Data Min Knowl Disc (2016) 30:511–549
DOI 10.1007/s10618-015-0424-z

Active exploration for large graphs

Meng Fang[1,2] · **Jie Yin**[2] · **Xingquan Zhu**[3]

Received: 21 November 2013 / Accepted: 16 June 2015 / Published online: 29 August 2015

Abstract Modern information networks, such as social networks, communication networks, and citation networks, are often characterized by very large sizes and dynamically changing structures. Common solutions to graph mining tasks (e.g., node classification) usually employ an unrestricted sampling-then-mining paradigm to reduce a large network to a manageable size, followed by subsequent mining tasks. However, real-world networks may be unaccessible at once and must be crawled progressively. This can be due to the fact that the size of the network is too large, or some privacy/legal concerns. In this paper, we propose an *Active Exploration* framework for large graphs, where the goal is to simultaneously carry out network sampling

Responsible editor: Thomas Seidl.

Most of this research was done while Meng Fang was at the University of Technology, Sydney.

Electronic supplementary material The online version of this article (doi:10.1007/s10618-015-0424-z) contains supplementary material, which is available to authorized users.

✉ Meng Fang
Meng.Fang@student.uts.edu.au; Meng.Fang@csiro.au

Jie Yin
Jie.Yin@csiro.au

Xingquan Zhu
xzhu3@fau.edu

1 Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia

2 CSIRO, Sydney, NSW, Australia

3 Department of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, USA

and node labeling in order to build a sampled network from which the trained classifier can have the maximum node classification accuracy. To achieve this goal, we consider a network as a Markov chain and compute the stationary distribution of the nodes by deriving supervised random walks. The stationary distribution helps identify specific nodes to be sampled in the next step, and the labeling process labels the most informative node which in turn strengthens the sampling of the network. To improve the scalability of active exploration for large graphs, we also propose a more efficient multi-seed algorithm that simultaneously runs multiple, parallel exploration processes, and makes joint decisions to determine which nodes are to be sampled and labeled next. The simultaneous, mutually enhanced sampling and labeling processes ensure that the final sampled network contains a maximum number of nodes directly related to the underlying mining tasks. Experiments on both synthetic and real-world networks demonstrate that our active exploration algorithms have much better chance to include target nodes in the sampled networks than baseline methods.

Keywords Active exploration · Supervised sampling · Random walks · Active learning · Networked data

1 Introduction

Recent years have witnessed the growth and popularity of information networks in all aspects of human society, such as scientific publications, business, biomedical research, and even people's daily lives (Wasserman and Faust 1995; Fang and Tao 2014). Typical examples include friendship networks in *Facebook*,[1] co-author and bibliography networks in *DBLP*,[2] and the World Wide Web. In these applications, a social network is represented as a large graph, in which nodes denote entities or instances (e.g., users or publications) and edges denote relationships between nodes (e.g., friendship, kinship or co-authorship). Such graphs can be very large in size and contain millions of nodes and edges. For example, some recent statistics[3] show that Facebook, the largest online social network, has 1.1 billion active users as of June 2013, and there are one million links shared between friends every 20 minutes. This is equivalent to a gigantic network with 1.1 billion nodes (i.e., users), involving one million linkage changes every 20 minutes. Other social networks such as Flickr also show similar trends, where their network involves 51 million unique users and 1.4 million pictures are uploaded every day.[4] Even for a small scientific research domain in Computer Science, *DBLP* indexes over 2,167,502 publications and their references, which easily form a million node scale network. For all these networks, the sheer number of nodes and edges makes analyzing the entire network computationally infeasible. Therefore, graph sampling (Gjoka et al. 2010; Hübler et al. 2008; Leskovec and Faloutsos 2006) becomes an important approach that generates a smaller, but represen-

[1] www.facebook.com.

[2] www.informatik.uni-trier.de/~ley/db/.

[3] www.statisticbrain.com/facebook-statistics/.

[4] www.flickr.com/photos/franckmichel/6855169886/.

tative subgraph to approximate the original large graph. Afterwards, more expensive and complicated analyses can be subsequently performed on the sampled graphs for various data mining tasks, such as labeling network nodes or edges to build node classification models or to classify the linkage relationships between nodes in a network.

For network sampling approaches, the existing *sampling-then-mining* paradigm assumes that the entire network is available for sampling and the sampled network retains equally useful information for all subsequent data mining tasks, such as node classification or link classification. However, this type of approach has several fundamental limitations. First, without knowing the underlying network analysis objective, the separated sampling process can hardly generate a quality network with a reduced size, tailored to various analytical needs for succeeding mining processes. Second, graph sampling processes typically operate on an entire static graph. However, real-world networks are rarely immediately available until a sampling process progressively crawls each node and its connections to form a network (Catanese et al. 2011). Thus, it would be beneficial to design algorithms that start from some specific nodes, explore their neighborhood, and acquire information about the network when necessary, such as labeling a particular subset of nodes. The obtained labeling information can be further fed to improve information collection while exploring the network. Third, existing graph sampling techniques have focused on generating a uniform random sample of nodes in the original graph. However, it is often the case that a data mining task aims to identify some significant nodes (i.e., positive/target instances) comprising only a small portion of the whole network. For example, in security surveillance, agents are more interested in identifying suspects and their relationships to other individuals. In disease monitoring, health analysts want to discover affected individuals in a large population. Ascertaining the information about individuals, such as their affiliations or infection status, would incur a prohibitive cost in terms of both time and resources. Therefore, it is highly desirable to minimize the cost of exploring the entire network while still acquiring a number of important nodes with a particular label.

Motivated by the above observations, we introduce a new *active exploration* problem for large graphs, Specifically, active exploration is an iterative task in network settings, where querying the labels of nodes is subject to a certain amount of labeling cost. Therefore, given a fixed amount of labeling budget, active exploration needs to explore the network and identify as many nodes with a particular label as possible. The problem of active exploration is related to active learning for networked data (Bilgic et al. 2010; Kuwadekar and Neville 2011) and semi-supervised classification (Belkin et al. 2004; Zhou et al. 2004; Zhu et al. 2003a), but has distinct objectives: Active learning intends to query the labels of nodes to improve the accuracy of a classifier, and semi-supervised classification leverages unlabeled data to reduce the amount of labeled data required to achieve the same level of classification accuracy. They both assume that the graph is fully observable (i.e., all nodes and edges are known). This differs from the active exploration problem proposed in this work, in which only a small portion of the network is observable. Our objective is to actively explore the unobserved portion of the network and query a small subset of nodes to maximize the identification of nodes with a particular label, such that the classifier trained from the obtained network can achieve the maximum classification accuracy.

To address this new problem, our key idea is to model a graph as a Markov chain and design two interleaved processes—sampling and labeling—that closely collaborate towards the goal of active exploration. At each iteration, the sampling process is guided by a supervised random walk that is more likely to visit positive nodes than negative nodes in the neighborhood. The labeling process is thereafter utilized to query a node's label when necessary. Only by querying, the true label of a node can be revealed. The node label along with the node features are used to update the stationary distribution of nodes in the explored network, so they can benefit supervised sampling at the next iteration. The tight coupling between the two processes allows them to interplay with each other and improve the exploration and retrieval of important nodes on large graphs. In our preliminary work (Fang et al. 2013), we have proposed a single-seed active exploration algorithm that starts with a single seed, and progressively samples and labels the rest of the network. Initial experimental results have shown that this algorithm can achieve a higher recall of identifying positive nodes while sampling a network.

In this work, we advance the single-seed active exploration to multiple seeds and also extend our active exploration framework to cope with large-scale networks. The single-seed algorithm is guided by a supervised random walk that computes the stationary probabilities of the nodes in the explored network and decides which node is to be sampled and labeled next. As the size of the explored network increases, sampling and labeling processes become computationally intensive and often infeasible; the sampling process needs to calculate the probability score of each node in the explored network, and more expensively, the labeling process needs to iterate over all the unlabeled nodes to identify the most informative node to be labeled, by recomputing the stationary distribution and comparing the difference before and after each node is presumably labeled. Hence, network size would significantly limit the efficiency and scalability of the single-seed active exploration algorithm on large networks. In order to scale the single-seed algorithm, we propose a multi-seed algorithm that simultaneously initializes multiple active exploration processes from k different seeds. At each iteration, each exploration process computes an independent supervised random walk and makes its *local* decision to determine which node to be sampled next in its own explored network. Furthermore, a *global* decision is made collectively to select the most informative node to be labeled from multiple explored networks. By using multiple, parallel exploration processes, our new multi-seed active exploration algorithm can significantly improve the efficiency of the single-seed algorithm, and scale well to large-size networks. To validate the effectiveness of our proposed algorithms, we carry out extensive experiments on both synthetic and real-world networks. Our experiments demonstrate that, first, our proposed algorithms achieve a higher recall of identifying positive nodes while sampling large networks than baseline methods, especially for networks with imbalanced class distributions, and second, the multi-seed algorithm can significantly improve the efficiency of the single-seed counterpart and scale well to large-scale networks.

The contributions of our work can be summarized as follows:

– We introduce a new active exploration problem on large graphs that iteratively samples and labels the nodes for identifying a specific subgroup of nodes and improving the classification accuracy;

– We formulate a supervised random walk as an optimization problem and derive its solution;
– Based on supervised random walks, we present a single-seed active exploration algorithm to simultaneously perform sampling and labeling on graphs and analyze its computational complexity;
– We further propose a multi-seed algorithm which offers great scalability of active exploration in handling large graphs.

The remainder of the paper is organized as follows. In Sect. 2, we briefly review the related literature. In Sect. 3, we give a formal definition of our active learning problem. In Sect. 4, we formulate a supervised random walk as an optimization problem, and then propose two active exploration algorithms based on the solutions derived from the optimization objective. We discuss the experimental results on both synthetic and real-world networks in Sect. 5, and conclude the paper in Sect. 6.

2 Related work

To position our work in the literature, we briefly review existing research work related to our active exploration problem. These include graph sampling, active learning in the context of graphs and social networks, semi-supervised classification, as well as multi-seed learning framework used in other different studies.

2.1 Graph sampling

In the following, we briefly review graph sampling techniques from static graph sampling versus streaming graph sampling perspectives.

2.1.1 Static graph sampling

Static graph sampling techniques can be roughly classified into two categories: *graph traversals* and *random walks* (Gjoka et al. 2010). For graph traversals, nodes are sampled without replacement; once a node is visited, it is never revisited again. Depending on the order in which nodes are visited, these methods include Breadth-First Search (BFS), Depth-First Search (DFS), forest fire, and snowball sampling (Wasserman and Faust 1995; Ahn et al. 2007; Mislove et al. 2007). Particularly, BFS is a popularly used technique for sampling social networks, which has been studied extensively (Ahn et al. 2007; Mislove et al. 2007, 2008; Viswanath et al. 2009; Wilson et al. 2009). However, some research work has shown that BFS is biased towards high degree nodes in real-world networks (Becchetti et al. 2006; Lee et al. 2006; Ye et al. 2010). When using graph traversals for sampling, the sampling process terminates after a pre-defined fraction of graph nodes are collected. The pre-defined sampling parameter therefore determines the size of the sampled network.

Random walks fall into the other category of graph sampling techniques, which are a natural and thoroughly studied approach to randomized graph exploration. A random walk is a stochastic process that starts at one node of a graph, and at each step moves

from the current node to an adjacent node chosen randomly and uniformly from the neighbors of the current node (Lovász 1993). Random walks have been widely used for sampling the Web (Henzinger et al. 2000), peer-to-peer networks (Gkantsidis et al. 2004; Stutzbach et al. 2009) and other large graphs (Gjoka et al. 2010; Leskovec and Faloutsos 2006), or condensing graphs to allow for better visualization (Rafiei and Curial 2005). Please refer to Lovász (1993) for an extensive survey. Similar to traversals, random walks are also found to be biased towards high degree nodes in the graph. However, the bias of random walks can be analyzed and corrected by using classical results from Markov chains. For example, in the context of peer-to-peer sampling, reweighting was proposed to correct the bias of random walks (Rasti et al. 2009). Gjoka et al. (2010) proposed a Metropolis-Hastings algorithm to collect an unbiased sample of *Facebook* users. Likewise, Hübler et al. (2008) presented a Metropolis algorithm for sampling a representative subgraph, requiring that the sampled graph preserves crucial graph properties of the original graph.

Some other studies have been proposed to achieve a faster estimation of random walks when the size of networks is large. For large graphs, a random walk on the graph often requires a large number of individual user queries. Zhou et al. (2013) proposed a method that increases the conductance of social network graphs by modifying the graph topology on the fly. Tong et al. (2006) proposed a fast method to random walk with restart using two important properties of graphs: linear correlations and block-wise, community-like structure. However, these algorithms are not suited to solve our problem because they treat all links equally important. In contrast, in our work, the strength of edges is defined as a function of edge features and we need to learn the parameters of the strength function. To handle the problem of large graphs, we alternatively propose a multi-seed strategy which can run in parallel to speed up the efficiency.

2.1.2 Streaming graph sampling

In large-scale social networks, the network structures may continuously change and evolve, which renders streaming networks or graphs rather than static graphs. Ahmed et al. (2014) extended graph sampling from static graphs to streaming graphs and used *induced edge sampling* to randomly sample edges from a streaming graph network, where the edges of the network are presented as a stream to form a sampled network. Streaming graph sampling can also be achieved by using streaming graph partitioning (Stanton and Kliot 2012) which uses hashing techniques to partition a large graph into small networks. Sarma et al. (2011) proposed a streaming model to estimate the probability distributions and PageRank scores for large-scale streaming graphs. All these techniques have provided solutions to tackle the changes and dynamics in large graphs for sampling.

2.1.3 Graph sampling for social network analysis

Graph sampling techniques provide an efficient, yet inexpensive solution for social network analysis. Leskovec and Faloutsos (2006) examined different sampling methods over different social networks and found that best performing methods are random walks and forest fires. Papagelis et al. (2013) introduced sampling-based algorithms

that given a user in a social network efficiently obtain a near-uniform random sample of nodes in its neighborhood. Maiya and Berger-Wolf (2010) described an online sampling technique to sample large social networks so as to discover the most influential individuals within the network.

There is a distinction between the aims of past work on graph sampling and our work. The earlier work on network sampling mainly seeks to obtain a smaller subgraph capturing the properties of the original graph. In other words, the sampled network can maximally preserve the original network statistics, but it may not be directly related to the succeeding mining tasks. In contrast, our work aims to supervise the sampling process to explore the network by visiting more important nodes belonging to a desired class, so we intend to form an explored network to benefit the succeeding mining tasks.

2.2 Active learning on graphs

Active learning aims to minimize the required labeled data by selectively choosing the most informative instances to query for their labels. Recently, graph-based active learning has been proposed to address the problem of classifying networked data. One line of research has focused on using graph-based metrics to define the informativeness of instances and then select the instances with the highest informative scores (Bilgic and Getoor 2008; Cesa-Bianchi et al. 2010). This is often achieved by enforcing the linkage structure of the network into the calculation of the informative scores of the network nodes (Fang et al. 2013). Other research has attempted to improve the accuracy of collective classification by combining link information with node-specific features (Bilgic et al. 2010; Kuwadekar and Neville 2011).

Prior work on active learning for networked data has focused on acquiring only the labels of nodes to improve the accuracy of a classifier, with the assumption that the entire network structure is directly observable. This differs from the active exploration problem as proposed in this work, in which only a small portion of the network is observable. Our goal is to actively explore and sample the unobserved portion of the network and query the labels of a small subset of nodes to form a small network comprising of as many target nodes as possible and improve the performance of classification.

Our work is also related to active sampling (Pfeiffer et al. 2012), in which both the instances' labels and edges are acquired through an iterative process to update a classifier for discovering the nodes with a specific label. This study assumes that a node has no other known attributes aside from its own label. In contrast, in our work, we formulate a supervised learning task by combining the network structure with rich node and edge attributes and use it to guide a random walk on the graph for discovering the nodes having a particular label while exploring the network.

2.3 Semi-supervised classification

Semi-supervised learning is a machine learning framework that learns from both labeled and unlabeled data to help reduce the number of labeled data needed to train a classification model with the maximum accuracy (Zhu and Ghahramani 2002). Many

existing methods fall into graph-based semi-supervised learning (Belkin et al. 2004; He et al. 2007; Zhu et al. 2003a, b), which regards the instance space as a weighted graph with both labeled and unlabeled instances being considered as nodes, and the similarities between instances are used to create weighted edges between nodes. By following this approach, any non-relational data can be represented as a graph, and the classification problems can thus be viewed as the estimation of a function on the graph. In network settings, when instances are explicitly linked to each other, such as a network of websites connected by hyperlinks, the edges simply correspond to the binary presence of a link (or are weighted by the number of links between two instances). When both local features and explicit links are available, some hybrid approaches are also used for semi-supervised classification (Macskassy 2007). Another family of semi-supervised classification is based on random graph walk, which relates itself to the methods that fall under the class of graph walk-based algorithms, such as Gyongyi et al. (2006), Lin and Cohen (2010) and Zhou et al. (2004). The theme of these methods usually involves finding the dominant eigenvectors of some form of affinity matrix or transition matrix of the graph and then performing within-network classification.

Our active exploration problem has a similar setting as semi-supervised classification in a transductive setting, which involves both labeled and unlabeled nodes over the graphs, but has distinct goals. For semi-supervised learning methods, as mentioned above, the assumption is that the final graph is fully specified (i.e., all nodes and edges are known) and that the labels of some nodes in the network are known. Therefore, the objective is to improve the classification accuracy using unlabeled data. In our work, we assume that a full graph is too large for its network structure to be known as input. Thus, only a partial network can be observed. We are concerned with actively exploring the unobserved portion of the network and querying a small subset of nodes to maximize the identification of important nodes with a particular class label.

2.4 Multi-seed learning framework

Our work is also related to multi-seed frameworks used in other different studies, such as local graph clustering or finding connected components in graphs (Alamgir and Von Luxburg 2010; Alon et al. 2008; Halperin and Zwick 1994; Karger et al. 1992). Most of these studies have focused on using multiple random or pseudo-random walks to identify significant subsets of networks. For example, Halperin and Zwick (1994) addressed the problem of finding connected components of an undirected graph, in which an optimal randomized algorithm was proposed by initializing random walks from a suitable initial sample of nodes in the network. Alamgir and Von Luxburg (2010) considered the problem of local graph clustering and proposed to construct a multi-agent random walk (MARW) from all network nodes to discover local clusters corresponding to points of interest. Alon et al. (2008) also demonstrated that using multiple random walks in parallel yields a speed-up in the cover time of visiting every node in a graph, which is linear in the number of parallel walks.

Our work is different from these studies in that, they assume the whole network already exists and has been collected for sampling at the beginning, so that multiple random walks can be designed by taking the whole network structure into consideration to optimize the sampling process. However, in our work, we only start from a very small substructure of the whole network (i.e., a few connected nodes) and progressively choose some nodes to visit and thereafter explore their neighborhood. More importantly, we consider that each node in the network has its own unique importance score, which is determined by the node's content features and link structure in the network. Such information is modeled by a supervised random walk, which provides guidance to visit and label more important nodes in the network during the exploration process. By taking existing multi-seed learning strategies into consideration, we propose to use multiple, parallel exploration processes to improve the scalability of active exploration on large graphs. Our design ensures that the parallelized process can make local or global decisions for determining the best nodes to be sampled and labeled during the active exploration process.

3 Problem definition

Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected graph where $\mathcal{V}$ denotes a set of nodes (or instances) and $\mathcal{E}$ denotes a set of edges between nodes. Each node $v_i \in \mathcal{V}$ is described by a feature vector x_i and a class label $y_i \in \mathcal{Y}$, where $\mathcal{Y}$ denotes a set of class labels. Each edge $(v_i, v_j) \in \mathcal{E}$ has a corresponding feature vector r_{v_i,v_j} which describes relationships between nodes v_i and v_j. The neighbors of a node v_i are denoted by $\mathcal{N}(v_i)$. We focus on a binary classification problem, in which each node v_i either belongs to a positive class ($y_i = +1$) or a negative class ($y_i = -1$), and the whole network only contains a very small number of positive nodes. We assume that a full graph is too large for its global network structure to be known as a whole. Therefore, only a partial network $G_t = (\mathcal{V}_t^l, \mathcal{V}_t^u, \mathcal{E}_t)$ can be observed at time t, where $\mathcal{V}_t^l$ denotes a set of the labeled nodes up to time t and $\mathcal{V}_t^u$ denotes a set of unlabeled nodes.

Given a fixed amount of labeling budget where querying the class label of each single node is subject to a certain amount of cost, the objective of active exploration is to obtain a sampled network with a small portion of labeled nodes, so that the classifier trained from the obtained network has the maximum classification accuracy. Accordingly, the active exploration problem aims to design an $\mathcal{AE}$ algorithm that (1) samples a representative subgraph $G^{'}$ from the original large graph G, and (2) selectively chooses a small set of nodes to query their labels, and acquires any new edges and nodes in order to identify as many positive nodes as possible before the limited labeling budget is exhausted.

We use Fig. 1 to illustrate key concepts behind the active exploration problem. Given a partially observed subgraph G_t, which is an explored network at time step t, we define two types of nodes: *Intra-acquired nodes* $\mathcal{I}_{intra}$ and *Border-acquired nodes* $\mathcal{I}_{border}$. In the figure, Intra-acquired nodes, denoted by double solid circles, are the nodes that have been sampled up to time step t. Once a node is sampled, the node itself, all of its neighboring nodes, and the edges between them are made observable. Border-acquired nodes, denoted by single solid circles, are those directly connected

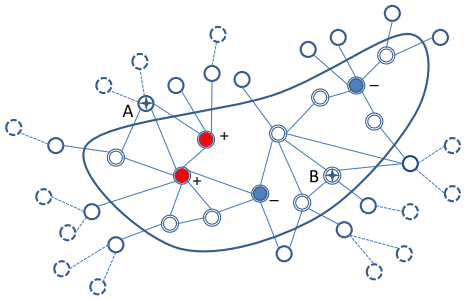

Fig. 1 A partially observed subgraph graph $G_t = (\mathcal{V}_t^l, \mathcal{V}_t^u, \mathcal{E}_t)$. Intra-acquired nodes are denoted by double *solid circles* and nodes directly connected to Intra-acquired nodes are border-acquired nodes. Labeled positive and negative nodes are marked with "+" and "−", respectively. Node A is selected from border-acquired nodes to be sampled next because it has maximum probability of belonging to positive class. Node B is selected to be labeled because it potentially provides a larger influence (according to our formulation) on the partially observed network G_t

to Intra-acquired nodes. Note that, when a subgraph G_t is explored at time step t, it means that all of its nodes, $\mathcal{V}_t = \mathcal{I}_{intra} \cup \mathcal{I}_{border}$, and the edges $\mathcal{E}_t$ between nodes are all observed. There are unlabeled nodes $\mathcal{V}_t^u$ and labeled nodes $\mathcal{V}_t^l$ in subgraph G_t, which can appear in both $\mathcal{I}_{intra}$ and $\mathcal{I}_{border}$. As shown Fig. 1, labeled positive and negative nodes are marked with "+" and "−", respectively.

Formally, an $\mathcal{AE}$ algorithm employs two interleaved processes—sampling and labeling—that iteratively collaborate towards the objective of active exploration. During each iteration, the *sampling* process aims to determine that, for a partially observed subgraph or explored network $G_t = (\mathcal{V}_t^l, \mathcal{V}_t^u, \mathcal{E}_t)$, which node v_i from $\mathcal{I}_{border}$ should be sampled next. At the end of this process, subgraph G_t is expanded to include the new node v_i, its neighbors $v_j \in \mathcal{N}(v_i)$, as well as new edges (v_i, v_j) between nodes v_i and v_j. The *labeling* process is, given a set of explored nodes $\mathcal{V}_t = \mathcal{I}_{intra} \cup \mathcal{I}_{border}$, to select a best node v_k from the unlabeled nodes $\mathcal{V}_t^u \in \mathcal{V}_t$, and queries its label y_k when necessary. The set of labeled nodes is then expanded to include the newly labeled data $\mathcal{V}_{t+1}^l = \mathcal{V}_t^l \cup (v_k, y_k)$. After each iteration, we have a newly updated network $G_{t+1} = (\mathcal{V}_{t+1}^l, \mathcal{V}_{t+1}^u, \mathcal{E}_{t+1})$. The two processes iterate until the limited budget is reached. Table 1 summarizes a list of notations used throughout this paper.

4 Active exploration algorithms

The aim of active exploration is to maximize the identification of important nodes belonging to a desired class while exploring the network under the limited budget constraint. Traditional graph sampling techniques can not be directly applied to achieve this objective, because they assume that nodes are equally important during the sampling process. Therefore, we propose novel algorithms to solve our active exploration problem.

Table 1 Table of notations

Symbol	Meaning
G_t	Explored network at time t, including labeled nodes, unlabeled nodes, and edges
$\mathcal{V}^l$	A set of labeled nodes
$\mathcal{V}^u$	A set of unlabeled nodes
$\mathcal{E}$	A set of edges
(u, v)	An edge between nodes u and v
$\mathcal{I}_{intra}$	Intra-acquired nodes
$\mathcal{I}_{border}$	Border-acquired nodes
$\mathcal{N}(v)$	Neighbors of node v
p_i	Probability of node v_i being positive
$\mathcal{AE}$	An active exploration algorithm
$f_w = (u, v)$	Strength function for edge (u, v)
w	Parameters of strength function
Dis_G	Stationary distribution of network G
$r_{u,v}$	Feature vector of edge (u, v)
$L+$	A set of labeled positive nodes
$L-$	A set of labeled negative nodes
P	Stationary distribution vector
Tr	Transition matrix

4.1 Motivation of our new algorithms

To ensure that our proposed algorithms can indeed explore more positive nodes, our idea is to sample the nodes which are more likely to be positive and choose to label the nodes which have the maximum influence in the network. As an example in Fig. 1, star node A has the maximum probability of being positive, because two of its direct neighbors are already labeled as positive, which are marked with "+" in the figure. Thus, node A is selected to be sampled next. Star node B is selected as the best node to be labeled next because it would have a larger influence on the partially observed network after its label is revealed. Therefore, one important issue is how to calculate the probabilities of nodes being positive and how to compute the influence of nodes in a partially observed network.

To achieve this goal, we model a graph as a Markov chain, where nodes are considered as different interior states and links are chains between states. In particular, we consider two virtual absorbing states: one virtual positive node *vs.* one virtual negative node. We assume that positive nodes are all connected to the virtual positive node, and negative nodes are all connected to the virtual negative node. Let p_i denote the probability of a node v_i being positive, which is calculated as the probability for node v_i to be transferred to the positive absorbing state in the Markov chain. To capture such transition probabilities, we consider a random walk on the Markov chain, in which a walk stops when it reaches an absorbing state. While traditional

random walks assume that transition probabilities of all edges to be the same, our proposed algorithms learn to assign each edge a transition probability such that the random walk is more likely to visit positive nodes than other negative nodes in a network.

In the following subsections, we first formulate supervised random walks as an optimization problem and derive its solution. Based on this, we then present two proposed algorithms to solve the active exploration problem.

4.2 Supervised random walks

Given an observed subgraph G_t, we propose a supervised random walk that naturally combines the information from the network structure with node and edge features. Motivated by Backstrom and Leskovec (2011), we consider biasing the random walk by assigning each edge a random walk transition probability (i.e., strength). Therefore, we aim to learn a strength function $f_w(v, u)$ for each edge (u, v), based on features of nodes u and v, as well as the features of the edge (u, v). Intuitively, a random walk is more likely to traverse an edge of high strength and thus the connected node via the path of the strong edge would be more likely visited by the random walk.

Based on the above problem setting, the task is now to learn the parameters w of a function $f_w(v, u)$ that assigns each edge a transition probability. To achieve this, we formulate an optimization problem:

$$\min_w F(w) = \sum_{y_i \in L+, y_j \in L-} h(p_j - p_i) + \sum_{y_i y_j = 1} \|p_i - p_j\|^2 + \|w\|^2, \tag{1}$$

where $L+$ and $L-$ is a set of labeled nodes with positive and negative labels, respectively. The random walk assigns each node a probability score p, which depends on $f_w(v, u)$ that is parameterized by w. $h(\cdot)$ is a loss function that assigns a non-negative penalty according to the difference of the scores $p_j - p_i$. If $p_j - p_i < 0$, then $h(\cdot) = 0$. If $p_j - p_i > 0$, then $h(\cdot) > 0$. Therefore, the first term indicates that we want the probability scores of nodes in $L+$ to be greater than the scores of nodes in $L-$. The second term indicates that nodes having the same class labels should have close probability scores. The third term is the norm of parameter vector w which indicates the complexity of w. In the following, we discuss how to solve this optimization problem.

As discussed before, each edge (u, v) in a graph has a corresponding feature vector $r_{u,v}$ that describes nodes u and v (e.g., words in paper titles) and the interaction attributes (e.g., the number of words shared between two paper titles). However, as a special case, the two virtual absorbing nodes are not associated with any feature vectors, so we need to find a way to compute the edge feature vectors $r_{s,v}$, where s is a virtual absorbing node and v a labeled node. Because two virtual absorbing nodes are only connected to labeled nodes having the same label, we simply assume the distances between each virtual absorbing node and its direct neighbors are the same. Thus, for the two virtual absorbing nodes, we define the edge feature vectors $r_{s,v} = \mathbf{1}$.

Then, we define the strength function for each edge (u, v) as $R_{u,v} = f_w(r_{u,v})$. For example, the strength function f_w can be a simple linear function of edge features,

that is, $f_w(r_{u,v}) = w^T r$. Function $f_w(\cdot)$ parameterized by w takes the edge feature vector $r_{u,v}$ as input and computes the corresponding edge strength $R_{u,v}$ that models the random walk transition probability. Note that, for each virtual absorbing node, its edges connecting to its neighbors share the same edge strength $R_{s,v}$. Based on edge strength $R_{u,v}$, we can build the random walk stochastic transition matrix Tr as follows:

$$Tr_{u,v} = \begin{cases} \frac{R_{u,v}}{\sum_v R_{u,v}} & \text{if } u, v \in \mathcal{E}, \\ 0 & otherwise. \end{cases} \tag{2}$$

To connect the node probability scores with the strength function $f_w(r_{u,v})$, below we first define the stationary distribution Dis_G for a network G.

Definition 1 (*Stationary distribution* Dis_G) Given a network G, its pseudo stationary distribution Dis_G is defined as $Dis_G = \{p_i | v_i \in G\}$, where p_i indicates the probability of node v_i being positive.

Let P be a stationary distribution vector of the random walk, in which each entry p_v is from the stationary distribution Dis_G. We have

$$1 = \sum_v p_v, \tag{3}$$

and P can be solved by the following eigenvector equation:

$$P^T = P^T Tr. \tag{4}$$

The above equation establishes the relationships between the node probability scores p_v and the parameter w of function $f_w(r_{u,v})$ via the random walk transition matrix Tr.

Now we can minimize Eq. (1) with respect to parameter vector w. The optimization problem can be solved by deriving the gradient of $F(w)$ with respect to w, and then using a gradient based method to find w that minimizes $F(w)$. First, we have derivative of $F(w)$ with respect to w as

$$\begin{aligned} \frac{\partial F(w)}{\partial w} &= \sum_{i \in L+, j \in L-} \frac{\partial h(p_j - p_i)}{\partial w} + \sum_{y_i y_j = 1} \frac{\partial (p_i - p_j)}{\partial w} + 2\|w\|, \\ &= \sum_{i \in L+, j \in L-} \frac{\partial h(p_j - p_i)}{\partial (p_j - p_i)} \left(\frac{\partial p_j}{\partial w} - \frac{\partial p_i}{\partial w} \right) + 2 \sum_{y_i y_j = 1} \left(\frac{\partial p_i}{\partial w} - \frac{\partial p_j}{\partial w} \right) + 2w. \end{aligned} \tag{5}$$

We can easily compute $\frac{\partial h(p_j - p_i)}{\partial (p_j - p_i)}$ when we define a differentiable loss function for $h(.)$, for example squared loss. However, it is difficult to compute $\frac{\partial p_v}{\partial w}$ because we do not have the exact function form of $p(w)$. Therefore, we compute the derivative of

p with respect to the vector w based on Eq. (4). Since Tr is a symmetric matrix, we have

$$p_v = \sum_i p_i Tr_{i,v}. \tag{6}$$

Therefore, the derivative of p_v is given as:

$$\frac{\partial p_v}{\partial w} = \sum_i Tr_{i,v} \frac{\partial p_v}{\partial w} + p_v \frac{\partial Tr_{i,v}}{\partial w}. \tag{7}$$

We can calculate this equation by iteratively computing p_v and $\frac{\partial p_v}{\partial w}$. Firstly, we compute p_v.

- Initialization: for $v \in \mathcal{V}$, let $p_v^{(0)} = \frac{1}{|V|}$.
- Iteration: at step n:

$$p_v^{(n)} = \sum_i p_i^{(n-1)} Tr_{i,v}. \tag{8}$$

Secondly, we compute $\frac{\partial p_v}{\partial w}$. For each $w_c \in w, c = 1, \ldots, |w|$, let $\frac{\partial p_v}{\partial w_c}^{(0)} = 0$ then for $v \in V$, we have

$$\frac{\partial p_v}{\partial w_c}^{(n)} = \sum_i Tr_{i,v} \frac{\partial p_v}{\partial w_c}^{(n-1)} + p_v^{(n-1)} \frac{\partial Tr_{i,v}}{\partial w_c}. \tag{9}$$

To solve Eq. (1), we need to further calculate $\frac{\partial Tr_{i,v}}{\partial w}$ as

$$\frac{\partial Tr_{i,v}}{\partial w} = \frac{\frac{\partial f_w(r_{v,u})}{\partial w}\left(\sum_u f_w(r_{v,u})\right) - f_w(r_{v,u})\left(\sum_u \frac{\partial f_w(r_{v,u})}{\partial w}\right)}{\left(\sum_u f_w(r_{v,u})\right)^2}, \tag{10}$$

where $f_w(r_{v,u})$ is the edge strength function. We define f_w to be differentiable, so $\frac{\partial f_w(r_{v,u})}{\partial w}$ can be easily computed.

We now have an iterative way to compute the derivation $\frac{\partial F(w)}{\partial w}$. Then we compute the updated parameters using a gradient descent based method to solve the optimization problem and obtain optimal values for p and w. We summarize the procedure of computing the stationary distribution in Algorithm 1.

4.3 Single-seed active exploration

Based on supervised random walks discussed in the previous section, we now present a single-seed active exploration algorithm in detail. A preliminary version of this algorithm has been discussed in Fang et al. (2013).

Algorithm 1 StationaryDis(G, $f_w(\cdot)$)

Input: (1) A network $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{V}^l \cup \mathcal{V}^u$;
(2) Strength function: $f_w(\cdot)$;
Output: Stationary distribution Dis_G of network G.

Initialize $w = \mathbf{1}$;
For $v \in \mathcal{V}$, initialize $p_v \leftarrow p_v^0$;
For each $w_c \in w$, $c = 1, \ldots, |w|$, initialize $\frac{\partial p_v}{\partial w_c} \leftarrow \frac{\partial p_v}{\partial w_c}^{(0)}$;
repeat
  Iteratively compute p_v and $\frac{\partial p_v}{\partial w}$ using Eqs. (8-10);
  Compute the Hessian approximation B;
  Compute quasi-Newton direction $\Delta w \leftarrow -B^{-1}\frac{\partial p_v}{\partial w}$;
  Update new parameter $w' \leftarrow w + \Delta w$;
until $|w' - w| < 10^{-7}$;
$Dis_G = \{p_v | v \in \mathcal{V}\}$;
Return Dis_G.

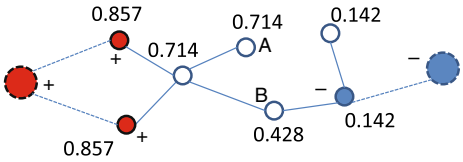

Fig. 2 A small explored network with an optimal stationary distribution. Positive nodes and negative nodes are marked with "+" and "−", respectively. Two *large dashed nodes* denote two virtual absorbing states. The sampling and labeling processes utilize the stationary distribution to select node A to be sampled and node B to be labeled next

4.3.1 Algorithm description

After solving the optimization problem in Eq. (1), we can obtain an optimal stationary distribution Dis_{G_t} for the explored network G_t at time step t, and construct a Markov chain with probabilities accordingly. Figure 2 illustrates a small explored network with an optimal stationary distribution Dis_{G_t}, where each node v is assigned with a probability score p_v, indicating the likelihood of the node being positive. Based on the estimated stationary distribution Dis_{G_t}, below we discuss the selection criteria used for sampling and labeling, respectively.

Sampling The sampling process of active exploration is to bias towards discovering more positive nodes. We select a node which is most likely to be positive and then explore its neighbors, including the nodes and edges. For example, in Fig. 2, node A with probability 0.714 is selected for sampling because it has the maximum probability of being positive in the border area. Intuitively, if a node has a higher value of p_v, it is more likely to be a positive node because it is closer to the virtual positive node. Therefore, we choose a node v^* from bBorder-acquired nodes to be sampled next such that it has the highest value of p_v.

$$v^* = \arg \max_{v \in \mathcal{I}_{border}} p_v. \tag{11}$$

Labeling Labeling is another important process of active exploration, which aims to obtain the labeling information of important nodes. Given a Markov chain with probabilities, we select the most influential node in the explored network G_t and make a query for its label only when necessary. That is, when an influential node is labeled, the stationary probabilities of nodes in the explored network would be largely affected. For example, in Fig. 2, center node B with probability 0.428 is selected to be labeled because its labeling information may largely influence the stationary probabilities of the nodes in the current network.

We measure a node's potential informativeness in terms of its ability to influence the network after being labeled. When we select a node for querying, its actual label is unknown. After labeling, we have a newly labeled node, which means that we change a node's state for our Markov chain. Accordingly, we define this difference as the informativeness of a node. Let Dis_{G_t} denote the stationary distribution of the network G_t, and v denote a node. Before labeling, we have

$$p_v \sim Dis_{G_t}(State = +1|v). \tag{12}$$

After labeling, we recompute the Markov chain as

$$p_v^{'} \sim Dis_{G_t^{'}}(State = +1|v). \tag{13}$$

Intuitively, we assume that a node's label is more important when there is a significant difference in the stationary distribution before and after this node is labeled. We use the KL-divergence to measure the difference between two stationary distributions. Thus we have

$$KL(Dis_{G_t^{'}}|Dis_{G_t}) = \sum_i \ln\left(\frac{p_v^{'}(v_i)}{p_v(v_i)}\right) p_v^{'}(v_i). \tag{14}$$

Before making the query, we do not know the true label of node v. However, we can use an estimate of the distribution from which v's true label would be chosen, p_v, given by the current Markov chain. Since we have two virtual states $+1$ and -1, we compute the expectation by calculating the estimated KL-divergence for the two classes. Let $G_t^{'} : (v, +1)$ denote the network G_t with a newly labeled node $(v, +1)$. According to Algorithm 1, we compute its corresponding stationary distribution and we have $Dis_{G_t^{'}:(v,+1)}$. In the same way, let $G_t^{'} : (v, -1)$ denote the network G_t with a newly labeled node $(v, -1)$. We compute its corresponding stationary distribution using Algorithm 1, and we have $Dis_{G_t^{'}:(v,-1)}$. Because the label of node v can be $+1$ or -1, we compute the average difference using the weight from previously estimated probability p_v. Thus, we have

$$E_{KL_v} = p_v KL\left(Dis_{G_t^{'}:(v,+1)}|Dis_{G_t}\right) + (1 - p_v)KL\left(Dis_{G_t^{'}:(v,-1)}|Dis_{G_t}\right). \tag{15}$$

Then we select a node v^* with the maximum expectation E_{KL_v} and query v^*'s label.

$$v^* = \arg\max_{v \in \mathcal{V}^u} E_{KL_v}. \tag{16}$$

Algorithm 2 Single-Seed Active Exploration

Input: (1) An explored seed network $G_t = (\mathcal{V}_t^l, \mathcal{V}_t^u, \mathcal{E}_t)$;
(2) Threshold of KL-divergence: Thr;
(3) The maximum number of labeled nodes (queries): Budget.
Output: The explored network $G_t = (\mathcal{V}_t^l, \mathcal{V}_t^u, \mathcal{E}_t)$.

1: t=1 and q=1; // t: number of exploration steps; q: number of queries.
2: **while** $q \leq$ Budget **do**
3: $Dis_{G_t} \leftarrow$ StationaryDis$(G_t, f_w(\cdot))$;
4: Select a node for sampling by Eq. (11) and update $\mathcal{V}_t^u$;
5: **for** Each node v in unlabeled node set $\mathcal{V}_t^u$ **do**
6: $\hat{y}_v \leftarrow$ Assign a label $y \in \mathcal{Y}$ to v;
7: $G_t^{'} \leftarrow$ Construct a temporary network with $\hat{y}_v$;
8: $Dis_{G_t^{'}} \leftarrow$ StationaryDis$(G_t^{'}, f_w(\cdot))$;
9: $E_{KL_v} \leftarrow$ Expected KL-divergence between Dis_{G_t} and $Dis_{G_t^{'}}$ by Eq. (15);
10: **end for**
11: $v \leftarrow$ Select a node for labeling by Eq. (16);
12: **if** $E_{KL_v} > Thr$ **then**
13: $y_v \leftarrow$ Query the label of node v;
14: $(\mathcal{V}_t^l, \mathcal{V}_t^u) \leftarrow$ Update with the newly labeled node (v, y_v);
15: $q \leftarrow q + 1$;
16: **end if**
17: $t \leftarrow t + 1$;
18: **end while**

Algorithm 2 lists the detailed procedure of the single-seed active exploration algorithm. This algorithm starts with a single seed (i.e., a few connected nodes), and iteratively samples and labels other nodes in the network. At each step t, we construct a Markov chain based on the subgraph obtained so far, and compute the optimal stationary distribution Dis_{G_t} (line 3). After that, the sampling process determines which node should be sampled next using Eq. (11) (line 4). The labeling process selects the most informative node using Eq. (16) and queries its label when necessary (lines 5–16).

Only by querying, the true label of a selected node can be observed. Therefore, if no query is issued at step t, the label of the selected node remains unknown. Because querying a node's true label incurs a cost, we employ a threshold Thr to determine whether or not to issue a query at step t (lines 12–16). Specifically, our algorithm issues a query when the expectation E_{KL_v} is larger than a given threshold Thr. Since E_{KL_v} indicates the influence of a selected node on the graph when its actual label is

observed, we progressively select to label a node which has a large value of E_{KL_v}. In Section 5.4, we empirically evaluate the impact of different Thr values on the algorithm performance.

4.3.2 Complexity analysis

As indicated in Algorithm 2, the most computationally expensive part of the single-seed active exploration algorithm is the labeling process, because it needs to iterate over all unlabeled nodes to identify the best node to be labeled next. For each unlabeled node, it requires to calculate the difference in the stationary distributions before and after this node is presumably labeled. Therefore, its complexity is $O(tn^3)$, where t is the number of exploration steps, and n is the size of the explored network. Clearly, the efficiency of single-seed active exploration is asymptotically bounded by the size of the explored network n.

4.4 Multi-seed active exploration

The above analysis shows that the complexity of single-seed active exploration is $O(n^3)$, where n is the size of the explored network. When n continuously increases on large graphs, the single-seed algorithm becomes computationally expensive and even infeasible. In order to scale up the algorithm, we propose to use a multi-seed framework for sampling and labeling, which simultaneously initializes multiple exploration processes rather than a single exploration process. Within this new framework, each exploration has its own explored network and only needs to calculate the stationary distribution of its local network. Since the size of each exploration network is much smaller than that of a single network, the scalability of the algorithm can be significantly improved. Below, we detail the multi-seed active exploration algorithm.

4.4.1 Algorithm description

Multi-seed active exploration initializes from k different seeds and runs all exploration processes in parallel. Each exploration process has its own explored network and computes the corresponding stationary distribution of its local network. Because different explored networks may overlap and share common nodes, the multi-seed algorithm allows each exploration process to make its local decision independently about which node should be sampled next in its own network, and also enable multiple exploration processes to make a global decision collectively to select the most informative node to be labeled next.

Suppose that we have $k \geq 2$ exploration processes, and each exploration process has one explored network $G_{i,t}$ up to time step t, where $1 \leq i \leq k$. For each explored network $G_{i,t}$, we optimize the function $F(w)$ and calculate the stationary distribution $Dis_{G_{i,t}}$ independently, as defined as

$$f_{w_{i(t)},t} : v \rightarrow Dis_{G_{i,t}}, where\ v \in G_{i,t}, \tag{17}$$

Thus, for each exploration, we obtain the probability score p_v drawn from $Dis_{G_{i,t}}$ independently from the others.

Sampling For each exploration process, we make a local decision to determine which node should be sampled next in each explored network. Similar to the single-seed algorithm, we choose to label a node which is most likely to be positive, and then explore its neighbors, including nodes and edges. Specifically, for each explored network $G_{i,t}$, we select a node v_i^* to be sampled next such that it has the highest probability score $p_{i,v}$:

$$v_i^* = \arg \max_{v \in \mathcal{I}_{border}^i} p_{i,v}. \tag{18}$$

All together, at each step, we sample k nodes $\{v_1^*, v_2^*, \ldots, v_k^*\}$, one from each exploration process.

Labeling Because different exploration processes may share overlapped areas in their explored networks, we resort to selecting one optimal node v^* to be labeled from multiple explorations,

$$v^* = \arg \max_{v \in \{v | v \in \mathcal{V}_{i,t}^u, i=1,\ldots,k\}} \Delta \mathcal{D}_v, \tag{19}$$

where $\Delta \mathcal{D}_v$ is the measure that indicates the influence of a node on the explored network after it is labeled. For a given node v, if $\Delta \mathcal{D}_v$ is large, that means once node v is labeled, it can largely affect other nodes in the network. Otherwise, if $\Delta \mathcal{D}_v$ is small, that means node v has little impact on other nodes. Therefore, we would like to select the most influential node among multiple explored networks and query for its label.

In order to measure a node's influence on the network, we consider each exploration network $G_{i,t}, i = 1, \ldots, k$ as a Markov chain with the stationary distribution $Dis_{G_{i,t}}$. If an influential node v is labeled, the status of other nodes in a Markov chain would be largely changed. By solving the optimization problem in Eq. (1), we can have an updated stationary distribution $Dis_{G'_{i,t}}$ assuming node v is labeled. Given two distributions $Dis_{G_{i,t}}$ and $Dis_{G'_{i,t}}$, we thus use the KL-divergence of two stationary distributions to define the influence of a node

$$\Delta \mathcal{D}_v = KL\left(Dis_{G'_{i,t}} | Dis_{G_{i,t}}\right). \tag{20}$$

When a node is selected for querying, its actual label is unknown. Therefore we calculate the expectation of $\Delta \mathcal{D}_v$

$$E_v[\Delta \mathcal{D}_v] = E_v\left(KL\left(Dis_{G'_{i,t}} | Dis_{G_{i,t}}\right)\right). \tag{21}$$

Similarly, before making the query, the true label of node v is unknown. Instead, we can use an estimate of the distribution from which v's true label would be chosen,

p_v, given by the current Markov chain. After labeling, the label of node v is y_v and $y_v \in \mathcal{Y}$. We rewrite Eq. (21) as

$$E_v[\Delta\mathcal{D}_v] = \sum_{y_v \in \mathcal{Y}} p_v KL\left(Dis_{G'_{i,t}} | Dis_{G_{i,t}}\right). \tag{22}$$

Because we have two virtual states $+1$ and -1, we compute the expectation by calculating the estimated KL-divergence for the two classes and take the average weighted by the previously estimated probability p_v. As a result, we have

$$E_v[\Delta\mathcal{D}_v] = p_v KL\left(Dis_{G'_{i,t}} | Dis_{G_{i,t}}\right)|_{y_v=+1} + (1-p_v) KL\left(Dis_{G'_{i,t}} | Dis_{G_{i,t}}\right)|_{y_v=-1}. \tag{23}$$

In the multi-seed setting, if node v only belongs to one explored network, we can directly compute $E_v[\Delta\mathcal{D}_v]$. In practice, it is possible that node v is sampled by multiple explored networks, $\mathcal{M}_v = \{G_{i,t} | \exists v \in G_{i,t}, i = 1, 2, \ldots, k\}$. We can compute the difference by two ways. One way is to calculate the average difference over different explorations. Given a set of explorations $\mathcal{M}_v$, we have

$$\begin{aligned}\hat{E}_v[\Delta\mathcal{D}_v] = \frac{1}{|\mathcal{M}_v|} \sum_{\mathcal{M}_v} \Bigg\{ & p_v KL\left(Dis_{G'_{i,t}} | Dis_{G_{i,t}}\right)|_{y_v=+1} \\ & + (1-p_v) KL\left(Dis_{G'_{i,t}} | Dis_{G_{i,t}}\right)|_{y_v=-1} \Bigg\}.\end{aligned} \tag{24}$$

Another way is to compute the maximum difference from different explorations. Therefore, we have

$$\begin{aligned}\hat{E}_v[\Delta\mathcal{D}_v] = \max_{\mathcal{M}_v} \Bigg\{ & p_v KL\left(Dis_{G'_{i,t}} | Dis_{G_{i,t}}\right)|_{y_v=+1} \\ & + (1-p_v) KL\left(Dis_{G'_{i,t}} | Dis_{G_{i,t}}\right)|_{y_v=-1} \Bigg\}.\end{aligned} \tag{25}$$

In our design, we use Eq. (25) to compute the expectation of $\Delta\mathcal{D}_v$ among multiple explorations. This is mainly because a node could have a large impact only on one exploration process, whereas using the average can smooth out its significance across multiple explorations. By doing so, we assume that if a node has a large impact on one exploration, it should be labeled accordingly. In order to find the best node to be labeled, we use the following equation which is calculated over all the explored networks

$$v^* = \arg \max_{v \in \{v | v \in \mathcal{V}^u_{i,t}, i=1,\ldots,k\}} \hat{E}_v[\Delta\mathcal{D}_v]. \tag{26}$$

Following the above process, we select a node v^* that has the maximum expectation $\hat{E}_v[\Delta\mathcal{D}_v]$ among all explorations and query its label. If node v^* is shared by multiple explorations, adding its new label will subsequently influence the sampling and labeling of these explorations at the next step.

The detailed procedure of multi-seed active exploration is given in Algorithm 3, where k parallel exploration processes run simultaneously. For each exploration, the stationary distribution is computed independently, a best node is selected to be sampled next (lines 4–7). A joint decision is made among multiple explorations to decide which node should be labeled next and query its label when necessary (lines 8–21). Similar to single-seed active exploration, the multi-seed algorithm also uses a threshold Thr to control whether or not to issue a query at step t (lines 17–21). For a selected node v, if the expectation $\hat{E}_v[\Delta\mathcal{D}_v]$ is larger than a given threshold Thr, the algorithm issues a query and obtains node v's label. Otherwise, the label of node v remains unknown. Because $\hat{E}_v[\Delta\mathcal{D}_v]$ indicates the influence of a selected node on the graph when its actual label is observed, we progressively select to label a node which has a large value of $\hat{E}_v[\Delta\mathcal{D}_v]$.

Algorithm 3 Multiple-Seed Active Exploration

Input: (1) k explored seed networks $G_{i,t} = (\mathcal{V}_t^l, \mathcal{V}_t^u, \mathcal{E}_t), i = 1, \cdots, k$;
(2) Threshold of KL-divergence: Thr;
(3) The maximum number of labeled nodes (queries): Budget.
Output: The explored network $G_t = (\mathcal{V}_t^l, \mathcal{V}_t^u, \mathcal{E}_t)$.

1: Initialize k seeds for k exploration;
2: t=1 and q=1; // t: number of exploration steps; q: number of queries.
3: **while** q $\leq$ Budget **do**
4: **for** Each exploration k **do**
5: $Dis_{G_{k,t}} \leftarrow$ StationaryDis($G_{k,t}$, $f_w(.)$);
6: Select a node for sampling by Eq. (18);
7: **end for**
8: **for** Each exploration k **do**
9: **for** Each node v in unlabeled node set $\mathcal{V}_{k,t}^u$ **do**
10: $\hat{y}_v \leftarrow$ Assign a label $y \in \mathcal{Y}$ to node v;
11: $G_t^{'} \leftarrow$ Construct a temporary network with $\hat{y}_v$;
12: $Dis_{G_{k,t}^{'}} \leftarrow$ StationaryDis($G_{k,t}^{'}$, $f_w(.)$);
13: $\hat{E}_v[\Delta\mathcal{D}_v] \leftarrow$ Expected KL-divergence between $Dis_{G_{k,t}}$ and $Dis_{G_{k,t}^{'}}$ by Eq. (25);
14: **end for**
15: **end for**
16: Select a node v for labeling by Eq. (26);
17: **if** $\hat{E}_v[\Delta\mathcal{D}_v] > Thr$ **then**
18: $y_v \leftarrow$ Query the label of v;
19: $(\mathcal{V}_t^l, \mathcal{V}_t^u) \leftarrow$ Update with the newly labeled node (v, y_v);
20: $q \leftarrow q + 1$;
21: **end if**
22: $t \leftarrow t + 1$;
23: **end while**

4.4.2 Complexity analysis

Now we analyze the complexity of the multi-seed active exploration algorithm, which runs k explorations in parallel. For each exploration, the complexity of computing the stationary distribution is $O(tn'^2)$, where t is the number of exploration steps, and n' is the size of each explored network. The labeling process uses a pool-based active learning strategy to examine each unlabeled node in the pool iteratively, and thus its complexity is $O(tkn'^3)$. Comparing with one-seed active exploration, we have $n' \sim n/k$ and $k \ll n'$. Thus, the time complexity of multi-seed active exploration can be reduced significantly, especially for a large number of exploration processes k. The multi-seed algorithm runs k^2 faster than a single-seed exploration. For example, when $n = 10{,}000$ and $k = 10$, multi-seed active exploration is 100 times faster than a single-seed active exploration.

5 Experiments

In this section, we empirically validate the effectiveness of our proposed active exploration algorithms. In Sect. 5.1, we first discuss general settings of our experiments, including datasets, performance metrics, and baseline methods. In Sect. 5.2 and Sect. 5.3, we evaluate the performance of single-seed active exploration on synthetic and real-world networks, respectively. Then, we investigate the impact of different Thr values on single-seed active exploration in Sect. 5.4. The comparison between single-seed active exploration and multi-seed active exploration is reported in Sect. 5.5.

5.1 Experimental settings

In our experiments, we use both synthetic networks and real-world networks to validate the performance of our proposed algorithms and compare with several baseline methods.

5.1.1 Benchmark networks

To study the algorithm performance with respect to different network features, we generate scale-free graphs with 400 nodes and 4000–6000 edges to simulate networks, including label information and features for the network nodes. Because real-world networks usually have community structures, we use a random graph to create network components, each containing a number of nodes, and then connect these components by randomly creating edges between different components (Erdős and Rényi 1959). To generate a class label for each node, we simply assign all nodes within one component as one class (we focus on binary classification problems so each node is labeled as either $+1$ or -1). Details about synthetic networks are described in Sect. 5.2.

In addition to synthetic networks, we also validate our proposed algorithms on three real-world networks, including PubMed, CiteSeer and Core citation networks.[5] Detailed information about the three citation networks is reported in Sect. 5.3.

[5] http://www.cs.umd.edu/projects/linqs/projects/lbc.

5.1.2 Baseline methods

In the experiments, our two proposed algorithms, single-seed active exploration and multi-seed active exploration, are referred to as **Single-Seed AEGraph** and **Multi-Seed AEGraph**, respectively. To the best of our knowledge, there is no existing method which addresses active exploration on graphs. To study the empirical performance of our proposed algorithms, we use four baseline methods for comparison:

- *Random* This method carries out network sampling and labeling in a uniform random manner. At each iteration, it simply samples a node chosen uniformly at random from the neighbors of already sampled nodes, and then selects an unlabeled node to label at random.
- *Degree* This method uses node degree as the measure to guide the sampling and labeling processes because many static graph sampling methods such as BFS are found biased towards high degree nodes in the graph. At each iteration, it samples a node with the maximum node degree and explores the neighbors of the selected node. The unlabeled node with the maximum degree is also labeled during the labeling process.
- *Unweighted* This is a variant of our proposed AEGraph algorithm by removing the weight optimization module. In other words, this method does not consider node features and there is no strength function for each edge. At each step, this method computes the standard stationary distribution of a random walk. It samples a node with the maximum probability score and also selects an unlabeled node with the maximum probability to be labeled.
- *Fixed* Instead of learning the weights for edge features as our propose AEGraph algorithm, this method simply computes the edge strength using a linear combination of the features with fixed weights and the weight of each feature is set to be one.

5.1.3 Performance metrics

We evaluate the performance of active exploration algorithms by using the following three metrics.

(1) *Recall* Because the goal of active exploration is to maximize the identification of positive nodes in the sampled subgraph, we use recall to compare different methods with respect to different sizes of explored networks. In our experiments, because we know genuine labels of all nodes in the network, we first withhold the labeling information of all nodes to carry out the sampling and labeling processes (i.e., without using the labels of nodes). Only after a node is selected for labeling, we assign its genuine label back to the node. We can thus compute recall as the number of positive nodes that have been explored divided by the number of genuine positive nodes. Because we always assign the genuine label (rather than the predicted label) to the node selected for querying, there is no need to use precision as a metric here.

(2) *Classification accuracy* To evaluate the quality of the explored network, we also assess the performance of the classifier trained from the obtained explored net-

work. We can collect different explored networks, including unlabeled nodes, labeled nodes and links, using different exploration strategies. After that, we train a classifier based on these networks solely using labeled nodes in the network, and then compare their performance. In our work, we choose the collective classification (Sen et al. 2008) as our classifier and predict the labels of unlabeled nodes in the explored network.

(3) *Processing time* To study the scalability of our proposed algorithms, we also compare the efficiency of Single-Seed AEGraph and Multi-Seed AEGraph with respect to their processing time.

5.1.4 General parameter settings

For active exploration, we need to select several initial nodes as a seed to start each exploration process. In our experiments, we started with a small connected network (with 5 % nodes) containing positive or negative nodes and unlabeled nodes which are randomly selected. After that, the algorithm iteratively explores the network by carrying out sampling and labeling simultaneously.

In the following, we detail the parameter settings of our proposed active exploration algorithms.

– *Edge strength function* We employ a linear function $f_w(\cdot)$ to calculate the edge strength. Let r denote the feature vector of the edge connecting nodes u and v, $f_w(\cdot)$ is defined as:

$$f_w(r_{u,v}) = w^T r. \tag{27}$$

– *Loss function* To define the penalty for the optimization function in Eq. (1), we use a common squared loss with margin b as:

$$h(x) = max\{x + b, 0\}^2. \tag{28}$$

– *Parameter* Thr The proposed active exploration algorithms contain the iterative sampling and labeling processes. In practice, labeling is considered much more expensive than sampling, so the algorithm does not need to query and label a node at each iteration. In our algorithms, the threshold Thr is used to control whether a selected node needs to be labeled or not. For synthetic networks, we empirically set $Thr = 0.1$ because we find this threshold value achieves a good balanced recall for positive nodes, with respect to the number of queries. For real-world networks, we use $Thr = 0.001$ and report the results in Sect. 5.3. In Sect. 5.4, we evaluate the impact of different Thr values on the algorithm performance.

5.2 Performance of single-seed active exploration on synthetic data

In this section, we focus on evaluating the performance of single-seed active exploration on three synthetic networks with different network characteristics.

5.2.1 Synthetic networks

In our experiments, we build three synthetic networks, each containing two labels: positive and negative. The three networks have different network features with different levels of biased node distributions. The details are summarized as follows:

- *P200-N200* The P200-N200 network contains two components, each having 200 nodes. Each node has six random edges on average. The two components each belongs to one class. We define one component as positive, and select 30 % of positive nodes to be connected with four nodes randomly selected from the other component.
- *P100-N300* The P100-N300 network contains two components, which have 100 and 300 nodes, respectively. The component with 100 nodes belongs to the positive class, and the second component belongs to the negative class. Each node in the network has six random edges on average. After that, we randomly create 480 edges between two components. This network is used to simulate real-world situation with moderately biased node distributions.
- *P50*2-N300* The P50*2-N300 network contains three components, where the largest one contains 300 nodes, belonging to the negative class, and the other two components each contains 50 nodes, belonging to the positive class. Meanwhile, each node has six randomly connected edges within its component. After that, we create 480 edges to randomly connect the three components. This network is used to simulate real-world situation with severely biased node distributions. A snapshot of the three networks is shown in Fig. 3.

For each node in the networks, we create two node features: (1) the first feature is a random variable which follows a zero mean (variance $\sigma = 1$) Gaussian distribution. It acts as a noisy feature without any specific meaning; and (2) the second feature is also a random variable with Gaussian distribution but is subject to different means. Specifically, if a node belongs to the positive class, it follows a Gaussian distribution with $\mathcal{N}(0, 1)$. If the node belongs to the negative class, it follows a Gaussian distribution with $\mathcal{N}(1, 1)$. In addition, given an edge (v, u) with two nodes u and v, we define the edge feature as:

$$r_{v,u} = |x_v - x_u|, \tag{29}$$

where $|\cdot|$ is 1-norm and $r_{v,u}$ indicates the difference between the node features of v and u. If the nodes have similar features, their difference is small, and vice versa.

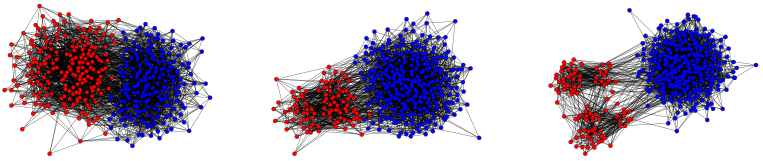

Fig. 3 A snapshot of three synthetic networks with different levels of biased node distributions. **a** P200-N200. **b** P100-N300. **c** P50*2-N300

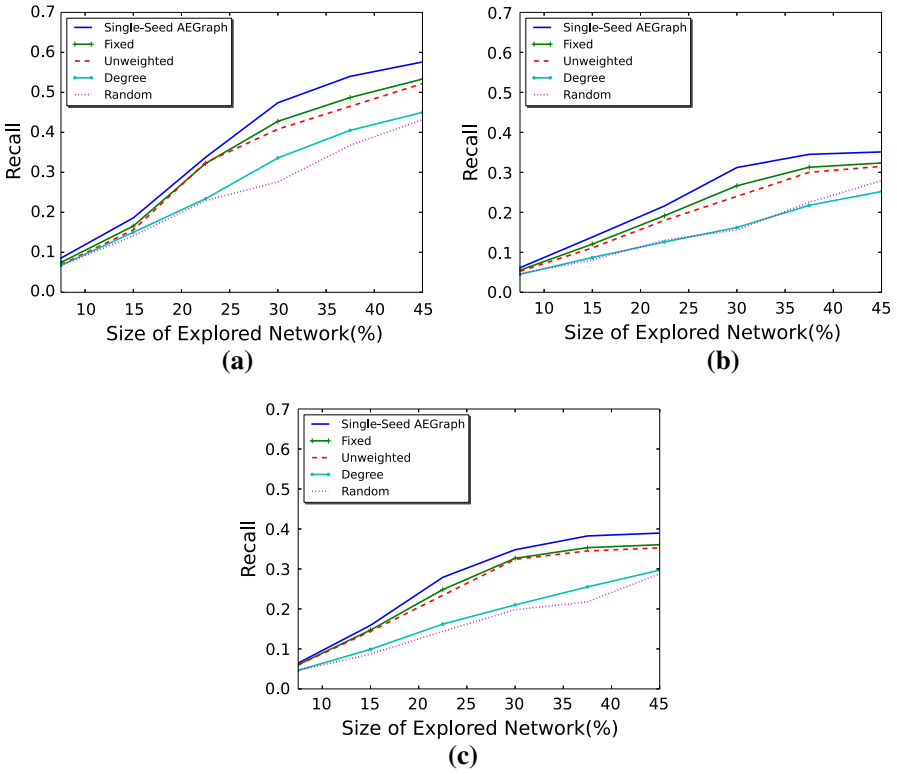

Fig. 4 Recall of positive nodes with respect to different sizes of explored networks. **a** P200-N200. **b** P100-N300. **c** P50*2-N300

5.2.2 Results

Figures 4 and 5 report the recall of positive nodes with respect to different sizes of explored networks (Fig. 4) and different numbers of labeled nodes (Fig. 5).

The results in Fig. 4 show that biased sampling can help acquire more positive nodes. Single-Seed AEGraph, fixed and unweighted all actively sample positive nodes by utilizing the probability score of each node, leading to higher recall values than others for the same size of explored network. Meanwhile, Single-Seed AEGraph performs better than fixed and unweighted. This is because nodes with the same class label in synthetic networks are correlated in the feature space. Single-Seed AEGraph leverages the correlations, whereas Unweighted discards the edge strength (which captures the node correlations) and fixed ignores different contributions of edge features to computing the edge strength. The recall achieved by degree and random are very close to each other, and are significantly inferior to Single-Seed AEGraph. Both degree and random select next node solely based on the structure of the explored network (without any active sampling strategy). While random completely follows a random approach to sample the network, the sampling process of degree is biased to favor nodes with a higher degree. In practice, positive nodes does not necessarily have a high degree,

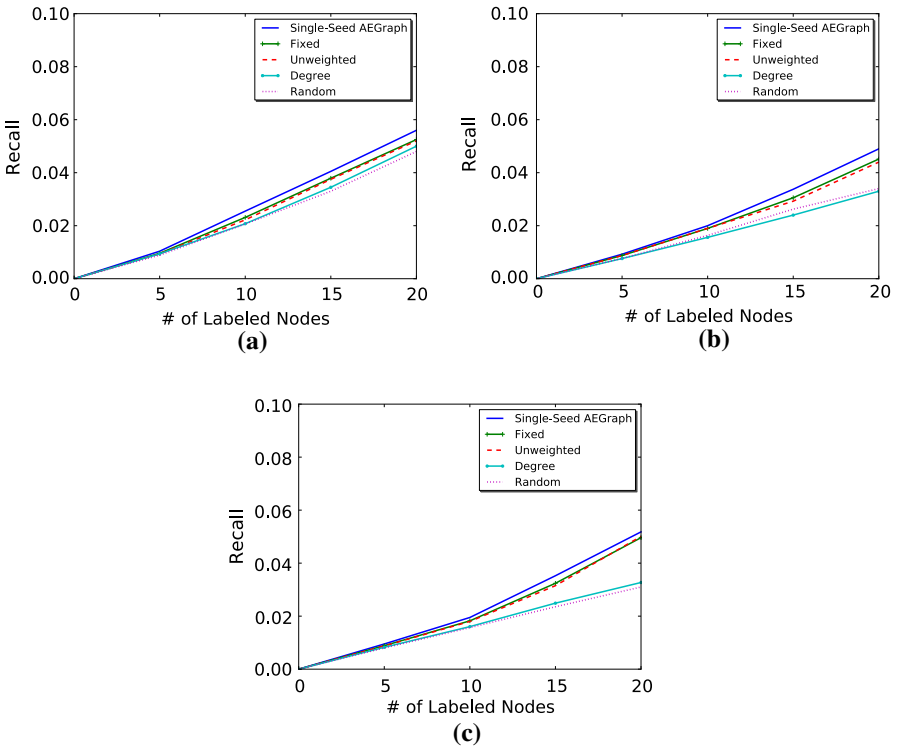

Fig. 5 Recall of positive nodes in explored networks with respect to different numbers of labeled nodes. **a** P200-N200. **b** P100-N300. **c** P50*2-N300

which explains why Degree fails in achieving good performance. Expectedly, Random is not effective in acquiring positive nodes.

Figure 4 also shows that the recall value achieved by Single-Seed AEGraph increases with a larger slope at the early exploration stage (compared to the later stages). This is because in our experiments we assume that the total number of positive nodes is given and fixed. As the exploration process continuously discovers positive nodes, the number of remaining undiscovered positive nodes would decrease, which makes it more difficult to identify them.

The recall of positive nodes with respect to different numbers of labeled nodes is shown in Fig. 5. The results show that our active exploration method has good performance in labeling positive nodes. Because Single-Seed AEGraph has better performance in finding positive nodes during the active exploration process, there are more discovered positive nodes in the pool to be evaluated in order to find and label the most important positive nodes, which in turn strengthens the chance of finding positive nodes in the future.

To evaluate the quality of explored networks, we compare the classification accuracies of the classifiers trained from the explored networks which are collected by different exploration strategies, such as Single-Seed AEGraph, fixed, weighted, degree and random. In Fig. 6, the results show that Single-Seed AEGraph achieves much better

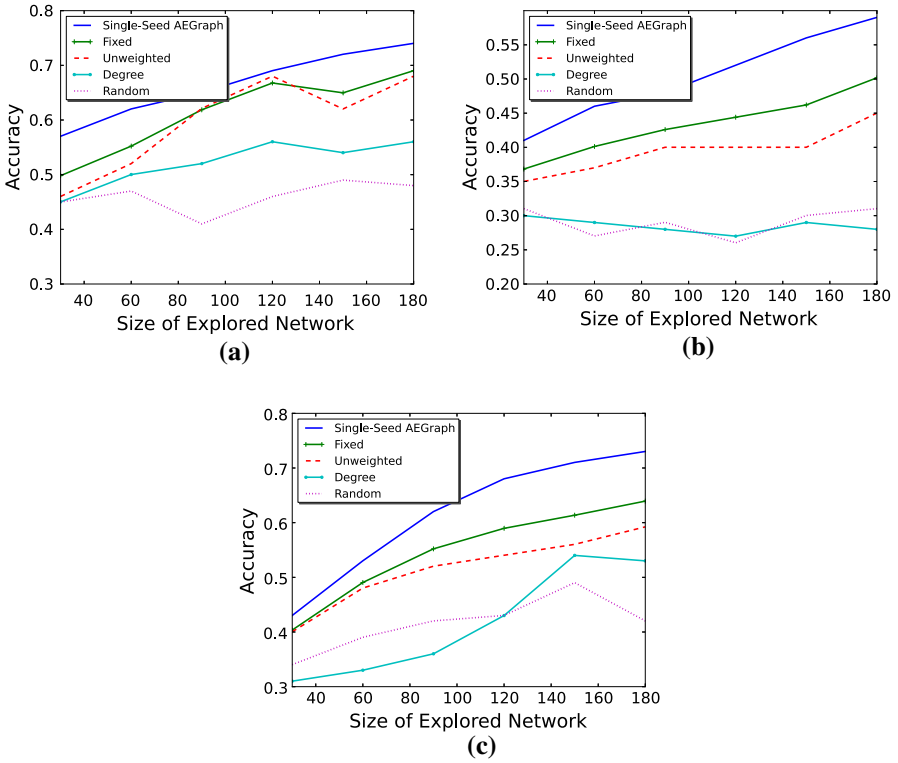

Fig. 6 Classification accuracies based on explored networks using different exploration strategies. **a** P200-N200. **b** P100-N300. **c** P50*2-N300

performance than others. Unweighted is worse because it compute a standard random walk without considering node/edge features. Although fixed takes the features into account, it lacks a learning module to optimize the weights. Thus, its performance is worse than that of Single-Seed AEGraph. Degree and random have the worst performance. These strategies can not collect sufficient positive nodes to train an accurate classifier. Single-Seed AEGraph has the ability to label more positive data than other methods, which helps greatly enhance the classification performance.

5.3 Performance of single-seed active exploration on real-world data

For real-world networks, we used three datasets: PubMed, CiteSeer, and Cora. The general information about these networks is given in Table 2 and their detailed descriptions are discussed as follows.

– *CiteSeer* The CiteSeer network consists of 3312 scientific publications and 4732 citation links. Each node is represented by a 0/1-valued word vector indicating absence/presence of the corresponding word from a dictionary of 3703 words, and is labeled as one of six classes: *Databases (DB)*, *Machine Learning (ML)*, *Information Retrieval (IR)*, *Artificial Intelligence (AI)*, *Human Computer Interaction*

Table 2 Summary of the three real-world networks: CiteSeer, Cora and PubMed

Data set	CiteSeer	Cora	PubMed
# of instances	3312	2708	19,717
# of links	4732	5429	44,338
# of instances of the largest class	701	818	7875
# of instances of the smallest class	249	180	4103

(HCI), and *Agents*. For our active exploration problem, we define the smallest class "*AI*" as the positive and the rest as negative, and explore a network for "*AI*".

- *Cora* The Cora network contains 2,708 scientific publications classified into one of seven classes: *Probabilistic Methods*, *Neural Networks*, *Case Based*, *Rule Learning*, *Reinforcement Learning*, *Genetic Algorithms*, and *Theory*. The citation network contains 5429 links. We consider the smallest class "*Rule Learning*" as positive and others as negative, and explore a network for "*Rule Learning*".
- *PubMed* The PubMed network consists of 19,717 scientific publications from the PubMed database pertaining to diabetes, and each of them belongs to one of three classes: "Diabetes Mellitus, Experimental" (7739), "Diabetes Mellitus Type 1" (7875), and "Diabetes Mellitus Type 2" (4103) (The number in the bracket denotes the number of papers in each class). The citation network consists of 44,338 links. We used the PubMed network as a case study to construct three exploration problems for its three classes, respectively:
  - *Problem 1* we define "Diabetes Mellitus, Experimental" as positive and others as negative, and explore a network for "Diabetes Mellitus, Experimental";
  - *Problem 2* we define "Diabetes Mellitus Type 1" as positive and others as negative, and explore a network for "Diabetes Mellitus Type 1";
  - *Problem 3* we define "Diabetes Mellitus Type 2" as positive and others as negative, and explore a network for "Diabetes Mellitus Type 2".

In our experiments, we use node features to construct edge features. For each edge between two nodes, each representing a paper, the first edge feature is the number of shared words between two papers, defined as:

$$r^1_{u,v} = k, k = \left|\{w|w \in W_u \bigcap W_v\}\right|, \tag{30}$$

where W denotes the words that a paper contains. The second edge feature is defined as the cosine similarity between two papers,

$$r^2_{u,v} = \cos(\mathbf{w}_u, \mathbf{w}_v), \tag{31}$$

where $\mathbf{w}$ is the bag-of-word vector to represent each paper using the occurrence of the words in the paper (Namata et al. 2009). The edge strength function and loss function are the same as the one used for synthetic networks.

Figure 7 reports the recall of positive nodes with respect to different sizes of explored networks. It shows that Single-Seed AEGraph, Fixed and Unweighted work better

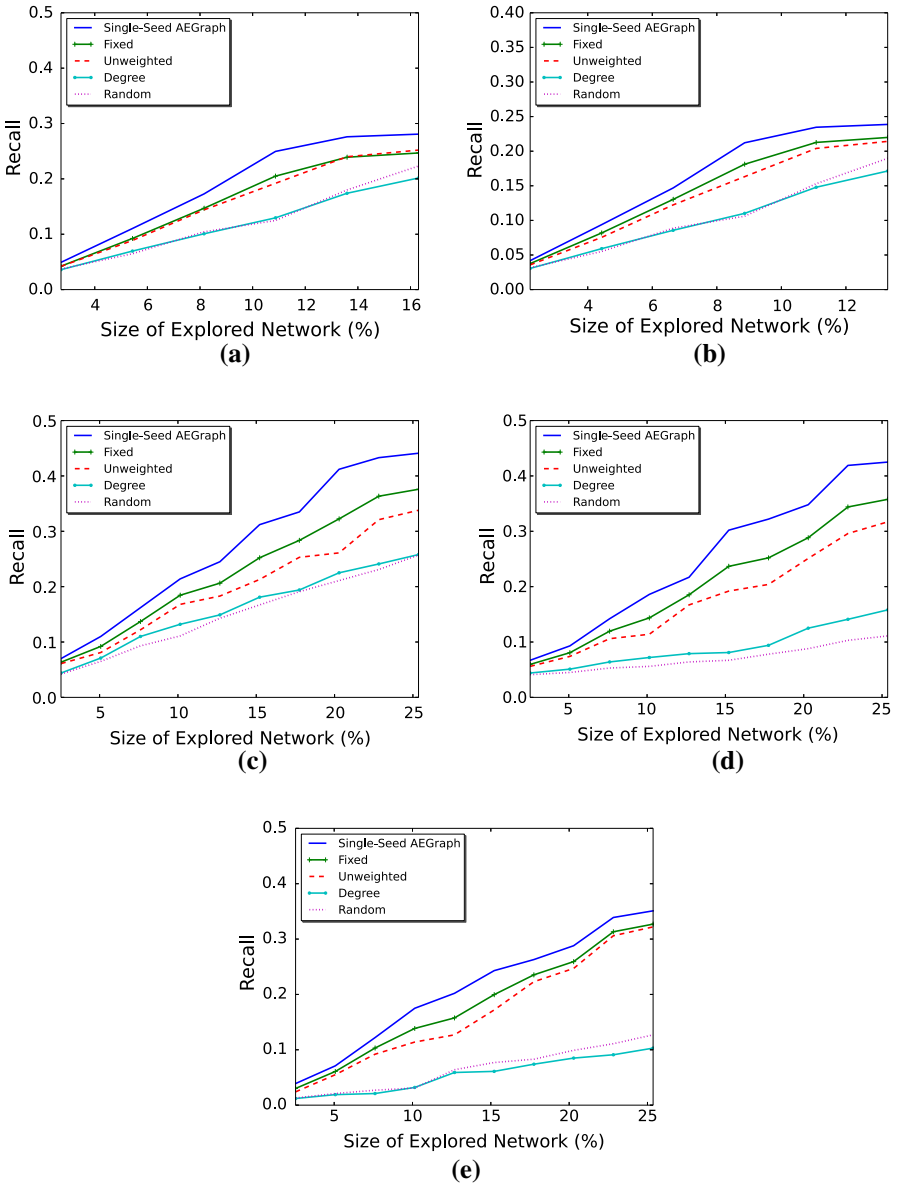

Fig. 7 Recall of positive nodes with respect to different sizes of explored networks. **a** CiteSeer. **b** Cora. **c** PubMed: diabetes mellitus, experimental. **d** PubMed: diabetes mellitus type 1. **e** PubMed: diabetes mellitus type 2

than degree and random, which do not use active sampling strategy for identifying positive nodes. In addition, Single-Seed AEGraph consistently outperforms fixed and unweighted in the five figures. This is because papers in the same class often share common keywords, which is captured by the edge strength function defined in Single-Seed AEGraph. In contrast, Fixed treats each edge feature equally important without con-

sidering their distinct contributions to calculating the edge strength, while Unweighted discards edge strength and therefore ignores the degree of correlations between papers during the sampling process.

The results in Fig. 7c, d show that Single-Seed AEGraph has a larger slope of improvement at the beginning of the sampling process. After 4000 exploration iterations, the recall values become relatively stable. This demonstrates that Single-Seed AEGraph has good performance when the exploration process starts. It can thus potentially find useful positive nodes with very little cost. The decreasing slope of performance improvement, at the latter stage of the sampling process, is mainly because the number of undiscovered positive nodes decreases so it becomes more difficult to find them.

In Fig. 8, we report the recall of positive nodes with respect to different numbers of labeled nodes. We can see that, the recall values of Single-Seed AEGraph increase quickly during the beginning and middle stage. This is mainly because there are many positive nodes with high clustering coefficients (i.e., many positive nodes are connected to each other). The dense network structures allow each labeled positive node to help discover more positive nodes in the next iteration. So our proposed algorithm shows a high recall improvement slope.

Figure 9 reports the node classification accuracies of the classifiers trained from different explored networks. The x-axis in the figures denotes the size of the explored network and the y-axis shows the classification accuracy based on the explored networks obtained by using different strategies. The results in Fig. 9 show that Single-Seed AEGraph outperforms other approaches. This is, in fact, understandable because the previous results have already demonstrated that our proposed algorithm can collect and label more positive data than other methods. As a result, it can effectively balance the positive and negative nodes in the explored network. This is especially helpful when positive nodes in the whole network are rather few, which is normally the case for real-world networks where the positive nodes (or events/nodes satisfying certain criteria) are only a small portion of the entire network. Although Fixed is the baseline method considering the features, it does not have a learning component to optimize the weights for the edge strength function. Consequently, its performance is inferior to that of our proposed algorithm. Unweighted performs even worse because it ignores the features while computing the standard random walk. Degree and Random have the worst performance because they never consider the roles of positive nodes in the exploration process.

Overall, the results in Fig. 9 demonstrate that our proposed algorithm can indeed advance the process of active exploration over large graphs. One can start from a very small set of labeled nodes to actively expand and explore the network, with the explored network covering as many positive nodes as possible. During the active exploration process, we can also provide the most promising nodes for the experts to label, with the labeled nodes being integrated into the active exploration process to further improve the exploration performance in the round.

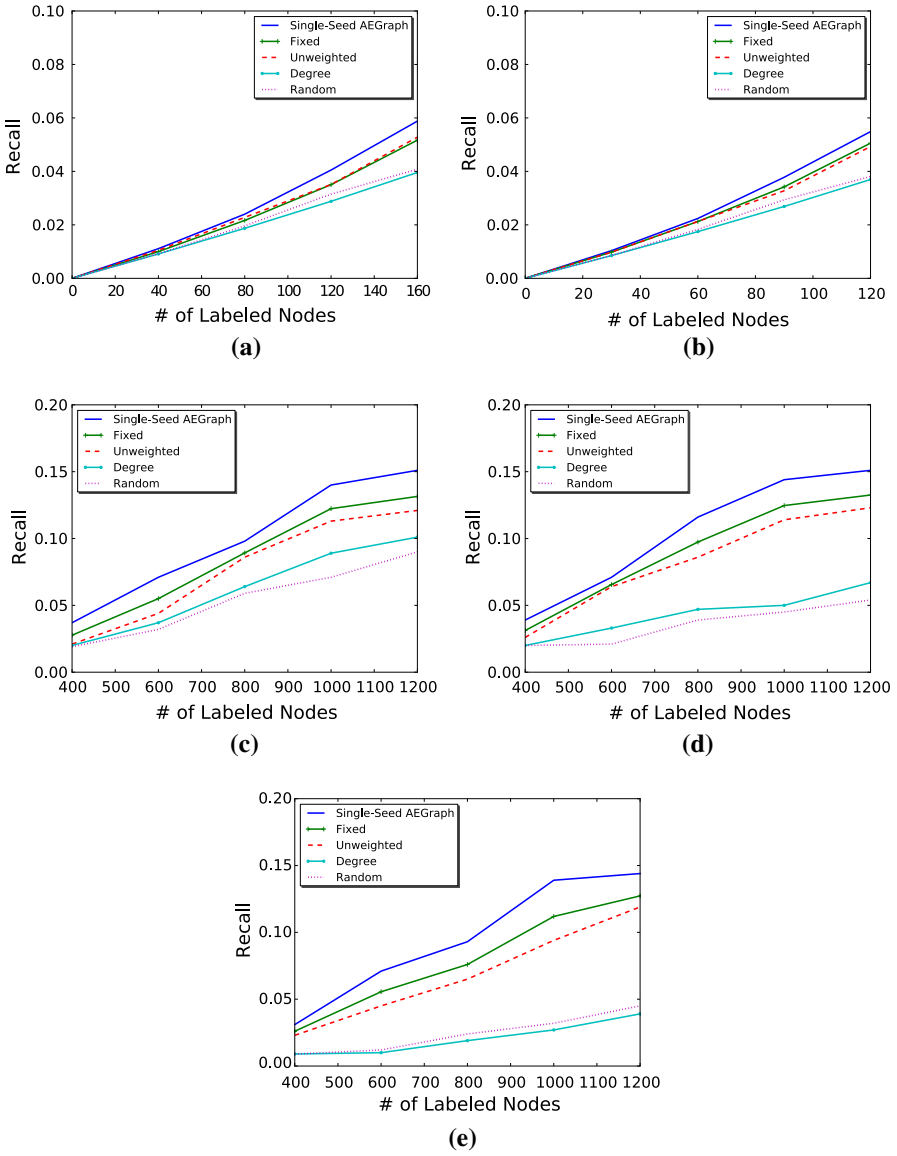

Fig. 8 Recall of positive nodes in explored networks with respect to different numbers of labeled nodes. **a** CiteSeer. **b** Cora. **c** PubMed: diabetes mellitus, experimental. **d** PubMed: diabetes mellitus type 1. **e** PubMed: diabetes mellitus type 2

5.4 Impact of threshold Thr

In our proposed Single-Seed AEGraph algorithm, threshold Thr controls whether a node with the maximum KL divergence E_{KL} should be labeled or not at each iteration. A smaller Thr value would result in a better chance for a node to be labeled, whereas a larger Thr value would allow the nodes to be less frequently labeled (and therefore

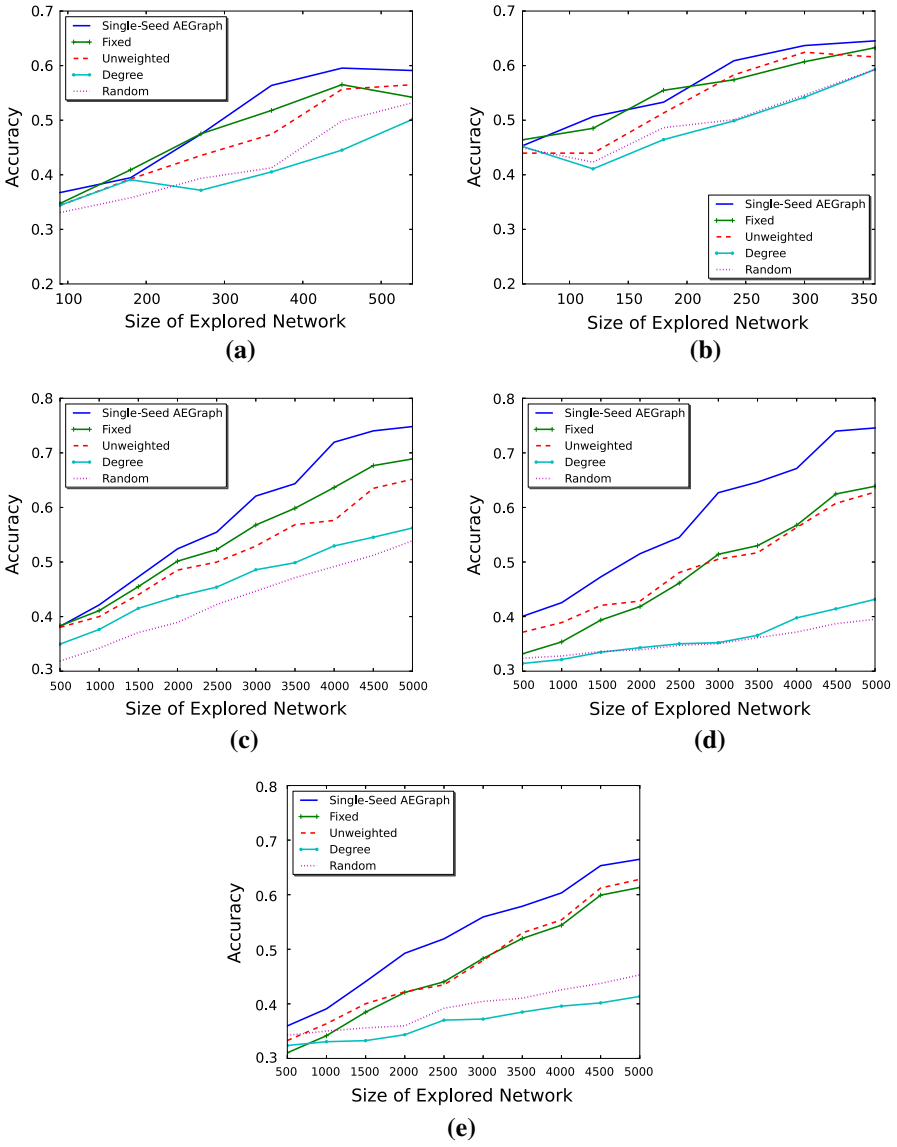

Fig. 9 Classification accuracies based on explored networks using different exploration strategies. **a** CiteSeer. **b** Cora. **c** PubMed: diabetes mellitus, experimental. **d** PubMed: diabetes mellitus type 1. **e** PubMed: diabetes mellitus type 2

reduce the labeling cost). Note that labeled nodes can provide valuable information to guide the active exploration process, so we empirically evaluate the impact of different Thr values on the algorithm performance in Figs. 10 and 11.

We use two networks as our case study to investigate the impact of Thr on our proposed Single-Seed AEGraph algorithm: The first one is P100-N300, with three threshold values 0.05, 0.1 and 0.2; and the second one is "PubMed: Diabetes Mellitus,

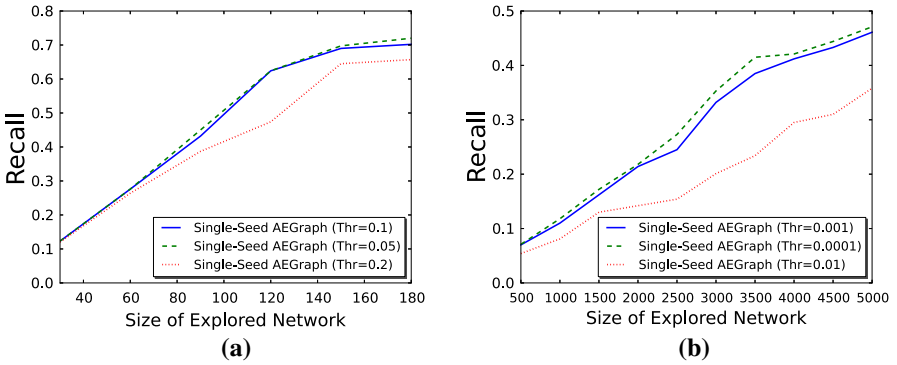

Fig. 10 Recall of positive nodes with respect to different sizes of explored networks. **a** P100-N300. **b** PubMed: diabetes mellitus, experimental

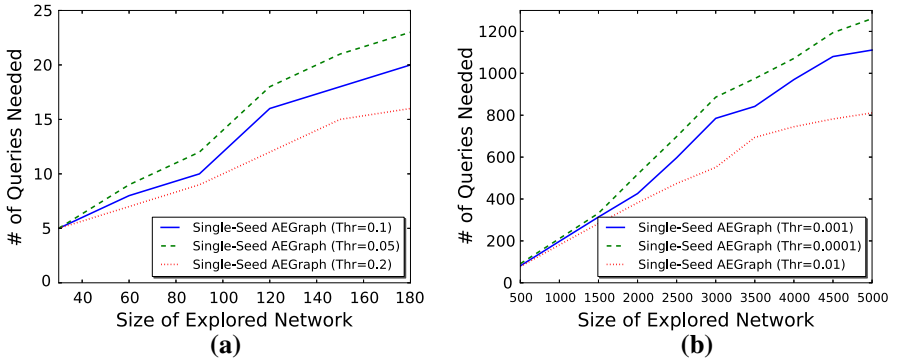

Fig. 11 Number of queries required in the y-axis in order to produce an explored network with the size indicated in the x-axis. The results are reported with respect to different Thr threshold values. **a** P100-N300. **b** PubMed: diabetes mellitus, experimental

Experimental", with three threshold values 0.0001, 0.001 and 0.01. The results in Fig. 10 show that the smaller the Thr value is, the higher recall the algorithm can achieve. This is because, a smaller Thr value would likely increases the chance of labeling more nodes at each iteration (In an extreme case, if $Thr = 0$, there is one node to be labeled at each iteration). Because including more labeled nodes can provide important knowledge to guide the future active exploration process, we can always observe better performance when lowering the Thr threshold value. A smaller Thr value, on the other hand, also implies a higher labeling cost (because more nodes need to be labeled in the explored network). Our results in Figs. 10 and 11 show that there is very little difference for small Thr values (such as 0.1 vs. 0.05 in Fig. 10). As a result, one can specify a relatively small Thr value to compromise the labeling cost and the overall active exploration performance.

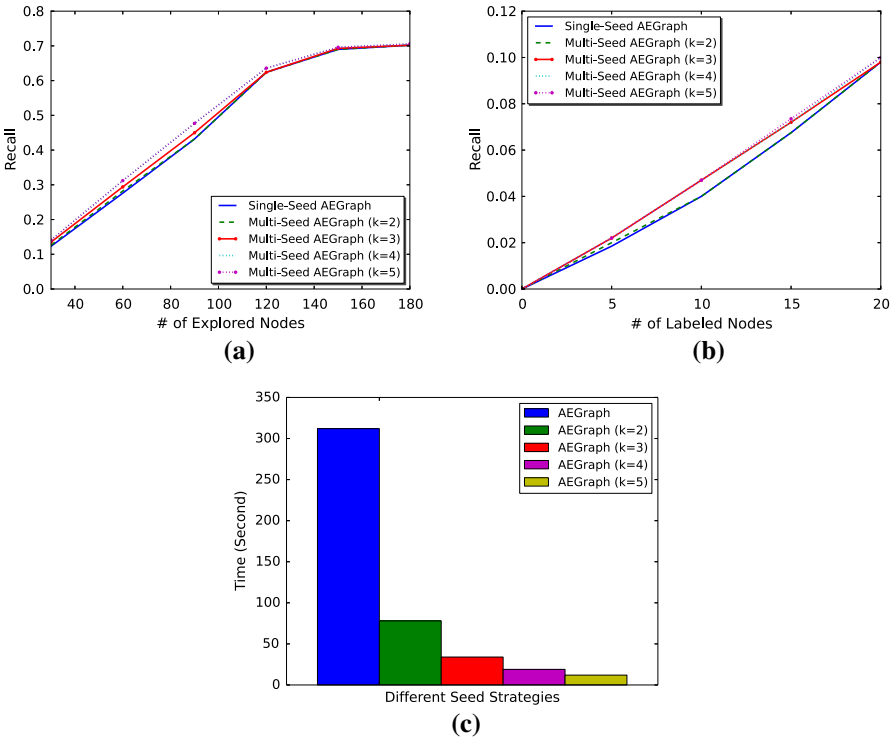

Fig. 12 Comparison of single-seed and multi-seed active exploration on the synthetic network P100-N300. **a** Sampled nodes. **b** Labeled nodes. **c** Processing time (Color figure online)

5.5 Comparison of single-seed versus multi-seed active exploration

We now perform experiments to compare the performance of single-seed active exploration and multi-seed active exploration. The aim of our experiments is to answer two questions: (1) To what extent the multi-seed algorithm can improve the efficiency of the single-seed algorithm? (2) Can the two algorithms achieve comparable active exploration performance? Considering the single-seed algorithm is a special case of the multi-seed algorithm with $k = 1$, we compare the performance of the two algorithms by increasing the number of exploration processes k. Again, we use two networks as our case study because similar observations can be made from other networks. The first one is the synthetic network P100-N300, with four k values: 2, 3, 4, and 5; and the other one is the PubMed network for exploring "Diabetes Mellitus, Experimental", with four k values 2, 5, 10 and 20.

Figures 12 and 13 report the comparison of the results on the synthetic network and the PubMed network, respectively. The results from Fig. 12 show that Single-Seed AEGraph and Multi-Seed AEGraph achieve comparable recall values with respect to different numbers of sampled and labeled nodes in the explored network. Similar observations can also be made on the PubMed network in Fig. 13. Interestingly, as the number of seeds increases, Multi-Seed AEGraph (i.e., $k = 20$) demonstrates

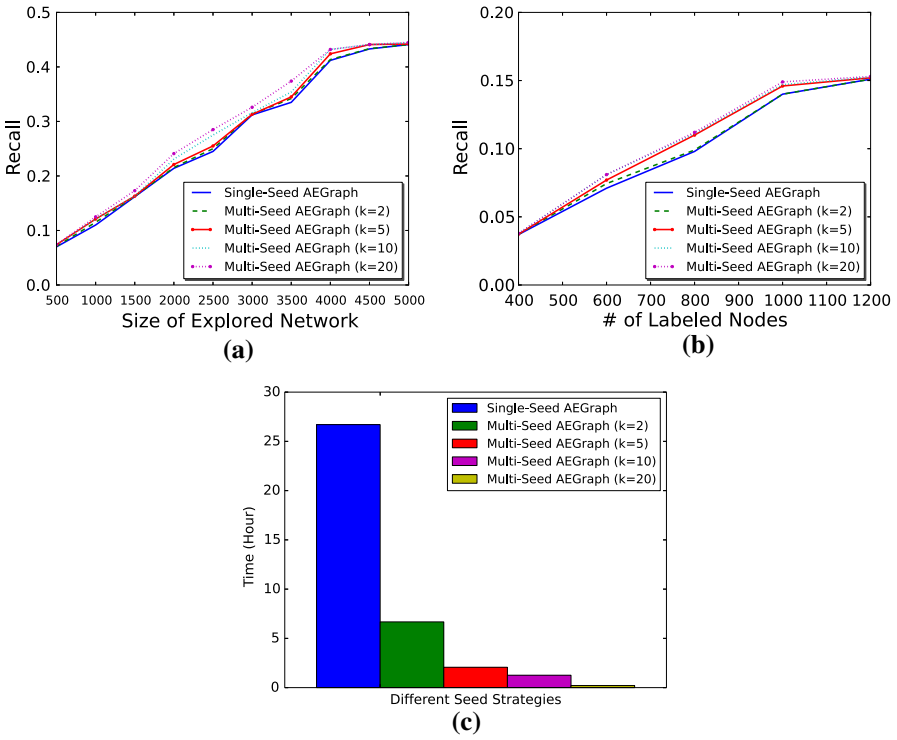

Fig. 13 Comparison of single-seed and multi-seed active exploration on the PubMed network for exploring "Diabetes Mellitus, Experimental". **a** Sampled nodes. **b** Labeled nodes. **c** Processing time (Color figure online)

increasing capabilities of yielding a higher recall of positive nodes in the explored network than Single-Seed AEGraph. This further asserts the effectiveness of the multi-seed algorithm by jointly exploring the nodes and querying the labels of nodes among all exploration processes.

On the other hand, Multi-Seed AEGraph can significantly improve the efficiency of Single-Seed AEGraph on both networks. In Fig. 12c, Multi-Seed AEGraph ($k = 5$) takes about 20 times less processing time than Single-Seed AEGraph. Similarly, in Fig. 13c, we can also observe that the processing time of Single-Seed AEGraph can be significantly reduced by using multiple, parallel exploration processes.

Based on our experiments, we can conclude that, Multi-Seed AEGraph can significantly enhance the efficiency and scalability of Singe-Seed AEGraph over large graphs, with comparable active exploration performance.

6 Conclusion

In this paper, we formulated a new active exploration framework which combines network sampling and active labeling to generate a small explored network from the original large graph. We argued that, existing network sampling approaches often

assume that the entire network is immediately available for sampling. However, the dynamic changes of the networks as well as the privacy and policy settings often forbid the entire network to be collected at once for sampling. Meanwhile, the separated sampling and labeling processes often produce inferior explored networks, especially when the networks contain very few nodes directly related to specific graph mining tasks. The proposed active exploration framework intends to combine network sampling and node labeling as a mutual, beneficial process to explore a network best suitable for the underlying mining tasks. To achieve this goal, we consider the network structure and node features to guide a supervised sampling and labeling process. A Markov chain with the stationary probability distribution is learned to assign proper probability scores to the nodes in the explored network. The sampling and labeling processes can utilize the probability scores of the nodes to explore future candidates for sampling and label important nodes for improving classification. To improve the scalability of active exploration over large graphs, we also proposed a multi-seed strategy which simultaneously runs multiple, parallel exploration processes, and makes local/global decisions about which node should be sampled and labeled next from multiple exploration processes. Experiments on both synthetic and real-world networks confirmed that our active exploration algorithms significantly outperform baseline approaches, especially for networks containing very few positive nodes. The main contributions of this paper, in comparison with existing work, are three-fold: (1) a supervised random walk with a clear optimization objective; (2) a seamless network sampling and labeling framework for maximum performance gain; and (3) a multi-seed algorithm for active exploration over large graphs.

References

Ahmed N, Neville J, Kompella R (2014) Network sampling: from static to streaming graphs. ACM Trans Knowl Discov Data 8(2):1–56

Ahn Y-Y, Han S, Kwak H, Moon S, Jeong H (2007) Analysis of topological characteristics of huge online social networking services. In: Proceedings of the 16th international conference on world wide web, pp 835–844. ACM, Banff, Alberta

Alamgir M, Von Luxburg U (2010) Multi-agent random walks for local clustering on graphs. In: Proceedings of the 10th IEEE international conference on data mining, pp 18–27. IEEE, Sydney

Alon N, Avin C, Koucky M, Kozma G, Lotker Z, Tuttle MR (2008) Many random walks are faster than one. In: Proceedings of the 20th annual symposium on parallelism in algorithms and architectures, pp. 119–128. ACM, Munich, Germany

Backstrom L, Leskovec J (2011) Supervised random walks: predicting and recommending links in social networks. In: Proceedings of the 4th ACM international conference on web search and data mining, pp 635–644. ACM, Hong Kong

Becchetti L, Castillo C, Donato D, Fazzone A (2006) A comparison of sampling techniques for web graph characterization. In: Proceedings of the 2006 workshop on link analysis: dynamics and static of large networks, Philadelphia, PA

Belkin M, Matveeva I, Niyogi P (2004) Regularization and semi-supervised learning on large graphs. Learning theory. Springer, Berlin, pp 624–638

Bilgic M, Getoor L (2008) Effective label acquisition for collective classification. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 43–51. ACM

Bilgic M, Mihalkova L, Getoor L (2010) Active learning for networked data. In: Proceedings of the 27th international conference on machine learning, pp 79–86. Haifa, Israel

Catanese S, Meo P, Ferrara E, Fiumara G, Provetti A (2011) Crawling facebook for social network analysis purposes. In: Proceedings of the 1st international conference on web intelligence, mining and semantics, Article No. 52. ACM, Sogndal

Cesa-Bianchi N, Gentile C, Vitale F, Zappella G (2010) Active learning on trees and graphs. In: Proceedings of the 23rd annual conference on learning theory, pp 320–332. Haifa, Israel

Erdős P, Rényi A (1959) On random graphs I. Publ Math Debrecen 6:290–297

Fang M, Tao D (2014) Networked bandits with disjoint linear payoffs. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 1106–1115. ACM

Fang M, Yin J, Zhu X (2013) Active exploration: Simultaneous sampling and labeling for large graphs. In: Proceedings of the 22nd ACM international conference on information and knowledge management, pp 829–834. ACM, Burlingame, CA

Gjoka M, Kurant M, Butts C, Markopoulou A (2010) Walking in facebook: a case study of unbiased sampling of osns. In: Proceedings of the 29th conference on computer communications, pp 1–9. San Diego, CA

Gkantsidis C, Mihail M, Saberi A (2004) Random walks in peer-to-peer networks. In: Proceedings of the 23rd annual joint conference of the IEEE computer and communications societie, pp 120–130. IEEE

Gyongyi Z, Garcia-Molina H, Pedersen J (2006) Web content categorization using link information. Technical report, Stanford

Halperin S, Zwick U (1994) An optimal randomized logarithmic time connectivity algorithm for the EREW PRAM. In: Proceedings of the 6th annual ACM symposium on parallel algorithms and architectures, pp 1–10. ACM, Cape May, NJ

He J, Carbonell JG, Liu Y (2007) Graph-based semi-supervised learning as a generative model. In: Proceedings of the 20th international joint conference on artificial intelligence, pp 2492–2497

Henzinger MR, Heydon A, Mitzenmacher M, Najork M (2000) On near-uniform URL sampling. Comput Netw 33(1):295–308

Hübler C, Kriegel H-P, Borgwardt K, Ghahramani Z (2008) Metropolis algorithms for representative subgraph sampling. In: Proceedings of the 8th IEEE international conference on data mining, pp 283–292. Pisa

Karger DR, Nisan N, Parnas M (1992) Fast connected components algorithms for the erew pram. In: Proceedings of the 4th annual ACM symposium on parallel algorithms and architectures, pp 373–381. ACM, San Diego, CA

Kuwadekar A, Neville J (2011) Relational active learning for joint collective classification models. In: Proceedings of the 28th international conference on machine learning, pp 385–392. Bellevue, WA

Lee S, Kim P, Jeong H (2006) Statistical properties of sampled networks. Phys Rev E 73(1):016102

Leskovec J, Faloutsos C (2006) Sampling from large graphs. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining, pp 631–636. Philadelphia, PA

Lin F, Cohen WW (2010) Semi-supervised classification of network data using very few labels. In: Proceedings of the 2010 international conference on advances in social networks analysis and mining, pp 192–199. IEEE, Adense

Lovász L (1993) Random walks on graphs: a survey. Combinatorics 2(1):1–46

Macskassy SA (2007) Improving learning in networked data by combining explicit and mined links. In: Proceedings of the 22nd conference on artificial intelligence, pp. 590–595. AAAI Press, Vancouver, BC

Maiya A, Berger-Wolf T (2010) Online sampling of high centrality individuals in social networks. In: Proceedings of the 14th Pacific-Asia conference on knowledge discovery and data mining, pp 91–98. Hyderabad, India

Mislove A, Koppula H, Gummadi K, Druschel P, Bhattacharjee B (2008) Growth of the flickr social network. In: Proceedings of the 1st workshop on online social networks, pp 25–30. ACM, Seattle, WA

Mislove A, Marcon M, Gummadi K, Druschel P, Bhattacharjee B (2007) Measurement and analysis of online social networks. In: Proceedings of the 7th ACM SIGCOMM conference on internet measurement, pp 29–42. ACM, San Diego, CA

Namata G, Sen P, Bilgic M, Getoor L, Sahami M, Srivastava A (2009) Collective classification for text classification. In: Text mining: classification, clustering, and applications. CRC Press, Boca Rotan, pp 51–69

Papagelis M, Das G, Koudas N (2013) Sampling online social networks. IEEE Trans Knowl Data Eng 25:662–676

Pfeiffer III J, Neville J, Bennett P (2012) Active sampling of networks. In:Proceedings of the ICML workshop on mining and learning with graphs, Edinburgh, Scotland

Rafiei D, Curial S (2005) Effectively visualizing large networks through sampling. In: Proceedings of IEEE visualization, pp 375–382. IEEE, Minneapolis, MN

Rasti A, Torkjazi M, Rejaie R, Duffield N, Willinger W, Stutzbach D (2009) Respondent-driven sampling for characterizing unstructured overlays. In: Proceedings of the 28th conference on computer communications, pp 2701–2705. IEEE, Rio de Janeiro

Sarma A, Collapudi S, Panigrahy R (2011) Estimating pagerank on graph streams. J ACM 58(3):13

Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93

Stanton I, Kliot G (2012) Streaming graph partitioning for large distributed graphs. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, Beijing

Stutzbach D, Rejaie R, Duffield N, Sen S, Willinger W (2009) On unbiased sampling for unstructured peer-to-peer networks. IEEE/ACM Trans Netw 17(2):377–390

Tong H, Faloutsos C, Pan J-Y (2006) Fast random walk with restart and its applications. In: Proceedings of the 6th international conference on data mining, pp 613–622. IEEE

Viswanath B, Mislove A, Cha M, Gummadi K (2009) On the evolution of user interaction in facebook. In: Proceedings of the 2nd ACM SIGCOMM workshop on online social networks, pp 37–42. ACM, Barcelona

Wasserman S, Faust K (1995) Social network analysis: methods and applications. Cambridge University Press, Cambridge

Wilson C, Boe B, Sala A, Puttaswamy K, Zhao B (2009) User interactions in social networks and their implications. In: Proceedings of the 4th ACM European conference on computer systems, pp 205–218. ACM, Nuremberg

Ye S, Lang J, Wu F (2010) Crawling online social graphs. In: Proceedings of the 12th International Asia-Pacific conference on web conference, pp 236–242. IEEE, Busan

Zhou D, Bousquet O, Lal TN, Weston J, Schölkopf B (2004) Learning with local and global consistency. Adv Neural Inf Process Syst 16(16):321–328

Zhou Z, Zhang N, Gong Z, Das G (2013) Faster random walks by rewiring online social networks on-the-fly. In: Proceedings of the 29th IEEE international conference on data engineering, pp 769–780. IEEE, Brisbane

Zhu X, Ghahramani Z (2002) Learning from labeled and unlabeled data with label propagation. Technical report, CMU-CALD-02-107, Carnegie Mellon University

Zhu X, Ghahramani Z, Lafferty J (2003) Semi-supervised learning using Gaussian fields and harmonic functions. In: Proceedings of the 20th International Conference on Machine Learning, vol 3, pp 912–919. Washington, DC

Zhu X, Lafferty J, Ghahramani Z (2003) Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In: Proceedings of the 2003 ICML workshop on the continuum from labeled to unlabeled data in machine learning and data mining, pp 58–65. Washington, DC