the experiment. As a result, existing methods are inapplicable to big graph data such as social network cascade data where the number of connected subgraphs can be extremely large.

Online incremental feature selection [6] was recently proposed to select features from high dimensional feature streams. For example, a recent work [9] involved tolerance of information for selecting incremental features and presented a method for selecting strongly relevant and non-redundant features on the fly. However, this method is also inapplicable to incremental subgraph features where the challenges of subgraph pattern mining and incremental feature selection are tangled. In addition, the method in [9] assumes that the prior knowledge on the structure of the feature space is known a priori, so heuristic rules can be applied. This strong assumption does not hold in general graph and social network settings, neither.

1.1 Challenges

Motivated by the above observations, our research mainly aim to solve the following challenges:

How to build a classifier from graphs with an high dimensional number of subgraph features, and how to design an efficient algorithm to rapidly solve the graph classification. The high dimensional number of features makes existing classifiers either inapplicable or ineffective. Numerous feature selection models have been proposed to select sparse features by filter, wrapper, or embedding approaches. However, these feature selection methods are inefficient in high dimensional subgraph feature space. For example, when using l_1 regularization, 1 TB memory is needed to store the feature coefficients of 10^{12} features.

How to join short-pattern subgraphs and design a feature selection model that prefers to select long-pattern subgraph features. We observe that the feature space is dominated by short-pattern subgraphs due to the *downward closure property* of the frequent subgraph mining algorithms [37], [44], i.e., short-pattern subgraph features are always more frequent

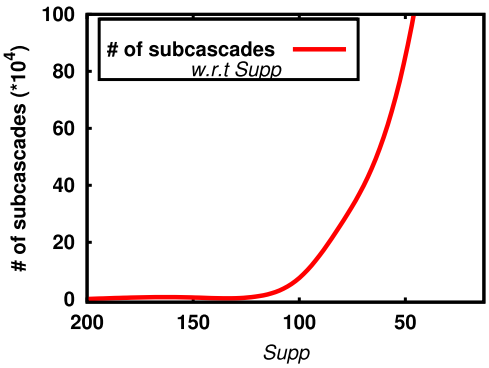


Fig. 2. The number of frequent subgraphs *w.r.t.* the support threshold value in frequent pattern mining. The cascade data, containing about 2.76 million cascades and 3.3 million nodes, are obtained from the SNAP data set (http://snap.stanford.edu/infopath/data.html/). When the parameter *Supp* is 50, the number of discovered subgraph features is more than $8 * 10^5$!

than long-pattern interesting subgraphs. To discover long-pattern subgraphs buried under a huge number of short-pattern subgraphs, we need to systematically design a new feature selection method. Generally, a small value of the frequency support threshold is employed such that all candidate subgraphs can be preserved in the feature space. However, this method will unavoidably generate an exponential number of short-pattern subgraphs which flood interesting long-pattern subgraph features. While there are many possible ways to join short-pattern subgraphs, the join rules between subgraph fragments need to be established for efficient computation. New constraints are required to force traditional max-margin based graph classifiers to select long-pattern subgraph features.

How to evaluate the performance of the proposed method. Both real-world and synthetic data sets are required to compare the proposed method with existing methods.

1.2 Our Work and Contributions

In this paper, we aim to address discriminative subgraph features selection from high dimensional subgraph feature space for max-margin graph classification (Section 4.1). Our research extends the max-margin graph classifier to handle high dimensional incremental subgraph features using a primal-dual subgraph feature selection which continuously selects subgraph features that are both primal and dual feasible (Section 4.2). Because the primal-dual subgraph feature selection algorithm converges quickly and tends to select short-pattern subgraphs, we further propose a long-pattern driven subgraph feature selection model for selecting interesting long-pattern subgraphs (Section 4.3). In experiments, we test the methods on both synthetic and real-world data sets. The results show that the proposed algorithms can both solve the incremental subgraph feature problem and select discriminative long-pattern subgraph features.

The major contribution of the paper is threefold:

- We study the problem of graph classification with incremental subgraph features. We first propose a general max-margin graph classifier, based on which we propose a *primal-dual incremental subgraph feature selection* algorithm. The incremental algorithm

constructs a sequence of solutions that are both primal and dual feasible. Each primal-dual pair shrinks the primal-dual gap and renders a better solution towards the optimal.
- We propose a new Incremental Subgraph Join Feature selection algorithm (ISJF for short). ISJF adds a new constraint on the max-margin graph classifier and forces the classifier to select long-pattern subgraphs by joining short-pattern subgraph features.
- The performance of the algorithms is validated on four synthetic networks and two real-world networks (DBLP graph data and social network cascade data set with 2.76 million information cascades). The results show that the proposed incremental algorithm enjoys the merit of early prediction which is more than 400 seconds faster than existing models for cascading outbreak prediction.

The rest of the paper is organized as follows. Section 2 surveys related work. Section 3 introduces the preliminaries including important definitions. Section 4 discusses the new classification model and extends the classification model into long-pattern feature mining. Section 5 conducts experiments and Section 6 concludes the paper.

2 Related Work

The increasing availability of networked data is creating great potential for knowledge discovery from graph data. Generally, real-world network graphs tend to be big and complex. One difficulty of big graph data is the transfer of graph data into proper formats for learning methods to train classifiers for graph classification.

Existing *graph classification* methods mainly fall into two categories: 1) Distance-based methods, which design a pairwise similarity measure between two graphs, such as graph kernel [10], graph embedding [22], and transformation [23]. The main shortcomings of these type of methods are that it is computationally expensive to calculate graph distance and prediction rules are hard to interpret, because graph features are numerous and implicit. 2) Subgraph feature-based methods, which identify significant subgraphs as signatures for one particular class. Fei and Huan [24] proposed to extract subgraph structural information for classification. Thoma et al. [25] formulated subgraph selection as a combinatorial optimization problem, used heuristic rules, and combined a frequent subgraph mining algorithm, gSpan [4], to find subgraph features. Kong and Yu [26] presented a semi-supervised subgraph feature selection method which uses unlabeled graphs to boost subgraph feature selection for classification. Several boosting methods [5], [27] use individual subgraph features as weak classifiers to build an ensemble for graph classification. None of the existing subgraph-based methods consider high dimensionality in subgraph features as streams.

To handle high dimensional features, many research efforts have been made to address the *incremental feature* challenge. The work in [28] proposed a grafting algorithm based on a stage-wise gradient descent approach for incremental feature selection. However, grafting is ineffective in dealing with incremental features with unknown feature size because choosing a suitable regularization parameter inevitably requires information about the global feature set. The work in [29] studied stream-wise feature selection and proposed two algorithms based on steam-wise regression, information-investing and Alpha-investing. However, this method only considers adding new features and never evaluates the redundancy of selected features after new features are added. The authors in [6] presented a framework based on feature relevance. The work [8] used a new, adaptive complexity penalty, the Information Investing Criterion (IIC), to dynamically adjust the threshold on the entropy reduction required for adding a new feature. The work in [9] involved the tolerance of information for selecting incremental features. These methods require prior knowledge about the structure of the feature space in order to heuristically control the choice of candidate feature selection. In real-world applications, obtaining such prior knowledge is difficult.

The *primal-dual* approach provides a powerful tool for designing approximate online and incremental algorithms. Typical primal-dual methods include primal-dual online linear programming [17], primal-dual online set cover [11], and primal-dual online SVMs [16]. By following the weak duality theorem [19], online learning algorithms quickly converge to approximate solutions by continuously updating both primal and dual feasible solutions which generate tight competitive bounds with respect to off-line algorithms.

Cascade outbreak prediction has been studied extensively. Most existing research efforts can be categorized into two classes: how to detect an outbreak cascade with minimum detection time or minimum affected population [31], and how to predict outbreaks in an early stage according to the cascading behaviors of a given network and dynamic cascades over the network [36]. The former assumers that a portion of nodes in a given cascade can be accessed and used to select some of them as sensors for outbreak detection, whereas the latter aims to predict whether an arbitrary given cascade may outbreak or not. However, both of them use the nodes in a network as features for classification and predication, ignoring the fact that cascades consist of sequential paths, while monitoring nodes in a network is costly because a user may post messages intensively.

High Dimensional Data Learning. High dimensionality is important and challenging because the immense growth of feature dimensionality in data analytics has exposed the inadequacies of many existing methodologies [43]. Directly learning a classifier from high dimensional subgraphs is infeasible. So far, many feature selection methods have been proposed to transform high dimensional data into a lower space representation while preserving the intrinsic data structure. The existing feature selection methods are often categorized as filter, wrapper, and embedding approaches [38]. Among the above sparsity induced methods, l_1 regularization has been popularly used in the literature [42]. However, l_1 regularization is inadequate in this work because it is inefficient when the data dimension is ultra-high. For example, 1 TB memory is needed to store the weight vector w when the data dimension is 10^{12}. Moreover, feature selection bias inevitably exists in the l_1 norm, and different levels of sparsity can be achieved by changing the regularization parameter C.

Data stream mining is one of the important data mining tasks. Existing data stream mining algorithms [1], [12], [39], [40], [41] focus on the challenges of high dimensional stream

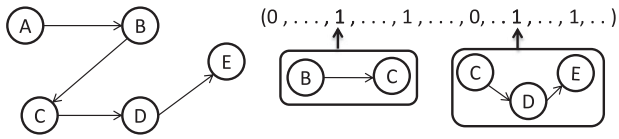


Fig. 3. Subgraph features. The graph (left) is converted into a binary feature vector (right) by examining the existence of subgraph features. The feature vector can be processed by traditional classifiers such as SVMs.

records, concept-drift, concept-evolution, and feature-evolution. However, none of these stream algorithms touch the problem of incremental subgraph features.

In contrast to incremental feature selection and subgraph mining methods, we have combined streaming feature selection and subgraph extraction to tackle high dimensional subgraph features for graph classification.

3 Preliminaries

A graph $G = (V, E)$ consists of a set of nodes $V = \{1, \ldots, p\}$ and a set of edges $E \subseteq V \times V$. A directed acyclic graph (DAG) is a graph G whose edges are directed and do not contain directed cycles. We use lower-case bold-faced letters to represent vectors and upper-case bold-faced letters to represent matrices. For example, symbol $\mathbf{e}$ represents a unit vector with all entries equal to 1.

Definition 1 (Subgraph). *Consider a directed acyclic graph $G = (V, E)$, $g_i = (V', E')$ is a subgraph of G, i.e., $g_i \subseteq G$, iff (1) $V' \subseteq V$, (2) $E' \subseteq E$. If g_i is a subgraph of G, then G is a supergraph of g_i.*

Definition 2 (Subgraphs Join (SgJ). *Consider two directed acyclic subgraphs g_i and g_j. The vertex sets are $V(g_i) = \{V_\alpha, \ldots, V_\beta\}$ and $V(g_j) = \{V_\alpha{}', \ldots, V_\beta{}'\}$. If $V_\beta = V_\alpha{}'$ or $V_\beta{}' = V_\alpha$, then $g_i \otimes g_j = \{V_\alpha, \ldots, V_\beta, V_\gamma{}', \ldots, V_\beta{}'\}$ is defined as SgJ, where $\otimes$ is a concatenation operation.*

Because subgraph mining often outputs many short-pattern subgraphs, Definition 2 joins these subgraphs to generate long-pattern subgraphs. The difficulty is that the result may be uncertain due to graph isomorphism [7]. In this paper, we only consider joining correlated subgraph features, as shown in Fig. 4. This is because correlated subgraph features have a high probability of generating interesting long-pattern subgraphs. As shown in Fig. 4, the join result is determined based on Definition 2.

In binary classification, the task is to learn a classification boundary from a training graph set $\{(G_k, y_k)\}$, $1 \le k \le n$, where each G_k is a training graph with class label $y_k \in \{-1, +1\}$.

Definition 3 (Subgraph Features). *Let $S = \{g_1, \ldots, g_m\}$ be a set of subgraphs in a training graph and $|S| = m$. Each graph G_k is encoded as an m-dimensional vector $\mathbf{x}_k$ with $\mathbf{x}_k^u (1 \le u \le m)$ denoted by*

$$\mathbf{x}_k^u = I(g_u \subseteq G_k), \ \forall g_u \in S,$$

where $I(\cdot)$ equals 1, if the condition is satisfied; otherwise, it is 0.

We use a simple example in Fig. 3 to explain the generation of the subgraph feature space. Consider a graph $A \to B \cdots \to E$ which contains subgraphs $B \to C$ and $C \to D \to E$, the corresponding elements in the subgraph feature space are set to 1.

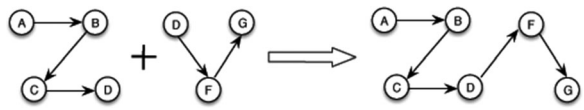


Fig. 4. Joining correlated subgraph fragments.

Given a set of training graphs, the subgraph feature space increases exponentially *w.r.t.* graph size. Therefore, it is impractical to use all subgraphs as features. We use frequent subgraphs to prune trivial subgraph patterns.

A handful of algorithms have been proposed to mine frequent subgraphs $\mathcal{F}_s$ from a set of graphs, such as the Depth-First-Search (DFS) algorithm gSpan [4]. The key idea of gSpan is that each subgraph has a unique DFS Code, which is defined by a lexicographic order during the search process. Two subgraphs are isomorphic 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.

4 Graph Classification

We present a *primal-dual incremental subgraph feature selection* algorithm based on a max-margin graph classifier. First, we assume that the subgraph feature space is finite, based on which we present a *max-margin graph classifier*. Then, we extend the classifier to handle high dimensional incremental features using primal-dual subgraph feature selection.

We introduce a feature scaling vector $\mathbf{d} \in [0, 1]^m$ with $||\mathbf{d}||_1 = \sum_{i=1}^m d_j \le B$ to encourage sparsity. This way, at most B subgraphs are selected. Given a graph G_i, we impose $\sqrt{\mathbf{d}} = [\sqrt{d_1}, \ldots, \sqrt{d_m}]^T$ on its features [13], [14] to a re-scaled example $\hat{\mathbf{x}}_i = \mathbf{x}_i \odot \sqrt{\mathbf{d}}$, where $\mathbf{x}_i \odot \sqrt{\mathbf{d}}$ represents the element-wise product between vectors $\mathbf{x}_i$ and $\sqrt{\mathbf{d}}$. Let $\mathcal{D} = \{d \in \mathbb{R}^m \,|\, ||\mathbf{d}||_1 \le B, d_j \in [0, 1], j = 1, \ldots, m\}$ be the domain of $\mathbf{d}$, the max-margin graph classifier can be formulated as follows:

$$\begin{aligned} \min_{\mathbf{d} \in \mathcal{D}} \min_{\mathbf{w}, \xi, b} \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^n \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) + b) \ge 1 - \xi_i \\ & \xi_i \ge 0, \ i = 1, \ldots, m, \end{aligned} \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^m$ and b determine the classification boundary, ξ_i is the empirical error of $\mathbf{x}_i$, and C is a trade-off parameter. The problem is non-convex *w.r.t.* w and d.

When $\mathbf{d}$ is fixed, Eq. (1) degenerates to a standard SVM model. By introducing the Lagrangian multiplier $\alpha_i \ge 0$ to each constraint $y_i(w^T(x_i \odot \sqrt{d}) + b) \ge 1 - \xi_i$, and setting the derivatives of the Lagrange function to be 0 with respect to parameters $\mathbf{w}, \xi$ and b, we obtain

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}}), \ \sum_{i=1}^n \alpha_i y_i = 0, \ 0 \le \alpha_i \le C.$$

Plugging the above results back into Eq. (1), we obtain the dual form of the original problem as follows:

$$\min_{\mathbf{d} \in \mathbf{D}} \max_{\boldsymbol{\alpha} \in \mathcal{A}} \quad -\frac{1}{2} \left\| \sum_{i=1}^n \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}}) \right\|^2 + \mathbf{e}^T \boldsymbol{\alpha}, \tag{2}$$

where $\mathcal{A} = \{\boldsymbol{\alpha} | \sum_{i=1}^{n} \alpha_i y_i = 0, \boldsymbol{\alpha} \geq 0\}$. Based on the minimax saddle-point theorem [15], we can interchange the order of $\min_{\mathbf{d}\in\mathbf{D}}$ and $\max_{\boldsymbol{\alpha}\in\mathcal{A}}$, and solve the following minimax problem instead,

$$\min_{\boldsymbol{\alpha}\in\mathcal{A}} \max_{\mathbf{d}\in\mathbf{D}} \quad \frac{1}{2}\left\|\sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \odot \sqrt{\mathbf{d}})\right\|^2 - \mathbf{e}^T\boldsymbol{\alpha}. \tag{3}$$

Apparently, the primal problem in Eq. (1) can be equivalently formulated as its dual in Eq. (3). Eq. (3) has only two variables $\mathbf{d}$ and $\boldsymbol{\alpha}$, and it is linear with respect to $\mathbf{d}$ and convex with respect to $\boldsymbol{\alpha}$, which can be solved by the *block coordinate descent algorithm* that alternates between the optimization of $\mathbf{d}$ and $\boldsymbol{\alpha}$.

Given a fixed $\mathbf{d}$, the optimization problem in Eq. (3) is reduced as follows,

(*Optimization 1*: fix $\mathbf{d}$ and solve $\boldsymbol{\alpha}$)

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\left\|\sum_{i=1}^{n} \alpha_i y_i \hat{\mathbf{x}}_i\right\|^2 - \mathbf{e}^T\boldsymbol{\alpha} \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0,\ \alpha_i \geq 0,\ i = 1, \ldots, n, \end{aligned} \tag{4}$$

where $\hat{\mathbf{x}}_i$ is re-scaled by the given $\mathbf{d}$, i.e., $\hat{\mathbf{x}}_i = \mathbf{x}_i \odot \sqrt{\mathbf{d}}$. Due to the sparsity of the scaler $\mathbf{d}$, the above problem can be solved by using standard quadratic programming with a small set of features $\hat{\mathbf{x}}_i$ (A formal solution is given in Appendix B).

When the variable $\boldsymbol{\alpha}$ is determined, we select the number of features B as in Eq. (5).

(*Optimization 2*: fix $\boldsymbol{\alpha}$ and solve $\mathbf{d}$)

$$\begin{aligned} \max_{\mathbf{d}} \quad & \left\|\sum_{j=1}^{n} \alpha_j y_j (\mathbf{x}_j \odot \sqrt{\mathbf{d}})\right\|^2 \\ \text{s.t.} \quad & \sum_{j=1}^{m} d_j \leq B,\ 0 \leq d_j \leq 1,\ j = 1, \ldots, m. \end{aligned} \tag{5}$$

To solve Eq. (5), we define a score function to denote the weight of each feature, i.e.,

$$\mathbf{c}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \in \mathbb{R}^m. \tag{6}$$

Based on the above definition, we have

$$\left\|\sum_{j=1}^{n} \alpha_j y_j (\mathbf{x}_j \odot \sqrt{\mathbf{d}})\right\|^2 = \sum_{j=1}^{m} [c_j(\boldsymbol{\alpha})]^2 d_j.$$

The optimization problem in Eq. (5) can then be converted to a linear programming problem with respect to $\mathbf{d}$ as follows:

$$\begin{aligned} \max_{\mathbf{d}} \quad & \sum_{j=1}^{m} [c_j(\alpha)]^2 d_j \\ \text{s.t.} \quad & \sum_{j=1}^{m} d_j \leq B,\ 0 \leq d_j \leq 1,\ j = 1, \ldots, m. \end{aligned} \tag{7}$$

Eq. (7) can be solved analytically. First, we construct a feasible solution by finding the top B largest scores $[c_j(\alpha)]^2$. Then, we set the number of B scaler d_j to 1 and the remaining $(m - B)$ d_j to 0. Clearly, such a feasible solution is also the optimal solution. The algorithm is given in Algorithm 1.

Algorithm 1 is equivalent to the *cutting-plane algorithm* for solving the semi-infinite programming problem. In each iteration, the cutting-plane algorithm removes nonactive constraints that correspond to redundant features in the primal problem. The algorithm iteratively finds active constraints that heavily violate the KKT condition. In the following, we prove the convergence of Algorithm 1.

Algorithm 1. The Max-Margin Graph Classifier

Input: Graph $\mathbf{G}$, parameters C, B
Output: Graph classifier f
$\boldsymbol{\alpha}^0 \leftarrow 1/C \cdot \mathbf{1}$, $U \leftarrow \emptyset$, $t \leftarrow 0$;
repeat
  Calculate $\mathbf{d}_t$ based on $\boldsymbol{\alpha}_t$ using Eq. (5)
  `// Optimization 2;`
  $U \leftarrow U \cup \mathbf{d}_t$;
  Calculate $\boldsymbol{\alpha}_{t+1}$ based on U using Eq. (4)
  `// Optimization 1;`
  $t \leftarrow t + 1$;
until *Convergence*;
Output $f \leftarrow SVM(U, C)$;

Theorem 1. *Let $\{\boldsymbol{\alpha}_t, \mathbf{d}_t\}$ be a sequence of solutions generated by Algorithm 1. If Algorithm 1 stops at iteration $\{t+1\}$, then $\{\boldsymbol{\alpha}_t, \mathbf{d}_t\}$ is the global optimal solution of Eq. (3).*

Proof. The objective function of Eq. (3) is convex *w.r.t.* $\boldsymbol{\alpha}$ and linear *w.r.t.* $\mathbf{d}$. Thus, Eq. (3) has a global optimum. The algorithm, by iteratively solving Eqs. (6) and (7), will converge to the global optimum. □

The stop criterion is based on the bound of $\mathbf{d}$. Specifically, Algorithm 1 stops when $\mathbf{d}$ becomes stable. In each iteration, the algorithm scans the entire feature space and chooses B features. In the worst case, the algorithm iterates m/B times and selects all m features, i.e., the algorithm takes $O(m^2)$ time at worst. The square time complexity is, unfortunately, unaffordable for ultra-high dimensional data, which is often the case in our problem setting.

4.1 Incremental Subgraph Features

Eq. (1) provides a basic classifier for solving high-dimensional but finite subgraph feature space. To process high dimensional subgraph feature streams, we introduce a *primal-dual incremental subgraph feature selection* algorithm.

Our method is based on the online linear programming of the feature selection function given in Eq. (7). We assume that the constraint matrix is revealed column by column along with the objective function, i.e., the features are processed column by column in a one-scan manner.

In incremental feature selection scenarios, a feature set is split and loaded into memory in a mini-batch manner. At each time window $t = \lceil m\epsilon \rceil$, the incremental algorithm learns both primal and dual feasible solutions. The primal problem is formulated as in Eq. (8) which allows $(1 - \epsilon)$ approximation to the offline solution given by Eq. (9) [20]. Eq. (10) is clearly a small linear program problem defined on the $\lceil m\epsilon \rceil$ features.

(Primal)

$$\begin{aligned} \max_{\mathbf{d}} \quad & \sum_{j=1}^{t} [c_j(\alpha)]^2 d_j \\ \text{s.t.} \quad & \sum_{j=1}^{t} d_j \leq (1-\epsilon)\frac{t}{m}B \\ & 0 \leq d_j \leq 1,\ j = 1, \ldots, t \end{aligned} \tag{8}$$

For Eq. (8), we use $p \in \mathbb{R}^m$ to denote the dual variable. The dual problem of Eq. (8) is as follows:

(Dual)

$$\begin{aligned} \min_{p} \quad & \sum_{i=1}^{m} b_i p_i (1-\epsilon)\frac{t}{m} + \sum_{i=m+1}^{m+t} p_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} p_i + p_{i+j} \geq [c_j(\alpha)]^2,\ j = 1, \ldots, t \\ & p_i \geq 0, 1 \leq i \leq m. \end{aligned} \tag{9}$$

The dual problem converts a high dimensional problem in Eq. (8) into a big constraint problem with respect to dual vector p. For any given p, we define the function $x_t(p)$ denoting dual feasibility as follows:

$$x_t(p) = \begin{cases} 0 & \text{if } b_i \leq p^T \\ 1 & \text{if } b_i > p^T. \end{cases} \tag{10}$$

The incremental algorithm constructs a sequence of solutions that are both primal and dual feasible. Each primal-dual pair shrinks the dual gap and renders a better solution towards the optimal.

In the online problem, at time t, the coefficient $c_j(\alpha)$ is updated, and the algorithm makes a decision x_t. Given the previous decisions $x_1, \ldots, x_{t-1}$, and $c_j(\alpha)^2$ till time t, the tth decision is to select an x_t such that $\sum_{j=1}^{t} x_j \leq B$, $0 \leq x_j \leq 1$. The goal of the online algorithm is to choose x_t such that the objective function $\sum_{t=1}^{m} c_j(\alpha)^2 x_j$ is maximized.

To evaluate the performance of an online algorithm, one approach is based on its performance on the worst-case input, e.g., completely robust to make input uncertainty [17]. This approach leads to gloomy bounds for the online problem: no online algorithm can achieve better than $O(1/m)$ approximation of the optimal offline solution [18]. In our problem settings, subgraph features arrive in a random order, and the total number of features m is known a priori. We consider the average behavior of the online algorithm over random permutations, and can use m to decide the length of history used to learn the dual bounds in the algorithm. In this case, the total number of m is a known priori [20], we can relate the approximate knowledge of m within at most $1 \pm \theta$ multiplicative error without affecting the final results.

The primal-dual incremental subgraph feature selection algorithm is given in Algorithm 2. The primal solution $x_t(p)$ constructed using sample dual solutions in Eq. (21) is a feasible solution to the linear program Eq. (8) with high probability. In fact, the iterative primal solutions constructed using sample dual solutions converge and approach to optimal, e.g., with probability $1-\xi$, $\sum_{t \in N} x_t(p) \geq (1-3\xi)OPT$ given $B \geq \frac{6m\log(n/\xi)}{\xi^3}$, where OPT denotes the optimal objective value for the offline problem [20].

Incremental feature selection is motivated by the limitations of Algorithm 1. First, the algorithm needs to fully scan all the features in each iteration, which requires $O(m^2)$ complexity in the worst case and is therefore unsuitable for big networks. Second, the number of selected features tB increases as the number of iterations t continue, which may become high dimensional. Therefore, we propose a new primal-dual incremental feature selection method.

Algorithm 2 is based on the observation that the optimal solution for the offline linear program is almost entirely determined by the optimal dual solution corresponding to the inequality constraints. The algorithm is $1 - O(\theta)$-competitive and the results can be stated as follows:

Algorithm 2. Primal-Dual Incremental Subgraph Feature Selection for Graph Classification

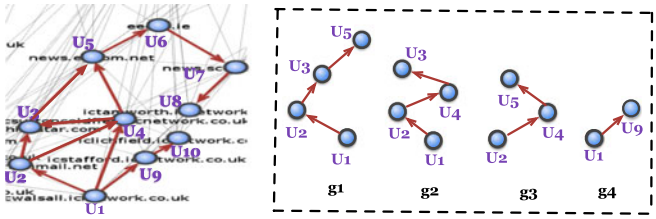
Input: Graph G, parameters C, B, mini-batch size K
Output: classifier c
$\alpha^0 \leftarrow 1/C \cdot \mathbf{1}$, $S \leftarrow \emptyset$, $t \leftarrow 0$;
repeat
  $S \leftarrow$ a mini-batch of K features;
  Calculate top-k candidate features based on α_t using Eq. (21) `// mini-batch scores;`
  for *each feature* x_t *in top-k* **do**
    if $x_t(p) \leq b_i - \sum_{j=1}^{t-1} d_j x_j$ **then**
      $x_t = x_t(p)$
    else
      $x_t = 0$
  Calculate α_{t+1} based on d_t using Eq. (4)
  `// Optimization 1;`
  $t \leftarrow t + 1$;
until *no feature left*;
Output $c \leftarrow SVM(S, C)$;

Because optimization 1 can be solved by using convex quadratic programming, there is polynomial time interior point algorithm in Matlab. The primal-dual incremental feature selection selects a mini-batch and only calculates the scores on the mini-batch. The algorithm firstly selects m features, and then calculates B features which has highest score, and it scans the feature set only once. Therefore, the time complexity of the algorithm is *O(m*BlogB)*. Since B is a small value, the time complexity can be a linear time complexity *O(m)*.

4.2 Long-Pattern Subgraph Features

In fact, the number and size of subgraph features crucially depend on the threshold parameters of frequent subgraph mining algorithms. Any improper setting of the parameters will generate many trivial short-pattern subgraph fragments which dominate the feature space, distort graph classifiers, and flood interesting long-pattern subgraphs.

The primal-dual incremental feature selection converges rapidly; however, the algorithm often gets trapped in generating a large number of short-pattern subgraph features which may flood interesting long-pattern subgraph features. Fig. 5 shows an example in which a long-pattern subgraph feature f_3 is buried under two subgraph features f_1



(a) A social network cascade data set from http://snap.stanford.edu/infopath

(b) Four graphs of cascade data (g1, g2, g3, g4)

	f1: U1→U2	f2: U2→U3	Label
g1	1	1	+1
g2	1	1	+1
g3	0	0	-1
g4	0	0	-1

(c) Subgraph features f1 and f2

	f3: U1→U2→U3	Label
g1	1	+1
g2	1	+1
g3	0	-1
g4	0	-1

(d) Subgraph feature f3

Fig. 5. An illustration of a long-pattern subgraph feature buried under two short-pattern subgraph features in the information cascade data. Consider four graphs $g_1, \ldots, g_4$. g_1 and g_2 from class "+1" while g_3 and g_4 from "−1". Assume we have two short-pattern subgraphs $f_1 : U_1 \rightarrow U_2$ and $f_2 : U_2 \rightarrow U_3$, and a long-pattern subgraph $f_3 : U_1 \rightarrow U_2 \rightarrow U_3$ by joining f_1 and f_2. If one feature is allowed to select for classification, then f_1 or f_2 is likely to be selected, instead of the more interesting f_3.

and f_2. A classifier is likely to choose the two subgraph features instead of the more informative long-pattern subgraph feature which reflects a latent social group. In this section, we design a long-pattern driven algorithm that prefers long-pattern subgraphs as features, with only one scanning of the feature set.

Intuitively, the size of a subgraph compromises the classification accuracy. In graph classification, based on the minimum description length theory, the relationship between the size of a subgraph and the size of the original graph is compromised when subgraphs are used as features.

Formally, we add extra constraints to allow the algorithm to choose long patterns. The new constraint can be stated as: *if two short patterns p_a and p_b are in the active feature set, i.e., $d_{p_a} = 1$ and $d_{p_b} = 1$, then their derived pattern $p_a \otimes p_b$ will be also selected, i.e., $d_{p_a \otimes p_b} = 1$.*

Based on the new constraints, we obtain a new classification model as follows:

$$\begin{aligned} \min_{d \in \mathbf{D}} \min_{w,\epsilon,b} \quad & \frac{1}{2}||w||^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(w^T(x_i \odot \sqrt{d} + b) \leq 1 - \xi_i \\ & \xi \geq 0,\ i = 1, \ldots, m. \\ & d_{p_a \otimes p_b} = 1, \forall\ d_{p_a} = 1 \wedge d_{p_b} = 1. \end{aligned} \tag{11}$$

The new constraints enforce that if two features can be concatenated into a long cascade, then the corresponding new cascade will be set to 1. The algorithm thus tends to select long patterns

$$\begin{aligned} \max_{d} \quad & \sum_{j=1}^{m}[c_j(\alpha)]^2 d_j \\ \text{s.t.} \quad & \sum_{j=1}^{m} d_j \leq B, 0 \leq d_j \leq 1, \\ & d_{p_a \otimes p_b} = 1, \forall\ d_{p_a} = 1 \wedge d_{p_b} = 1. \end{aligned} \tag{12}$$

TABLE 1
Analysis of the New Constraint

	Equal			Improve			Reduce		
X	p_a	P_b	p_{ab}	p_a	P_b	p_{ab}	p_a	P_b	p_{ab}
x_1	1	1	1	1	1	1	1	0	0
x_2	1	1	1	1	1	1	1	0	0
x_3	0	0	0	1	0	0	0	1	0
x_4	0	0	0	0	1	1	0	1	0

TABLE 2
List of the Synthetic and Real-World Data Sets

Data Set	Nodes	Edges / Cascades	Other parameters
Albert-Barabasi	5,000	19,990	$n = 4$
Forest Fire	5,000	21,124	$f = 0.35, b = 0.32$
Small World	5,000	19,996	$\alpha = 4, p = 0.1$
Erdos-Renyi	5,000	6,000	$[0.5, 0.5; 0.5, 0.5]$
DBLP	2,000	162,466	\
MemeTracker	3.3 mil.	27,559,952	\

Eq. (12) can be solved analytically. First, we construct a feasible solution by finding the largest scores $[c_j(\alpha)]^2$. Then, we set the scaler δ_j to 1 and the remaining to 0.

Algorithm 3. Long-Pattern Classifier

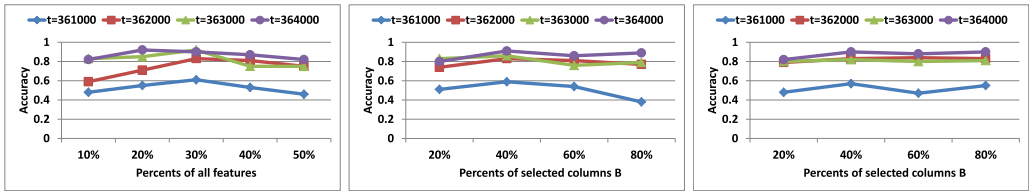
Input: Graph G, parameters C, B, mini-batch size K
Output: classifier c
$\alpha^0 \leftarrow 1/C \cdot \mathbf{1}$, $S \leftarrow \emptyset$, $t \leftarrow 0$;
repeat
  $S \leftarrow$ a mini-batch of K features;
  Calculate candidate top-k features based on α_t using Eq. (21) `// mini-batch scores;`
  Calculate α_{t+1} based on d_t using Eq. (4)
  `// Optimization 1;`
  $t \leftarrow t + 1$;
until *no feature left*;
Output $c \leftarrow SVM(S, C)$;

Table 1 shows three different types of result when a long-pattern subgraph p_{ab} is generated from two short-pattern subgraph fragments p_a and p_b. For example, consider the four training graphs $g_1, \ldots, g_4$, where g_1 and g_2 belong to the same group, while g_3 and g_4 fall into another group. Assume we have obtained two subgraph fragments $P(a) = A \rightarrow B$ and $P(b) = B \rightarrow C$. We can generate a long-pattern subgraph $A \rightarrow B \rightarrow C$ based on Definition 3. The classifier may have three different types of result, Equal, Improve and Reduce. Therefore, we have the following intuitive conclusion.

Theorem 2. *Consider two subgraph fragments $p_a = 1$ and $p_b = 1$, if $p_{ab} = p_a \otimes p_b$, then the generated long-pattern subgraphs p_{ab} can be used to replace the original two patterns p_a and p_b.*

Proof. Evidently, if $p_{ab} = p_a \otimes p_b$, then $p_{ab} = 1$ is the solution of Eq. (13). □

Based on the above analysis, we design the long-pattern driven incremental feature selection algorithm given in Algorithm 2. The algorithm processes data in mini-batches. The algorithm examines the concatenation of short-pattern



(i) ISF: accuracy *w.r.t.* the observed features. (ii) OSF: accuracy *w.r.t.* the selected columns B. (iii) ISF: accuracy *w.r.t.* the selected columns B.

Fig. 6. Parameter study on min-batch size B at each iteration and value k in top_k.

subgraphs and generates long-pattern subgraphs that comply with Theorem 3. The complexity of the algorithm is still $o(m)$ in the worst case.

5 Experiments

We test the proposed algorithms on two real-world networks and four synthetic network data sets. The purpose of the experiments is to: 1) conduct a parameter study for the purpose of choosing optimal parameter settings for our experiments; 2) compare the proposed algorithms with benchmark methods to validate the performance of our method; and 3) test our method on real-world social network applications. The source codes and data sets are available online.[1] The frequent subgraph mining is implemented in Java, the feature generation is implemented in Python, and the optimization algorithms are implemented in Matlab. All experiments are tested on a Linux Ubuntu server with 16*2.9 GHz CPU and 64 G memory.

Data. We use two real-world networks and four synthetic data sets for testing. The data sets are summarized in Table 2.

Real-World Data. MemeTracker data set [21] is downloaded from the SNAP cascade data website, and is the topic-based MemeTracker containing 27.6 million news articles and blog posts from 3.3 million online sources with time-stamps over a one-year period, from March 2011 to February 2012. The data format is as follows, $<meme\ id>; <website\ id>, <timestamp>, \ldots, <website\ \ id>, <timestamp>$. The time-stamp indicates the information arrival time of a node from its parent nodes. We generate information propagation graphs as shown in Fig. 1. We treat each website as a graph node, there is an edge between two nodes if a website forward articles or blogs from another website. Thus, the propagation network forms a graph at a speicific observation time stamp (e.g., 361,000(s)). All the graphs at different time stamps in a cascade have the same label with the cascade (outbreak or non-outbreak). Predicting each graph label forms a graph classification task.

Synthetic Data. We use four well-known models to generate synthetic networks for testing and comparison, namely, the *Erdos Renyi [34], Albert Barabasi [33], Forest Fire [32] and Small World [35]* models.

Erdos Renyi [34] generates random graphs with arbitrary degree distributions. Each edge is included in the graph with a probability p independent of other edges. All graphs with N nodes and L edges have equal probability of $p^L(1-p)^{\binom{N}{2}-L}$. The parameter p is a weighting function. In particular, $p = 0.5$ corresponds to the case in which all $2^{\binom{N}{2}}$ graphs on N vertices are chosen with equal probability.

Albert-Barabasi (Scale-free network) [33] generates random scale-free networks using a preferential attachment mechanism. The network begins with an initial connected network containing β_0 nodes. New nodes are added to the network one at a time. Each new node is connected to $\beta \leq \beta_0$ existing nodes with a probability proportional to the number of links that existing nodes already have. For this model, we need to set parameter n, which denotes the number of edges created by each new node.

Small-world (Watts-Strogatz model) [35] is defined as a network in which the typical distance ζ between two randomly chosen nodes grows proportionally to the logarithm of the number of nodes N in the network, that is $\zeta \propto \log N$. We use parameter α to denote that each node is connected to α nearest neighbors in topology, and p denotes the rewiring probability.

Forest Fire model (Scale-free network) [32] is defined as a cellular automaton on a grid with γ^d cells. γ is the side-length of the grid and d is its dimension. In this model, we need to set the parameters of the forward burning probability f, and the backward burning probability b.

To better simulate real-world network diffusion, we generate synthetic networks under a power-law degree distribution exponent $\alpha = 1.5$, which corresponds to the power-law degree exponent of the MemeTracker network.

Synthetic cascades are generated using the following methods. First, we randomly select a root node r with a non-zero out-degree. The node r is then added to the initially empty list of the infected nodes I and all outgoing edges (r, s) are added to the initially empty FIFO queue of the infected-susceptible node pairs S. We choose an edge from the candidate set each time and calculate the time delay for

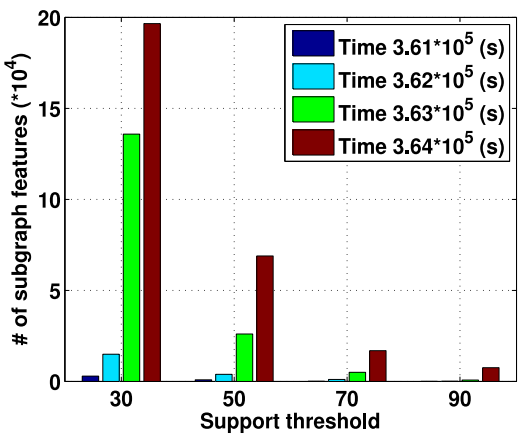
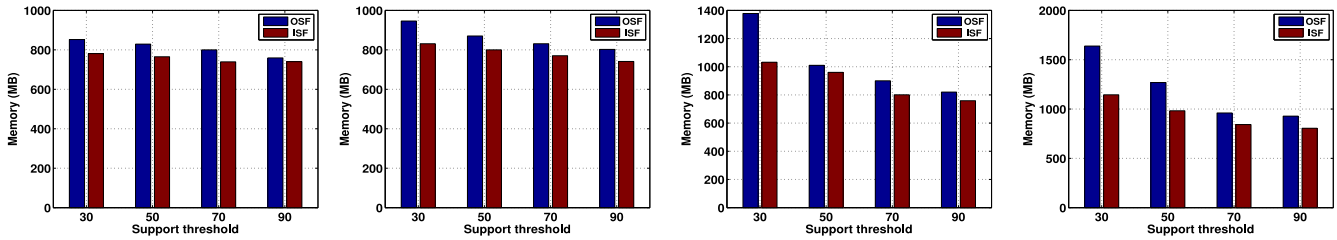


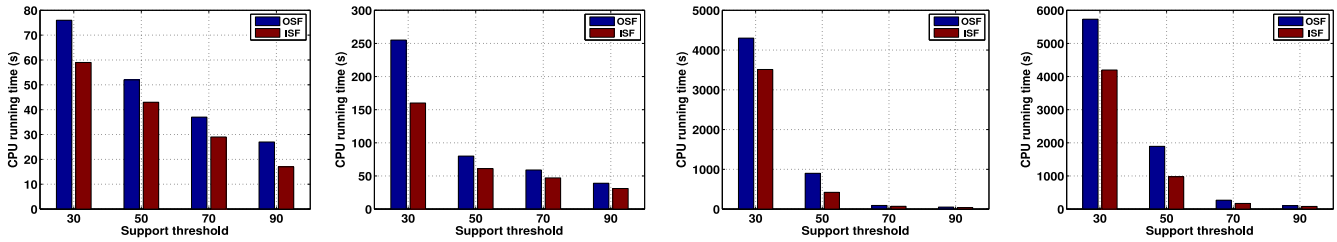
Fig. 7. # of subgraph features *w.r.t.* support threshold.

1. https://github.com/BlindReview/streaming for the source codes and the synthetic data sets. http://snap.stanford.edu/infopath/ for the cascade data set.



(A) Time $3.61 * 10^5$ (s). (B) Time $3.62 * 10^5$ (s). (C) Time $3.63 * 10^5$ (s). (D) Time $3.64 * 10^5$ (s).

Fig. 8. Memory cost *w.r.t.* the support threshold *Supp* under different propagation time stamps.



(A) Time $3.61 * 10^5$ (s). (B) Time $3.62 * 10^5$ (s). (C) Time $3.63 * 10^5$ (s). (D) Time $3.64 * 10^5$ (s).

Fig. 9. Running time *w.r.t.* the support threshold *Supp* under different propagation time stamps.

the edge until the time delay exceeds a given time window (we use [0, 1,000] as the time window). The data are generated by repeating the above steps for each source node.

In the synthetic networks, nodes are continuously included in the network, we separate each cascade by different time stamp (e.g., 100, 200), and each network at a specific time stamp (e.g., time is 100) can be treated as a graph. All the graphs in a cascade have the same label with the cascade (outbreak or non-outbreak). We aim to classify outbreak cascades from non-outbreak ones by using graph classification.

Benchmark Methods. We compare the proposed *ISF* and the Incremental Subgraph Join Feature selection algorithm with the following three methods: 1) Meta Subgraph Feature selection (*MSF*) which randomly selects meta subgraphs (nodes) as features for graph classification [36]. In MSF, each node of a graph is taken as a feature; 2) Off-line Subgraph Feature selection (*OSF*) which loads all features into memory at one time and selects top_k columns (with top k scores) at each iteration until the condition $\|\alpha_{t+1} - \alpha_t\| < \epsilon$ is met, where $\epsilon = 0.001$ in our experiments; 3) Random Subgraph Feature selection (*RSF*) which randomly selects B features for graph classification.

Measures. We compare the algorithms with respect to running time, prediction accuracy (*tp* / (*tp* + *fp* + *fn* + *tn*)), precision(*tp* / (*tp* + *fp*)), recall (*tp* / (*tp* + *fn*)), and the F1 Score (2 * (*precision* * *recall* / (*precision* + *recall*))) to evaluate classification performance, where *tp* is true positive, *fp* is false positive, *fn* is false negative, and *tn* is true negative.

5.1 Parameter Study

We first test the parameters in Eq. (8) *w.r.t.* the number of features B, and the parameter k of the top_k features in OSF and ISF.

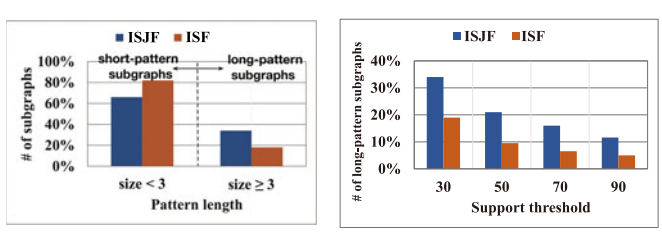
Fig. 6 shows the performance of the two algorithms under different parameter settings on the real-world data.

The mini-batch size B: $B > 0$ represents the portion of selected features *w.r.t.* the number of features at each iteration. If B is too large, the algorithm suffers from large computation and memory cost. In the worst case, B equals to the number of all features which degenerate to the off-line method. Fig. 6 shows that the best prediction accuracy is achieved when B is 30 percent of all features.

The Parameter Top_k. We use the optimal value of B = 30 percent to study parameter k, which indicates the number of features being selected for the next mini-batch of feature selection. Fig. 6 shows that both OSF and ISF have the highest accuracy when k equals to 40 percent of mini-batch B.

5.2 Experimental Results

The Dimension of Subgraph Features and Running Time. Fig. 7 shows the dimension of subgraph features *w.r.t.* the support threshold *Supp*. Figs. 8 and 9 show the memory consumption and running time comparisons for ISF and OSF *w.r.t.* the support threshold under different propagation time stamps. The different propagation time stamps and support thresholds indicate the different dimensions of the subgraph features. The number of features approximates to 2×10^5 when the support threshold is 30 and the propagation time reaches 364,000. Figs. 8 and 9 show that our ISF algorithm can handle high dimensional features faster than OSF. This is because ISF uses a primal-dual subgraph feature selection which continuously selects subgraph features to quickly solve the graph classifier. In contrast, OSF loads



(i) # of subgraph patterns *w.r.t.* pattern length. (ii) # of long-pattern subgraphs *w.r.t.* support threshold.

Fig. 10. Percentage of patterns *w.r.t.* the support threshold and pattern length.

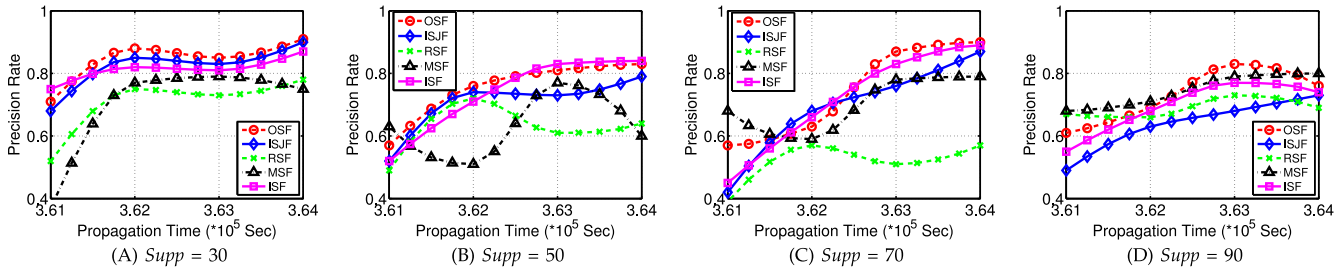


Fig. 11. Precision comparison under different *Supp*.

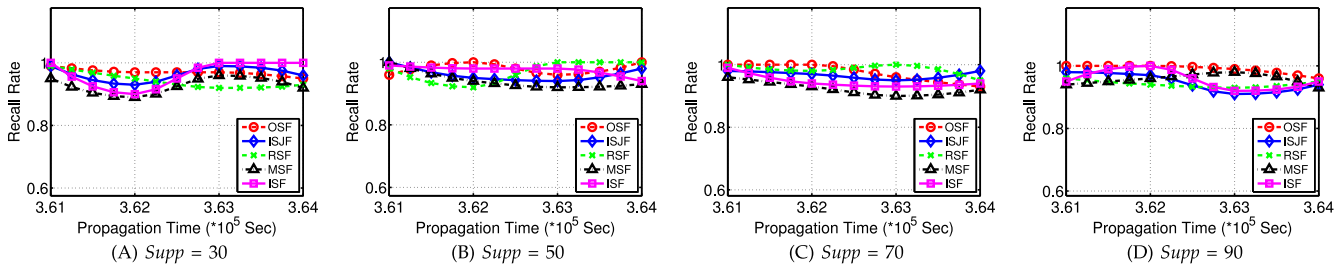


Fig. 12. Recall comparison under different *Supp*.

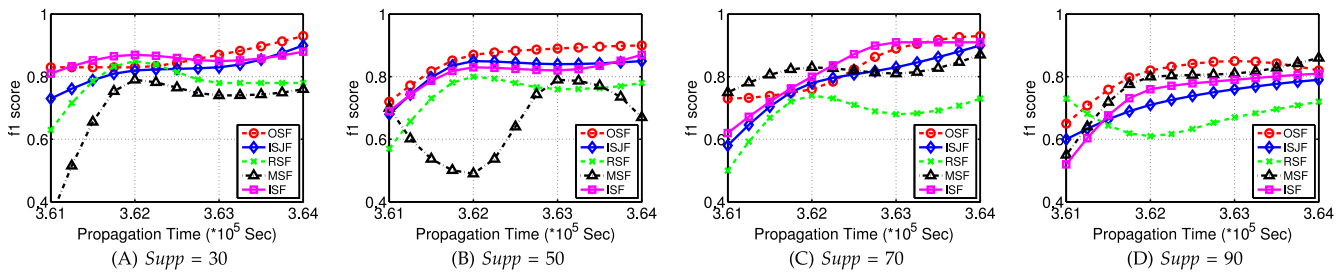


Fig. 13. F1 score comparison under different *Supp*.

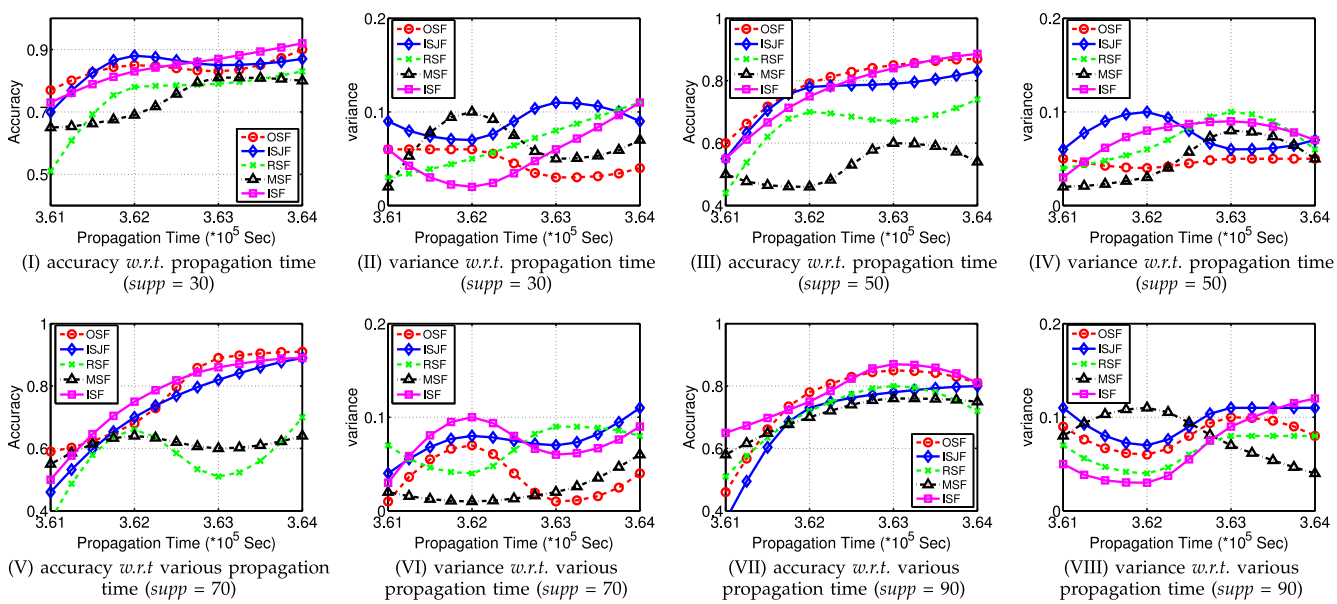


Fig. 14. Accuracy and variance comparisons *w.r.t.* time stamp on the real-world data set.

all features into memory at one time which costs heavy in both memory and time.

Prediction Accuracy. The five algorithms are compared under different support thresholds (30, 50, 70 and 90) in the subgraph mining. The settings for the time stamps are 3.61, 3.62, 3.63 and $3.64 * 10^5$ seconds. We report the performance of ISF, ISJF and three benchmarks on the real-world data set in Figs. 11, 12, 13, and 14 and on the synthetic data sets listed in Table 3. Fig. 10i shows the proportion of short-pattern subgraph features (length < 3) and long-pattern subgraph features (length ≥ 3) when the support is equal to 30. Fig. 10ii shows the comparison of the

TABLE 3
F1 Score under the Parameter Support = 30 on the Four Synthetic Data Sets

Data Sets	Time	OSF	RSF	MSF	ISF	**ISJF**
Albert-Barabasi	100	**0.711 ± 0.146**	0.611 ± 0.070	0.564 ± 0.019	0.682 ± 0.077	0.651 ± 0.126
	300	0.762 ± 0.077	0.700 ± 0.094	0.743 ± 0.026	0.762 ± 0.027	**0.791 ± 0.102**
	500	**0.885 ± 0.037**	0.739 ± 0.094	0.664 ± 0.109	0.785 ± 0.094	0.863 ± 0.082
	700	0.903 ± 0.006	0.667 ± 0.059	0.782 ± 0.011	**0.912 ± 0.028**	0.910 ± 0.006
Erdos Renyi	100	0.750 ± 0.125	0.667 ± 0.178	0.712 ± 0.169	**0.750 ± 0.067**	0.750 ± 0.125
	300	0.824 ± 0.122	0.703 ± 0.067	0.624 ± 0.058	0.817 ± 0.058	**0.824 ± 0.058**
	500	0.807 ± 0.122	0.798 ± 0.044	0.766 ± 0.044	0.815 ± 0.100	**0.889 ± 0.044**
	700	0.889 ± 0.104	0.796 ± 0.114	0.813 ± 0.044	**0.889 ± 0.103**	0.824 ± 0.181
Forest Fire	100	0.891 ± 0.007	0.793 ± 0.027	0.796 ± 0.131	**0.910 ± 0.067**	0.891 ± 0.017
	300	**0.880 ± 0.004**	0.799 ± 0.035	0.761 ± 0.021	0.846 ± 0.001	0.873 ± 0.027
	500	**0.897 ± 0.002**	0.826 ± 0.044	0.801 ± 0.061	0.885 ± 0.030	0.849 ± 0.021
	700	0.879 ± 0.105	**0.916 ± 0.08**	0.862 ± 0.065	0.889 ± 0.117	0.870 ± 0.121
Small-world	100	0.526 ± 0.005	0.404 ± 0.003	0.470 ± 0.004	**0.579 ± 0.008**	0.406 ± 0.013
	300	0.563 ± 0.007	0.484 ± 0.009	0.519 ± 0.006	0.585 ± 0.016	**0.592 ± 0.009**
	500	0.746 ± 0.012	0.498 ± 0.005	0.479 ± 0.007	0.595 ± 0.007	**0.746 ± 0.003**
	700	0.699 ± 0.021	0.552 ± 0.107	0.519 ± 0.003	0.760 ± 0.143	**0.763 ± 0.006**

long-pattern subgraphs. From these results, we make the following observations.

1) The classification precision and F1 score increase with propagation time. This is because more nodes and paths in the cascade graphs are available for classification as time increases. Because the incremental cascade classifier selects top_k features at each iteration and uses the dual problem to solve the high dimensional problem, our method is more advantageous than the benchmark methods.
2) The precision, F1 score, and accuracy show that OSF outperforms other methods. This is because OSF always selects the top_k columns from all the features while ISF selects the top_k of $t * B \leqslant N$ features, where t is the number of iterations, B is the number of selected columns, and N is the number of all the used features.
3) In terms of prediction variance, OSF and ISF have lower variance error, which means they are more stable than the benchmark methods.
4) From the four synthetic data sets in Table 3, when the time is around 300-500, ISJF is usually better than ISF, because there are more short-pattern subgraphs and useful long-pattern subgraphs when the time is around 300-500. In addition, the long patterns are useful for small world data than scale free networks, as any vertices in the small world be connected by at most six vertices [30].

5.3 Case Study on Cascading Outbreak Early Prediction

We test the incremental subgraph features method on real-world social networks to predict outbreak cascades. Cascade data represents a new type of graph data that reflect information flows. Because there are no cascade descriptors (features) directly available to reflect the direction of information flows, we resort to subcascades as features. Existing works on cascade outbreak prediction are based on node features and *MSF* can be used as the solution. Fig. 1 (Section 1) samples several information cascades in a social network. Each cascade can be denoted as a graph, and subcascades correspond to subgraphs. For example, the cascade $n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow n_4$ is a propagation graph which contains a simple subgraph of $n_1 \rightarrow n_2 \rightarrow n_3$. If a cascade becomes an outbreak, we label it as the positive class +1; otherwise, -1. The subgraph features and class labels are used to build classifiers.

The problem of cascade outbreak prediction is defined as: *Given a network graph G, consider a cascade* $\mathbf{x} = \{xID; \langle V, T\rangle\}$ *where V is a network of nodes, T is a time-stamp, and a threshold parameter* $\gamma \in (0, 1)$, *if* $|V| \geq \lceil \gamma |G| \rceil$, *then x is labeled as outbreak, i.e.,* $y = +1$; *otherwise,* $y = -1$.

Fig. 15 shows the log-log distribution of the cascade size and its node number. As shown in Fig. 15, the size of the MemeTracker cascade data set follows the power-law with long-tails, which indicates that only a small proportion of these cascade become outbreak cascades.

We select cascades having more than 300 nodes as outbreaks (877 positive examples), and cascades having less than 100 nodes as non-outbreaks (27,515,721 negative

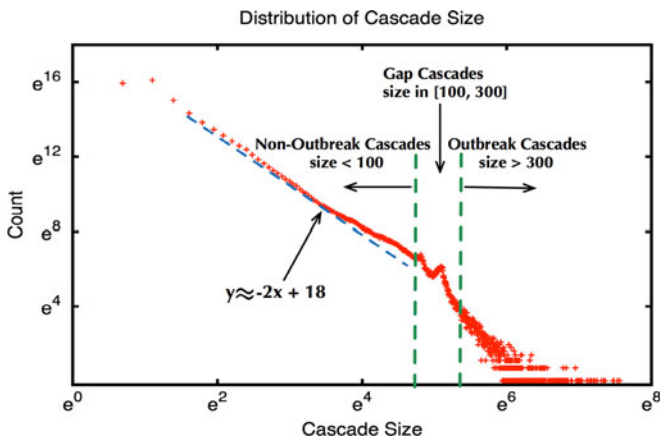


Fig. 15. The probability distribution of cascades. The dotted line is the linear fitting result to the red curve, showing that the distribution fits the power-law. The two dotted vertical lines indicate the threshold which discriminates outbreaks from non-outbreak cascades. The sizes in [100, 300] are the gap cascades which are not used in our experiments.

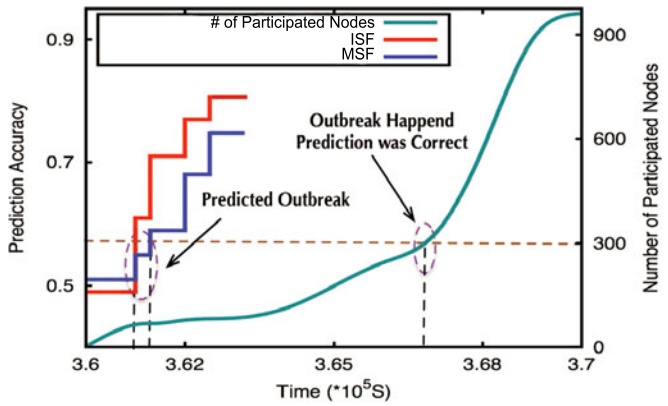


Fig. 16. Early prediction of information cascade outbreaks. We compare the subcascade-based method (red line) with the node-based method (blue line). The figure shows that the subcascade-based method provides better prediction accuracy than the node-based method.

examples). We randomly select outbreak and non-outbreak examples to construct balanced training sets.

Fig. 16 shows that ISF outperforms MSF, which validates the superiority of the proposed subcascade-based outbreak prediction method. This is mainly because node features ignore the path (graph) information. For example, cascade $\alpha \rightarrow \beta$ and $\beta \rightarrow \alpha$ will be predicted as the same class label because they share the same node features.

6 Conclusion

In this paper, we study graph classification with incremental subgraph features. Based on the observation that subgraph features follow the *downward closure property* and long-pattern subgraph features are often buried underneath short-pattern subgraph features, we propose a *primal-dual incremental subgraph feature selection* algorithm (*ISF*) for mining incremental subgraph features, and a subgraph join feature selection algorithm (*ISJF*) to exact long-pattern subgraphs. Experiments on real-world cascade outbreak prediction in social networks demonstrate the effectiveness of the proposed models.

Appendix

A. The Derivation from Eq. (1) to Eq. (3)

When $\mathbf{d}$ is fixed, the inner minimization in Eq. (1) degenerates to a standard SVM model *w.r.t.* $\mathbf{w}$ and $\boldsymbol{\xi}$

$$\begin{aligned} \min_{\mathbf{w},\xi,b} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0,\ i = 1, \ldots, m. \end{aligned} \tag{13}$$

By introducing the Lagrangian multiplier $\alpha_i \geq 0$ to each constraint $y_i(w^T(x_i \odot \sqrt{d}) + b) \geq 1 - \xi_i$, we obtain

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \xi, \alpha) = & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ & - \sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}^T(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) + b) - 1 + \xi_i),\ \alpha_i \geq 0. \end{aligned} \tag{14}$$

Then by setting the derivatives of the Lagrange function to be 0 with respect to parameters $\mathbf{w}, \xi$ and b, we obtain

$$\begin{cases} \frac{\partial \mathcal{L}(\mathbf{w},b,\xi,\alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}}) = 0 \\ \frac{\partial \mathcal{L}(\mathbf{w},b,\xi,\alpha)}{\partial b} = -\sum_{i=1}^{n}\alpha_i y_i = 0 \\ \frac{\partial \mathcal{L}(\mathbf{w},b,\xi,\alpha)}{\partial \xi_i} = C - \alpha_i = 0. \end{cases} \tag{15}$$

That is,

$$\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}}),\ \sum_{i=1}^{n}\alpha_i y_i = 0,\ 0 \leq \alpha_i \leq C.$$

Plugging the above results back into Eq. (14), we obtain the dual form of the original problem as follows:

$$\max_{\boldsymbol{\alpha}\in\mathcal{A}} \quad -\frac{1}{2}\left\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\right\|^2 + \mathbf{e}^T\boldsymbol{\alpha}. \tag{16}$$

As the objective function $-\frac{1}{2}\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\|^2 + \mathbf{e}^T\boldsymbol{\alpha}$ in Eq. (16) is linear in $\mathbf{d}$ and convex in $\boldsymbol{\alpha}$, and both $\mathcal{A}$ and $\mathbf{D}$ are compact domains, Eq. (1) can be equivalently reformulated as follows:

$$\min_{\boldsymbol{\alpha}\in\mathcal{A}}\max_{\mathbf{d}\in\mathbf{D}} \quad \frac{1}{2}\left\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\right\|^2 - \mathbf{e}^T\boldsymbol{\alpha}. \tag{17}$$

As both $\mathcal{A}$ and $\mathbf{D}$ are convex compact sets, the following equivalence holds by interchanging the order of $\min_{\mathbf{d}\in\mathbf{D}}$ and $\max_{\boldsymbol{\alpha}\in\mathcal{A}}$ in Eq. (2) based on the minimax saddle-point theorem [15],

$$\begin{aligned} & \min_{\mathbf{d}\in\mathbf{D}}\max_{\boldsymbol{\alpha}\in\mathcal{A}} \quad -\frac{1}{2}\left\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\right\|^2 + \mathbf{e}^T\boldsymbol{\alpha} \\ = & \max_{\boldsymbol{\alpha}\in\mathcal{A}}\min_{\mathbf{d}\in\mathbf{D}} \quad -\frac{1}{2}\left\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\right\|^2 + \mathbf{e}^T\boldsymbol{\alpha} \\ \Longleftrightarrow & \min_{\boldsymbol{\alpha}\in\mathcal{A}}\max_{\mathbf{d}\in\mathbf{D}} \quad \frac{1}{2}\left\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\right\|^2 - \mathbf{e}^T\boldsymbol{\alpha}. \end{aligned}$$

B. Optimization 1 Can Be Solved by Quadratic Programming with Small Set of Features

Let $f(\boldsymbol{\alpha}, \mathbf{d}) = \frac{1}{2}\|\sum_{i=1}^{n}\alpha_i y_i(\mathbf{x}_i \odot \sqrt{\mathbf{d}})\|^2 - \mathbf{e}^T\boldsymbol{\alpha}$. By introducing an additional variable $\theta \in \mathbb{R}$, the problem can be reformulated as follows:

$$\min_{\boldsymbol{\alpha}\in\mathcal{A},\theta\in\mathbb{R}} \ \theta \quad \text{s.t. } \theta \geq f(\boldsymbol{\alpha}, \mathbf{d}), \forall\ \mathbf{d} \in \mathcal{D} \tag{18}$$

which is a convex quadratic programming problem. Each nonzero $\mathbf{d} \in \mathcal{D}$ defines a quadratic constraint with respect to $\boldsymbol{\alpha}$. There are as many as $(\sum_{i=0}^{B}\binom{m}{i})$ quadratic constrains in Eq. (4).

Because the scaling vector d is fixed, and we use $\|\mathbf{d}\| \leq B$ to encourage sparsity so that at most B subgraph features are selected. Since B is a small value, it can be solved with small set of features.

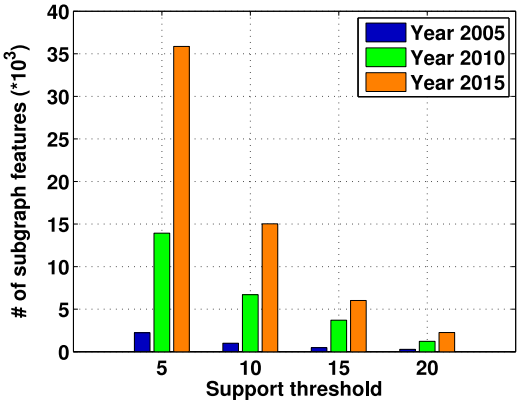


Fig. 17. # of subgraph features *w.r.t.* support threshold on the DBLP data set.

C. Derivation from Eq. (10) to Eq. (11)

By introducing the Lagrangian multiplier $p_i, q_j, r_j \geq 0$ to each constraint in Eq. (8), we have

$$\mathcal{L}(\mathbf{d}) = \sum_{j=1}^{t}[c_j(\alpha)]^2 d_j + \sum_{i=1}^{m} p_i \left[(1-\epsilon)\frac{t}{m}b_i - \sum_{j=1}^{t} d_j\right] + \sum_{j=1}^{t} q_j d_j + \sum_{j=1}^{t} r_j(1-d_j). \tag{19}$$

Then by setting the derivatives of the Lagrange function to be 0 with respect to parameters $\mathbf{d}$, we obtain

$$\nabla_{\mathbf{d}}\mathcal{L}(\mathbf{d}) = [c_j(\alpha)]^2 - \sum_{i=1}^{m} p_i + q_j - r_j = 0. \tag{20}$$

That is, $[c_j(\alpha)]^2 = \sum_{i=1}^{m} p_i - q_j + r_j$.

As $r_j \geq 0$, we have $\sum_{i=1}^{m} p_i + r_j \geq [c_j(\alpha)]^2$. By plugging the above result to Eq. (20), we have the dual problem as follows:

$$\begin{aligned} \min_{p,r} \quad & \sum_{i=1}^{m} b_i p_i (1-\epsilon)\frac{t}{m} + \sum_{j=1}^{t} r_j \\ \text{s.t.} \quad & \sum_{i=1}^{m} p_i + r_j \geq [c_j(\alpha)]^2, \ j = 1, \ldots, t \\ & p_i, r_j \geq 0, 1 \leq i \leq m. \end{aligned} \tag{21}$$

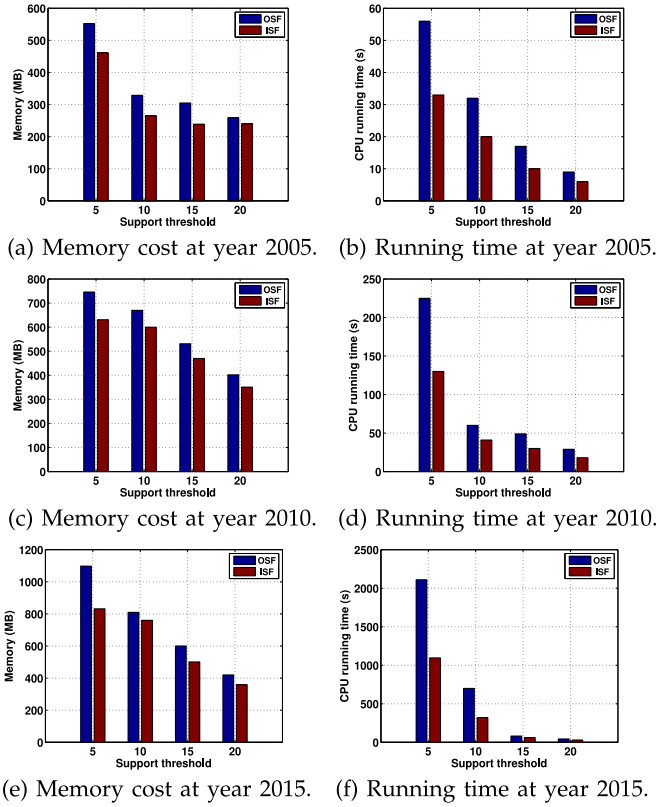
For simplicity, we use only one Lagrange multiplier and map the space of r_j to the same space of p_i, the dual problem can be rewritten as follows:

$$\begin{aligned} \min_{p} \quad & \sum_{i=1}^{m} b_i p_i (1-\epsilon)\frac{t}{m} + \sum_{i=m+1}^{m+t} p_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} p_i + p_{i+j} \geq [c_j(\alpha)]^2, \ j = 1, \ldots, t \\ & p_i \geq 0, 1 \leq i \leq m. \end{aligned} \tag{22}$$

D. Experiments on the DBLP Data Set

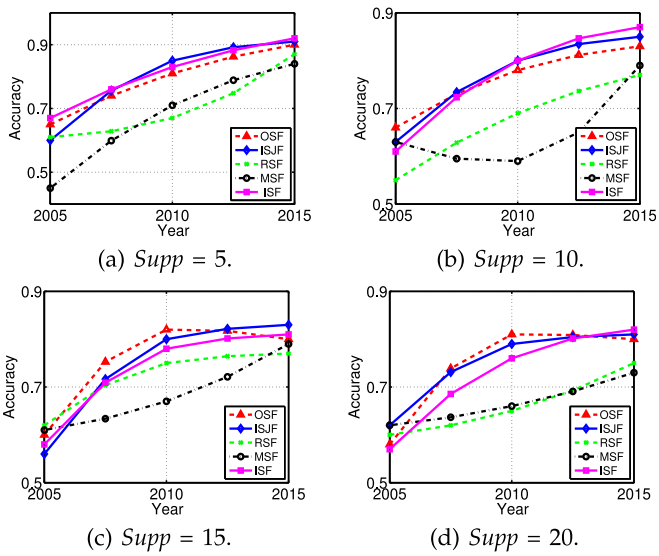
Due to page limitations, we report experimental results on real-world data set (DBLP) in the Appendix.

DBLP. DBLP author classification refers to the task of predicting author's research areas based on co-authorship network where two authors are connected if they publish at least one paper together. We retrieve around 2,000 authors from DBLP and 95,411 coauthors. The training sets are drawn from different domain, i.e., the conferences in each research areas. We mainly crawl the authors from the areas in Software Engineering and Data Mining. We fetch the co-authorship network graphs of each author from 2001 to 2015, and formulate the co-authorship graphs at different year. Specifically, we have three phases' graphs, i.e., 2001 to 2005, 2006 to 2010, and 2011 to 2015. Subgraph features are commonly used in author classification that is based on co-



(a) Memory cost at year 2005. (b) Running time at year 2005.
(c) Memory cost at year 2010. (d) Running time at year 2010.
(e) Memory cost at year 2015. (f) Running time at year 2015.

Fig. 18. Memory cost and running time w.r.t. the support threshold at different year on the DBLP data set.



(a) *Supp* = 5. (b) *Supp* = 10.
(c) *Supp* = 15. (d) *Supp* = 20.

Fig. 19. Accuracy comparison under different *Supp* on the DBLP data set.

authorships. However, predicting an authors research areas using only a single coauthor or conference is infeasible due to the interdisciplinary nature of research.

The number of subgraph features with respect to support threshold, the memory cost with respect to the support threshold, the running time with respect to the support threshold, and the accuracy comparison in each year bracket are reported in Figs. 17, 18, and 19,

From the results on the DBLP data set, we make observations similar to the MemeTracker data set. The classification accuracy increases with each year bracket. This is because more nodes and paths in the co-authorship graph are available for classification as time goes by. The proposed approach also shows significant improvement with respect to the running time and memory cost, especially when the subgraphs' dimensions are high.

Acknowledgments

This work is partially supported by the ARC Future Fellowship FT130100746, ARC grant LP150100671 and DP140100545, the US National Science Foundation under grant IIS-1613950, and by the program for professor of special appointment (Eastern Scholar) at the Shanghai Institutions of Higher Learning.

References

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for on demand classification of evolving data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 5, pp. 577–589, May 2006.

[2] X. Yan, F. Zhu, P. S. Yu, and J. Han, "Feature-based similarity search in graph structures," *ACM Trans. Database Syst.*, vol. 31, no. 4, pp. 1418–1453, 2006.

[3] A. Inokuchi, "Mining generalized substructures from a set of labeled graphs," in *Proc. IEEE Int. Conf. Data Mining*, 2005, pp. 415–418.

[4] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *Proc. IEEE Int. Conf. Data Mining*, 2002, pp. 721–724.

[5] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda, "gBoost: A mathematical programming approach to graph classification and regression," *J. Mach. Learn.*, vol. 75, no. 1, pp. 69–89, 2009.

[6] X. Wu, K. Yu, H. Wang, and W. Ding, "Online streaming feature selection," presented at the 27th Int. Conf. Mach. Learn., Haifa, Israel, 2010.

[7] Y. Zhu, L. Qin, J. X. Yu, Y. Ke, and X. Lin, "High efficiency and quality: Large graphs matching," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage.*, 2011, pp. 1755–1764.

[8] L. H. Ungar, J. Zhou, D. P. Foster, and B. A. Stine, "Streaming feature selection using IIC," in *Proc. 10th Int. Workshop Artif. Intell. Statistics*, 2005, Art. no. 357.

[9] X. Wu, K. Yu, W. Ding, H. Wang, and X. Zhu, "Online feature selection with streaming features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 5, pp. 1178–1192, May 2013.

[10] H. Kashima, K. Tsuda, and A. Inokuch, "Marginalized kernels between labeled graphs," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 321–328.

[11] M. Lopez, and G. Still, "Semi-infinite programming," *Eur. J. Oper. Res.*, vol. 180, no. 2, pp. 491–518, 2007.

[12] M. M. Masud, et al., "Classification and adaptive novel class detection of feature-evolving data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 7, pp. 1484–1497, Jul. 2013.

[13] M. Varma and B. R. Babu, "More generality in efficient multiple kernel learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 1065–1072.

[14] M. Tan, I. W. Tsang, and L. Wang, "Towards ultrahigh dimensional feature selection for big data," *J. Mach. Learn. Res.*, vol. 15, pp. 1371–1429, 2014.

[15] M. Sion, "On general minimax theorems," *Pacific J. Math.*, vol. 8, pp. 171–176, 1958.

[16] M. Takac, A. Bijral, P. Richtarik, and N. Srebro, "Mini-batch primal and dual methods for SVMs," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 2059–2067.

[17] N. Buchbinder and J. Naor, "Online primal-dual algorithms for covering and packing," *Math. Operations Res.*, 2009, pp. 270–286.

[18] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg, "Online auctions and generalized secretary problems," *SIGecom Exchanges*, vol. 7, no. 2, pp. 1–11, 2008.

[19] S. Shalev-Shwartz and Y. Singer, "Online learning meets optimization in the dual," in *Proc. 19th Annu. Conf. Learn. Theory*, 2006, pp. 423–437.

[20] S. Agrawal, Z. Wang, and Y. Ye, "A dynamic near-optimal algorithm for online linear programming," *Math. Operations Res.*, vol. 62, pp. 876–890, 2009.

[21] J. Leskovec, L. Backstrom, and J. Kleinberg, "Meme-tracking and the dynamics of the news cycle," in *Proc. 15th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, 2009, pp. 497–506.

[22] K. Riesen and H. Bunke, "Graph classification by means of Lipschitz embedding," *IEEE Trans. Syst. Man Cybern. Part B: Cybern.*, vol. 39, no. 6, pp. 1472–1483, Dec. 2009.

[23] K. Riesen and H. Bunke, *Graph Classification and Clustering Based on Vector Space Embedding*. River Edge, NJ, USA: World Sci. Publishing Company, 2010.

[24] H. Fei and J. Huan, "Structured sparse boosting for graph classification," *ACM Trans. Knowl. Discovery Data*, vol. 9, 2014, Art. no. 4.

[25] M. Thoma, et al., "Near-optimal supervised feature selection among frequent subgraphs," in *Proc. 9th SIAM Int. Conf. Data Mining*, 2009, pp. 1069–1080.

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

[27] 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.

[28] S. Perkins and J. Theiler, "Online feature selection using grafting," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 592–599.

[29] J. Zhou, D. P. Foster, R. A. Stine, and L. H. Ungar, "Streamwise feature selection," *J. Mach. Learn. Res.*, vol. 7, pp. 1861–1885, 2006.

[30] S. Zhang, X. Luo, J. Xuan, X. Chen, and W. Xu, "Discovering small-world in association link networks for association learning," *World Wide Web*, vol. 17, no. 2, pp. 229–254, 2014.

[31] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2007, pp. 420–429.

[32] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: Densification laws, shrinking diameters and possible explanations," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2005, pp. 177–187.

[33] A. L. Barabasi and R. Alber, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, 1999.

[34] B. Bollobas, *Random Graphs*. Cambridge, U.K.: Cambridge Univ. Press, 2001.

[35] D. Watts and S. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, pp. 440–442, 1998.

[36] P. Cui, S. Jin, L. Yu, F. Wang, W. Zhu, and S. Yang, "Cascading outbreak prediction in networks: A data-driven approach," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 901–909.

[37] X. Yan, X. Zhou, and J. Han, "Mining closed relational graphs with connectivity constraints," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2005, pp. 324–333.

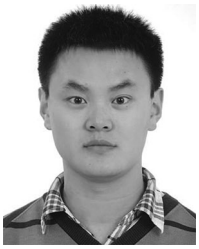
[38] H. Liu, H. Motoda, R. Setiono, and Z. Zhao, "Feature selection: An ever evolving frontier in data mining," *J. Mach. Learn. Res.*, vol. 10, pp. 4–13, 2010.

[39] 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.

[40] M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham, "Classification and novel class detection in concept-drifting data streams under time constraints," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 6, pp. 859–874, Jun. 2011.

[41] X. Zhang, C. Furtlehner, C. Germain-Renaud, and M. Sebag, "Data stream clustering with affinity propagation," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 7, pp. 1644–1656, Jul. 2014.

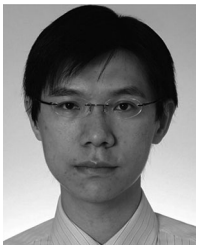
[42] M. Tan, L. Wang, and I. W. Tsang, "Learning sparse SVM for feature selection on very high dimensional datasets," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 1047–1054.
[43] Y. Zhai, Y.-S. Ong, and I. W. Tsang, "The emerging "Big Dimensionality"," *IEEE Comput. Intell. Mag.*, vol. 9, no. 3, pp. 14–26, Aug. 2014.
[44] H. Wang, P. Zhang, I. Tsang, L. Chen, and C. Zhang, "Defragging subgraph features for graph classification," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 1687–1690.



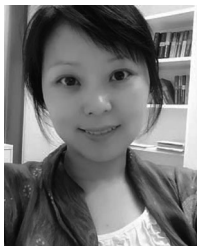
Haishuai Wang received the master's degree in computer science from Beijing Jiaotong University, Beijing, China, in 2013. He has been working toward the PhD degree in the Centre for Quantum Computation and Intelligent Systems, University of Technology Sydney. His research focuses on data mining and machine learning.

Peng Zhang received the PhD degree from the Chinese Academy of Sciences, in July 2009. Since then, he has worked in one National Engineering Laboratory, Chinese Academy of Sciences and two universities in the USA. He is now with the University of Technology Sydney, as a lecturer. He is an associate editor of two Springer journals, the *Journal of Big Data* and the *Annals of Data Science*.

Xingquan Zhu received the PhD degree in computer science from Fudan University, Shanghai, China. He is an associate professor in the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, Florida, and a distinguished visiting professor (Eastern Scholar) with the Shanghai Institutions of Higher Learning. His current research interests include data mining, machine learning, and multimedia systems. He is an associate editor of the *IEEE Transactions on Knowledge and Data Engineering*. He is a senior member of the IEEE.



Ivor Wai-Hung Tsang received the PhD degree in computer science from the Hong Kong University of Science and Technology, in 2007. He is an Australian Future fellow and associate professor in the Centre for Quantum Computation & Intelligent Systems, University of Technology, Sydney. He received the prestigious IEEE Transactions on Neural Networks Outstanding 2004 Paper Award in 2006, the 2014 IEEE Transactions on Multimedia Prized Paper Award, and a number of best paper awards and honors from reputable international conferences, including the Best Student Paper Award at CVPR 2010, and the Best Paper Award at ICTAI 2011, etc.



Ling Chen received the PhD degree from Nanyang Technological University, Singapore. She is a senior lecturer in the Centre for Quantum Computation and Intelligent Systems, University of Technology, Sydney. Before joining the University of Technology, Sydney, she was a postdoc research fellow in the L3S Research Center, University of Hannover, Germany. Her research interests include data mining and machine learning, social network analysis, and recommender systems.

Chengqi Zhang is a professor of information technology with the University of Technology, Sydney (UTS), Sydney, where 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 has been serving as an associate editor of three international journals, including the *IEEE Transactions on Knowledge and Data Engineering*. He is senior member of the IEEE.

Xindong Wu received the PhD degree in artificial intelligence from the University of Edinburgh, United Kingdom. He is a Yangtze River scholar in the School of Computer Science and Information Engineering, Hefei University of Technology, China, and a Alfred and Helen Lamson Endowed professor of computer science with the University of Louisiana at Lafayette. His research interests include data mining and knowledge-based systems. He is a fellow of the IEEE, the AAAS, and the American Association for the Advancement of Science.