Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 385265, 13 pages
http://dx.doi.org/10.1155/2013/385265

Research Article

Semisupervised Clustering for Networks Based on Fast Affinity Propagation

Mu Zhu, Fanrong Meng, and Yong Zhou

Department of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221008, China

Correspondence should be addressed to Mu Zhu; zhumubest@163.com

Received 3 May 2013; Accepted 1 July 2013

Academic Editor: Orwa Jaber Housheya

Most of the existing clustering algorithms for networks are unsupervised, which cannot help improve the clustering quality by utilizing a small number of prior knowledge. We propose a semisupervised clustering algorithm for networks based on fast affinity propagation (SCAN-FAP), which is essentially a kind of similarity metric learning method. Firstly, we define a new constraint similarity measure integrating the structural information and the pairwise constraints, which reflects the effective similarities between nodes in networks. Then, taking the constraint similarities as input, we propose a fast affinity propagation algorithm which keeps the advantages of the original affinity propagation algorithm while increasing the time efficiency by passing only the messages between certain nodes. Finally, by extensive experimental studies, we demonstrate that the proposed algorithm can take fully advantage of the prior knowledge and improve the clustering quality significantly. Furthermore, our algorithm has a superior performance to some of the state-of-art approaches.

1. Introduction

Network as an expressive data structure is popularly used to model structural relationships between objects in many real-world applications. Examples include social networks, web, and citation networks. In these networks, individuals represented by nodes are linked with some special relationships, such as the friendships between people, hyperlinks between web pages, and references among papers.

Network clustering, as a key technology for network analysis, can discover the hidden structures and functions in networks [1], which is attracting a considerable amount of attention from researchers in various domains. A lot of related research achievements, including modularity optimization [2–5], spectral clustering [6–8], and similarities-based algorithms [9–11] have been implemented to partition the networks into clusters (or communities, groups), such that there are a dense set of edges within each cluster and few edges between clusters. However, most of the existing algorithms are unsupervised and cannot utilize any prior knowledge to improve the clustering quality. Usually, prior knowledge related to the data such as label information or pairwise constraints can be obtained in many real applications.

Semisupervised clustering is a very useful strategy that can guide the clustering process to get a high clustering quality in the presence of the obtained prior knowledge. There have been a lot of semisupervised clustering methods in data mining and machine learning domains, such as the constraint-based methods [12–14], the distance (or similarity) metric learning methods [15–18], and the methods integrating the two above [19–21]. However, most of them are designed for the traditional vector data and not well fit for the current network data. For example, some distance-based algorithms use the Euclidean distance, which is invalid in measuring the distances between nodes in networks. Only a few graph-based semisupervised methods, such as semisupervised spectral clustering algorithm [22–24] and semisupervised kernel k-means algorithms [25, 26], can be applied in network clustering, but they cannot always get clustering result with high and steady quality.

In this paper, we propose a novel semisupervised clustering algorithm for networks based on fast affinity propagation (SCAN-FAP), which has a strong ability to improve the

clustering quality by making full use of a little of prior knowledge. The basic innovation of our approach consists of two important components. The first is to define an effective similarity measure taking into account both the structural and the pairwise constraint information, while the second is to provide a performable clustering algorithm taking the similarities as input. Our contributions are threefold.

(i) We propose a constraint similarity measuring method, which is extending the idea of the basic SimRank [27]. This method adopts the most commonly used must-link and cannot-link pairwise constraints as the prior knowledge and embodies them into the SimRank equation, which tactfully integrates the structural information and the prior knowledge.

(ii) Taking the constraint similarities as input, we propose a fast affinity propagation algorithm for network clustering. By the analysis of the factor graph, the theoretical basis of the affinity propagation, we modify the binary model by reducing the number of variables to be optimized. The proposed algorithm derived from the new model can generate clusters with high quality while speed up the running time compared with the original affinity propagation algorithm [28].

(iii) We demonstrate, by using various real networks, that the proposed algorithm method has the advantages of both the effectiveness and efficiency on improving the clustering quality, by making use of a small number of pairwise constraint information.

The paper is organized as follows. Section 2 reviews some related work including network clustering and semisupervised clustering. Section 3 describes the proposed constraint similarities and its calculations. Section 4 proposes the fast affinity propagation algorithm for networks. Section 5 analyzes the time complexity of our algorithm. We present the experimental results to show the advantages of our algorithm in Section 6. Finally, in Section 7, we present the conclusions of our work.

2. Related Work

2.1. Network Clustering. Recently, there exists an increasing body of literature on network clustering methods, such as modularity optimization, spectral clustering, and similarity-based algorithms.

Modularity [2] is a very famous objective function for measuring the strength of dividing a network into clusters. The value of modularity is the fraction of the edges that fall within the given clusters minus the expected such fraction if edges were distributed at random. Networks with high modularity have dense links between nodes within clusters but sparse connections between nodes in different clusters. Therefore, lots of optimization methods, such as Mod-CSA [3], LPAm [4], and MIGA [5], are designed for getting the maximum modularity as their objective. It is worth mentioning that these methods have a resolution limit and may fail to identify some smaller clusters [29].

Spectral clustering algorithms are also popularly used for networks. They cut the networks by transforming the initial set of nodes into a set of points in space, whose coordinates are elements of eigenvectors. The set of points is then clustered via standard techniques, such as k-means. There are various kinds of spectral clustering, which are distinguished from each other by their "cut functions," which will be then solved as quadratic optimization problems. The traditional objective functions include the famous "Ratio Cut" [7] and "Normalized Cut" [8]. Other recent "cut functions" proposed in [30, 31] are set equal to the modularity for real networks. In addition, Dhillon et al. [32] proved that, in certain conditions, spectral clustering algorithms can seem as the corresponding kernel-based k-means formations.

A nature property in networks is the similarity between nodes. Therefore, some similarity-based algorithms are proposed to utilize the similarities to partition the networks. SCAN [9], a density-based algorithm, clusters the networks by using the transitivity of the cosine similarities between nodes. It can not only find the clusters with connectivity and maximality but also detect the hubs and outliers. Liu [10] made use of the famous affinity propagation algorithm to partition the networks with various similarity measures. Cheng et al. [11] measured the similarities between nodes by considering the information of both the content and links and then proposed an efficient incremental computing approach for network clustering.

However, most of algorithms introduced above are unsupervised, which do not have the ability of utilizing a small number of prior knowledge to improve the clustering quality.

2.2. Semisupervised Clustering. Semisupervised clustering is a kind of learning methods, which can guide the clustering process by using a small number of prior knowledge. It becomes growing popular in machine learning and data mining due to its important ability of improving the clustering quality. Generally, there are two kinds of prior knowledge: label information and pairwise constraints. The pairs of constraints are more popular than the label information since they are faster or cheaper to be obtained [19]. Must-link and cannot-link are two common used pairwise constraints. The former indicates the two individuals must be in the same cluster while the latter indicates the individuals must be in different clusters.

There are two kinds of strategies about semisupervised clustering: constraint-based optimization and distance (or similarity) metric learning. The constraint-based optimization methods, such as [12–14], modify the objective function by using the obtained label information or pairwise constraints and then solve the new objective optimization problem. The metric learning methods always use the prior information to adjust the original distances (or similarities) between objects and then cluster the dataset by some original method. Relevant examples are demonstrated in [15–18]. Some other methods, such as [19–21], combine the two strategies together to make use of the prior knowledge.

However, most of the existing semisupervised clustering methods will be invalid in network clustering, since they are

only designed for the traditional vector data. A few graph-based methods in their semisupervised formation can be used to partition networks. For example, semisupervised spectral clustering [22] modifies the adjacent matrix of the network using the pairwise constraints at the beginning of the original algorithm; semisupervised kernel-based k-means method proposed by Kulis et al. [25] defines a penalized kernel matrix subject to pairwise constraints, which will guide the assignments of the clusters; the method in [26] defines a new penalized kernel matrix taking the density modularity [33] as the objective function.

3. Constraint Similarity Measure

One of the most important motivations of our method is to adjust the original similarities between nodes when some prior knowledge consists in the networks. For the prior knowledge, we adopt the pairwise constraints since it is more popular in real applications. For the similarities adjustments, we define a novel measure integrating the pairwise constraints and the basic SimRank similarities.

3.1. Basic SimRank. SimRank is a general and a global similarity measure, which takes fully into account the structural context of all the nodes in networks. The basic idea of SimRank is "two objects are similar if they are related to similar objects."

Let $G = (V, E)$ represent a network, where V is the set of nodes and E is the set of edges. For a node v, $I(v) = \{u \mid (u, v) \in E\}$ represents the set of neighbors of v; $|I(v)|$ is the number of elements in $I(v)$; $I_i(v)$ is the ith element in $I(v)$. Let us denote the similarity between nodes a and b by $s(a, b) \in [0, 1]$. Following the earlier definitions, a recursive equation is defined for $s(a, b)$. If $a = b$, then $s(a, b)$ is set to be 1.

Otherwise,

$$s(a,b) = \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)||I(b)|} s\left(I_i(a), I_j(b)\right), \quad (1)$$

where C is a decay factor between 0 and 1. Note that either a or b may have no neighbors. In this situation, $s(a, b)$ is set to be 0 since there is no way to infer any similarity between them. Formally, when $I(a) = \emptyset$ or $I(b) = \emptyset$, $s(a, b) = 0$.

3.2. Definition of the Constraint Similarity Measure. The basic SimRank is an unsupervised measure. However, in semisupervised network clustering, the prior knowledge should be taken fully into account for the similarity measure. The pairwise constraints are adopted to express the prior knowledge in our method. The formation of the pairwise constraints is a constraint matrix Q, the element of which represents the relationship between two nodes:

$$Q(i,j) = \begin{cases} 1 & \text{must-link}(i,j) \\ -1 & \text{cannot-link}(i,j) \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where must-link means the pairs of two nodes must be in the same cluster and cannot-link means the pairs of two nodes must be in different clusters.

In particular to deserve to be mentioned, the must-link relationship satisfies three properties: reflexivity, symmetry, and transitivity. Formally,

(1) reflexivity: for any node v, $Q(v, v) = 1$;

(2) symmetry: for any pair of two nodes v and w, if $Q(v, w) = 1$, then $Q(w, v) = 1$;

(3) transitivity: for any three nodes u, v, and w, if $Q(u, v) = 1$ and $Q(v, w) = 1$, then $Q(u, w) = 1$.

Extending the basic SimRank and combining with the known information of the pairs of constraints, we define a novel constraint similarity measure composed of three composed three components as follows:

$$s(a,b) = \begin{cases} 1 & \text{if } Q(a,b) = 1 \\ 0 & \text{if } Q(a,b) = -1 \\ \dfrac{C}{|I(a)|\,|I(b)|} \displaystyle\sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s\left(I_i(a), I_j(b)\right) & \text{if } Q(a,b) = 0. \end{cases} \quad (3)$$

In (3), the similarity between two nodes is set to 1 as the max value when the must-link relationship is satisfied and 0 as the min value when cannot-link relationship is satisfied. Note that any node has the must-link relationship with itself and any isolated node with no neighbors has the cannot-link relationship with all the remaining nodes. The similarity between two nonconstraint nodes, similar to the basic SimRank equation, is calculated according to the similarities among their neighbors. The score will be increasing or decreasing when the two non-constraint nodes are linked with more must-link or cannot-link relationships. As a result, the constraint similarity measure considering the prior constraints information can help to improve the clustering result efficiently.

3.3. Calculation of the Constraint Similarity Measure. A solution to the SimRank equations for a network can be reached by iteration to a fix value. Let G be a given network with n nodes. For each iteration k, we store the n^2 elements $R_k(*, *)$, where $R_k(a, b)$ implies the score of the similarity between a and b in the current iteration. We successively calculate $R_{k+1}(*, *)$ using $R_k(*, *)$. Initially, each $R_0(a, b)$ is defined to be

$$R_0(a,b) = \begin{cases} 1, & \text{if } Q(a,b) = 1 \\ 0, & \text{else.} \end{cases} \quad (4)$$

To calculate $R_{k+1}(a, b)$ from $R_k(a, b)$, we use (5), which has the same form as the basic SimRank equation, when there is no constraint between a and b; namely, $Q(a, b) = 0$. On each

iteration $k+1$, $R_{k+1}(a,b)$ keeps 1 when $Q(a,b) = 1$ and 0 when $Q(a,b) = -1$:

$$R_{k+1}(a,b) = \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_k\left(I_i(a), I_j(b)\right). \quad (5)$$

As shown previously, the current score of the similarity between a and b is updated by the score from previous iteration. We argue that this calculation method has the property of convergence.

Proposition 1. *The similarity score calculated by the method introduced before will converge to the true value as the iteration k increases; namely, $s(a,b) = \lim_{k\to\infty} R_k(a,b)$.*

Proof. Certainly, $s(a,b)$ is always keeping the score 1 and 0 when $Q(a,b) = 1$ and $Q(a,b) = -1$, respectively. So, we only need to prove the convergence for $s(a,b)$ when $Q(a,b) = 0$. On the iteration $k = 1$, for any pair of nodes a and b,

$$\begin{aligned} R_1(a,b) - R_0(a,b) &= R_1(a,b) \\ &= \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_0\left(I_i(a), I_j(b)\right), \\ &\qquad R_0\left(I_i(a), I_j(b)\right) \in \{0,1\} \\ &\le \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} |I(a)|\,|I(b)| \\ &= C. \end{aligned} \quad (6)$$

When $k > 1$,

$$\begin{aligned} &R_{k+1}(a,b) - R_k(a,b) \\ &= \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_k\left(I_i(a), I_j(b)\right) \\ &\quad - \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_{k-1}\left(I_i(a), I_j(b)\right) \\ &= \frac{C}{|I(a)|\,|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} \left(R_k\left(I_i(a), I_j(b)\right)\right. \\ &\qquad \left. - R_{k-1}\left(I_i(a), I_j(b)\right)\right) \\ &= \left(\frac{C}{|I(a)|\,|I(b)|}\right)^2 \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} \left(R_{k-1}\left(I_i(a), I_j(b)\right)\right. \\ &\qquad \left. - R_{k-2}\left(I_i(a), I_j(b)\right)\right) \\ &= \left(\frac{C}{|I(a)|\,|I(b)|}\right)^k \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} \cdots \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} \left(R_1\left(I_i(a), I_j(b)\right)\right. \\ &\qquad \left. - R_0\left(I_i(a), I_j(b)\right)\right). \end{aligned} \quad (7)$$

For $\forall i \in I(a)$, $j \in I(b)$, i and j satisfy $0 \le R_1(I_i(a), I_j(b)) - R_0(I_i(a), I_j(b)) \le C$; therefore,

$$\begin{aligned} 0 &\le R_{k+1}(a,b) - R_k(a,b) \\ &\le \left(\frac{C}{|I(a)|\,|I(b)|}\right)^k (|I(a)|\,|I(b)|)^k C^k = C^{k+1}. \end{aligned} \quad (8)$$

Since $C \in (0,1)$, $\lim_{k\to\infty} R_{k+1}(a,b) - R_k(a,b) = 0$. So, it follows from the previous proposition that $R(a,b)$ will converge to $s(a,b)$. □

4. Fast Affinity Propagation Algorithm for Network Clustering

After the similarities are calculated, the following work we should do is to design an efficient and effective clustering method making use of the similarities. Although there are a number of candidate algorithms to be chosen, we adopt the famous affinity propagation algorithm (AP) [28], which has the ability to get a stable and reliable clustering result. Furthermore, we observe that the efficiency of AP can be increased by ingenious design, due to the sparse similarities in networks. So, in this section, we propose a fast affinity propagation algorithm for network clustering.

4.1. Factor Graph. We start from the optimization problem of the factor graph, which is the theoretical basis of the affinity propagation algorithm. Figure 1 is the binary model for affinity propagation demonstrated in [34]. There are two types of nodes representing variables and functions, respectively, in the binary model. Let $C = \{c_{11}, \ldots, c_{nn}\}$ be the set of N^2 binary variables in the graph, such that $c_{ij} = 1$ if the exemplar for node i is node j. In this notion, $c_{jj} = 1$ indicates that j is an exemplar itself. For each variable c_{ij}, there are a similarity function node and two constraint function nodes. The similarity function is written in (9), where the value is equal to the input similarity between nodes i and j if $c_{ij} = 1$. The 1-of-N constraint function I denotes each node in affinity propagation clustering must be assigned to only a single exemplar. The consistency constraint function E means that if a node itself is not an exemplar, it cannot be the exemplar for any other node. The goal of the affinity propagation is to maximize $S(c_{11}, \ldots, c_{nn}) = \sum_{i,j} S_{ij}(c_{ij})$ by finding the optimal assignment for C. Obviously, the number of the variables to be solved is n^2:

$$S_{ij}\left(c_{ij}\right) = \begin{cases} s(i,j) & c_{ij} = 1, \\ 0 & c_{ij} = 0. \end{cases} \quad (9)$$

For semisupervised network clustering, the similarity between two nodes will be set to 0 when they are satisfied with

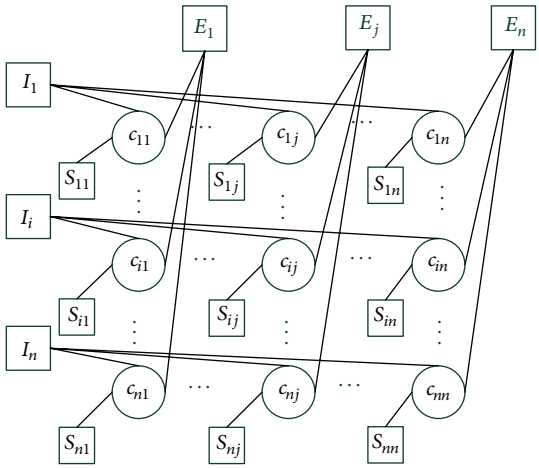


Figure 1: Binary model for factor graph.

cannot-link relationship. In the opposite way, there are lots of pairs of nodes with 0 similarity scores due to the influence both of the cannot-link relationships and the sparse structure of the real networks. Therefore, we argue that the pairs of nodes with 0 similarity scores are also constrained by cannot-link relationships. Formally, for nodes i and j, $s(i, j) = 0 \Leftrightarrow Q(i, j) = 0$. Since the pairs of nodes must be assigned into different clusters, if the similarity score between i and j is 0, then i and j are not represented by each other as the exemplar, such that both c_{ij} and c_{ji} are set to 0. In this way, parts of the binary variables will be set to 0 in the optimum assignment of C, which makes the original factor graph in Figure 1 adjusted.

Firstly, to facilitate the description, we give the definition for the similarity neighborhoods. For a node v, $\Gamma(v)$ is defined as the similarity neighborhoods of v such that $\Gamma(v) = \{w \mid s(v, w) \neq 0\}$; namely, the similarity score between v and any element in $\Gamma(v)$ is greater than 0. In addition, $\Gamma_k(v)$ denotes the kth item in $\Gamma(v)$, and $K_v = |\Gamma(v)|$ denotes the number of the items in $\Gamma(v)$. Then, using the similarity neighborhoods we give the local factor graph for nodes i and j in Figures 2 and 3.

Figure 2 indicates the relationships between the function node I_i and the related binary variable nodes. If node $j \notin \Gamma(i)$, which means $s(i, j) = 0$, the corresponding c_{ij} is set to 0, so i cannot choose j as its exemplar. Otherwise, if $j \in \Gamma(i)$, i is satisfied with the 1-of-N constraint in (10), which means that i can only choose its exemplar in its similarity neighborhoods,

$$I_i\left(c_{ij}\right) = \begin{cases} -\infty & \text{if } \sum_{j\in\Gamma(i)} c_{ij} \neq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Figure 3 indicates the relationships between the function nodes E_j and the related binary variable nodes. If node $i \notin \Gamma(j)$, which means $s(i, j) = 0$, the corresponding c_{ij} is set to 0. Otherwise, if $i \in \Gamma(j)$, then j is satisfied with the consistency constraint in (11), which means that all the nodes in $\Gamma(j)$ cannot choose j as their exemplars if j is not an exemplar itself ($c_{jj} = 0$),

$$E_j\left(c_{ij}\right) = \begin{cases} -\infty & \text{if } c_{jj} = 0, \ \sum_{i\in\Gamma(j)} c_{ij} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

The whole factor graph is integrating all the function nodes I and E like the ones represented in Figures 2 and 3. Each variable node c_{ij} becomes an isolated node without any constraint, and its value is set to 0. As a result, the objective function of this new factor graph is defined to be

$$F\left(c_{ij} \mid s(i,j)\neq 0\right) = \sum_{i,j} S_{ij}\left(c_{ij} \mid s(i,j)\neq 0\right) + \sum_{i} I_i\left(c_{ij} \mid j \in \Gamma(i)\right) + \sum_{j} E_j\left(c_{ij} \mid i \in \Gamma(j)\right). \tag{12}$$

The problem of maximizing (12) can be solved by the max-sum algorithm [35], which passes messages between function nodes and variable nodes in the factor graph. As depicted in Figure 4, there are five message types passed between nonisolated variable nodes and function nodes. The corresponding message update rules are defined as (13) to (17), when executing the max-sum algorithm on the new factor graph,

$$\rho_{ij}\left(i \mid i \in \Gamma(j)\right) = s_{j\in\Gamma(i)}(i, j) + \eta_{j\in\Gamma(i)}(i, j), \tag{13}$$

$$\beta_{ij}\left(j \mid j \in \Gamma(i)\right) = s_{i\in\Gamma(j)}(i, j) + \alpha_{i\in\Gamma(j)}(i, j), \tag{14}$$

$$\eta_{ij}\left(j \mid j \in \Gamma(i)\right) = -\max_{k\neq j \& k\in\Gamma(i)} \beta_{ik}, \tag{15}$$

$$\alpha_{ij}\left(j \mid j \in \Gamma(i)\right) = \begin{cases} \sum_{k\neq j \& k\in\Gamma(j)} \max\left[\rho_{kj}, 0\right] & i = j \\ \min\left[0, \rho_{jj} + \sum_{k\notin\{i,j\} \& k\in\Gamma(j)} \max\left[\rho_{kj}, 0\right]\right] & i \neq j. \end{cases} \tag{16}$$

We eliminate β and η, while only keeping α and ρ. As a result, α keeps the formation as (16) being only dependent on ρ, while ρ is calculated by (17) being only dependent on α,

$$\rho_{ij}\left(i \mid i \in \Gamma(j)\right) = s(i, j) - \max_{k\neq j \& k\in\Gamma(i)} \left(s(i, k) + \alpha_{ik}\right). \tag{17}$$

Then, the optimal solution of the objective function can be obtained by updating α and ρ alternately until the convergence. We can see that the number of messages passed is reduced on the factor graph due to the decrease of the elements of the variable nodes to be assigned. Therefore the time efficiency will be increasing for the objective optimizing.

4.2. Clustering Algorithm. As the analysis in the previous factor graph, we extend the original affinity propagation, an exemplar-based algorithm, into a fast version for network clustering. There are some basic parameters.

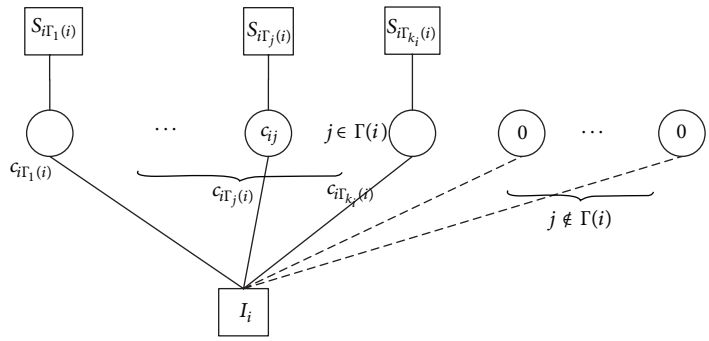


Figure 2: The relationships between function node I_i and the related binary variable nodes.

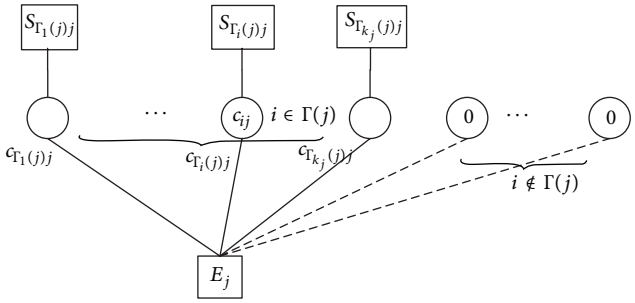


Figure 3: The relationships between function node E_j and the related binary variable nodes.

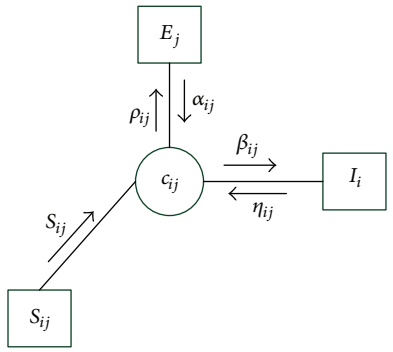


Figure 4: Messages passing between function nodes and variable nodes.

Responsibility $r(i, j)$, equivalent to ρ_{ij} in the factor graph, represents the evidence for how well-suited node j is to serve as the exemplar for node i.

Availability $a(i, j)$, equivalent to α_{ij} in the factor graph, represents the evidence for how appropriate it would be for node i to choose node j as its exemplar.

Preference p, the initialized value $s(i, i)$ for every node, represents the tendency of node i being chosen as an exemplar. At the beginning of the algorithm, all nodes share the same preference, which controls the number of the final clusters. The larger value of the preference results in a larger number of clusters.

Damping factor λ is used to smooth the responsibility and the availability so as to avoid numerical oscillations. In our algorithm, λ is set to 0.8. The smoothed responsibility and the availability are calculated by (18) and (19),

$$\begin{aligned} a(i, j \mid j \in \Gamma(i))^{(t)} &= (1-\lambda)\, a(i, j \mid j \in \Gamma(i))^{(t)} \\ &\quad + \lambda a(i, j \mid j \in \Gamma(i))^{(t-1)}, \end{aligned} \tag{18}$$

$$\begin{aligned} r(i, j \mid j \in \Gamma(i))^{(t)} &= (1-\lambda)\, r(i, j \mid j \in \Gamma(i))^{(t)} \\ &\quad + \lambda r(i, j \mid j \in \Gamma(i))^{(t-1)}. \end{aligned} \tag{19}$$

According to the parameters defined before, for a given network $G = (V, E)$ and the constraint matrix Q, the basic process of our algorithm is as follows.

Step 1. Combined with the constraint matrix Q, calculate the similarities for all pairs of nodes in G.

Step 2. Construct the similarity neighborhoods for each node.

Step 3. Initialize the preference p and the maximum iteration number; for each node i and node j in the similarity neighborhoods of i, set $r(i, j) = a(i, j) = 0$.

Step 4. For each node i and node j in the similarity neighborhoods of i, update $a(i, j)$ by (16) and (18); update $r(i, j)$ by (17) and (19).

Step 5. For each node i, calculate its exemplar by (20):

$$c^* = \arg \max_{j \in \Gamma(i)} \{r(i, j) + a(i, j)\}. \tag{20}$$

Step 6. When the exemplars for all nodes are not changed any more, or the iteration reaches the maximum value, the algorithm will be terminated, and the nodes with the same exemplar will be partitioned into the same cluster. Otherwise, go to Step 4.

5. Complexity Analysis

In this section, we present an analysis of the computation complexity of the proposed algorithm. Given a network with n nodes, the algorithm is composed of two parts: the constraint similarities calculation and the fast affinity propagation clustering. We analyze the computation complexity of the two parts, respectively, in the following.

For the constraint similarities calculation, if there is no any constraint condition, the running cost is $O(k_s d N^2)$, where k_s is the number of iterations ($k_s \ll N$) of the calculation and d is the average degree of all nodes in the network ($d \ll N$ in many real networks). If some constraint information is obtained, the similarities between the pairs of nodes satisfied with constraint conditions need not to be calculated. This is because the value will be fixed to 1 and 0 for the must-link and cannot-link pairs all the time. Assuming the number of pairwise constraints is C, the total cost of the constraint similarities calculation is $O(k_s d(N - C)^2)$.

For the fast affinity propagation algorithm, the worst case is that the similarities between all pairs of nodes are greater than 0. In this situation, the time cost is $O(k_A N^2)$, where k_A is the number of iterations of the algorithm. However, the edges are sparse in many real networks, which results in the fact that the similarity scores are 0 between many pairs of nodes. The fast affinity propagation algorithm does not take into account the messages passing between the pairs of nodes provided with 0 similarity scores. As a result, let M ($M < N^2$) represent the number of pairs of nodes provided with similarity scores greater than 0. Then, the total cost of the fast affinity propagation algorithm is $O(k_A M)$.

6. Evaluation

In this section, we evaluate the performance of the proposed semisupervised network clustering algorithm based on fast affinity propagation (SCAN-FAP). We conduct all the experiments on a Pentium Core2 Duo 2.8 GHz PC with 2 GBytes of main memory, running on Windows 7. We implement our algorithm in C++, using Microsoft Visual Studio 2008.

6.1. Dataset. We use six real network datasets to evaluate the performance of SCAN-FAP. Table 1 lists the information about these networks.

Table 1: Information about the experimental datasets.

Networks	Nodes	Edges	Clusters	References
Football	115	616	12	[36]
Polbooks	105	441	3	[36]
Adjnoun	112	425	2	[36]
PolBlogs	1490	19090	2	[36]
Cora	2708	5429	7	[37, 38]
Citeseer	3312	4732	6	[37, 38]

6.2. Effectiveness

6.2.1. Evaluation Criteria. Since the underlying class labels of all the datasets are already known, we adopt the Normalized Mutual Information (NMI) [25, 32] and the F-Measure Score [17] in our experiments to evaluate the quality of the clusters generated by various methods.

NMI is currently widely used in measuring the performance of network clustering algorithms. Formally, the measurement of NMI can be defined as follows:

$$\mathrm{NMI} = \frac{-2\sum_{i=1}^{r}\sum_{j=1}^{k} N_{ij}\log\left(N_{ij}N/N_{i.}N_{.j}\right)}{\sum_{i=1}^{r} N_{i.}\log\left(N_{i.}/n\right) + \sum_{j=1}^{k} N_{.j}\log\left(N_{.j}/N\right)}, \quad (21)$$

where N is the confusion matrix, N_{ij} is the number of nodes both in the ith class and the jth cluster, r and k are the number of classes and clusters, respectively, and $N_{i.}$ and $N_{.j}$ are the number of nodes in the ith class and the jth cluster, respectively.

F-Measure Score is another commonly used measure for evaluating clustering algorithms. Assume T is the set of node pairs (i, j) where nodes i and j belong to the same classes in the ground truth and S is the set of node pairs that belong to the same clusters generated by an algorithm. Then the F-Measure Score is computed from both the *precision* and the *recall* synthetically:

$$F\text{-Measure} = \frac{2 \times \mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}, \quad (22)$$

where *precision* and *recall* are written as follows (23):

$$\mathrm{precision} = \frac{|S \cap T|}{|S|}, \qquad \mathrm{recall} = \frac{|S \cap T|}{|T|}. \quad (23)$$

6.2.2. Experimental Methods. We compare SCAN-FAP with the other four representative semisupervised clustering algorithms, which are briefly described in the following.

SS-SP, a semisupervised spectral clustering algorithm [22], uses the known constraint relationships to modify the adjacent matrix of the original network. If the nodes i and j satisfy the must-link constraint, A_{ij} is set to 1; if i and j satisfy the cannot-link relationship, A_{ij} is set to 0. SS-SP will then execute the conventional spectral clustering algorithm on the modified adjacent matrix.

SS-NCut-KK [25] is a semisupervised kernel k-means algorithm based on Normalized Cut. In this algorithm, the original kernel matrix and the penalty term with constraint relationships are combined to construct a new kernel matrix, such that the wrong cluster assignments will go against the objective function. The kernel matrix is defined as $K = \sigma I - L + W$, where σ is a constant large enough to ensure that K is positive definite, I is the identity matrix, L is the corresponding Laplacian matrix, and W is the matrix of the penalty term. SS-NCut-KK executes the kernel k-means algorithm using the matrix K.

SS-DM-KK [26] is another semisupervised kernel k-means algorithm, which is similar to SS-NCut-KK. It takes

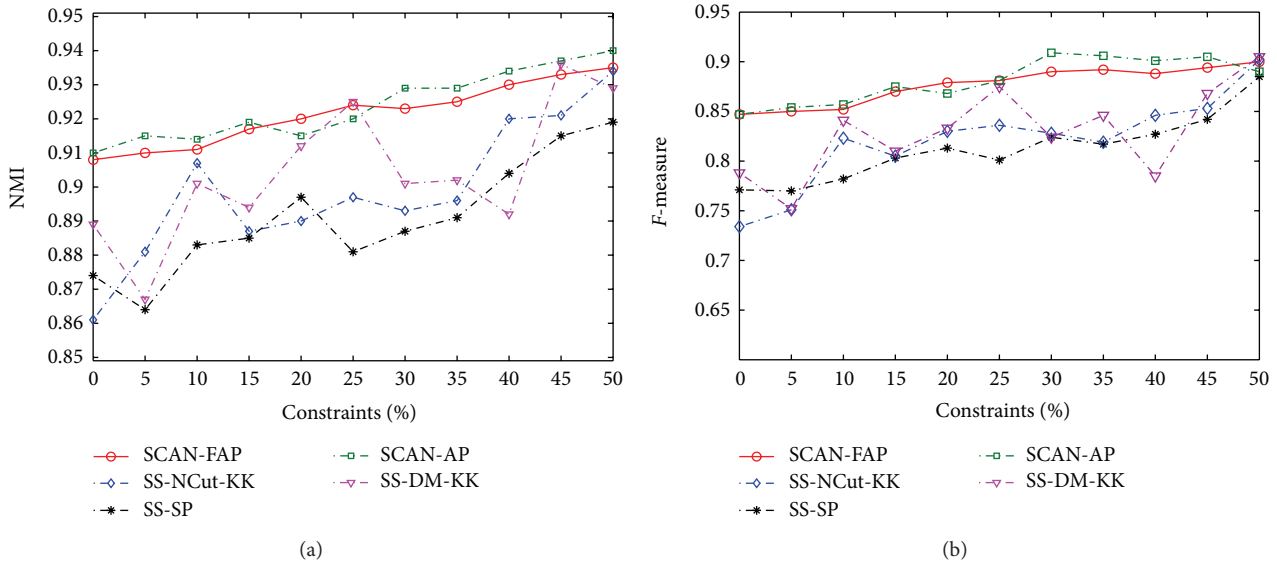


Figure 5: Clustering results on Football.

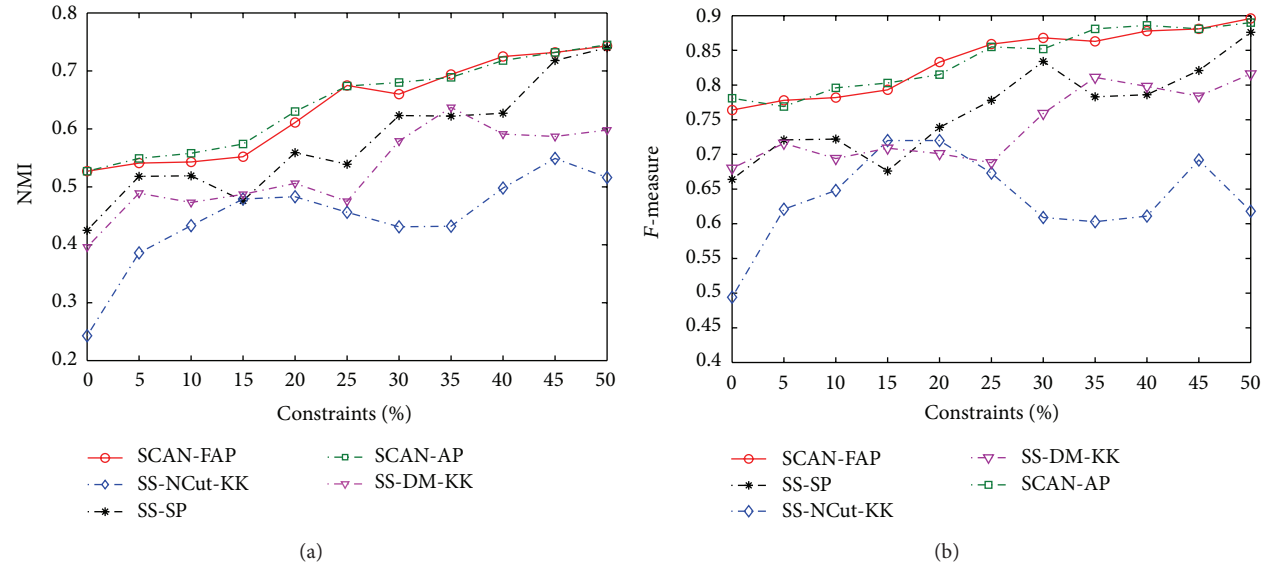


Figure 6: Clustering results on Polbooks.

the density modularity [33] integrating the constraint information as its objective function, which is the only difference from SS-NCut-KK.

SCAN-AP is a semisupervised network clustering algorithm most related to our work. The original affinity propagation algorithm is used taking as input the constraint similarities between all the nodes in the network.

We select pairwise constraints randomly. The proportion of the pairwise constraints to all pairs in the network is set to 5%, 10%, ..., 50%. Due to the randomness of the pairwise constraints, the clustering result of each experiment is the average value generated by running the corresponding algorithm over 20 times. The number of the clusters is set to the number of real class labels in the networks. Since the number of clusters generated by SCAN-AP and SCAN-FAP depends on the value of the preference parameter p, we choose different value of p to adjust the number of clusters to the ground truth.

6.2.3. Experimental Results. Figures 5, 6, 7, 8, 9, and 10 demonstrate the curves of the clustering results on the six real datasets generated by different algorithms, where the horizontal coordinates represent the proportion of pairwise constraints and the vertical coordinates represent the NMI and F-Measure scores, respectively. We can get some observations as follows.

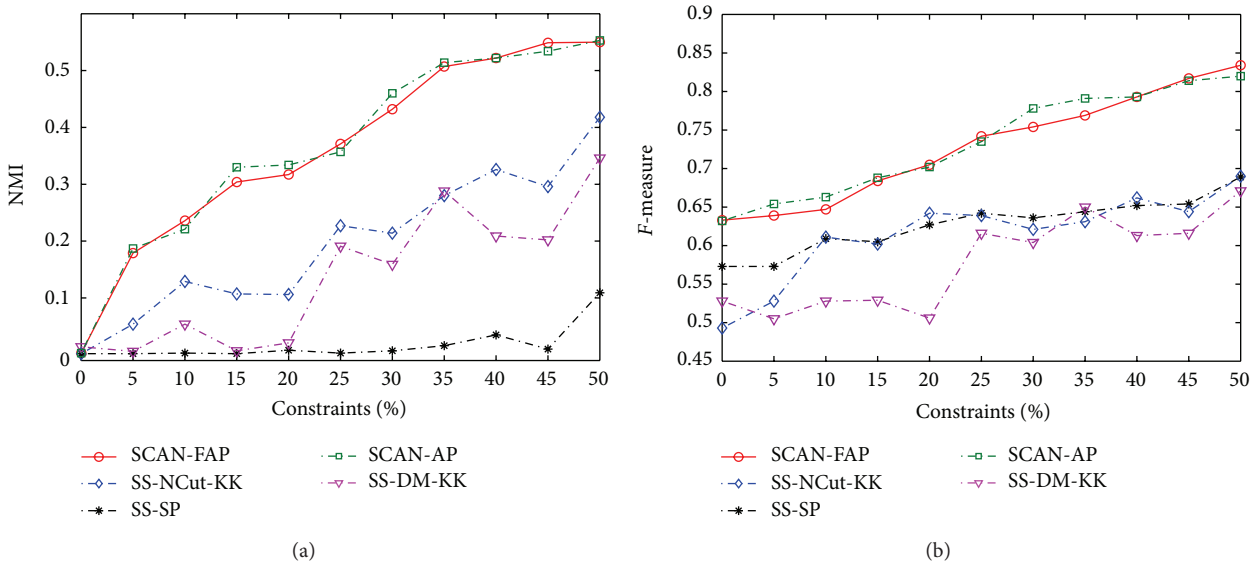


Figure 7: Clustering results on Adjnoun.

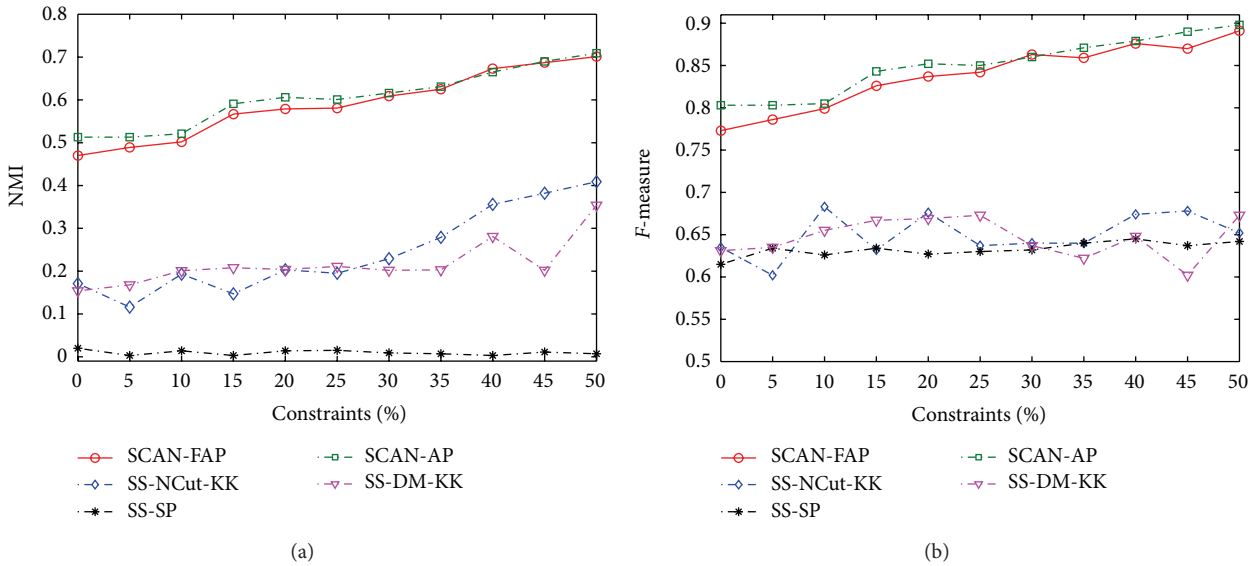


Figure 8: Clustering results on Polblogs.

Firstly, we compare all the algorithms in their unsupervised condition, which is reflected in the curves that the horizontal coordinates are 0. We can observe that SCAN-FAP and SCAN-AP get higher quality results on all the networks except Adjnoun than the other three algorithms. This indicates that SimRank is a good measure for similarities between nodes in networks. Note that the NMI of the results on Adjnoun obtained by all algorithms is close to 0, which implies that only the unsupervised structural information is insufficient for the clusters.

Then, we compare each semisupervised algorithm with its unsupervised version. When the pairwise constraints are provided, we can see from both NMI and *F*-Measure scores that the clustering results of SCAN-FAP and SCAN-AP are better than the ones without any pairwise constraints provided. So, it can be considered that the proposed constraint similarity measure can take full advantage of the prior knowledge to guide the clustering process so as to improve the quality of the clustering results. For the other algorithms, SS-NCut-KK and SS-DM-KK can improve the quality of clustering in most cases, but SS-SP fails to utilize the pairwise constraints on Adjnoun and PolBlogs.

We also compare the performance of SCAN-FAP with other semisupervised clustering algorithms except SCAN-AP. The NMI and *F*-Measure curves of SCAN-FAP can display ascending trend together with increasing the pairwise constraints. However, the SS-SP, SS-NCut-KK, and SS-DM-KK show occasionally fluctuant or unascending states, which

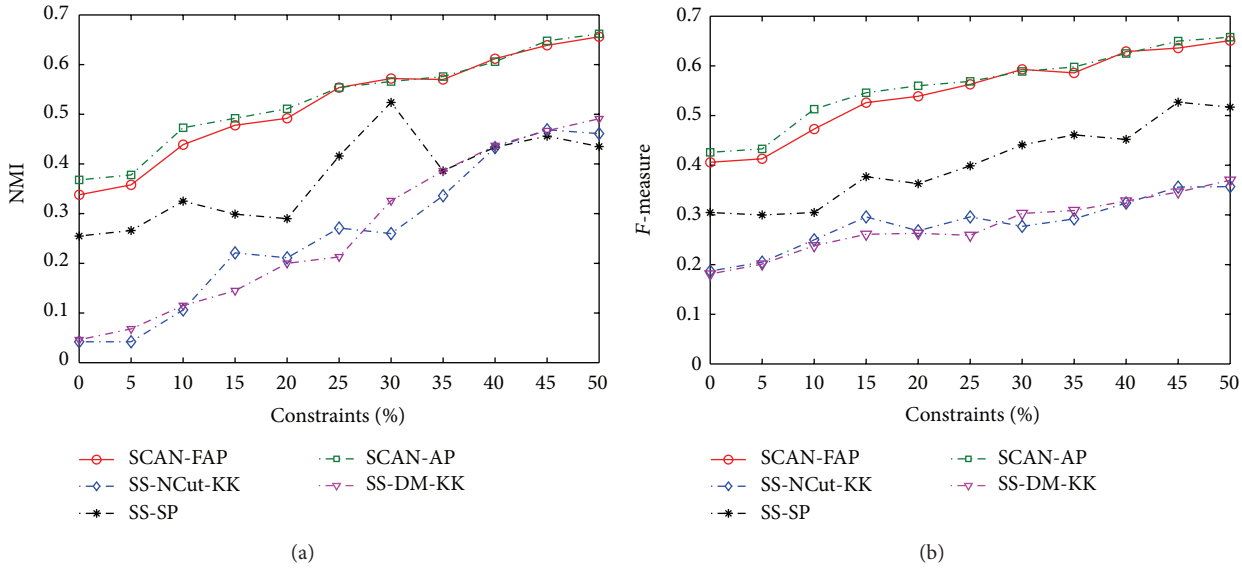


Figure 9: Clustering results on Cora.

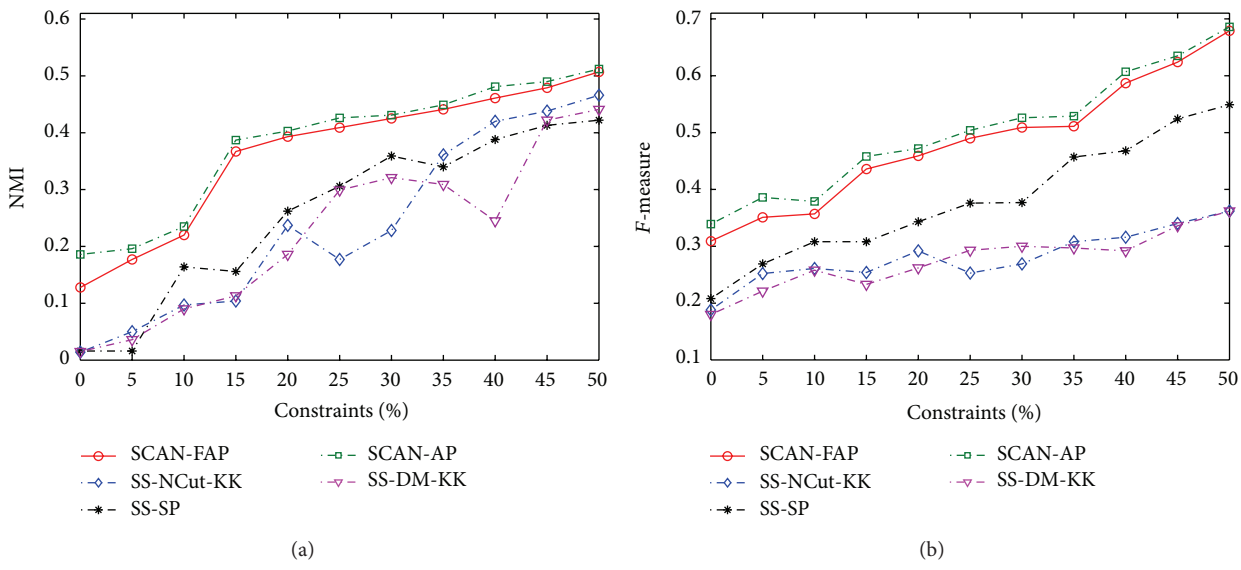
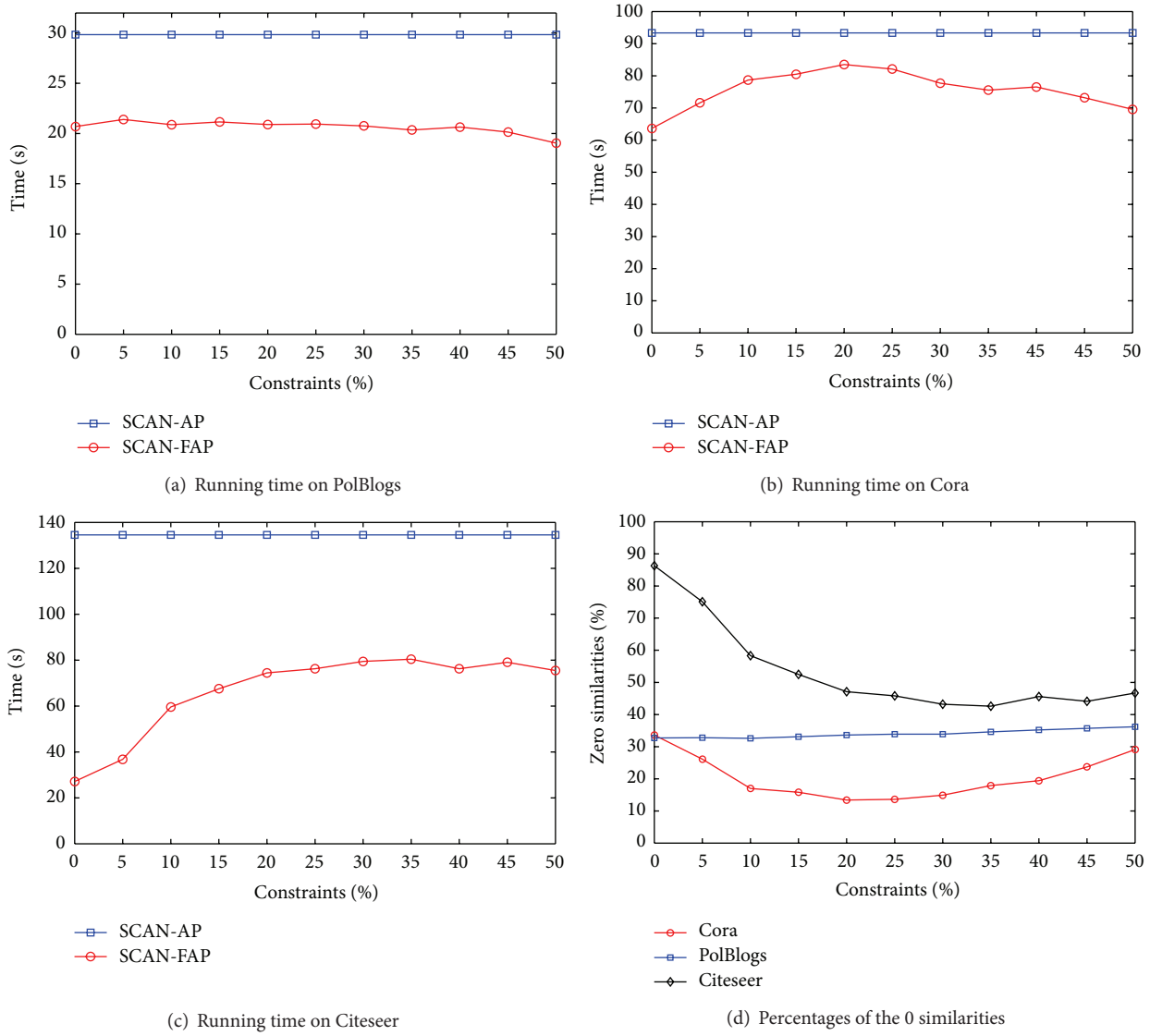


Figure 10: Clustering results on Citeseer.

reduces their reliability. For example, SS-SP on political blogs appears invalid with any percentage of constraints, while SS-NCut-KK and SS-DM-KK on Citeseer reveal decreased NMI with 25% and 40% constraints provided, respectively, and so on. One of the reasons for analyzing the instability of the three algorithms is their random initializations. For SS-SP, we should initialize the cluster centers when the k-means algorithm runs on the data composed of the eigenvectors and for SS-NCut-KK and SS-DM-KK, we should initialize the cluster assignment for each node.

Finally, we compare the performance of SCAN-FAP with its similar algorithm SCAN-AP. Meaningfully, the clustering quality of SCAN-FAP is always very close to SCAN-AP, which confirms the effectiveness of the messages passed only in the pairs of nodes with similarity scores greater than 0. We demonstrate that SCAN-FAP can take the place of SCAN-AP under keeping the clustering quality. At the same time, the time efficiency is much increased, which will be demonstrated next.

6.3. Efficiency. From the effectiveness evaluations, we can see that SCAN-FAP and SCAN-AP have the ability of getting clusters with high quality. They only need to be executed one time in real applications due to the stability of them. However, the other three methods have to be run many times and average the results, since they are influenced by some factors such as the initializations. Therefore, we only compare the time efficiency of SCAN-FAP and SCAN-AP. In addition,



(a) Running time on PolBlogs

(b) Running time on Cora

(c) Running time on Citeseer

(d) Percentages of the 0 similarities

Figure 11: Efficiency comparison between SCAN-FAP and SCAN-AP.

because both the two algorithms share the same module that calculates the constraint similarities, we only take the running time of the clustering process for comparisons. The datasets are adopted by the three relatively large networks: PolBlogs, Cora, and Citeseer.

Figure 11 demonstrates the results about the efficiency comparisons, where (a), (b), and (c) are the running time of the two algorithms on three datasets, respectively, and (d) illustrates the proportion of the number of the pairwise nodes with 0 similarity scores to all the pairs in each network. We can observe that SCAN-AP keeps the running time consistently, since the messages are passed between all the pairs of nodes. SCAN-FAP can increase the time efficiency obviously, and the degree of the increasing is inversely proportional to the ratio of the pairs of nodes with 0 similarity scores. This is because each node only chooses the exemplar from its similarity neighborhoods when using fast affinity propagation algorithm in SCAN-FAP.

7. Conclusions

In this paper, we propose a novel semisupervised clustering algorithm for networks based on fast affinity propagation, which can utilize the prior knowledge to guide the unsupervised clustering process. The algorithm firstly defines a constraint similarity measure, which takes the pairwise constraints information to extend the basic SimRank measure. The similarity scores calculated with the new measure are influenced by the must-link and cannot-link relationships, such that the semisupervised property is achieved. Then, taking as input the constraint similarities, a fast affinity propagation algorithm is proposed to partition the networks. This algorithm maintains the advantages of the original affinity propagation algorithm. Moreover, it increases the running efficiency by passing messages between only the nodes with similarity scores greater than 0. Experimental studies, by various kinds of real networks, demonstrate that

our algorithm has a stronger ability of making use of a small number of prior knowledge than the other representative methods, to improve the clustering quality effectively and efficiently.

References

[1] M. Coscia, F. Giannotti, and D. Pedreschi, "A classification for community discovery methods in complex networks," *Statistical Analysis and Data Mining*, vol. 4, no. 5, pp. 512–546, 2011.

[2] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, Article ID 026113, 2004.

[3] J. Lee, S. P. Gross, and J. Lee, "Modularity optimization by conformational space annealing," *Physical Review E*, vol. 85, no. 5, Article ID 056702, 5 pages, 2012.

[4] X. Liu and T. Murata, "Advanced modularity-specialized label propagation algorithm for detecting communities in networks," *Physica A*, vol. 389, no. 7, pp. 1493–1500, 2010.

[5] R. Shang, J. Bai, L. Jiao, and C. Jin, "Community detection based on modularity and an improved genetic algorithm," *Physica A*, vol. 392, no. 5, pp. 1215–1231, 2013.

[6] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[7] Y.-C. Wei and C.-K. Cheng, "Ratio cut partitioning for hierarchical designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 7, pp. 911–921, 1991.

[8] J. B. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[9] X. W. Xu, N. Yuruk, Z. D. Feng, and T. A. J. Schweiger, "SCAN: a structural clustering algorithm for networks," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*, pp. 824–833, ACM, August 2007.

[10] H.-W. Liu, "Community detection by affinity propagation with various similarity measures," in *Proceedings of the 4th International Joint Conference on Computational Sciences and Optimization (CSO '11)*, pp. 182–186, April 2011.

[11] H. Cheng, Y. Zhou, X. Huang, and J. X. Yu, "Clustering large attributed information networks: an efficient incremental computing approach," *Data Mining and Knowledge Discovery*, vol. 25, no. 3, pp. 450–477, 2012.

[12] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl et al., "Constrained K-means clustering with background knowledge," in *Proceedings of the 18th International Conference on Machine Learning (ICML '01)*, pp. 577–584, Morgan Kaufmann Publishers, 2001.

[13] K. Wagstaff and C. Cardie, "Clustering with instance-level constraints," in *Proceedings of the 17th International Conference on Machine Learning (ICML '00)*, pp. 1103–1110, Morgan Kaufmann Publishers, 2000.

[14] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semisupervised clustering," in *Proceedings of the 21st International Conference on Machine Learning (ICML '04)*, pp. 81–88, ACM, July 2004.

[15] D. Klein, S. D. Kamvar, and C. D. Manning, "From instance-level constraints to space-level constraints: making the most of prior knowledge in data clustering," in *Proceedings of the 19th International Conference on Machine Learning (ICML '02)*, pp. 307–314, Morgan Kaufmann Publishers, 2002.

[16] W. Tang, H. Xiong, S. Zhong, and J. Wu, "Enhancing semisupervised clustering: a feature projection perspective," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*, pp. 707–716, ACM, August 2007.

[17] Y. Xiao and J. Yu, "Semisupervised clustering based on affinity propagation algorithm," *Journal of Software*, vol. 19, no. 11, pp. 2803–2813, 2008.

[18] C. Ruiz, M. Spiliopoulou, and E. Menasalvas, "C-DBSCAN: density-Based clustering with constraints," in *Proceedings of the 11th Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, pp. 216–223, Springer, 2007.

[19] I. E. Givoni and B. J. Frey, "Semisupervised Affinity Propagation with Instance-Level Constraints," in *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pp. 161–168, JMLR, 2009.

[20] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning Distance Functions using Equivalence Relations," in *Proceedings of the 20th International Conference on Machine Learning*, pp. 11–18, ACM, August 2003.

[21] S. Basu, M. Bilenko, and R. J. Mooney, "A probabilistic framework for semisupervised clustering," in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04)*, pp. 59–68, August 2004.

[22] S. D. Kamvar, D. Klein, and C. Manning, "Spectral learning," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pp. 561–566, ACM, 2003.

[23] X. Wang and I. Davidson, "Flexible constrained spectral clustering," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*, pp. 563–572, ACM, July 2010.

[24] W. F. Chen and G. C. Feng, "Spectral clustering: a semisupervised approach," *Neurocomputing*, vol. 77, no. 1, pp. 229–242, 2012.

[25] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semisupervised graph clustering: a kernel approach," in *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*, pp. 457–464, ACM, August 2005.

[26] X. K. Ma, L. Gao, X. R. Yong, and L. Fu, "Semisupervised clustering algorithm for community structure detection in complex networks," *Physica A*, vol. 389, no. 1, pp. 187–197, 2010.

[27] G. Jeh and J. Widom, "SimRank: a measure of structural-context similarity," in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*, pp. 538–543, ACM, July 2002.

[28] B. F. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 305, no. 5814, pp. 972–976, 2007.

[29] A. Lancichinetti and S. Fortunato, "Limits of modularity maximization in community detection," *Physical Review E*, vol. 84, no. 6, Article ID 066122, 2011.

[30] S. White and P. Smyth, "A spectral clustering approach to finding communities in graphs," in *Proceedings of SIAM Conference on Data Mining (SDM '05)*, pp. 76–84, 2005.

[31] J. Q. Jiang, A. W. M. Dress, and G. Yang, "A spectral clustering-based framework for detecting community structures in complex networks," *Applied Mathematics Letters*, vol. 22, no. 9, pp. 1479–1482, 2009.

[32] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means, spectral clustering and normalized cuts," in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04)*, pp. 551–556, ACM, August 2004.

[33] Z. P. Li, S. H. Zhang, R. S. Wang, X.-S. Zhang, and L. Chen, "Quantitative function for community detection," *Physical Review E*, vol. 77, no. 3, Article ID 036109, 9 pages, 2008.

[34] I. E. Givoni and B. J. Frey, "A binary variable model for affinity propagation," *Neural Computation*, vol. 21, no. 6, pp. 1589–1600, 2009.

[35] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, NY, USA, 2006.

[36] http://www-personal.umich.edu/~mejn/netdata/.

[37] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008.

[38] http://www.cs.umd.edu/~sen/lbc-proj/LBC.html.