Particle Competition and Cooperation in Networks for Semi-Supervised Learning

Fabricio Breve, *Member, IEEE,* Liang Zhao, *Member, IEEE,* Marcos Quiles, *Member, IEEE,* Witold Pedrycz, *Fellow, IEEE,* and Jiming Liu, *Senior Member, IEEE*

Abstract—Semi-supervised learning is one of the important topics in machine learning, concerning with pattern classification where only a small subset of data is labeled. In this paper, a new network-based (or graph-based) semi-supervised classification model is proposed. It employs a combined random-greedy walk of particles, with competition and cooperation mechanisms, to propagate class labels to the whole network. Due to the competition mechanism, the proposed model has a local label spreading fashion, i.e., each particle only visits a portion of nodes potentially belonging to it, while it is not allowed to visit those nodes definitely occupied by particles of other classes. In this way, a "divide-and-conquer" effect is naturally embedded in the model. As a result, the proposed model can achieve a good classification rate while exhibiting low computational complexity order in comparison to other network-based semi-supervised algorithms. Computer simulations carried out for synthetic and real-world data sets provide a numeric quantification of the performance of the method.

Index Terms—Semi-Supervised Learning, Particles Competition and Cooperation, Network-Based Methods, Label Propagation.

1 Introduction

Semi-Supervised learning has received increasing interest in the recent years. It is one of the classes of machine learning techniques lying between the two major categories: *supervised learning* and *unsupervised learning*. In supervised learning, an algorithm learns a function from the training data, which consists of pairs of data items and their respective labels, so after being learnt from a number of training examples, the algorithm can predict the labels of new datum. On the other hand, in unsupervised learning, all data items are unlabeled and the objective is to determine their intrinsic structure. One form of unsupervised learning is clustering where we partition a data set into subsets (clusters), so that data in the same cluster share some commonalities. Machine learning algorithms have been successfully applied to solve many practical problems such as data mining, pattern recognition, bioinformatics, time-series prediction and others [1], [2], [3], [4], [5], [6].

Nowadays, data sets under processing become larger and larger. In many situations only a small subset of data items can be effectively labeled. This is because the labeling process is often expensive, time consuming, and requires intensive human involvement. As a result, partially labeled data sets become more frequently encountered. Supervised learning techniques cannot handle this kind of problems because they assume all or almost all data items are labeled for the training process. On the other hand, unsupervised learning techniques ignore label information of the data items. Therefore, in such cases semi-supervised learning methods are of interest. They addresses the specificity of the problem by combining a few labeled data items with a large number of unlabeled data to produce better classifiers while requiring less human effort [7], [8]. In the following, we will give a review of semi-supervised methods, including the graph-based category, in which the proposed method is included.

- *F. Breve and L. Zhao are with the Department of Computation, Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, SP, Brazil, 13560-970.*
  E-mail: {fabricio,zhao}@icmc.usp.br
- *M. Quiles is with the Department of Science and Technology (DCT), Federal University of São Paulo (Unifesp), São José dos Campos, SP, Brazil.*
  E-mail: quiles@unifesp.br
- *W. Pedrycz is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, T6R 2V4, Canada and Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland*
  E-mail: pedrycz@ee.ualberta.ca
- *J. Liu is with the Computer Science Department, Hong Kong Baptist University, Kowloon, Hong Kong*
  E-mail: jiming@comp.hkbu.edu.hk

1.1 Related Work

Semi-supervised methods include generative models [9], [10], cluster-and-label techniques [11], [12], co-training and tri-training techniques [13], [14], [15], [16], low-density separation models, like Transductive Support Vector Machines (TSVM) [17], and graph-based methods, like Mincut [18], Local and Global Consistency [19], Local Learning Regularization [20], Local and Global Regularization [21], random walk techniques [22], [23], [24], and label propagation techniques [25], [26]. Recently, a link was established between the two major categories of semi-supervised learning approaches (graph-based methods and co-tri-training) [27].

Traditional semi-supervised techniques, such as Transductive Support Vector Machine (TSVM) [17], can identify data classes of well defined forms, but usually fail

to identify classes of irregular forms. Thus, assumptions on class distribution have to be made and unfortunately such class distribution is usually unknown a priori. For the last years, many graph based methods have been developed. Most of them share the regularization framework, differing only in the particular choice of the loss function and the regularizer [18], [19], [28], [29], [30], [31]. The main advantage of graph-based methods is the ability of identifying classes of different distributions. Recently, some methods inspired in physics concepts were presented. Wang et. al. proposed a semi-supervised learning method by considering each data point as an Ising spin and the labeling process of the data points is the determination of the directions of the corresponding spins [32]. Getz et. al. developed another method based on statistical physics [33]. In their work, they estimate the distribution of classifications and yields more accurate and robust results. Wang and Zhang proposed a semi-supervised learning approach based on electrostatic field model. In their model, they treat the labeled data points as point charges, the unlabeled data points are placed in the electrostatic fields generated by these charges and the labeling process is considered as the electric potentials of the electrostatic field at their corresponding places [34]. Although many graph-based semi-supervised learning methods have been developed, most of them have a high order of computational complexity ($O(n^3)$), making their applicability limited to small or middle size data sets [7]. As data sets get larger and larger, the development of efficient semi-supervised learning methods is still meaningful. An interesting graph-based work to treat large scale data set is presented in [35], in which a set of anchor points are firstly determined, a sparse adjacency matrix is designed and, then the learning is conducted by graph regularization. Such an adjacency matrix generating method is a general framework for graph construction, which can be used in various graph-based semi-supervised learning methods, including the method proposed in this paper, to improve their efficiency.

With the emergence of the complex network field, the study of large networks with non-trivial topological structures, such as computer networks, transportation networks, social networks, and biological networks [36], [37], triggers much interests by researchers. Many of these networks are found to be divided naturally into communities or modules, thus discovering of these communities structure became one of the main issues in complex network study (see [38], [39] and references there in). Among many community detection techniques, the particle competition approach proposed in [40] is strongly related to this work. In that model, particles walk in the network and compete with each other in such a way that each of them tries to possess as many nodes as possible. At the same time, each particle prevents other particles to invade its territory. Finally, each particle is confined inside a network community.

1.2 Contributions

In this paper, we present semi-supervised learning technique based on particle competition and cooperation in the network constructed from the input data set [40]. Among several other improvements, in the present model competitive and cooperative mechanisms are introduced and combined in a unique scheme. Particles of the same class proceed in the network in a cooperative manner to propagate their labels. Particles of different classes compete with each other to determine class borders.

The proposed semi-supervised learning method is fundamentally different from other graph-based techniques in the following way. Traditional graph-based semi-supervised learning techniques spread labels in a global fashion, i.e., at each time step the label information is propagated from all nodes to all other nodes accordingly to the edge weights. In contrast, the technique proposed in this paper is based on the particle competition-collaboration approach and it has a local spreading fashion in the network, i.e., at each time step, each particle spreads its label to a neighbor node chosen by the random-greedy rule. In other words, the label propagation approaches used by other graph-based methods manipulate all nodes of the network at each iteration, including those nodes which are already correctly labeled and do not need any modification. In the proposed method, due to the competition mechanism, each particle only visits a portion of nodes potentially belonging to the current particle or its teammates, while it is not allowed to visit those nodes definitely occupied by other teams of particles. It can be roughly understood that our method has a "divide-and-conquer" effect embedded in the competition-cooperation scheme. In this way, each particle walks only in a sub-network and many long-range redundant operations are avoided. As a result, the proposed method has a lower computational complexity order as will be shown later. Computer experiments offer a detailed numeric insight into the performance of the method.

As occurred in any semi-supervised learning algorithm, more labeled data items are provided, higher precision of data class detection can be achieved. However, there is a difference between the proposed method and many other semi-supervised learning methods. The proposed method without particle cooperation (with only particle competition mechanism) is already a network-based data clustering technique. It means that even without any labeled data, it can be used to detect data clusters (corresponding to data classes in semi-supervised learning) with a good precision. On the other hand, many other semi-supervised learning techniques are modeled in such a way that they explicitly depend on the labeled data. For example, in [19], the conformation to labeled data in the label propagation process is explicitly represented by one of the two terms of the energy function to be minimized. Thus, other techniques have

stronger dependence on the percentage of labeled data than the proposed method. In other words, our approach aggregates the advantage of underlying data clustering and label propagation methods. As a result, the proposed method has less degradation in term of class detection accuracy rate than other methods when less data items are labeled.

1.3 Organization

The remainder of this paper is organized as follows: Section 2 describes the proposed model in detail. In Section 3 we analyze the computational and storage complexity of the proposed model. Section 4 covers some results of computer experimentation. Finally, in Section 5 we draw some conclusions.

2 Model Description

In this section, we introduce a network based semi-supervised learning algorithm. Firstly, the input data set is transformed to an undirected unweighed network by using the rules presented in Subsection 2.1. Then, a set of particles, each of them representing a labeled data item, are put in the network. The subset of particles having the same label is called a *team*. Each node in the network possesses a vector of elements with each of them representing a domination level of each particle team. As the system runs, each particle chooses a neighbor node to visit by using the combined random-greedy rule (Subsection 2.3). At the target node (the node being visited), the domination level of the team of the current particle is increased, while the domination levels of other teams are decreased. Each team of particles tries to dominate as many nodes as possible in a cooperative way and at the same time prevent intrusion of particles of other teams. The node and particle updating rules are presented in Subsection 2.2. In Subsection 2.4 we discuss a stop criterion. At the end of iterative process, each unlabeled data item is labeled by the team with the highest domination level. Finally, a label propagation algorithm based on the particle cooperation-competition is presented in Subsection 2.5.

2.1 Initial Configuration

Given a data set $\chi = \{x_1, x_2, \ldots, x_l, x_{l+1}, \ldots, x_n\} \subset \mathbb{R}^m$ and the corresponding label set $L = \{1, 2, \ldots, c\}$, the first l points $x_i (i \leq l)$ are labeled as $y_i \in L$ and the remaining points $x_u (l < u \leq n)$ are left unlabeled, i.e, $y_u = \emptyset$. The goal is to assign a label to each of these unlabeled samples.

Firstly, we show the network formation for a given data set. Define a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. $\mathbf{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, where each one v_i corresponds to a sample x_i. $\mathbf{E}$ is the set of links (v_i, v_j), which can be represented by an adjacency matrix $\mathbf{W}$:

$$W_{ij} = \begin{cases} 1 & \text{if} \quad ||x_i - x_j||^2 \leq \sigma \quad \text{and} \quad i \neq j \\ 0 & \text{if} \quad ||x_i - x_j||^2 > \sigma \quad \text{or} \quad i = j \end{cases}, \quad (1)$$

where W_{ij} specifies whether there is an edge between the pair of nodes x_i and x_j. σ is a threshold which defines the maximum distance between x_i and x_j at which the nodes v_i and v_j can be connected, and $||.||$ is a distance function. In this paper, we have used the Euclidean distance in all computer simulations.

Another method for building the adjacency matrix is connecting each node v_i to its k-nearest neighbors, therefore:

$$W_{ij} = \begin{cases} 1 & \text{if } x_j \text{ is among the } k\text{-nearest} \\ & \text{neighbors of } x_i \text{ or vice-versa} \\ 0 & \text{otherwise} \end{cases}. \quad (2)$$

For each labeled data $x_i \in \{x_1, x_2, \ldots, x_l\}$ or its corresponding node in the network $v_i \in \{v_1, v_2, \ldots, v_l\}$, a particle $\rho_i \in \{\rho_1, \rho_2, \ldots, \rho_l\}$ is generated and its initial position is at v_i. Therefore, the number of particles is equal to the number of labeled sample in the data set. For simplicity, we call the node v_i is the *home node* of particle ρ_i if v_i is the initial position of particle ρ_i. Each particle changes its position at each iteration and the distance from its home node is registered. Each subset of particles generated from the samples with the same class labels form a *team*, collaborating with each other and competing with particles from other teams.

In this model, there are two types of dynamics: particle dynamics and node dynamics. Each particle ρ_j comes with two variables: $\rho_j^{\omega}(t)$ and $\rho_{\mathbf{j}}^{\mathbf{d}}(\mathbf{t})$. The first variable $\rho_j^{\omega}(t) \in [0, 1]$ is the particle strength characterizing how much the particle can dominate a node at time t. The second variable is a distance table, i.e., a vector $\rho_{\mathbf{j}}^{\mathbf{d}}(\mathbf{t}) = \{\rho_i^{d_1}(t), \rho_i^{d_2}(t), \ldots, \rho_i^{d_n}(t)\}$, where each element $\rho_j^{d_i}(t) \in [0, \quad n-1]$ corresponds to the distance measured between the particle's home node v_j and the node v_i (current position of particle ρ_j).

Each node v_i has one vector variable $\mathbf{v_i^{\omega}(t)} = \{v_i^{\omega_1}(t), v_i^{\omega_2}(t), \ldots, v_i^{\omega_c}(t)\}$, a vector of the same size of L, where each element $v_i^{\omega_\ell}(t) \in [0, 1]$ corresponds to the level of domination of team ℓ over node v_i. The sum of the domination levels of each node is always constant, because a particle increases the node domination level of its own team and, at the same time, decreases the other teams' domination levels. Thus, the following relationship holds:

$$\sum_{\ell=1}^{c} v_i^{\omega_\ell} = 1. \quad (3)$$

The initial level of domination vector $\mathbf{v_i^{\omega}}$ of each node v_i is set as follows:

$$v_i^{\omega_\ell}(0) = \begin{cases} 1 & \text{if } y_i = \ell \\ 0 & \text{if } y_i \neq \ell \text{ and } y_i \in L \\ \frac{1}{c} & \text{if } y_i = \emptyset \end{cases}, \quad (4)$$

The initial domination level of each node defined by Eq. (4) can be understood by the following way: 1) For each node corresponding to a labeled data, the domination level of the dominating team is set to the highest value

1, while the domination levels of other teams are set to the lowest value 0. 2) For each node corresponding to an unlabeled sample, the domination levels of all particle teams are set to the same value $\frac{1}{c}$, where c is the number of classes or number of teams of particles. An illustration of the initial domination level configuration is shown by Fig. 1a.

The initial position of each particle is set to its home node and the respective initial strength is set as follows:

$$\rho_j^{\omega}(0) = 1, \tag{5}$$

i.e., each particle starts walking with maximum strength.

Finally, the distance table of each particle is initially set as follows:

$$\rho_j^{d_i}(t) = \begin{cases} 0 & \text{if} \quad i = j \\ n-1 & \text{if} \quad i \neq j \end{cases}, \tag{6}$$

which means the particles only know the distances to the nodes they already visited or targeted; these distances are recalculated dynamically at each particle movement. Therefore, initially, all the distances are assumed to be the largest possible value $n-1$, except, of course, for the distance from its home node, which is zero at the beginning.

2.2 Node and Particle Dynamics

Regarding the node dynamics, at iteration t, each particle selects a target neighbor node to visit and each node holds a vector where each element represents the domination level of a particle team. Particles of different teams compete for owning the network nodes, thus a particle will increase the domination level of its team in the target node, at the same time it will decrease the domination level of the other teams in this same node. The exception are the labeled nodes, which domination levels are fixed. Therefore, for each selected target node v_i, the domination level $v_i^{\omega_\ell}(t)$ is updated as follows:

$$v_i^{\omega_\ell}(t+1) = \begin{cases} \max\{0, v_i^{\omega_\ell}(t) - \frac{\Delta_v \rho_j^{\omega}(t)}{c-1}\} \\ \quad \text{if } y_i = \emptyset \text{ and } \ell \neq \rho_j^f \\ v_i^{\omega_\ell}(t) + \sum_{q \neq \ell} v_i^{\omega_q}(t) - v_i^{\omega_q}(t+1) \\ \quad \text{if } y_i = \emptyset \text{ and } \ell = \rho_j^f \\ v_i^{\omega_\ell}(t) \quad \text{if } y_i \in L \end{cases}, \tag{7}$$

where $0 < \Delta_v \leq 1$ is a parameter to control changing rate of the domination levels and ρ_j^f represents the class label of particle ρ_j. If Δ_v takes a low value, the node domination levels change slowly, while if it takes a high value, the node domination levels change quickly. Each particle ρ_j increases the domination level of its team ($v_i^{\omega_\ell}$, $\ell = \rho_j^f$) at the node v_i which it is targeting, while it decreases the domination levels of this same node of other teams ($v_i^{\omega_\ell}$, $\ell \neq \rho_j^f$), always respecting the conservation law defined by Eq. (3). The domination level of all labeled node v_i^{ω} are always fixed, this situation is defined by the third case expressed by Eq. (7). Figure 1d illustrates what happens to the domination levels when a particle visits a node corresponding to an unlabeled sample.

Regarding the particle dynamics, a particle will get stronger when it is targeting a node being dominated by its own team and it will get weaker when it is targeting a node dominated by other teams. Thus, at each iteration t, a particle strength $\rho_j^{\omega}(t)$ is updated as follows:

$$\rho_j^{\omega}(t+1) = v_i^{\omega_\ell}(t+1), \tag{8}$$

where v_i is the target node, and $\ell = \rho_j^f$, i.e., ℓ is the class label of particle ρ_j. Therefore, each particle ρ_j has its strength ρ_j^{ω} set to the value of its team domination level $v_i^{\omega_j}$ of the node v_i. In this sense, a particle gets strong if it visits a node with high domination level of its own team. This situation is illustrated by Fig. 1b, where a particle of "red" team is targeting a node dominated by the same team. Consequently, the particle's strength is increased. On the other hand, a particle will be weakened if it tries to invade a node dominated by another team. This behavior is illustrated by Fig. 1c, where a particle of "red" team is targeting a node dominated by the "yellow" team. In this case, the particle's strength is reduced.

The distance table is introduced in order to keep the particle aware of how far it is from its home node, so it will not go too far away, which would let its neighborhood susceptible to be attacked by other particles. The distances together with the domination levels will prevent the particle from losing all its strength when walking into enemies' neighborhoods and at the same time it will keep them around to protect their own neighborhood. Each particle ρ_j updates its distance table $\rho_j^{d_k}(t)$ at each iteration t as follows:

$$\rho_j^{d_k}(t+1) = \begin{cases} \rho_j^{d_i}(t) + 1 & \text{if } \rho_j^{d_i}(t) + 1 < \rho_j^{d_k}(t) \\ \rho_j^{d_k}(t) & \text{otherwise} \end{cases}, \tag{9}$$

where $\rho_j^{d_i}(t)$ and $\rho_j^{d_k}(t)$ are the distances to its home node from the current node and the target node, respectively.

The distance calculation is simple: we assume that the particles initially have limited knowledge of the network, i.e., they know how many nodes in the network, but they do not know how the nodes are connected, so they assume all the nodes can be reached in at most $n-1$ steps (the largest possible distance). Every time a particle moves from a current node to a target node, it checks the current distance table. If the target node distance is higher than the current node distance, the target node distance is updated to the distance of the current node plus 1. This method has advantage to use already known distances without recalculation.

It should be noted that a particle really visits a target node only if the domination level of its team is higher than others; otherwise, a shock happens and the particle stays at the current node until next iteration. This is the competition mechanism implemented in the present model.

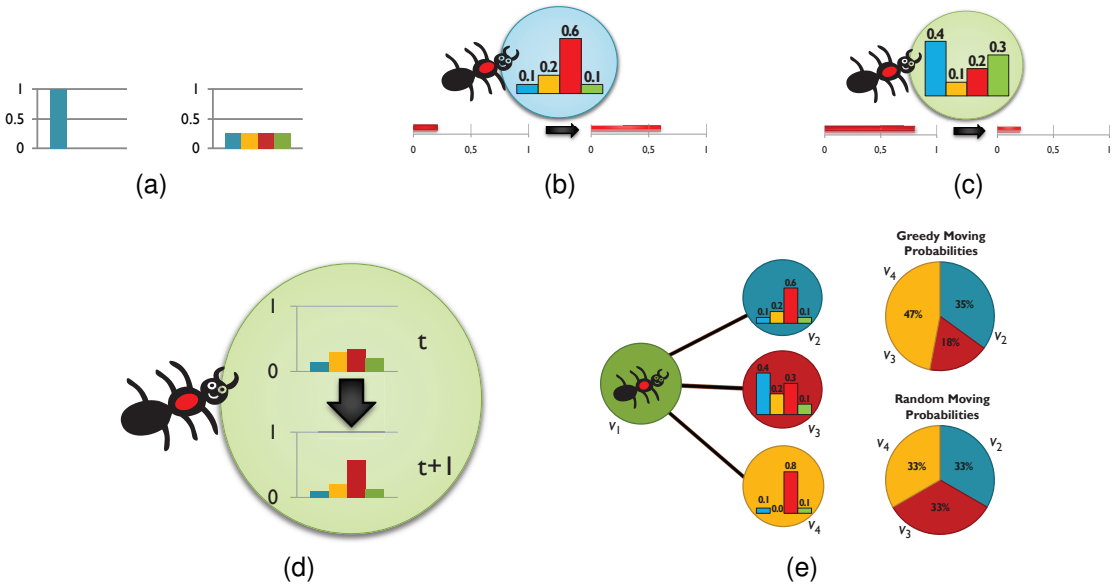


Fig. 1. Illustrations of node and particle dynamics. (a) initial domination levels for a node corresponding to a labeled sample (left) and an unlabeled node (right) in a problem of 4 classes; (b) a particle gets stronger as it is targeting a node being dominated by its own team; (c) a particle gets weaker as it is targeting a node being dominated by another team; (d) a particle increases its team domination level in the target node while decreasing domination level of other teams; (e) node probabilities of being chosen by a particle with greedy and random movement, all candidate nodes have the same distance from the particle home node.

2.3 Random-Greedy Walk

How does a particle choose a neighbor node to visit? We introduce two rules governing its behavior: *random walk* and *greedy walk*. In *random walk*, the particle randomly chooses any neighbor to visit without concerning domination levels or distance from its home node. This movement is useful for exploration and acquisition of new nodes. Meanwhile, in *greedy walk*, each particle prefers visiting those nodes that have been already dominated by its own team and that are closer to their home nodes. This movement is useful for defense of its team's territory. The particles should exhibit both movements in order to achieve an equilibrium between exploratory and defensive behavior.

Therefore, in *random walk* the particle ρ_j tries to move to any node v_i with the probabilities defined as:

$$p(v_i|\rho_j) = \frac{W_{qi}}{\sum_{\mu=1}^{n} W_{q\mu}}, \tag{10}$$

where q is the index of the current node of particle ρ_j, so $W_{qi} = 1$ if there is an edge between the current node and any node v_i, and $W_{qi} = 0$ otherwise. In *greedy movement* the particle tries to move to a neighbor with probabilities defined according to its team domination level on that neighbor $\rho_j^{\omega_\ell}$ and inverse of the distance $(\rho_j^{d_i})$ from that neighbor v_i to its home node v_j by the following expression,

$$p(v_i|\rho_j) = \frac{W_{qi} v_i^{\omega_\ell} \frac{1}{(1+\rho_j^{d_i})^2}}{\sum_{\mu=1}^{n} W_{q\mu} v_i^{\omega_\ell} \frac{1}{(1+\rho_j^{d_i})^2}}. \tag{11}$$

Again, q is the index of the current node of particle ρ_j and $\ell = \rho_j^f$, where ρ_j^f is the class label of particle ρ_j. Figure 1e illustrates some probabilities calculated using these rules.

At each iteration, each particle has probability p_{grd} to take greedy movement and probability $1 - p_{\text{grd}}$ to take random movement, with $0 \leq p_{\text{grd}} \leq 1$. Once the random movement or greedy movement is determined, the target neighbor node $\rho_j^{\tau}(t)$ will be chosen with probabilities defined by Eq. (10) or Eq. (11), respectively.

2.4 Stop Criterion

In most cases, after a sufficient number of steps, the inner nodes of each class will be completely dominated by a single team. However, outer nodes will still change their domination level, even change their domination team. Therefore, we cannot expect a convergence of all nodes labels in every cases, so we monitor the average maximum domination levels of the nodes ($\langle v_i^{\omega_\ell} \rangle$, $\ell = \arg\max_q v_i^{\omega_q}$) and stop the algorithm when there is no increasing of this quantity. Really, the number of iterations required for stabilizing of average maximum domination levels of all nodes is roughly proportional to the network size (this will be shown in Section 4), therefore one can just run the algorithm for a pre-defined number of steps proportional to the network size. But notice that network mixture, which is usually unknown a priori in real-world problems, also affects the amount of steps. Thus, monitoring the average maximum domination levels of nodes is a safer and more efficient stop

criterion.

2.5 The Algorithm

Overall, the proposed algorithm can be outlined as follows:

Algorithm 1: Particle Competition and Cooperation

1 Build the adjacent matrix W by using Eq. (1) or (2);
2 Set nodes' domination levels by using Eq. (4);
3 Set initial positions of particles at their corresponding home nodes by using Eq. (5);
4 Set particle strength and distance tables by using Eq. (6);
5 **repeat**
6 **for** *each particle* **do**
7 Select between random or greedy rule with probability defined by p_{grd};
8 Select the target node by using the combined random-greedy rule described in Subsection 2.3;
9 Update target node domination levels by using Eq. (7);
10 Update particle strength by using Eq. (8);
11 Update particle distance tables by using Eq. (9);
12 **until** *the stopping criterion is satisfied*;
13 Label each unlabeled data item by the team of maximum level of domination: $y_i = \arg\max_\ell v_i^{\omega_\ell}(\infty)$;

The particles from the same team collaborate with each other in the sense that they work together to raise their domination level on a node in order to raise their team flag in that node. But they still put the defense of its neighborhood in the first place, i.e., the nodes around their home node. However, once a while, they may visit their team-mates and help them defending their territory, and they will also eventually receive help from its team-mates in their own area.

Turning each labeled data point into a particle is also convenient, as the distribution of labeled nodes among classes is usually proportional to the distribution of unlabeled nodes among classes, as the labeled nodes are usually nodes that were randomly chosen from the whole data set to be labeled by a specialist.

3 Computational Time and Storage Complexity

In this section, we provide analysis on the time and storage complexity order of the algorithm presented in Subsection 2.5.

3.1 Computational Time Complexity

At the beginning of the algorithm, Steps 1 to 4 are executed and their execution time is dominated by Step 1: building the adjacent matrix from the input data set. This step has complexity order $O(n^2)$. Step 5 defines a loop that is repeated until convergence, i.e., the number of iterations of the algorithm. The order of this quantity will be determined later. Step 6 is an inner loop and all the remaining steps (Steps 7 to 12) are inside this inner loop. In other words, Steps 7 to 12 are actions to be done by each of the l particles at each iteration. Specifically, in Step 7 the algorithm chooses between random or greedy rule for selecting a neighbor for the particle to visit, which executes in constant time ($O(1)$). In Step 8, the algorithm has to calculate probability of being selected of each neighbor, this operation depends on the number of neighbors of each node, i.e. its order of complexity is $O(\langle k\rangle)$, where $\langle k\rangle$ is the average node degree of the network. Steps 9 to 11 perform updates to node domination levels, particle strength and particle distance table respectively, these are only scalar operations for each particle and each chosen node, thus they take constant time ($O(1)$). At this point, we conclude that each inner loop iteration has order of complexity $O(\langle k\rangle)$. Notice that $\langle k\rangle$ is defined by the choice of σ or k in Eq. (1) or Eq. (2), respectively, and it can be controlled to be a small number ($\langle k\rangle \ll n$). Usually, the algorithm has good classification performance when the network is sparse. Therefore, the inner loop usually runs in constant time $O(1)$. However, if the network is not sparse, i.e., $\langle k\rangle$ is proportional to n, the inner loop runs at the order of $O(n)$.

The inner loop (Step 6) should be repeated for each of the l particles. The number of particles is equal to the number of labeled data points, which is usually a small one in semi-supervised learning problems ($l \ll n$), i.e., we consider that l is a constant. Therefore, the inner loop for all particles runs in constant time $O(1)$. Finally, the outer loop (Step 5) is repeated until the average maximum domination levels of nodes ($\langle v_i^{\omega_\ell}\rangle$, $\ell = \arg\max_q v_i^{\omega_q}$) converge. Consider a network with some completely separated classes and each class (a subgraph) has one or more particles. The ownership of each node can be determined by only one visit, thus the number of iterations of the outer loop (Step 5) is certainly $O(n) = Cn$, where C is a positive constant proportional to the random level of particle walking. If the classes are connected but well defined, the ownership of each node can be determined by a small number of visits. Then, in order to have all n nodes dominated by particles, the number of iterations is again $O(n) = Cn$. For the same reason, we expect the number of iterations of the outer loop (Step 5) to be $O(n) = Cn$, i.e., each of the n nodes are dominated by a particle through constant times of visits.

In summary, if average degree $\langle k\rangle$ of the network is constant, Steps 5 to 12 run at linear time $O(n)$; if the average degree is proportional to n, Steps 5 to 12 run at $O(n^2)$. In both cases, the time complexity of the whole algorithm is determined by Step 1: building the adjacent matrix from the input data set, which is at $O(n^2)$.

Now we provide some experimental results to check our analysis. In the following simulations, we perform label propagation by using the proposed algorithm directly on networks, i.e., Step 1 is not executed. Without considering Step 1, we expect our algorithm to run at linear time order. We first generate a set of ramdom

clustered networks by using the method proposed by [38] with increasing sizes ($n = \{500, 1000, \ldots, 5000\}$). In this network generation method, pairs of nodes belonging to the same class are linked with probability p_{in}. In each of these networks, the average node degree $\langle k \rangle$ keeps constant. The $z_{out}/\langle k \rangle$ defines the mixture of the classes, i.e., as $z_{out}/\langle k \rangle$ increases, the classes become more and more mixed and harder to be identified. In all experiments, the nodes are equally divided into $c = 4$ classes, average node degree is set to $\langle k \rangle = 25$, and the labeled nodes subset size is set to $l = 50$, with their components randomly chosen. The proposed algorithm is executed on these networks measuring the number of iterations and time needed to achieve the convergence of average maximum domination levels of nodes ($\langle v_i^{\omega^\ell} \rangle$, $\ell = \arg\max_q v_i^{\omega_q}$), this stop criterion has been explained in Section 2.4. Time is measured in a regular desktop computer with an Intel Core 2 Quad Processor model Q9450 and 4GB of RAM. The algorithm is implemented by using MATLAB. The parameters are set to $p_{\text{grd}} = 0.70$ and $\Delta_v = 0.35$, which usually present good results, as it will be shown in Subsection 4.1. The results are shown in Figs. 2a and 2b for networks with medium ($z_{out}/\langle k \rangle = 0.2$) and high ($z_{out}/\langle k \rangle = 0.4$) network mixtures, respectively. By analyzing these figures, we notice that the amount of iterations and the time increases linearly as the network size increases, which confirm our analysis.

Our analysis shows that the order of time complexity of the proposed algorithm is at most $O(n^2)$. This feature is quite attractive because most graph-based semi-supervised methods have cubic complexity order, that is $O(n^3)$ [7]. This is because that graph-based methods usually have a global label propagation mechanism, i.e., at each iteration all nodes of the graph are updated by predefined rules, leading to matrix operations with cubic computational complexity. The proposed method is unique in the sense that it propagates the labels locally due to particle competition, avoiding many unnecessary walking or visits.

In order to further verify the above complexity analysis, we compare the execution time between the proposed method and other three representative graph-based methods. In these simulations, we use a sequence of artificial data sets with increasing sizes and the elements of each data set are equally divided into 4 normally distributed classes (Gaussian distribution). These data sets are generated by using function `gauss` from PRTools [41]. The Gaussian kernels are positioned in coordinates $(-2, -2)$, $(-2, +2)$, $(+2, +2)$, and $(+2, +2)$. Other parameters are set to the default values of the tool. For each generated data set, four graph-based semi-supervised methods are applied: Local and Global Consistency (LGC) [19], Label Propagation (LP) [25], Linear Neighborhood Propagation (LNP) [26], and the proposed method. For LGC and LNP, we have fixed $\alpha = 0.99$, as done in [19] and [26], respectively. The other parameters are set empirically, for LGC and LP, $\sigma = 3$; and for LNP and the proposed method $k = 25$. For the proposed method, graphs are built by using Eq. (2). The execution time results are shown in Fig. 3, and each point is an average of at least 20 realizations. All of the four algorithms are run by using a desktop computer with an Intel Core 2 Quad Processor model Q9450 and 4GB of RAM. These results show that the proposed method runs much faster than the others.

3.2 Storage Complexity

Regarding the memory requirements and storage complexity, the proposed algorithm uses the following data structures: the adjacency matrix used to represent the graph, the distance tables, and the nodes domination levels. The adjacency matrix has n^2 size, however the proposed method has the advantage of working with sparse non-weighted networks, which can be implemented by using linked lists of binary variables to save memory space. The average node degree $\langle k \rangle$ is defined by the choice of σ or k in Eq. (1) or Eq. (2), respectively, and it is usually a small number ($\langle k \rangle \ll n$). Therefore, we can consider that adjacency matrix implemented through linked lists grows only linearly as a function of the network size, i.e., O(n). The particles distances tables can be implemented using a single $n.l$ matrix of unsigned integers, however it is unlikely that a particle will visit all the nodes in a network. Usually, each particle will visit only a small subset of the nodes during the whole execution. Therefore we do not need to store the distances for nodes that were never visited. This leads to a sparse distance matrix, which can also be implemented using linked lists to save memory space. The amount of labeled samples l, and therefore the number of particles, is also generally much smaller than the network size ($l \ll n$), therefore it is $O(n)$ as well. Finally, the nodes domination levels can be implemented using a single matrix with size $n.c$, in which the number of classes is much smaller than the network size ($c \ll n$), i.e., $O(n)$. In addition, three other structures are stored using single vectors: node labels with n size and particle strength and position, both with l size. Most graph-based algorithms use non-sparse weighted networks and thus have quadratic storage complexity, i.e. $O(n^2)$. By analyzing the variables in our algorithm we can notice that the storage complexity will be no higher than $O(n^2)$ in the worst scenario, and it is going to be only $O(n)$ in most practical cases.

4 Computer Simulations

In this section, we present some experimental results by using the proposed algorithm. Firstly, we study how the parameters influence the performance of the algorithm (Subsection 4.1) and temporal dynamics of nodes and particles (Subsection 4.2). Then, some simulation results of semi-supervised classification tasks by applying the proposed model to some toy data sets with different

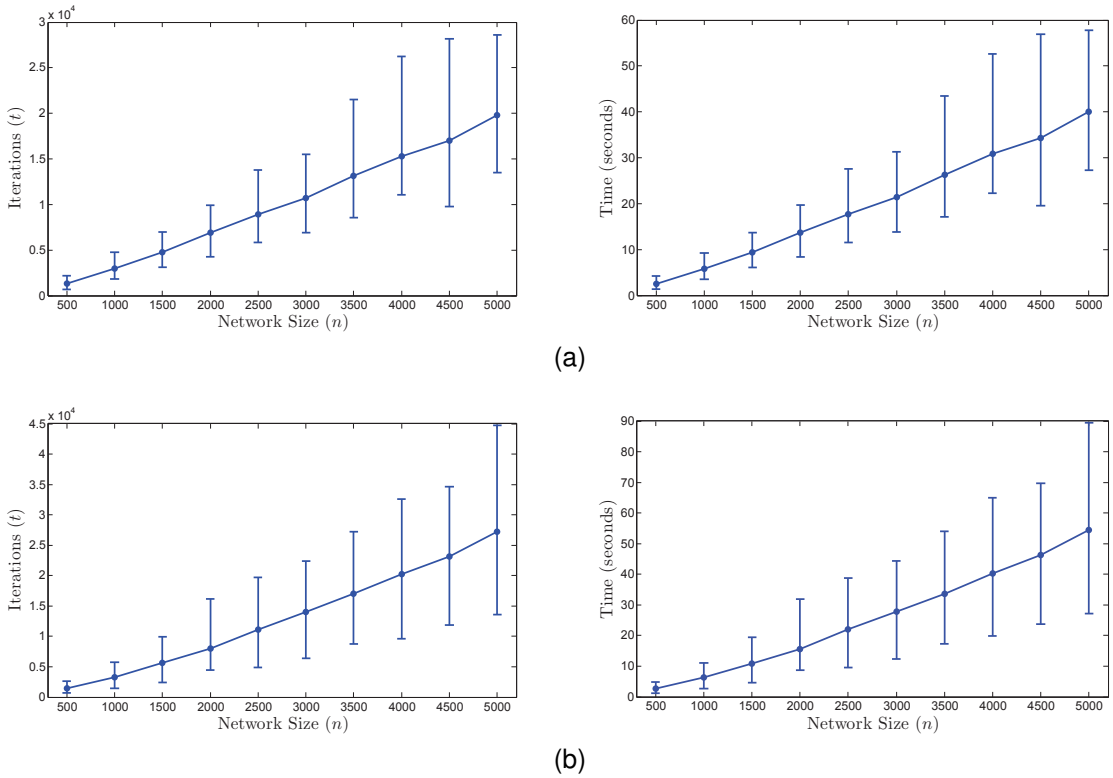


Fig. 2. Number of iterations (left) and time (right) required to the convergence of average maximum domination levels of nodes ($\langle v_i^{\omega_\ell} \rangle$, $\ell = \arg\max_q v_i^{\omega_q}$) with increasing network size (n), $l = 50$ and $\langle k \rangle = 25$. (a) $z_{out} = 5$, $z_{out}/\langle k \rangle = 0.2$; (b) $z_{out}/\langle k \rangle = 0.4$. Each point in the trace is averaged by 200 realizations. The error bars represent the maximum and minimum values.

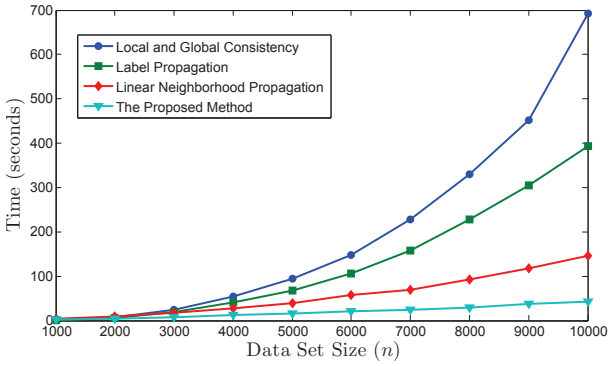


Fig. 3. Execution time of four graph-based semi-supervised classification methods, including the proposed method, to classify a sequence of artificial data sets with 4 normally distributed classes and with increasing sizes.

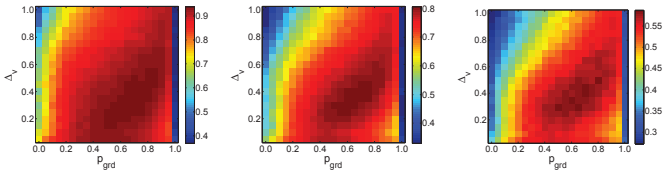
data distributions is presented in Subsection 4.3. Experiments with some artificial and real-world data sets are presented in Subsection 4.4, these results are compared to those obtained with state-of-the-art methods applied to the same data sets.

4.1 Parameter Selection

In this section, we present some computer simulations in order to find out how the parameters p_{grd} and Δ_v affects the algorithm's performance. We use networks with different mixtures and connectivity, which are generated using the method proposed by [38], already explained on Section 3.

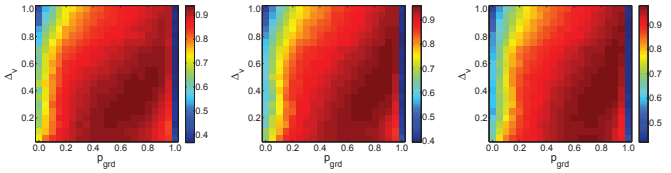
In the first set of experiments, we generate networks with increasing mixture, $z_{out}/\langle k \rangle = \{0.250, 0.375, 0.500\}$, while we keep both the network size $n = 128$ and average node degree $\langle k \rangle = 16$ constant. All nodes are equally divided into $c = 4$ classes. 10% of the nodes are randomly selected and presented to the algorithm with their respective labels while the others are unlabeled. We impose at least one labeled node per class. The parameters are tested with increasing values $p_{\text{grd}} = \{0.00, 0.05, 0.10, \ldots, 1.00\}$ and $\Delta_v = \{0.05, 0.10, 0.15, \ldots, 1.00\}$. All the 420 combinations are tested and each of them is repeated 100 times to take an average. The results are presented in Figs. 4a, 4b and 4c for $z_{out}/\langle k \rangle = 0.250$, 0.375 and 0.500 respectively.

By analyzing Fig. 4 we notice that both $p_{\text{grd}} = 0$ and $p_{\text{grd}} = 1$ lead to bad classification rates. Those cases correspond to complete random walking or complete greedy walking of particles, respectively, which confirms that the combination of random-greedy walking can improve the model's performance. We also notice that parameters p_{grd} and Δ_v depend on each other, because as the p_{grd} increases, the algorithm reduces its exploration behavior, therefore a higher value of Δ_v is required in



(a) (b) (c)

Fig. 4. Correct classification rate ϕ vs. p_{grd} vs. Δ_v for different network mixture $z_{out}/\langle k\rangle$. (a) $z_{out}/\langle k\rangle = 0.250$; (b) $z_{out}/\langle k\rangle = 0.375$; (c) $z_{out}/\langle k\rangle = 0.500$. The following parameters are held constant in these simulations: $n = 128$, $c = 4$, $\langle k\rangle = 16$. Each point is averaged by 100 realizations.



(a) (b) (c)

Fig. 5. Correct classification rate ϕ vs. p_{grd} vs. Δ_v for different node average degree $\langle k\rangle$. (a) $\langle k\rangle = 16$; (b) $\langle k\rangle = 24$; (c) $\langle k\rangle = 32$. The following parameters are held constant in these simulations: $n = 128$, $z_{out}/\langle k\rangle = 0.25$, $c = 4$, $\langle k\rangle = n/8$. Each point is averaged by 100 realizations.

order to magnify the changes caused by exploratory movements. By observing Figs. 4a to 4c we notice that there is a certain area in the parameter space where the algorithm reaches high classification rates. This area is quite large when the network mixture is low and it gets smaller as the network mixture increases, indicating that a finer tuning of parameters is required when facing a difficult problem. Interestingly, no matter the increasing of data mixture, the combination of p_{grd} and Δ_v to get good classification rate is almost fixed.

In the second set of experiments, we generate networks with increasing average node degree, $\langle k\rangle = \{16, 24, 32\}$, while we keep both the network size $n = 128$ and mixture $z_{out}/\langle k\rangle = 0.250$ constant. Number of classes, labeled subset size, parameter values and amount of repetitions are the same as in the previous experiments. The results are presented in Figs. 5a, 5b and 5c for $\langle k\rangle = 16$, 24 and 32 respectively. In this case, the area in the parameter space where the algorithm has good performance slightly dislocates to the right, which means that higher average node degree benefits from slightly higher p_{grd}.

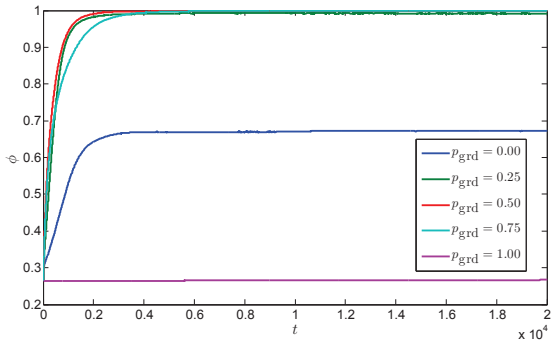
From Figs. 4 and 5 we observe that there is an area in the parameter space around $p_{\text{grd}} = 0.70$ and $\Delta_v = 0.35$ in which the algorithm usually presents good results no matter mixture or average node degree. Therefore, these parameter values can be considered in real problems if there is no time for model selection. Notice that the average node degree is always known a priori, thus one may prefer to set parameter values heuristically according to these measures. If there are sufficient labeled nodes, one can optimize the parameters by using cross-validation too.

4.2 Temporal Dynamics

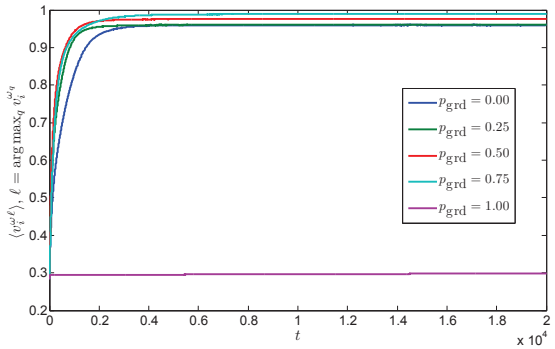
In order to better understand the temporal dynamics of the model and the importance of the random-greedy rule, we run another set of experiments. Figures 6a to 6c shows the temporal evolution of the model for five different values of p_{grd} ($p_{\text{grd}} = \{0.00, 0.25, 0.50, 0.75, 1.00\}$). In these cases, the algorithm is applied to randomly generated networks using the method proposed by [38], as explained in Section 3, with the following parameters: $n = 2048$, $l = 64$, $\langle k\rangle = 64$, $z_{out}/\langle k\rangle = 0.25$, and $\Delta_v = 0.35$.

Figure 6a shows the time series of correct detection rate (ϕ). Here we see that for complete random moving ($p_{\text{grd}} = 0.00$) or for complete greedy moving ($p_{\text{grd}} = 1.00$), the correct detection rate is low. This phenomenon has been expected because when $p_{\text{grd}} = 0.00$, the particles keep walking randomly all the time and the distance to their respective home nodes are not taken into account. In this case, each team of particles still tends to dominate a class and stay inside it due to the domination levels and the flags, but there is no guarantee that the particles will stay in the right classes, i.e., teams may exchange their territories because they do not have their home nodes as reference. On the other hand, if only the greedy rule is applied ($p_{\text{grd}} = 1$), the results are always poor. This is because, without the random rule, the particles are trapped in a small region of the already explored nodes and they rarely try to explore unknown nodes. Moreover, once a node has been fully dominated by a team, other teams will not have chance to visit it, because the probabilities calculated by Eq. (11) will always be zero. Therefore, a proper combination of randomness and determinism is the best alternative where the greedy rule holds the particles around their respective home nodes, defending their neighborhood, while the random rule allows the particles to explore previously unknown nodes and to compete for nodes dominated by another team. The complete greedy moving case ($p_{\text{grd}} = 1.00$) takes long time to converge and, at the same time, it gets bad classification results.

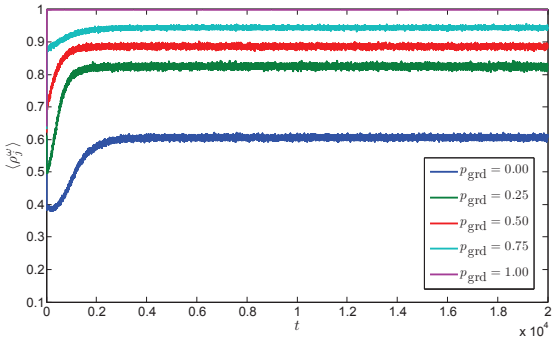
Figure 6b shows the average maximum domination level among all nodes ($\langle v_i^{\omega\ell}\rangle$, $\ell = \arg\max_q v_i^{\omega_q}$). Here we see that the domination levels quickly increase at the beginning and after a number of iterations stay at an almost constant level. The first stage corresponds to quick domination of particles to their respective neighborhoods without much competition and the second stage corresponds to high competition at the nodes near class borders before they are fully dominated. The exception is the complete greedy moving case ($p_{\text{grd}} = 1.00$), where the average maximum domination level increases very slowly.



(a)



(b)



(c)

Fig. 6. Time series for different values of p_{grd}: (a) correct detection rate (b) nodes' maximum domination level (c) average particle strength. Each point is the average of 100 realizations using networks generated by the method proposed by [38] with the following parameters: $n = 2048$, $l = 64$, $\langle k \rangle = 64$ and $z_{out}/\langle k \rangle = 0.25$.

Finally, Figure 6c shows the time series of average particle strength ($\langle \rho_j^{\omega} \rangle$), we see that the average strength of the particles gets higher as p_{grd} increases. This is because the particles lose their strengthes when they try to invade nodes dominated by another team, and that usually happens during random movement. In the complete greedy moving case ($p_{\text{grd}} = 1.00$) the competition is low so that the average particle strength is hardly lowered from one.

4.3 Toy Data Sets

In this section, we apply the proposed method to some toy data sets with different data distributions, which are generated by using PRTools [41] and are presented in Fig. 7, in order to verify the method's efficacy when classifying these types of data. The first data set (Fig. 7a) consists of $2,000$ samples divided equally into two banana-shaped classes, 20 of these samples (1%) are randomly selected as the pre-labeled subset. The second data set (Fig. 7b) consists of $1,000$ samples divided equally into two Highleyman classes, 100 of these samples (10%) are randomly selected as the pre-labeled samples. The third data set (Fig. 7c) consists of $1,200$ samples divided into two Lithuanian classes, with 800 and 400 samples respectively, in this case 60 of these samples (5%) are randomly selected as the pre-labeled subset. Finally, the forth data set (Figure 7d) consists of $1,200$ samples equally divided into three Gaussian distributed classes, 24 of these samples (2%) were randomly selected as the pre-labeled samples. In all experiments the following parameters were held constant: $\Delta_v = 0.35$ and $p_{\text{grd}} = 0.70$, corresponding to values in the *safe range* as described in Section 4.1. The graphs are built from the data sets by using Eq. (2), with the parameter k empirically set to $k = 20$ for all the experiments, except for the second one where $k = 5$ is used. The right side of Fig. 7 shows the final states of label propagation. We see that good results are obtained in all of the 4 cases, indicating that the proposed algorithm is robust to process data sets of various distributions.

4.4 Benchmark

In order to measure the performance of the proposed method, we have applied it to 7 standard semi-supervised data sets[1]. Some basic information about these data sets are available at Table 1. For detailed description about each of them, one can refer to [8].

TABLE 1
Basic Information of the Benchmark Data Sets

Data set	Classes	Dimension	Points	Type
g241c	2	241	1500	artificial
g241d	2	241	1500	artificial
Digit1	2	241	1500	artificial
USPS	2	241	1500	imbalanced
COIL	6	241	1500	
BCI	2	117	400	
Text	2	11,960	1500	sparse

For each data set, there are 10 or 100 labeled data points, and for each case 12 random splits are performed in order to partition the data set into labeled and unlabeled points. It is ensured that each split contains at least one point from each class. For comparison purpose,

1. Available at http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.html

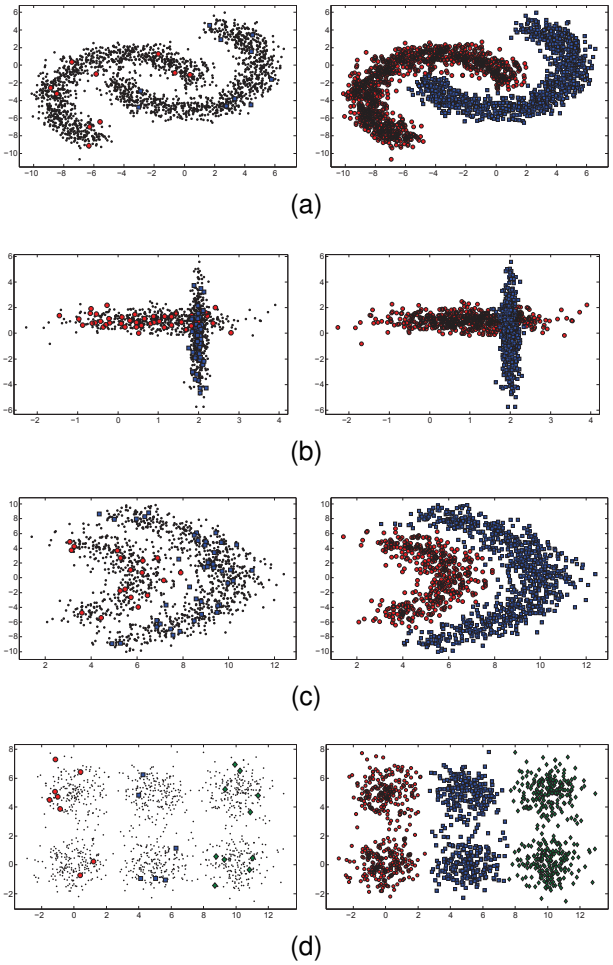


Fig. 7. Classification of toy data sets with: (a) $2,000$ samples divided equally into two banana-shaped classes; (b) $1,000$ samples divided equally into two Highleyman classes; (c) $1,200$ samples divided into two Lithuanian classes, with 800 and 400 samples respectively; (d) $1,200$ samples equally divided into three Gaussian distributed classes. The input data sets are represented on the left and the algorithm output are represented on the right. Circles (red), squares (blue), and lozenges (green) represent the labeled nodes; small dots (black) represent the unlabeled nodes.

we have included the classification results of 13 semi-supervised learning methods presented in [8]. The Nearest Neighbor (1-NN) and Linear SVM (SVM) [42] are used as the base line algorithms. The other 11 algorithms are those that presented the best performance on their respective category and they are presented in the top 11 lines of Table 2. The detailed configurations of each method are described in [8]. Besides of these 13 techniques, we have also included other 3 semi-supervised learning graph-based methods in the comparison, which can be considered as the techniques of the same subcategory of the proposed method. These are presented at the last 3 lines of Table 2.

In these experiments, we have used Eq. (2) to build graphs from the data sets, therefore we have to determine the values of three parameters, namely p_{grd} and Δ_v of the algorithm, and k for building graphs. Since the parameters of the 13 algorithms (from [8]) in comparison are optimized, we optimize the parameters of the proposed method too. The optimization is realized by using the genetic algorithm available in Global Optimization Toolbox of MATLAB with its default parameters. The parameters of the proposed algorithm are optimized over the following range: $0 \leq p_{\text{grd}} \leq 1$, $0 < \Delta_v \leq 1$, $1 \leq k \leq 100$.

TABLE 2
Semi-Supervised Learning Methods used for Performance Comparison

Abbreviation	Method	References
MVU + 1-NN	Maximum Variance Unfolding	[43], [44]
LEM + 1-NN	Laplacian Eigenmaps	[45]
QC + CMN	Quadratic Criterion and Class Mass Regularization	[29], [46]
Discrete Reg.	Discrete Regularization	[47]
TSVM	Transductive Support Vector Machines	[31], [48]
SGT	Spectral Graph Transducer	[31]
Cluster-Kernel	Cluster Kernels	[49]
Data-Dep. Reg.	Data-Dependent Regularization	[50]
LDS	Low-Density Separation	[48]
Laplacian RLS	Laplacian Regularized Least Squares	[51]
CHM (normed)	Conditional Harmonic Mixing	[52]
LGC	Local and Global Consistency	[19]
LP	Label Propagation	[25]
LNP	Linear Neighborhood Propagation	[26]

The same optimization process is used for the 3 graph-based methods. For both the LGC and the LP methods, σ is optimized over the following range: $0 \leq \sigma \leq 100$; and for the LNP method, k is optimized over the following range: $1 \leq \sigma \leq 100$. For the LGC and LNP methods, we have fixed $\alpha = 0.99$, as done in [19] and [26], respectively.

Tables 3 and 4 report the average test errors of the 16 methods and the proposed method applied to the data sets presented in Table 1. The values obtained for the proposed method are averaged by 200 realizations on each of the 12 subsets. Tables 3 and 4 show test errors (%) with 10 and 100 labeled training points respectively. By observing these results we notice that the performance of the proposed method is comparable to the state-of-the-art methods, and interestingly, its performance gets even better when less labeled training points are provided. This feature is quite desirable, since semi-supervised learning usually treats data sets with few labeled data points.

From Tables 3 and 4, we see that some methods are among the best for some data sets, but at the same time, they are among the worst for other data sets. However, the proposed method is more balanced than all others and it is well ranked specially in the cases which have few labeled data. Tables 5 and 6 show performance ranks for 10 and 100 labeled training points, respectively. We see that the proposed method has the best average rank and the lowest standard deviation in the 10 labeled training points case and it is within the best ones in the 100 labeled training points case. Comparing Table 5 and

TABLE 3
Test errors (%) with 10 labeled training points.

	g241c	g241d	Digit1	USPS	COIL	BCI	Text
1-NN	47.88	46.72	13.65	16.66	63.36	49.00	38.12
SVM	47.32	46.66	30.60	20.03	68.36	49.85	45.37
MVU + 1-NN	47.15	45.56	14.42	23.34	62.62	47.95	45.32
LEM + 1-NN	44.05	43.22	23.47	19.82	65.91	48.74	39.44
QC + CMN	39.96	46.55	9.80	13.61	59.63	50.36	40.79
Discrete Reg.	49.59	49.05	12.64	16.07	63.38	49.51	40.37
TSVM	24.71	50.08	17.77	25.20	67.50	49.15	31.21
SGT	**22.76**	**18.64**	8.92	25.36	-	49.59	29.02
Cluster-Kernel	48.28	42.05	18.73	19.41	67.32	48.31	42.72
Data-Dep. Reg.	41.25	45.89	12.49	17.96	63.65	50.21	-
LDS	28.85	50.63	15.63	17.57	61.90	49.27	**27.15**
Laplacian RLS	43.95	45.68	**5.44**	18.99	**54.54**	48.97	33.68
CHM (normed)	39.03	43.01	14.86	20.53	-	46.90	-
LGC	45.82	44.09	9.89	**9.03**	63.45	47.09	45.50
LP	42.61	41.93	11.31	14.83	55.82	**46.37**	49.53
LNP	47.82	46.24	8.58	17.87	55.50	47.65	41.06
Proposed Method	37.57	43.94	9.94	17.44	58.65	47.66	31.15

TABLE 4
Test errors (%) with 100 labeled training points.

	g241c	g241d	Digit1	USPS	COIL	BCI	Text
1-NN	43.93	42.45	3.89	5.81	17.35	48.67	30.11
SVM	23.11	24.64	5.53	9.75	22.93	34.31	26.45
MVU + 1-NN	43.01	38.20	2.83	6.50	28.71	47.89	32.83
LEM + 1-NN	40.28	37.49	6.12	7.64	23.27	44.83	30.77
QC + CMN	22.05	28.20	3.15	6.36	10.03	46.22	25.71
Discrete Reg.	43.65	41.65	2.77	4.68	**9.61**	47.67	24.00
TSVM	18.46	22.42	6.15	9.77	25.80	33.25	24.52
SGT	17.41	9.11	2.61	6.80	-	45.03	**23.09**
Cluster-Kernel	**13.49**	**4.95**	3.79	9.68	21.99	35.17	24.38
Data-Dep. Reg.	20.31	32.82	**2.44**	5.10	11.46	47.47	-
LDS	18.04	23.74	3.46	4.96	13.72	43.97	23.15
Laplacian RLS	24.36	26.46	2.92	4.68	11.92	**31.36**	23.57
CHM (normed)	24.82	25.67	3.79	7.65	-	36.03	-
LGC	41.64	40.08	2.72	**3.68**	45.55	43.50	46.83
LP	30.39	29.22	3.05	6.98	11.14	42.69	40.79
LNP	44.13	38.30	3.27	17.22	11.01	46.22	38.48
Proposed Method	24.20	23.93	2.65	4.65	14.85	44.38	25.03

TABLE 5
Ranking of semi-supervised methods with 10 labeled training points

	g241c	g241d	Digit1	USPS	COIL	BCI	Text	Mean	Std. Dev.
1-NN	15	14	10	5	8	10	6	**9.7**	**2.99**
SVM	13	13	17	13	15	15	13	**14.1**	**1.49**
MVU + 1-NN	12	8	11	15	7	6	12	**10.1**	**3.13**
LEM + 1-NN	10	5	16	12	12	8	7	**10.0**	**3.70**
QC + CMN	6	12	4	2	5	17	9	**7.9**	**5.15**
Discrete Reg.	17	15	9	4	9	13	8	**10.7**	**3.57**
TSVM	2	16	13	16	14	11	4	**10.9**	**4.15**
SGT	1	1	3	17	-	14	2	**6.3**	**6.73**
Cluster-Kernel	16	3	15	11	13	7	10	**10.7**	**3.95**
Data-Dep. Reg.	7	10	8	9	11	16	-	**10.2**	**2.80**
LDS	3	17	14	7	6	12	1	**8.6**	**5.39**
Laplacian RLS	9	9	1	10	1	9	5	**6.3**	**3.76**
CHM (normed)	5	4	12	14	-	2	-	**7.4**	**5.11**
LGC	11	7	5	1	10	3	14	**7.3**	**4.35**
LP	8	2	7	3	3	1	15	**5.6**	**4.78**
LNP	14	11	2	8	2	4	11	**7.4**	**3.88**
Proposed Method	4	6	6	6	4	5	3	**4.9**	**1.16**

TABLE 6
Ranking of semi-supervised methods with 100 labeled training points

	g241c	g241d	Digit1	USPS	COIL	BCI	Text	Mean	Std. Dev.
1-NN	16	16	14	7	9	17	10	**12.7**	**3.72**
SVM	7	6	15	15	11	3	9	**9.4**	**4.42**
MVU + 1-NN	14	13	6	9	14	16	12	**12.0**	**3.30**
LEM + 1-NN	12	12	16	12	12	10	11	**12.1**	**1.86**
QC + CMN	6	9	9	8	2	12	8	**7.7**	**3.00**
Discrete Reg.	15	15	5	3	1	15	4	**8.3**	**5.68**
TSVM	4	3	17	16	13	2	6	**8.7**	**6.08**
SGT	2	2	2	10	-	11	1	**4.7**	**4.36**
Cluster-Kernel	1	1	12	14	10	4	5	**6.7**	**4.66**
Data-Dep. Reg.	5	11	1	6	5	14	-	**7.0**	**4.59**
LDS	3	4	11	5	7	8	2	**5.7**	**2.92**
Laplacian RLS	9	8	7	3	6	1	3	**5.3**	**2.51**
CHM (normed)	10	7	12	13	-	5	-	**9.4**	**3.35**
LGC	13	15	4	1	15	7	15	**10.0**	**5.77**
LP	11	10	8	11	4	6	14	**9.1**	**3.29**
LNP	17	14	10	17	3	12	13	**12.3**	**4.36**
Proposed Method	8	5	3	2	8	9	7	**6.0**	**2.56**

6, the rank of our technique is a little bit lowered than others when more labeled data items are provided. It means that our method depends less on the percentage of labeled data than other ones. For example, in the Local and Global Consistency model, the conformation to labeled data in the label propagation process is explicitly represented by one of the two terms of the energy function to be minimized. For this reason, the techniques under comparison may have a bigger improvement of performance than our technique. It is worth to note that semi-supervised learning usually considers few labeled data items, since it is usually difficult to obtain labeled data. In this way, our method represents an advantage.

Finally, we present experimental results of our method and three representative graph-based methods (LGC, LP, and LNP) applied to three large real-world data set: the USPS Handwritten Digit Data Set [53], the Insurance Company Benchmark (COIL 2000) Data Set [54], and the Letter Recognition Data Set [55]. These data sets have $9,298$, $9,000$ and $20,000$ samples, respectively. For each data set, three different labeled subsets are build by randomly selecting 1%, 5%, and 10% from the total of elements, respectively. The four algorithms are applied to each of these configurations of each data set, and Table 7 report the test errors and execution times of this experiment. Time is measured in a laptop computer with an Intel Core i7-840QM Processor and 8GB of RAM. The algorithms are implemented by using MATLAB. Again, we see that our method can get good results compared to the others. Notice that the test system memory is not sufficient to run the LGC method in the Letter Recognition Data Set, as it is a large data set and the algorithm has high memory requirements.

5 Conclusions

This paper presents a new network-based method for semi-supervised classification by using combined random-greedy walking of particles, where each of them

TABLE 7
Class detection errors (%) and execution times (seconds) of four methods applied to large data sets

	LGC		LP		LNP		Prop. Method	
	Error	Time	Error	Time	Error	Time	Error	Time
USPS - 10%	**3.73**	457	7.27	154	12.80	105	4.37	**69**
USPS - 5%	**4.72**	456	9.77	243	5.99	107	5.13	**76**
USPS - 1%	9.38	461	23.77	579	22.54	120	**9.29**	**103**
COIL - 10%	7.64	445	8.14	54	11.09	73	**5.79**	**23**
COIL - 5%	7.48	443	7.92	59	11.38	68	**5.84**	**28**
COIL - 1%	7.36	472	7.16	83	8.03	65	**5.94**	**33**
Letter - 10%	-	-	**10.94**	2232	24.04	1010	11.74	**484**
Letter - 5%	-	-	18.99	1901	33.89	982	**16.86**	**497**
Letter - 1%	-	-	46.94	5495	54.28	1274	**39.76**	**522**

corresponds to a labeled data point. Starting from a small territory corresponding to the few labeled samples, these particles expand their domain walking in the network, collaborating each other with the particles of the same class, and competing to particles of other classes from intruding their territory.

Due to the competition mechanism, there is a divide-and-conquer effect embedded in our method. In this way, the particles are avoided to visit a considerable quantity of nodes which are definitely belonging to other teams of particles. In other words, traditional graph-based semi-supervised learning models spread labels in a global fashion, while the method proposed in this paper spread labels in a local fashion. Consequently, the proposed method has a time complexity lower than other graph-based models, our analysis shows that its order of time complexity is at most $O(n^2)$, while most graph-based semi-supervised methods have cubic complexity order ($O(n^3)$) [7]. Therefore, the proposed method can be used to classify larger data sets. Computer simulations show that the proposed model is promising for semi-supervised learning, resulting in good classification accuracy for both synthetic and real-world data sets, specially in the cases which few labeled data are available.

ACKNOWLEDGMENTS

This work is supported by the State of São Paulo Research Foundation (FAPESP) and the Brazilian National Council of Technological and Scientific Development (CNPq).

REFERENCES

[1] T. Mitchell, *Machine Learning*. McGraw Hill, 1997.
[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[3] E. Alpaydin, *Introduction to machine learning*. MIT Press, 2004.
[4] K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan, *Data Mining: A Knowledge Discovery Approach*. Springer, 2007.
[5] C. Aggarwal and P. Yu, "A survey of uncertain data algorithms and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 5, pp. 609–623, May 2009.
[6] R. Wolff, K. Bhaduri, and H. Kargupta, "A generic local algorithm for mining data streams in large distributed systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 4, pp. 465–478, April 2009.
[7] X. Zhu, "Semi-supervised learning literature survey," Computer Sciences, University of Wisconsin-Madison, Tech. Rep. 1530, 2005.
[8] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, MA: The MIT Press, 2006.
[9] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," in *Machine Learning*, vol. 39, 2000, pp. 103–134.
[10] A. Fujino, N. Ueda, and K. Saito, "A hybrid generative/discriminative approach to semi-supervised classifier design," in *AAAI-05, Proceedings of the Twentieth National Conference on Artificial Intelligence*, 2005, pp. 764–769.
[11] A. Demiriz, K. P. Bennett, and M. J. Embrechts, "Semi-supervised clustering using genetic algorithms," in *Proceedings of Artificial Neural Networks in Engineering (ANNIE-99*. ASME Press, 1999, pp. 809–814.
[12] R. Dara, S. Kremer, and D. Stacey, "Clustering unlabeled data with soms improves classification of labeled real-world data," in *Proceedings of the World Congress on Computational Intelligence (WCCI)*, 2002, pp. 2237–2242.
[13] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *COLT: Proceedings of the Workshop on Computational Learning Theory*, 1998, pp. 92–100.
[14] T. M. Mitchell, "The role of unlabeled data in supervised learning," in *Proceedings of the Sixth International Colloquium on Cognitive Science*, 1999.
[15] Z.-H. Zhou and M. Li, "Tri-training: exploiting unlabeled data using three classifiers," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 11, pp. 1529–1541, Nov. 2005.
[16] ——, "Semisupervised regression with cotraining-style algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 11, pp. 1479–1493, Nov. 2007.
[17] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley-Interscience, September 1998.
[18] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the Twentieth International Conference on Machine Learning*, 2003, pp. 912–919.
[19] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems*, vol. 16. MIT Press, 2004, pp. 321–328.
[20] M. Wu and B. Schölkopf, "Transductive classification via local learning regularization." Microtome, 03 2007, pp. 628–635.
[21] F. Wang, T. Li, G. Wang, and C. Zhang, "Semi-supervised classification using local and global regularization," in *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*. AAAI Press, 2008, pp. 726–731.
[22] M. Szummer and T. Jaakkola, "Partially labeled classification with markov random walks," in *Advances in Neural Information Processing Systems*, vol. 14, 2002.
[23] L. Grady, "Random walks for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1768–1783, 2006.
[24] M. Maila and J. Shi, "Learning segmentation with random walk," in *Advances Neural Information Processing Systems, NIPS*, 2001.
[25] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Carnegie Mellon University, Pittsburgh, Tech. Rep. CMU-CALD-02-107, 2002.
[26] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 1, pp. 55–67, Jan. 2008.
[27] W. Wang and Z.-H. Zhou, "A new analysis of co-training," in *ICML*, J. Fürnkranz and T. Joachims, Eds. Omnipress, 2010, pp. 1135–1142.
[28] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proceedings of the Eighteenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann, 2001, pp. 19–26.
[29] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semisupervised learning on large graphs," in *Conference on Learning Theory*. Springer, 2004, pp. 624–638.
[30] M. Belkin, N. P., and V. Sindhwani, "On manifold regularization," in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, 2005, pp. 17–24.
[31] T. Joachims, "Transductive learning via spectral graph partitioning," in *Proceedings of International Conference on Machine Learning*. AAAI Press, 2003, pp. 290–297.

[32] F. Wang, S. Wang, C. Zhang, and O. Winther, "Semi-supervised mean fields," in *AISTATS2007: Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, San Juan, Puerto Rico, March 2007.

[33] G. Getz, N. Shental, and E. Domany, "Semi-supervised learning - a statistical physics approach," in *Proc. of the 22nd ICML Workshop on Learning with Partially Classified Training Data*, Bonn, Germany, 2005.

[34] F. Wang and C. Zhang, "Semi-supervised learning based on generalized point charge models," *IEEE Transactions on Neural Networks*, vol. 19, no. 7, pp. 1307 – 1311, July 2008.

[35] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *ICML*, J. Fürnkranz and T. Joachims, Eds. Omnipress, 2010, pp. 679–686.

[36] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, pp. 167–256.

[37] S. Bornholdt. and H. Schuster, *Handbook of Graphs and Networks: From the Genome to the Internet*. Wiley-VCH, 2006.

[38] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 9, p. P09008, 2005.

[39] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, pp. 75–174, 2010.

[40] M. G. Quiles, L. Zhao, R. L. Alonso, and R. A. F. Romero, "Particle competition for complex network community detection," *Chaos*, vol. 18, no. 3, p. 033107, 2008.

[41] R. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D. Tax, and S. Verzakov, "Prtools4.1, a matlab toolbox for pattern recognition," Delft University of Technology.

[42] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[43] K. Q. Weinberger and L. K. Saul, "Unsupervised learning of image manifolds by semidefinite programming," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, Washington D.C., 2004, pp. 988–995.

[44] J. Sun, S. Boyd, L. Xiao, and P. Diaconis, "The fastest mixing markov process on a graph and a connection to a maximum variance unfolding problem," *SIAM Review*, vol. 48, no. 4, pp. 681–699, 2006.

[45] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computuation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[46] O. Delalleau, Y. Bengio, and N. L. Roux, "Efficient non-parametric function induction in semi-supervised learning," in *Artificial Intelligence and Statistics*, 2005, pp. 96–103.

[47] D. Zhou and B. Schölkopf, *Semi-supervised Learning*. Cambridge, MA: MIT Press, 2006, ch. Discrete Regularization, pp. 237–250.

[48] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *Tenth International Workshop on Artificial Intelligence and Statistics*, 2005, pp. 57–64.

[49] O. Chapelle, J. Weston, and B. Schölkopf, "Cluster kernels for semi-supervised learning," in *Advances in Neural Information Processing Systems*, vol. 15, 2003.

[50] A. Corduneanu and T. Jaakkola, *Semi-supervised Learning*. Cambridge, MA: MIT Press, 2006, ch. Data-Dependent Regularization, pp. 163–190.

[51] V. Sindhwani, P. Niyogi, and M. Belkin, "Beyond the point cloud: from transductive to semi-supervised learning," in *ICML '05: Proceedings of the 22nd international conference on Machine learning*. New York, NY, USA: ACM, 2005, pp. 824–831.

[52] C. J. C. Burges and J. C. Platt, *Semi-supervised Learning*. Cambridge, MA: MIT Press, 2006, ch. Semi-Supervised Learning with Conditional Harmonic Mixing, pp. 251–273.

[53] J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 550–554, 1994.

[54] P. van der Putten and M. van Someren, "Coil challenge 2000: The insurance company case," Sentient Machine Research and Leiden Institute of Advanced Computer Science, Amsterdam and Leiden, Tech. Rep. 2000-09, 6 2000.

[55] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[56] J. Fürnkranz and T. Joachims, Eds., *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. Omnipress, 2010.

Fabricio Breve is a post-doctoral researcher in the University of São Paulo, Brazil. He received his bachelor's degree from the Methodist University of Piracicaba, Brazil in 2001, his master's degree from the Federal University of São Carlos, Brazil in 2006, and his Ph.D. from the University of São Paulo, Brazil in 2010, all in Computer Science. His research interests include machine learning, image processing, artificial neural networks, complex networks, and nature-inspired computation.

Liang Zhao is an associate professor in the Department of Computer Science at University of São Paulo, Brazil. He received the BS degree in 1988 from Wuhan University, China, and the PhD degree from Aeronautic Institute of Technology, Brazil, in 1998. His research interests include artificial neural networks, nonlinear dynamical systems, complex networks, nature-inspired computation, and pattern recognition. He has received a Brazilian Research Productivity Award and he is an Associate Editor of IEEE Transactions on Neural Networks.

Marcos Quiles received the B.S. degree in 2003 from the State University of Londrina, Brazil, and Ph.D. degree from the University of São Paulo, Brazil, in 2009, both in Computer Science. He is currently an Assistant Professor at the Department of Science and Technology of the Federal University of São Paulo, Brazil. His current research interests include neural networks, computer vision, machine learning, and complex networks.

Witold Pedrycz received the M.Sc., Ph.D., and D.Sci. degrees from the Silesian University of Technology, Poland. He is a Professor and Canada Research Chair (CRC) with the Department of Electrical and Computer Engineering, University of Alberta, Canada. He is also with the Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland. He is the author of nine research monographs on computational intelligence and software engineering. He is the Editor-in-Chief of Information Sciences and a member of the Editorial Board of several other international journals. His research interests include computational intelligence, fuzzy modeling and control, knowledge discovery and data mining, bioinformatics, and software engineering.

Jiming Liu is the Chair Professor and Head of Computer Science Department at Hong Kong Baptist University. His current research interests include: Autonomy-Oriented Computing, Web Intelligence, self-organizing systems, and complex networks. He has published over 250 journal and conference papers, and 5 authored research monographs. He has served as the Editor-in-Chief of Web Intelligence and Agent Systems, Associate Editor of several international journals.