Semi-supervised classification and betweenness computation on large, sparse, directed graphs

Amin Mantrach a,*, Nicolas van Zeebroeck a, Pascal Francq b, Masashi Shimbo c, Hugues Bersini a, Marco Saerens b

a IRIDIA Laboratory, Université Libre de Bruxelles, 50 Av. Fr. Roosevelt, B-1050 Brussels, Belgium
b ÉSIS/ISM 6 Machine Learning Group, Université de Louvain, Place des Doyens 1, B-1348 Louvain-la-Neuve, Belgium
c Graduate School of Information Science, Nara Institute of Science and Technology, 8516-5 Takayama, Ikoma, Nara, Japan

ABSTRACT

This work addresses graph-based semi-supervised classification and betweenness computation in large, sparse, networks (several millions of nodes). The objective of semi-supervised classification is to assign a label to unlabeled nodes using the whole topology of the graph and the labeling at our disposal. Two approaches are developed to avoid explicit computation of pairwise proximity between the nodes of the graph, which would be impractical for graphs containing millions of nodes. The first approach directly computes, for each class, the sum of the similarities between the nodes to classify and the labeled nodes of the class, as suggested initially in [1,2]. Along this approach, two algorithms exploiting different state-of-the-art kernels on a graph are developed. The second approach works on a trellis structure built from biased random walks on the graph, extending an idea introduced in [3]. These random walks allow to define a biased bounded betweenness for the nodes of interest, defined separately for each class. All the proposed algorithms have a linear computing time in the number of edges while providing good results, and hence are applicable to large sparse networks. They are empirically validated on medium-size standard data sets and are shown to be competitive with state-of-the-art techniques. Finally, we processed a novel data set, which is made available for benchmarking, for multi-class classification in a large network: the U.S. patents citation network containing 3M nodes (of six different classes) and 38M edges. The three proposed algorithms achieve competitive results (around 85% classification rate) on this large network—they classify the unlabeled nodes within a few minutes on a standard workstation.

© 2010 Elsevier Ltd. All rights reserved.

1. Introduction

Within-network semi-supervised classification has received a growing focus in recent years (see [4,5] for a comprehensive survey of the topic). In such a setting, one tries to assign a label to the unlabeled nodes of a graph. Since the topology of the entire graph is used (including the unlabeled nodes), the problem is semi-supervised. Despite the growing need for dealing with huge real-world networks, few of the existing methods scale up to large graphs so that semi-supervised classification on large graphs has become one of the current central issues; see the survey [4, Section 6.3]. Indeed, the techniques that scale well [6] are not always competitive when compared to state-of-the-art graph-based metrics [7] such as the regularized Laplacian kernel [8], the sum-over-paths (SoP) covariance [9], the random walk with restart similarity and its normalized version [10,11,7], or the Markov diffusion kernel [12]. A naive application of these graph kernel-based approaches does not scale well since it relies on the computation of a dense similarity matrix, which usually requires a matrix inversion. Techniques approximating the inverse of the matrix usually require some strong properties on the matrix, like the positive semi-definiteness [13], and are only conceivable for medium-size graphs (up to 50,000 nodes)—for larger graphs, a dense similarity matrix cannot be computed and stored into memory. This paper tackles this problem with two different approaches. The first approach is based on existing, competitive, kernels on a graph, but it explicitly avoids the computation of the pairwise similarities between the nodes (following an idea suggested by Zhou et al. [1,2]). Indeed, as opposed to [11,14,9], Zhou et al. suggest...
to avoid computing each pairwise measure and solving a system of linear equations instead. We design two iterative algorithms along this approach, each based on a different state-of-the-art similarity metric: the SoP covariance kernel [9] and the normalized random walk with restart. This kernel on a graph was called regularized commute-time kernel in [7] and is closely related to the modified Laplacian matrix [15], as well as the random walk with restart similarity [10,11]. As suggested initially in [1,2], the algorithms directly approximate the sum of similarities to labeled nodes. The second approach takes its inspiration from the randomized shortest path framework of [16,9] and the D–walk algorithm based on bounded random walks [3]. In this case, a random walk betweenness [17], measuring how well a node is “in-between” each class, is derived from a trellis structure constructed from a biased random walk on the graph.

1.1. Contribution and organization of the paper

This work makes three main contributions:

- It provides three algorithms to address within-network semi-supervised classification tasks on large, sparse, directed graphs. All these algorithms have a computing time linear in the number of edges of the graph. Moreover, an algorithm allowing to compute the SoP betweenness centrality [9] is also proposed.
- It validates the three proposed algorithms on eight medium-size standard data sets and compares them to state-of-the-art techniques. Their performances are shown to be competitive in comparison with the other techniques. Results are also computed on a standard large-scale data set [18].
- It introduces a novel benchmark data set: The U.S. patents citation network, on which our three algorithms obtain competitive results.

The subsequent part, Section 2, introduces the necessary background and notations. Then, in Section 3, two iterative algorithms are derived from the assumptions of local and global consistency. Further, Section 4 defines a biased bounded betweenness and are derived from the assumptions of local and global consistency. The resulting d

k

0, representing the immediate cost of following this arc, is associated. The cost matrix C is the matrix containing the immediate costs c

kk

as elements.

A random walk on this graph is defined in the standard way. In node k, the random walker chooses the next arc to follow according to transition probabilities representing the probability of jumping from node k to node k ∈ S(k), the set of successor nodes (successors S). These transition probabilities will be denoted as p

kk

with k ∈ S(k). If there is no arc between k and k, we simply consider that c

kk

takes an arbitrary large value, denoted by ∞; in this case, the corresponding transition probability will be set to zero, p

kk

= 0. The natural random walk on this graph is defined in the following way: it corresponds to a random walk with transition probabilities

\[
p_{kk} = \frac{1}{c_{kk}} \sum_{k \in S(k)} \frac{1}{c_{kk}}
\]

(1)

The corresponding transition matrix will be denoted as P. In other words, the random walker chooses to follow an arc with a probability proportional to the inverse of the immediate cost (apart from the sum-to-one normalization), therefore locally favoring arcs having a low cost. Instead of C, we might be given an adjacency matrix A with elements a

kk

≥ 0 indicating the affinity between node k and node k. In this case, the corresponding costs could be computed from c

kk

= 1/a

kk

and the transition probabilities associated to each node are simply proportional to the affinities (and normalized).

3. Kernel-based semi-supervised classification on large sparse graphs

Three approaches for semi-supervised classification on large sparse graphs are investigated in this paper. The first two approaches (detailed in Sections 3.2 and 3.3) are based on approximating, or bounding, standard kernel-based techniques, and are developed in this section. They will therefore be referred to as approximate approaches. The third approach, discussed in detail in Section 4, is a generalization of the discriminative random walks classifier (D–walks, [3]).

3.1. Kernel-based classification

The approximate approaches are kernel-based and adopt the simple following classification procedure (the consistency method), initially proposed by Zhou et al. in [1,2] (see also [19–22]). Based on a regularization framework for the optimization a loss function, this classification procedure takes both available class labels and smoothness into account. The resulting decision procedure is based on a simple sum of similarities (each similarity being provided by an element of the graph kernel matrix) with the labeled nodes (as described in [1,2] for instance). This technique has been used with other kernels than those initially proposed by Zhou et al. with competitive results [9,7]. It corresponds to a simple alignment between the kernel matrix and the class membership vector.

More precisely, suppose that we are given a meaningful proximity matrix K (usually a graph kernel matrix; see e.g. [23]) providing similarities k

c

between each pair of nodes of the graph G. For each node, its total similarity with nodes belonging to class c is contained in the column vector x

c

= Ky

. Then, each node is assigned to the class showing the largest similarity; the predicted class index is thus provided by argmax

c

(x

c

) for all nodes. In this section, we propose to directly estimate this sum of similarities x

c

for two different metrics, i.e. the sum-over-paths (SoP) covariance [9] and the so-called regularized commute-time kernel [7], called in this paper the normalized random walk with restart.

The SoP covariance kernel is related to other, already available, kernels on a graph [23,12,7], such as, e.g., the commute-time kernel [24,25]. The commute-time kernel is the natural kernel associated to the commute-time distance (also called resistance distance [26]), the average number of steps that a random walker, starting from a given node, takes for entering another node for the first time and afterward going back to the starting node. As explained in [7,24], most of these kernels on a graph define a similarity measure.
taking into account all paths — both direct and indirect — between graph nodes. They have the nice property of increasing when the number of paths connecting two nodes increases and when the "length" of any path decreases. In short, the more short paths connect two nodes, the more similar those nodes are. On the contrary, the usual "shortest path" (also called "geodesic" or "Dijkstra distance") between nodes of a graph does not necessarily decrease when connections between nodes are added and thus it does not capture the fact that strongly connected nodes are at a smaller distance than weakly connected ones. These similarity measures are usually easy to compute and have an intuitive interpretation, although they do not scale well for large networks.

The regularized commute-time kernel is closely related to the modified Laplacian matrix [15] and the random walk with restart (RWR) similarity measure [10,11], that have been shown to be competitive on such tasks [9,7,2]. Because of the widespread use and the popularity of the RWR measure, the regularized commute-time kernel will be called the normalized random walk with restart kernel in this paper. Notice that such iterative updates for semi-supervised classification have been seen earlier in [1,27–29]; for a good review of these techniques see [18, Chapter 11]. In the remainder, we propose to apply such iterative procedure to two recently introduced graph kernels — that have been shown to perform well previously on medium size graphs [9,7] — not yet applied on large-scale networks. Finally, notice also that column $i$ of the kernel $K_e$ (the similarities from node $i$), can be approximated with the same method by using $e_i$, i.e. column $i$ of the identity matrix, instead of $y$.

### 3.2. The approximate sum-over-paths covariance

This first approach starts from the SoP covariance kernel. According to this measure [9], two nodes are considered as highly similar if they often co-occur together on the same – preferably short – paths. This leads to the definition of a covariance kernel capturing similarities between pairs of nodes. Let us first introduce the matrix $W$, corresponding to Eq. (23) of [9],

$$W = \exp(-\mathcal{C}) = \exp(-\mathcal{C} + \ln \mathbf{P})$$

(2)

where $\mathbf{P}$ is the transition matrix containing the $p_{kl}$, and the logarithm/exponential functions are taken elementwise. Moreover, $\mathcal{C}$ is the elementwise (Hadamard) matrix product. $W$ contains as elements $W_{kl} = \rho_{kl} \exp(-\mathcal{C}_{kl})$. If we set $Z = (I - W)^{-1}$, then the SoP covariance between node $k$ and node $l$ (see [9, Eq. (41)]) is

$$\text{cov}(k,l) = \frac{1}{Z} \left( \frac{Z_{kl} - 1}{Z_{kl}} \right) - \frac{Z_{kl} - 1}{Z_{kl}} \left( \frac{Z_{kl} - 1}{Z_{kl}} - 1 \right) \left( \frac{Z_{kl} - 1}{Z_{kl}} - 1 \right)$$

(3)

where $Z = \sum_{k=1}^{n} 1_{:k}$, $Z_{kl} = \sum_{k=1}^{n} 1_{:k} 1_{l:k}$, $Z_{kl} = \sum_{k=1}^{n} 1_{:k} 1_{l:k}$, $Z_{kl}$ is element $k$ of $Z$, $Z_{kl}$ is the Kronecker delta whose value is $1$ if $k = l$ and $0$ otherwise. On the other hand, the SoP betweenness centrality measure (see [9, Eq. (40)]), is

$$\text{bet}(k) = \frac{Z_{kl} - 1}{Z}$$

(4)

and corresponds to the expected number of times node $k$ appears on a path through the network.

As already mentioned, the goal here is to directly approximate $\text{s} = \mathbf{K} \cdot \mathbf{y}$ where $\mathbf{K}$ is the SoP covariance kernel matrix containing the elements $\text{cov}(k,l)$ (see Eq. (3)). For the sake of readability, let us fix a specific class $c$ in the remainder of this Section 3, and omit the superscript $c$ from $\mathbf{y}$. Now the sum of similarities between node $k$ and the labeled nodes (of class $c$) is

$$\sum_{l} \text{cov}(k,l) y_l = \frac{1}{Z} \left( Z_{kl} - 1 \right) y_l + Z_{kl} \sum_{l} Z_{kl} y_l - Z_{kl} \sum_{l} Z_{kl} y_l - Z_{kl} \sum_{l} Z_{kl} y_l - Z_{kl} \sum_{l} Z_{kl} y_l$$

(5)

If we denote element $k$ of a vector $x$ as $x_k$, and the diagonal matrix constructed from the vector $y$ as $\text{Diag}(y)$, and define

$$x_0 = Ze \quad \text{thus } Z_{kl} = [x_0]_k$$

(6)

$$x_1 = Z^1 e \quad \text{thus } z_{kl} = [x_1]_k$$

(7)

$$x_2 = Z^{2} \text{Diag}(y) x_1 \quad \text{thus } z_{kl} y_l = [x_2]_k$$

(8)

$$x_3 = Z^{3} \text{Diag}(y) x_0 \quad \text{thus } z_{kl} y_l = [x_3]_k$$

(9)

$$x_4 = Z^4 y \quad \text{thus } Z_{kl} y_l = [x_4]_k$$

(10)

$$x_5 = Z^5 x_0 \text{Diag}(y) x_0 \quad \text{thus } Z_{kl} y_l = [x_5]_k$$

(11)

$$x_6 = e^T \text{Diag}(y) x_0 \quad \text{thus } Z_{kl} y_l = [x_6]_k$$

(12)

then Eq. (5) can be rewritten as

$$\sum_{l} \text{cov}(k,l) y_l = \frac{1}{Z} \left[ [x_0]_k + [x_1]_k + [x_2]_k - [x_3]_k - [x_5]_k \right] - [x_4]_k [x_1]_k - [x_6]_k [x_1]_k [x_0]_k - [x_6]_k [x_0]_k - [x_6]_k [x_5]_k [x_0]_k$$

(13)

where the partition function $Z$ is computed by $z_{kl} - n = z_{kl} e - n e$ (a column vector full of $1$’s). The algorithm for computing this quantity consists in first solving the two systems of linear equations (6) and (7), which may be solved iteratively (Indeed, $\rho(W) < 1$, see [9]). For example, here is the way Eq. (6) is solved:

$$Ze = x_0 = \left( I - W \right)^{-1} e = x_0$$

$$== e = (I - W)x_0$$

$$== x_0 = Wx_0 + e$$

(14)

from which we obtain the iterative updating scheme

$$\begin{cases}
  x_0(t) = e \\
  x_0(t+1) = Wx_0(t) + e
\end{cases}$$

(15)

Of course, more sophisticated methods, like conjugate gradient techniques [30], could be used instead. Afterwards, using the same trick, we solve the systems of linear equations (8)–(10) which are again solved iteratively. Finally, Eqs. (11) and (12) are computed directly. The complexity of each iteration is $O(n)$ since the matrix is assumed to be sparse. So the global complexity is approximately $O(T|E|)|C|$ where $T$ is the number of iterations. Note that we might directly compute the results for all the $\mathbf{y}$ labeling vector classes by
using (instead of a column vector $y'$) the concatenation of these vectors in a matrix $X$. In this case, depending on the architecture, solving the equations may be transparently parallelized. Indeed, we can easily compute the covariance for each class independently.

Moreover, from Eq. (4), the SoP betweenness centrality of node $k$ can be computed from $\text{bet}(k) = (X_k \mathbf{1} - 1) \mathbf{1} / Z$.

### 3.3. The bounded normalized random walk with restart

In this second approach, we will approximate a second kernel-based method, referred to as the normalized random walk with restart (and called the regularized commute-time kernel in [7]), by bounding the underlying random walk. The notion of approximating stationary quantities for random walks by only considering paths up to a specified length $t$ has already been investigated in [3,31,27,32] (for more details, see the related work in Section 6). We propose to apply this idea on the normalized random walk with restart kernel, and provide an interesting interpretation of the bounding. This kernel on a graph is closely related to the modified Laplacian matrix [15], the commute-time kernel [24,25] and the well-known random walk with restart similarity [10,11].

The normalized random walk with restart matrix $K$ is given by

$$K = (D - \alpha P)^{-1}$$

(16)

where $A$ is the adjacency matrix, $D = \text{Diag}(A \mathbf{1})$, and $\mathbf{a}$ is a column vector full of 1's. If matrix $A$ is symmetric, Eq. (16) defines a valid kernel on a graph. As shown now, the parameter $\alpha \in [0,1]$ denotes, at each time step of a random walk, the probability that the random walker continues his walk. We are looking for a way to bound the sum-of-similarities $s = Ky$ up to a priori specified walk length $t$. Following [7], since the transition matrix of the natural random walk on the graph is $P = D^{-1}A$, we may note that

$$Ky = (D - \alpha P)^{-1} y = D^{-1} (D - \alpha P)^{-1} y = D^{-1} (1 - \alpha P) y = D^{-1} \left( \sum_{t=0}^{\infty} (\alpha P)^t \right) y = \bar{s}(\infty)$$

Thus, intuitively, random walkers start from the labeled nodes with an initial distribution $X(0) = y$. Then, they diffuse with transition matrix $\alpha P$, which is substochastic. Therefore, these random walkers have a non-zero probability of disappearing (giving up the walk) at each time step. Let us denote by $x(t)$ the column vector containing the expected number of random walkers in a specific node of the network after $t$ steps of the random walk; thus, $x(t) = \alpha P^t x(0)$ and $\mathbf{1}^\top x(0) = y$. Eq. (21) tells us that the sum of similarities $K y$ is simply the normalized (by $D^{-1}$) cumulated sum of expected visits to each node, $Ky = D^{-1} \sum_{t=0}^{\infty} \alpha^t \mathbf{1} x(t)$. Bounding the walks aims to truncate this series up to term $t$, $s(t) = D^{-1} \sum_{t=0}^{t} \alpha^t \mathbf{1} x(t)$. The normalizing factor $D^{-1}$ has the effect of weighting the intrinsic importance of popular nodes [33,34]. Since $P = D^{-1} A$, it can easily be shown that $s(t)$ may be computed using the following recurrence equation:

$$s(0) = y$$

$$s(t) = s(t-1) + \alpha^t \mathbf{1} x(t) \quad \text{for} \quad t = 1, \ldots, \tau$$

$$s(t) = D^{-1} s(t)$$

(22)

This recurrence scheme can be iterated until convergence, but, in our experiments, we stop the iteration at $t = \tau$ steps, which is equivalent to bounding the random walk up to $\tau$ steps. The parameter $\tau$ will be tuned by an internal cross-validation. Notice that the matrix $(I - \alpha P)^{-1}$ in Eq. (20) coincides with the well-known random-walk with restart similarity matrix; it was used, e.g., in [10] for computing similarities between nodes and was inspired from Page et al.'s famous PageRank algorithm [35].

### 4. The biased $D$–walks

Callut et al. [3] recently proposed still another random-walk based approach: the discriminative random walks ($D$–walks), providing a class betweenness measure for classifying nodes in a graph. It computes a group betweenness index [36] with respect to a set of nodes—in this case, the labeled nodes belonging to the same class. This model performed well on a number of semi-supervised tasks [3]. In this section, we propose an extension of the $D$–walks by using the randomized shortest path (RSP) framework introduced in [16,37,9]. By defining an entropy-related parameter $\theta$ that controls the global entropy spread by the random walker in the network, we may gradually bias the random walk towards short paths. For a parameter value of $\theta = 0$, our extension reduces to the $D$–walks. On the other hand, for intermediate values of $\theta$, the random walk is biased towards short paths, therefore avoiding, to a certain extend, loops or broad, irrelevant, walks. For additional motivations, the reader is advised to turn to [16].

A $D$–walk relative to class $c$ (see [3] for further details) is a random walk on a graph that starts (at $t = 0$) from a node belonging to class $c$ and ends in a node of the same class $c$. Therefore, the nodes of class $c$ are transformed into absorbing nodes when $t \geq 1$. More precisely, the approach considered here consists in applying the randomized shortest path framework on a lattice structure constructed from the original network whose weight matrix is referred to as $W$. Any vector or matrix with a tilde will be $n$-dimensional.

This matrix is directly computed from the exponential cost matrix $W$ associated to the network (Eq. (2)), as will be explained below. Inspired by hidden Markov models [38], the main idea is the following: the original network $G$ is unfolded in time in order to build a lattice $L$ made of the network nodes at time steps $0,1,\ldots,t$. Transitions are only allowed from nodes at time $t$ to successor nodes at time $t+1$ (see Fig. 1). The interpretation of this lattice is immediate: it represents a bounded random walk on the graph $G$ with $t = 0,1,\ldots$. The random walker starts in a node belonging to class $c$ at time $t = 0$ and walks until either he reaches a node in class $c$, and is absorbed by this node, or stops at time $t = \tau$, that is, $t = \tau$ is the maximum walk length. The lattice $L$ therefore contains $N \times n \times (c+1) + n$ nodes in total (Fig. 1).

This lattice will be considered as a new graph – which is acyclic this time – on which the randomized shortest-path framework can be applied [16]. Its $N \times N$ exponential costs matrix is denoted by $W^e$. 
and is organized in $(\tau + 1) \times (\tau + 1)$ blocks of size $n \times n$: 

$$W = \begin{pmatrix} 0 & \tilde{W}^{(1)} & 0 & \ldots & 0 \\ 0 & 0 & \tilde{W}^{(2)} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & \tilde{W}^{(\tau)} \end{pmatrix}$$

where $0$ is a matrix full of zeroes of the appropriate size, i.e., $n \times n$ in this case.

Most blocks are null matrices since the graph considered here is a lattice and only transitions between time steps $t$ and $t+1$ are allowed. Hence, only submatrices at block $(t, t+1)$, for $t = 0, 1, \ldots, \tau$ may hold non-zero elements. Moreover, the matrices $W^t$ are set equal to the $n \times n$ matrix $W$ computed from Eq. (2), with one important modification: when $t=1$, $W(t) = W$, but for $t > 1$, the rows of the matrix corresponding to nodes labeled as class $c$ are replaced by zero rows. This method aims at making these nodes absorbing: when a random walker hits one of these nodes, he stops his walk and disappears, according to the strategy of D-walks.

Two $N$-dimensional vectors, $h^c_0$ and $h^c_1$, are also defined, 

$$h^c_0 = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ h^c_0 \end{pmatrix}, \quad h^c_1 = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ h^c_1 \end{pmatrix}$$

These column vectors are indicator vectors defining the sets of starting and ending nodes of the random walks. The entries of the vector $h^c_0$ ($h^c_1$) are set to 1 if a path can start from (end into) the corresponding node, and 0 otherwise. They are also decomposed into $n \times 1$ blocks: $h$ is an $n \times 1$ vector whose values are equal to 1 if the node considered belongs to class $c$ and 0 otherwise. In other words, $h = y$. Since the paths (or walks) considered start at time step 0, the vector $h^c_0$ holds zero elements for all time steps $t > 0$. The vector $h^c_1$ contains the $n \times 1$ vector $h$ for every time step larger than 0 since the random walker may stop at each time step $t = 1, 2, \ldots, \tau$ (except $t=0$), when hitting a node of class $c$. For readability reasons, the superscript $c$ of $h^c_0$ and $h^c_1$ is omitted in the remaining of this section.

The betweenness we are looking for is computed for each class $c$ independently. Indeed, for each class, we consider all paths of length up to $\tau$ (in number of steps), starting from a node belonging to class $c$, and ending in a node of the same class. In other words, the betweenness measures to which extend a node is located in-between the nodes of the class of interest.

As detailed in [16,37,9] and already mentioned in Section 3.2, an important quantity appearing in the RSP framework is $Z^c = (I - W^c)^{-1}$, called the fundamental matrix. Every quantity of interest can be obtained from this matrix (see [16] for details). For instance, the expected number of transitions through link $k \rightarrow k'$ (see [9, Appendix A, Eq. (49)]), from which our betweenness will be derived, is

$$\pi_{lk} = \frac{1}{\theta} \hat{h}^l_z Z^c (W^c)^{0<} Z^c h^c$$

where $(W^c)^{0<}$ is the partial derivative of $W^c$ with respect to $c$, and $Z = h^c Z^c h^c$ if we define $(x)^c = h^c Z^c$ as the forward parameters vector by reference to hidden Markov models [38], then

$$Z^c (W^c)^{0<} = \hat{h}^c$$

Symmetrically, we define $b^c = Z^c h^c$ as the backward parameters vector,

$$b^c = Z^c h^c$$

Decomposing $x^c$ into blocks in Eq. (26) yields

$$\begin{pmatrix} \tilde{x}^c(0) \\ \vdots \\ \tilde{x}^c(\tau) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} (W^c)(1) \\ \vdots \\ (W^c)(\tau) \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} \tilde{x}^c(0) \\ \vdots \\ \tilde{x}^c(\tau) \end{pmatrix} \begin{pmatrix} \hat{h}^c \\vdots \\hat{h}^c \end{pmatrix}$$

which allows to obtain the following forward recurrence relations:

$$\begin{cases} \tilde{x}^c(0) = h^c \\ \tilde{x}^c(t+1) = (\tilde{W}^c(t+1)\tilde{x}^c(t) \end{cases}$$

Similarly, we do the same for $b^c$ to obtain the backward recurrence relations:

$$\begin{cases} b^c(t) = \hat{b}^c \\ b^c(t-1) = \tilde{W}^c(t) b^c(t) + \hat{b}^c, \quad t > 0. \end{cases}$$

Replacing $\hat{h}^c Z^c$ (Eq. (24)) and $Z^c h^c$ (Eq. (27)) in Eq. (23) yields

$$\eta_{lk} = \frac{1}{\theta} \frac{\partial Z^c}{\partial c} (W^c)^{0<} Z^c h^c$$

The partition function $Z^c$ corresponds to the contribution of all the paths starting and ending in class $c$. As already stated, $Z^c = h^c Z^c h^c$ and $b^c = Z^c h^c$; hence, $Z^c = \hat{h}^c b^c$. Since the $n \times 1$ column vector $h^c_0$ is formed by an $n \times 1$ block which is $h^c$ and $0$
values in the remaining positions, the product $h^j \beta^e$ equals $(h^j)^T \beta^e (0)$. Intuitively, it means that only paths starting at time step 0 contribute in the partition function. Therefore,

$$\pi_{ik}^j = \frac{1}{\theta} \left( \frac{1}{\theta} \right)^T \frac{1}{\theta} \beta^e = \frac{1}{\theta} \exp \left( -\theta \log \left( \frac{1}{\theta} \right)^T \beta^e \right)$$

(32)

Now, differentiating $W'$ with respect to $c_{ik}$ and assuming that node $k$ does not belong to class $c$, in which we obtain the trivial result $\pi_{ik}^j = 0$ since $k$ is absorbing yields

$$\left( W' \right)_{ik} = -\theta \exp \left( -\theta \log \left( \frac{1}{\theta} \right)^T \beta^e \right)$$

where $\beta^e$ is an $n \times 1$ column vector containing a 1 in position $k$, and 0's everywhere else. Therefore, the expected number of transitions through link $k \rightarrow k'$ is simply the sum of the expected number of transitions over all time steps,

$$\pi_{ik}^j = \sum_{t=0}^{T} \frac{1}{\theta} \exp \left( -\theta \log \left( \frac{1}{\theta} \right)^T \beta^e \right)$$

(33)

with $\beta^e = \exp ( -\theta \log \left( \frac{1}{\theta} \right)^T \beta^e )$. Consequently, the betweenness of node $k$ with respect to class $c$ is simply the sum of incoming transitions.

$$\beta_{ik}^c = -\sum_{t=0}^{T} \frac{1}{\theta} \exp \left( -\theta \log \left( \frac{1}{\theta} \right)^T \beta^e \right)$$

(34)

By observing from Eq. (29) that $\sum_{t=0}^{T} \frac{1}{\theta} \exp \left( -\theta \log \left( \frac{1}{\theta} \right)^T \beta^e \right)$, we finally obtain for the biased $D$–walk betweenness vector

$$\beta^c = -\sum_{t=1}^{T} \frac{1}{\theta} \exp \left( -\theta \log \left( \frac{1}{\theta} \right)^T \beta^e \right)$$

(35)

where $\beta^c$ is the elementwise (Hadamard) matrix product. This betweenness is very similar to the $\gamma^c$ variable computed in hidden Markov models [38,39] where it can be interpreted as the probability of being in some node after a walk of $t$ steps without having visited any node of class $c$. Once the class betweenness has been computed, each node is assigned to the class showing the largest betweenness. The algorithm is detailed in Algorithm 1. The time complexity is $O(\theta |E| |C|)$ since for each class $c \in |C|$ we have to compute the forward and backward vectors which require $t \times |V|$ steps. Each vector computation implies a product between a sparse matrix and a column vector which is $O(\theta |E| + t |V|)$. The space complexity is $O(\theta |E| + t |V|)$ since we have to store in memory both the matrix $W$ and the matrix $W'$ and at the same time we have to keep in memory $t$ times the forward and backward vectors.

Algorithm 1. Computation of the biased $D$–walk betweenness.

Input:

- A graph $G$ containing $n$ nodes.
- $\theta > 0$: the parameter controlling the degree of exploration.
- $C$: the $n \times n$ cost matrix associated to $G$, containing elements $c_{ij} > 0$.
- $P$: the $n \times n$ transitions–probabilities matrix of a natural random walk.
- $\gamma^c$: the $n \times 1$ class membership vector containing 1 for nodes belonging to the class $c$ and 0 otherwise.

Output:

- The betweenness matrix $B$ containing $|C|$ columns where each column contains the betweenness of each node relatively to a class.

1. $B = P \exp (-\theta C)$, where the exponential is taken elementwise and $\cdot$ is the elementwise (Hadamard) matrix product.
2. for $t=1$ to $|C|$ do
3. $W' = W$ and set the rows for which $\gamma^c = 1$ to $0^t$.
4. $E^t = y^c$ (initialization of the backward vector).
5. for $t=1$ to 1 do
6. $E^t = E^t + \gamma^c$ (initialization of the forward vector).
7. for $t=1$ to $|C|$ do
8. $E^t(t+1) = \left( W' \right)^T E^t(t)$
9. end for
10. $B(:,c) = \sum_{t=0}^{T} \gamma^c(t) E^t(t)$
11. end for
12. $B(:,c) = \exp \left( -\theta \log \left( \frac{1}{\theta} \right)^T \beta^c \right)$
13. end for
14. return $B$

5. Experiments

This experimental section has two main goals. Firstly, the three approximate algorithms introduced in this paper are compared to their exact counterpart (without approximating or bounding) and to some state-of-the art techniques on several graph-based semi-supervised classification tasks over medium-size data sets. Secondly, the performance of our three proposed algorithms are further assessed and compared to two state-of-the art techniques on (i) a large 6-classes sparse network, consisting of the graph of citations between about 3 million U.S. patents and (ii) a standard large-scale protein secondary structure data set.

5.1 First experiment: validation of the approximate approaches

In this first part of the experiments, we address the task of classifying unlabeled nodes in partially labeled graphs on eight medium-size data sets (up to 5000 nodes). The goal is to compare the approximate approaches to the exact kernel-based techniques in terms of classification rate. This comparison is performed on medium-size networks only since kernel approaches cannot be computed on large networks.

Data sets. The different classification models, referred to as classifiers, are compared on eight data sets that were used previously for semi-supervised classification: the four universities WebKB co-citation data sets [2,6], the two industry data sets [6], the IMDB products data set [6] and the CORA citation data set [6].

IMDb: The Collaborative Internet Movie Database (IMDB, [6]) has several applications such as making movie recommendations or movie category classification. The classification problem focuses on the prediction of the movie notoriety (whether the movie is a box-office or not). It contains a graph of movies linked together.

---

*The preprocessed version in Matlab/Octave format of the data sets used in this first experiment is available from http://iriia.ullab.ac.be/~amantrac/pub/bop_TPAMI.zip [9].*
whenever they share the same production company. The weight of an edge in the resulting graph is the number of production companies two movies have in common. The IMDb-proco graph contains 1169 movies and has the class distribution shown in Table 1.

**Industry:** Industry regroups two data sets [6]. The industry-pr data set is based on 35,318 Newswire press releases. The companies mentioned in each press release were extracted and an edge was placed between two companies if they appeared together in a press release. The same applies to the industry-yc data set based on 22,170 business news stories collected from the web. The weight of an edge is the number of such co-occurrences found in the complete corpus. To classify a company, Yahoo!’s 12 industry sectors have been used in the two industry data sets. The details about the two industry data sets are reported in Table 2.

**CoRA:** CoRA cite is a graph of 3583 nodes collected from machine learning research papers labeled into seven different topics [6]. Papers are linked if they share a common author, or if one cites the other. The composition of the CoRA cite data set is reported in Table 3.

---

### Table 1

Class distribution for the IMDb-proco data set.

<table>
<thead>
<tr>
<th>Category</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-revenue</td>
<td>572</td>
</tr>
<tr>
<td>Low-revenue</td>
<td>597</td>
</tr>
<tr>
<td>Total</td>
<td>1169</td>
</tr>
<tr>
<td>Majority class proportion (%)</td>
<td>50.67</td>
</tr>
</tbody>
</table>

### Table 2

Class distribution for the industry-yc and industry-pr data sets.

<table>
<thead>
<tr>
<th>Category</th>
<th>Size</th>
<th>industry-yc</th>
<th>industry-pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic materials</td>
<td>104</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>Capital Goods</td>
<td>83</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>Conglomerates</td>
<td>14</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>Consumer cyclical</td>
<td>99</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>Consumer noncyclical</td>
<td>60</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>Energy</td>
<td>71</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>Financial</td>
<td>170</td>
<td>268</td>
<td></td>
</tr>
<tr>
<td>Healthcare</td>
<td>180</td>
<td>279</td>
<td></td>
</tr>
<tr>
<td>Services</td>
<td>444</td>
<td>478</td>
<td></td>
</tr>
<tr>
<td>Technology</td>
<td>505</td>
<td>609</td>
<td></td>
</tr>
<tr>
<td>Transportation</td>
<td>38</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>Utilities</td>
<td>30</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1798</td>
<td>2189</td>
<td></td>
</tr>
<tr>
<td>Majority class proportion (%)</td>
<td>28.1</td>
<td>27.8</td>
<td></td>
</tr>
</tbody>
</table>

### Table 3

Class distribution for the CoRA cite data set.

<table>
<thead>
<tr>
<th>Category</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case-based</td>
<td>402</td>
</tr>
<tr>
<td>Genetic methods</td>
<td>551</td>
</tr>
<tr>
<td>Neural networks</td>
<td>1064</td>
</tr>
<tr>
<td>Probabilistic methods</td>
<td>529</td>
</tr>
<tr>
<td>Reinforcement learning</td>
<td>335</td>
</tr>
<tr>
<td>Rule learning</td>
<td>230</td>
</tr>
<tr>
<td>Theory</td>
<td>472</td>
</tr>
<tr>
<td>Total</td>
<td>3583</td>
</tr>
<tr>
<td>Majority class proportion (%)</td>
<td>28.70</td>
</tr>
</tbody>
</table>

### Table 4

Class distribution for the four WebKB cocite data sets.

<table>
<thead>
<tr>
<th>Category</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cornell</td>
<td></td>
</tr>
<tr>
<td>Texas</td>
<td></td>
</tr>
<tr>
<td>Washington</td>
<td></td>
</tr>
<tr>
<td>Wisconsin</td>
<td></td>
</tr>
<tr>
<td>Course</td>
<td>54</td>
</tr>
<tr>
<td>Department</td>
<td>25</td>
</tr>
<tr>
<td>Faculty</td>
<td>62</td>
</tr>
<tr>
<td>Project</td>
<td>54</td>
</tr>
<tr>
<td>Staff</td>
<td>6</td>
</tr>
<tr>
<td>Student</td>
<td>145</td>
</tr>
<tr>
<td>Total</td>
<td>346</td>
</tr>
<tr>
<td>Majority class proportion (%)</td>
<td>41.9</td>
</tr>
</tbody>
</table>
performed (10 runs), on which performances are averaged. For each unlabeled node, the various classifiers predict the most suitable category according to the procedures described previously. During each run, a 10-fold nested cross-validation is performed. The external folds are obtained by 10 successive rotations of the nodes and the performance of one run is averaged on these 10 folds. For each fold of the external cross-validation, a 5-fold internal cross-validation is performed on the remaining labeled nodes in order to tune the hyper-parameters of each classifier (i.e. the parameter \( \theta \) for SoP, aSoP and bDWALK, the parameters \( z \) for NRWR, bNRWR, NRL and aNRL, the parameter \( k \) of KNN). For each method and each labeling rate, we report the average classification rate averaged on the 10 runs.

**Results and discussion:** Comparative results for each method on the eight different data sets are reported in Figs. 2(a), (b), 3(a) and (b). Clearly, whatever the labeling rate considered, the approximate methods achieve almost the same performance as their exact kernel-based counterpart on all data sets (i.e. there is no significative difference in terms of accuracy according to a sign test for a labeling rate of 10% with a \( p \)-value of 0.01). By analyzing the statistical significance of the results (Table 5), we observe that the best methods overall are the NRWR (and its bounded counterpart), the biased \( D/C_0 \) walk, and the NRL (and its approximate counterpart) since they range among the top methods on all the benchmarked data sets. The SoP approach obtains good results in general, but it achieves sometimes least competitive results for a low labeling rate. This is the case on the two industries data set (Fig. 3(a)) and on the washington-cocite data set (Fig. 2(a)). Finally the hitting time approach results seems to be least competitive on these data sets except for the texas-cocite.

### 5.2. Second experiment: application to large-scale networks

In this second part of the experiments, we address the same task of classification of unlabeled nodes in partially labeled graphs, but this time on two large-scale data sets. The goal is to compare the results obtained by the three new algorithms introduced in this paper. The first data set considered is the US patents network which consists of more than 3 millions of nodes interconnected by 38 millions of links. The second data set consists of an amino acid sequence window associated to a target secondary structure [18]. This data set has already been used to investigate how far state-of-the-art semi-supervised methods can cope with large-scale application in [18].

**Data sets:** The patent data set introduced in this work is based on two publicly available databases: the NBER data set [40,41] and the PatStat database [42]. The resulting set is made of 3,416,966 U.S. patents granted between 1963 and 2002 and contains bibliographic data on each patent such as filing and grant dates, priority numbers (in case a U.S. patent was filed following preceding national or international applications), number of claims, and the list of countries of extension (countries other than the U.S. where the same patent was filed). In the economic literature, these
More importantly, patents are classified according to different U.S. and international classifications. The main U.S. class used in this work contains six broad industrial areas (chemicals, ICT, drugs and medical, electrical and electronic, mechanical, others), each of which contains up to nine subclasses. On the other hand, the international patent classification (IPC), maintained by the World Intellectual Property Organization (WIPO), provides a hierarchical representation of all technological fields. The first level contains eight classes (labeled from A to H: human necessities, transporting, chemistry and metallurgy, textiles and paper, fixed constructions, electrical and electronic, mechanical, others), each of which contains up to nine subclasses. On the other hand, the IPC class is made of four digits (from 01 to 20). At the third level, the IPC class is made of four digits (from A01A to H10G). Additional levels are available up to 11 digits. It is to be noted however that patents are frequently assigned to several classes. In this case, one class is referred to as the main category. Although it is difficult to link such technological classes to industrial sectors, the economic literature has used them intensively to analyze innovation activities across industries (see [43]).

The graph structure in the data is made of bibliographic references between patents. In order to obtain a patent, an inventor must provide the list of references to patents and scholarly publications upon which the invention is based. This list is then completed by an examiner at the patent office in an attempt to delineate as clearly as possible the territory covered by each patent. What had been published earlier can indeed no longer be patented. The network of citations between patents can be seen as indicative

### Table 5
Compilation of statistical sign tests computed for the aSoP, bNRWR, bNRWR, bNRWR, bNRWR, hit Time and KNN classification methods. The graphs show the results obtained on the two industries data sets and on the IMDb and Cora data sets [6].

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>aSoP</th>
<th>bNRWR</th>
<th>bNRWR</th>
<th>hit Time</th>
<th>aNRL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aSoP</td>
<td>$(7),&lt; (1)$</td>
<td>$(8)$</td>
<td>$(9)$</td>
<td>$(10)$</td>
<td>$(11)$</td>
</tr>
<tr>
<td>bNRWR</td>
<td>$(11)$</td>
<td>$(12)$</td>
<td>$(13)$</td>
<td>$(14)$</td>
<td>$(15)$</td>
</tr>
<tr>
<td>bNRWR</td>
<td>$(16)$</td>
<td>$(17)$</td>
<td>$(18)$</td>
<td>$(19)$</td>
<td>$(20)$</td>
</tr>
<tr>
<td>hit Time</td>
<td>$(21)$</td>
<td>$(22)$</td>
<td>$(23)$</td>
<td>$(24)$</td>
<td>$(25)$</td>
</tr>
</tbody>
</table>

---

Fig. 3. Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5%, 10%, 20%, 35%, 50%, 65%, 80% and 95%. Results are reported for the aSoP, bNRWR, bNRWR, bNRWR, bNRWR, hit Time and KNN classification methods. The graphs show the results obtained on the two industries data sets and on the IMDb and Cora data sets [6].

The results are reported for the aSoP, bNRWR, bNRWR, bNRWR, bNRWR, hit Time and KNN classification methods. A signed test has been performed on each of the eight medium-size data sets, based on the results of the 10 runs of the semi-supervised classification task, for a fixed labeling rate of 10%. Each entry of the table shows the number of times the row method is respectively significantly (i.e. p-value < 0.01) better (>), the same (=) or worse (<) than the column method. For instance, bNRWR performs significantly better than hit Time on seven data sets.
of knowledge flows between companies, countries or industries. They have therefore been intensively used as indicators of knowledge spillovers in the economic literature (see e.g. [44–48]). In addition, since patent citations indicate downstream research activities, hence investments around the same technology, they suggest that an intensively cited patent denotes a particularly important or valuable invention. Citation counts have therefore been used to produce value-weighted counts of patents (see [49,40,41]). One can naturally expect patents to preferably cite patents from the same technological class. An interesting topic, not covered in this paper, concerns clustering applied to patents that official patent numbers have been replaced by sequential numbers (from 1 to 3,245,005) to ease computations. The class distribution for the U.S. patents data set is shown in Table 6.

Furthermore, the SoP betweenness has been computed according to Eq. (4). The histogram of the base-10 logarithm of this betweenness is displayed in Fig. 4, for a θ parameter equal to 0.1 according to previous tests (see [9]). The betweenness scores are located in the interval [7.3172e–08, 2.7947e–04]. For information, the patents that obtain the highest betweenness scores are, respectively, the U.S. patent 4340581 on DSCG binding protein and process for preparing same of July 20, 1982; the U.S. patent 4683202 on Process for amplifying nucleic acid sequences of July 28, 1987 and the U.S. patent 4723129 on Bubble jet recording method and apparatus in which a heating element generates bubbles in a liquid flow path to project droplets of February 2, 1988.

The secondary structure (SecStr) is a large data set that has been benchmarked in [18]. It consists of 83,679 amino acids around which an amino acids window [−7, +7] is centered. The target is composed of two main classes: the α-helical and β-sheet secondary structure form one class of 47,856 protein positions while the remaining coil structural motif positions form the other class. Hence, the main task is to predict the secondary structure of a given amino acid in a protein based on a sequence window. The class distribution of this data set is shown in Table 7:

<table>
<thead>
<tr>
<th>Category</th>
<th>Size</th>
<th>Proportion (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>aH</td>
<td>35,823</td>
<td>42.81</td>
</tr>
<tr>
<td>cH</td>
<td>47,856</td>
<td>57.19</td>
</tr>
<tr>
<td>Total</td>
<td>83,679</td>
<td>100</td>
</tr>
<tr>
<td><em>Majority class proportion</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>57.19</td>
<td></td>
</tr>
</tbody>
</table>

The performance on each run is assessed on the remaining labeled nodes in order to tune the hyper-parameters of each classifier (see the first experiment in Section 5.1 for details). For each run, a 10-fold cross-validation is performed on the U.S. patents (Fig. 5(a)), we observe that the results are very stable across the 10 runs. The bounded normalized random walk with restart kernel-based method (bNRWR) achieves the best performance (significant, p < 0.01, at a labeling rate of 5% and 10%, according to a t-test performed on the 10 runs) for all labeling rates, very closely followed by the b7WALK (except for a very low labeling rate, where the b7WALK performance drops), and closely followed by the aSoP. It can be observed that these three proposed techniques are not very sensitive to the labeling rate. Indeed, the drop in performance from 95% to 5% labeling rate is lower than 6%.

<table>
<thead>
<tr>
<th>Category</th>
<th>Size</th>
<th>Proportion (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>α-helical</td>
<td>35,823</td>
<td>42.81</td>
</tr>
<tr>
<td>coil</td>
<td>47,856</td>
<td>57.19</td>
</tr>
<tr>
<td>Total</td>
<td>83,679</td>
<td>100</td>
</tr>
<tr>
<td><em>Majority class proportion</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>57.19</td>
<td></td>
</tr>
</tbody>
</table>

### Notes

1. This preprocessed data set in Matlab/Octave format is available from http://iridia.ulb.ac.be/~amantrac/pub/patents.mat.tar.gz.

---

**Table 6**

<table>
<thead>
<tr>
<th>Category</th>
<th>Size</th>
<th>Proportion (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chemicals</td>
<td>630,107</td>
<td>19.42</td>
</tr>
<tr>
<td>ICT</td>
<td>381,537</td>
<td>11.76</td>
</tr>
<tr>
<td>Drugs and medical</td>
<td>245,595</td>
<td>7.57</td>
</tr>
<tr>
<td>Electrical and electronic</td>
<td>575,369</td>
<td>17.73</td>
</tr>
<tr>
<td>Mechanical</td>
<td>724,022</td>
<td>22.31</td>
</tr>
<tr>
<td>Others</td>
<td>688,375</td>
<td>21.21</td>
</tr>
<tr>
<td>Total</td>
<td>3,245,005</td>
<td>100</td>
</tr>
</tbody>
</table>

**Table 7**

<table>
<thead>
<tr>
<th>Category</th>
<th>Size</th>
<th>Proportion (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>aH</td>
<td>35,823</td>
<td>42.81</td>
</tr>
<tr>
<td>cH</td>
<td>47,856</td>
<td>57.19</td>
</tr>
<tr>
<td>Total</td>
<td>83,679</td>
<td>100</td>
</tr>
<tr>
<td><em>Majority class proportion</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>57.19</td>
<td></td>
</tr>
</tbody>
</table>
whatever the method considered. The hitting time approach is also stable but it underperforms in comparison with the three top leading methods. Actually, we already observed on medium size data sets that the hitting time performance was somewhat below the top methods [see experiments on medium size data set Section 5.1]. In contrast, we observe a significant drop in performance for the aNRL and the KNN when the labeling rate decreases (for the sake of readability the results of the KNN and the aNRL are not shown for low labeling rates). Actually, both methods obtain a classification rate around 30% and around 8% for respectively a labeling rate of 5% and 1%.

On the SecStr data set (Fig. 5(a)), we observe a sudden large drop of the aNRL curve below 5% of labeling rate while the drop is more gentle on the U.S. patents network. The hitting time (hit Time) may be considered as the leading method on this data set, very closely followed by the bWRWALK and the bNRWR—all these three algorithms are rather robust with respect to changes in labeling rate (i.e. the drop in performance is lower than 10%). Actually, only the bNRWR and the bWRWALK achieved consistently very stable results on the U.S. patents graph as well as on the eight medium-size networks. Thus, according to our benchmarks, these two approaches seem to be quite robust. Finally, the aSoP results are a bit disappointing on this data set since its performance curve drops more sharply than the leading algorithms for low labeling rates (as before, for the sake of readability the results of the KNN and the aNRL are not reported for low labeling rates).

An analysis of the stability of the algorithms on all benchmarked data sets (medium-size and large-scale) is provided in Table 8. It shows, for all the implemented methods, the average accuracy drop when the labeling rate decreases from 95% to 5% on medium-size graphs, and from 95% to 1% on large-scale networks.

Table 9 provides a comparison of the running time of all methods, averaged on 10 runs for the U.S. patents network. We observe that the classification task only takes a few minutes, whatever the method used. The aSoP method is significantly slower, but is still able to classify the whole graph in a few minutes.

6. Related work

Graph-based semi-supervised classification has been the subject of intensive research in recent years. A wide range of approaches have been developed in order to address the problem. Among them, we may cite random walks [51,29,3], graph mincuts [52], spectral methods [53–56], regularization frameworks [19,21,22,1,2], transductive and spectral SVMs [57]. For a comprehensive survey of semi-supervised classification, including graph-based approaches, see [4,5]. Some of these approaches tackle the problem by random-walk-based methods. However, it has been shown that hitting times and commute times approaches suffer from several problems; for example, they take too long paths (hence irrelevant) into account so that popular entities are intrinsically favored [33,34]. On the other hand, random-walk-based techniques often require to inverse a matrix in order to compute measures on walks that are potentially of infinite length.

These shortcomings led some authors to consider bounded (or truncated) walks [58,31,3]. Sarkar et al. [58,31], for instance, used a truncated commute-time approach and showed experimentally that the truncation boosts the results while providing a competitive computing time in a proximity search task on a large graph with 600 K nodes. In the same spirit, Callut et al. suggested to bound

![Figure 5](https://example.com/image.png)

**Fig. 5.** Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 1%, 5%, 10%, 20%, 35%, 50%, 65%, 80% and 95%. Results are reported for the aSoP (approximate sum-over-paths), the bNRWR (bounded normalized random walk with restart), the bWRWALK (biased walk), the aNRWL (approximate normalized regularized Laplacian), the hit Time and the KNN methods. The graphs show the results obtained on (a) the U.S. patents data set and (b) the SecStr data set.

<table>
<thead>
<tr>
<th>Size</th>
<th>aSoP</th>
<th>aNRL</th>
<th>bNRWR</th>
<th>bWRWALK</th>
<th>hit Time</th>
<th>KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium-size</td>
<td>15.90</td>
<td>14.43</td>
<td>14.22</td>
<td>15.09</td>
<td>17.44</td>
<td>28.09</td>
</tr>
<tr>
<td>Large-scale</td>
<td>11.17</td>
<td>68.66</td>
<td>8.53</td>
<td>8.49</td>
<td>5.50</td>
<td>64.30</td>
</tr>
</tbody>
</table>

Table 8

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1%</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>35%</th>
<th>50%</th>
<th>65%</th>
<th>80%</th>
<th>95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>aSoP</td>
<td>768</td>
<td>749</td>
<td>972</td>
<td>883</td>
<td>658</td>
<td>291</td>
<td>313</td>
<td>351</td>
<td>337</td>
</tr>
<tr>
<td>aNRL</td>
<td>45</td>
<td>15</td>
<td>17</td>
<td>25</td>
<td>46</td>
<td>77</td>
<td>134</td>
<td>179</td>
<td>246</td>
</tr>
<tr>
<td>bNRWR</td>
<td>41</td>
<td>42</td>
<td>31</td>
<td>82</td>
<td>118</td>
<td>178</td>
<td>261</td>
<td>380</td>
<td>505</td>
</tr>
<tr>
<td>bWRWALK</td>
<td>55</td>
<td>58</td>
<td>63</td>
<td>79</td>
<td>120</td>
<td>184</td>
<td>271</td>
<td>379</td>
<td>511</td>
</tr>
</tbody>
</table>
walks for tackling graph-based semi-supervised problems [3]. Their approach offered a time complexity $O(|E|)$, but no experimental results on large graphs were presented. In this paper, we propose a generalization of the algorithm introduced by Callut et al. using the randomized shortest path framework [16,937]. In addition, we present experimental results on the large-scale U.S. patents data set and on a large-scale protein secondary structure data set.

Tong et al. suggested a method avoiding to inverse the complete matrix for computing the random walk with restart measure [50]. Their idea is to reduce the computing time by partitioning the input graph into smaller communities. Then, by applying a low rank approximation, they were able to approximate the random walk with restart. The approximated matrix obtained is sparse and hence can be kept into memory. Furthermore, they reduce the computing time by the number of communities initially found. Thanks to the Sherman–Morrison lemma, the precomputed inverse is updated on-the-fly for a new query. This method suffers from the fact that it adds an hyperparameter $k$, e.g. the number of communities, that depends on the network. Furthermore, the computing time is reduced by the factor $k$ which is still untractable for large graphs with millions of nodes.

Moreover, Tong et al. recently developed a direction-aware proximity method based on the concept of escape probability [14,11]. They presented two methods to compute efficiently this proximity measure. The first one, FastAllDAP, allows to compute directly the proposed measure between all pairs of nodes by reducing the cost of solving a large number of linear systems to one matrix inversion. This method can only be applied on medium-size graphs (i.e. < 10K nodes). The second method, FastOneDAP, may be applied to large graphs, but only computes one measure. In this case, the matrix inversion problem is approximated by a Taylor expansion of the concerned matrix and the complexity is reduced to $O(|E|)$. In this work, we avoid the direct computation of the proximity measure by taking advantage of the property of our targeted task, i.e. semi-supervised classification. Using the FastOneDAP directly for this task would require computing the measure $|V|$ times (i.e., for each node), so that its time complexity would be $O(|V|(|E|))$ (recall that the time complexity of our proposed method is $O(|E| + |E|)$, since the number $|E|$ of edges is usually much smaller than the number $|V|$ of nodes, we save an important amount of computing time.

Finally, Herbster et al. [60] proposed a technique for fast label prediction on a generic graph through the approximation of the graph with either a minimum spanning tree or a shortest path tree. Once the tree has been extracted, they are able to compute the pseudoinverse of the Laplacian matrix – a well-known kernel on a graph [25,24,61] – efficiently. The fast computation of the pseudoinverse enables to address prediction problems on large graphs.

7. Conclusion

This work investigated several approaches for tackling semi-supervised problems as well as betweenness computation on large, sparse, graphs. While this paper focuses on semi-supervised classification and betweenness computation, the same approaches (bounding or approximating random walks) could easily be applied in order to compute other graph measures, such as group degree centrality, closeness centrality, etc. [36]. This will undoubtedly be the subject of further work.

We will also, as future work, pursue the analysis of the U.S. patents data set introduced in this work. For instance, the correlation between various measures of importance of nodes (such as centrality/prestige) and econometric indicators of the value of a patent will be investigated. We will also study the impact of using additional patent information such as the abstract, various econometric indexes, etc., on the results of semi-supervised classification.

Another interesting issue in this respect is how to combine the information provided by the graph and the node features in a clever, preferably optimal, way.

Yet another interesting topic to be investigated is the clustering of large-scale graphs—we are currently working on extensions of graph kernel clustering applicable to large-scale graphs [62,63]. For a survey of community detection refer to [50].

Still another interesting application of the techniques presented in this paper is collaborative recommendation. Indeed, the same methods could be used for large-scale recommendation (like the Netflix challenge) using graph kernels in the same way as [25].

Acknowledgments

We thank the anonymous reviewers for their interesting remarks and suggestions that allowed to improve significantly the quality of the paper.

Part of this work has been funded by projects with the "Région wallonne" and the Belgian "Politique Scientifique Fédérale". We thank these institutions for giving us the opportunity to conduct both fundamental and applied research.

References
