ARTICLE IN PRESS

Information Sciences xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Greedy discrete particle swarm optimization for large-scale social network clustering

Qing Cai, Maoguo Gong *, Lijia Ma, Shasha Ruan, Fuyan Yuan, Licheng Jiao

Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, International Research Center for Intelligent Perception and Computation, Xidian University, Xi'an, Shaanxi Province 710071, China

ARTICLE INFO

Article history:
Received 19 November 2013
Received in revised form 12 September 2014
Accepted 25 September 2014
Available online xxxx

Keywords:
Social network
Clustering
Community structure
Particle swarm optimization
Large-scale global optimization

ABSTRACT

Social computing is a new paradigm for information and communication technology. Social network analysis is one of the theoretical underpinnings of social computing. Community structure detection is believed to be an effective tool for social network analysis. Uncovering community structures in social networks can be regarded as clustering optimization problems. Because social networks are characterized by dynamics and huge volumes of data, conventional nature-inspired optimization algorithms encounter serious challenges when applied to solving large-scale social network clustering optimization problems. In this study, we put forward a novel particle swarm optimization algorithm to reveal community structures in social networks. The particle statuses are redefined under a discrete scenario. The status updating rules are reconsidered based on the network topology. A greedy strategy is designed to drive particles to a promising region. To this end, a greedy discrete particle swarm optimization framework for large-scale social network clustering is suggested. To determine the performance of the algorithm, extensive experiments on both synthetic and real-world social networks are carried out. We also compare the proposed algorithm with several state-of-the-art network community clustering methods. The experimental results demonstrate that the proposed method is effective and promising for social network clustering.

1. Introduction

With the rapid development of the Internet and the Web 2.0 technologies, diverse types of social media networks, such as Blog, Wiki, Facebook, Twitter, and Weixin, have emerged and are changing the fundamental methods through which people share information and communicate. It is universally acknowledged that social media are profoundly affecting not only the global economy but also every aspect of our daily lives. Social media data are characterized by large data volumes, dynamics, interactivity and heterogeneity, which makes conventional computing models difficult to utilize. Social media data analysis challenges computing techniques in an unprecedented manner.

In 2009, scholars from the fields of social science, physical science, information science, etc., cooperated in an iconic work in [22], and since then, a new academic term has come into existence: social computing. Social computing, a new computing paradigm, is a cross-disciplinary research area integrating the computational and social sciences. According to Wikipedia,

* Corresponding author.
E-mail address: gong@ieee.org (M. Gong).

http://dx.doi.org/10.1016/j.ins.2014.09.041
0020-0255/

social computing refers to the use of social software, the result of a growing trend in information and communication technology usage of tools that support social interaction and communication. Another definition can be found in [46]. Social computing aims at making use of computing techniques to help people to communicate and cooperate and to help to understand how a society operates and functions so as to direct decision making. The primary task for social computing is the model construction. Currently, the mainstream model is the social network model in which nodes denote the social objects, and edges represent social interactions among them. From this point of view, social computing largely depends on the analysis of the constructed social networks. Social networks have numerous features, among which the community structure is an eminent one. In academic domains, communities, also called clusters or modules, are groups of vertices that most likely share common properties and/or play similar roles within the graph. Probing the community structures of networks can help us to understand how a network functions.

The discovery of community structures in networks can be considered as an optimization problem [29]. In the past few decades, numerous nature-inspired optimization algorithms characterized by good local learning and by global searching abilities have gathered momentum through both theoretical and empirical studies. Evolutionary algorithms (EAs) are some representative optimization paradigms among the nature-inspired optimization algorithms. EAs have been developed and successfully applied to a wide range of optimization problems, including network clustering optimization. Recently, scholars have successfully applied both single- and multi-objective EAs to discover community structures in networks [32,33,13]. In addition to the EA-based metaheuristic optimization techniques, another class is swarm-intelligence-based avenues, among which the outstanding paradigm is particle swarm optimization (PSO) [18]. PSO originated from the behavior of social animals, such as fish schooling and birds flocking. PSO optimizes a problem by employing a group of particles. Each particle is a candidate solution to the problem. The candidate solutions are updated with simple rules learnt by the particles. Due to its efficacy and its extremely easy implementation, PSO is prevalent in the optimization field, and diverse variants have been proposed [43,28,42]. However, canonical PSO is specially designed for continuous optimization problems. Although some discrete successors have emerged in the literature, their applications are still limited.

Note that due to the huge volume of data sets, social network clustering optimization is a large-scale global optimization (LSGO) problem. LSGO problems refer to optimization problems that involve a large number of decision variables. LSGO is difficult for existing optimization techniques. Studies on scaling up nature-inspired optimization algorithms to solve LSGO problems have attracted much attention [53,47,50,23]. In this paper, a discrete PSO algorithm for large scale social network clustering is put forward for the first time. The main highlights of the proposed algorithm are as follows. First, in order to handle the LSGO network clustering problem, the particles' velocity and position have been carefully redefined under discrete context so as to make them as easier as possible to encode/decode. Second, for the sake of better exhaustive global searching in the vast searching space, to drive the particles to promising regions, the particle-status-update principles have been thoroughly reconsidered by taking the advantage of the network topology. Third, to avoid being trapped into local optima, a greedy strategy specially designed for the particles to adjust their positions is newly suggested. The proposed algorithm is denoted by GDPSO. To check the performance of GDPSO, experiments on computer-generated and real-world social networks are done. On the real-world networks, we test GDPSO on both small scale and large scale networks. We also compare GDPSO with several state-of-the-art network clustering methods. The experimental results show that the proposed GDPSO algorithm is very effective and promising.

The remainder of this paper is organized as follows. Section 2 illustrates the related background and gives the motivation for this work. In Section 3, the proposed GDPSO framework is presented in detail. The designs of the particle representation and the update principles are described. Section 4 shows the performance testing of the proposed method, and the concluding remarks are subsequently summarized in Section 5.

2. Related background

2.1. Network clustering

Social networks are constructed based on social behaviors and relations. In graph theory, a social network can be expressed by a graph that is composed of nodes and edges, as shown in Fig. 1a. Social network clustering separates the whole network into small parts, within which the similarities are maximized and between which the dissimilarities are maximized, as in the toy model shown in Fig. 1b. These small parts are called communities in the literature.

Based on the node degree, Radicchi et al. in [34] gave a qualitative definition of a community. Let $G = (V, E)$ denote a network, where V and E are the aggregations of vertices and links, respectively. Let $k_i^{in} = \sum_{i,j \in S} A_{ij}$ and $k_i^{out} = \sum_{i \in S, j \notin S} A_{ij}$ be the internal and external degree of node i, A is the adjacency matrix of G, and $S \subset G$ is the subgraph that node i belongs to. Then, S is a community in a strong sense if $\forall i \in S,\ k_i^{in} > k_i^{out}$ and in a weak sense if $\sum_{i \in S} k_i^{in} > \sum_{i \in S} k_i^{out}$.

Thus far, a large number of clustering methods have been developed to discover communities in networks. These clustering methods can be categorized into three classes: (1) hierarchical clustering methods [20,15], which iteratively merge or split clusters according to vertex similarities; (2) partitional clustering approaches, such as the k-means [8] and fuzzy k-means methods [52]; and (3) spectral clustering methods [30,49], which use the spectra information of the adjacency matrix to partition networks. A recent survey on network community detection can be found in [9].

Please cite this article in press as: Q. Cai et al., Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.09.041

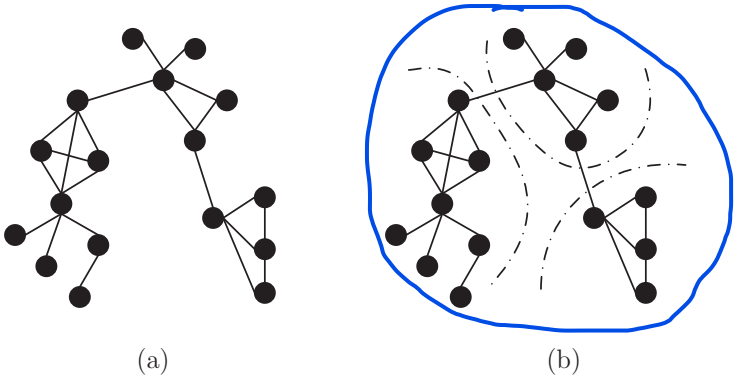


Fig. 1. Graphical illustration of a (a) graph-modeled social network and (b) social network clustering.

2.2. Particle swarm optimization

PSO is a population-based stochastic searching algorithm. It was proposed by Eberhart and Kennedy in 1995 [18]. PSO originated from the social behavior seen in, for example, fish schooling and birds flocking. The easy implementation, concise framework and fast computational convergence make PSO a popular optimization technique for solving continuous optimization problems [39,41,6,45,24].

PSO works with a group of individuals, which are typically called "particles". In PSO, each particle has a position and a velocity vector. The position vector usually simulates a candidate solution to the optimized problem, and the velocity vector denotes the position-changing tendency. To search for the optimal solution, a particle iteratively updates its flying velocity and current position according to its own flying experience and according to the other particles' flying experiences.

Assume that the particle swarm size is *pop*. Let $\boldsymbol{V}_i = \{v_i^1, v_i^2, \ldots, v_i^D\}$ and $\boldsymbol{X}_i = \{x_i^1, x_i^2, \ldots, x_i^D\}$ be the i-th ($i = 1, 2, \ldots, pop$) particle's velocity vector and position vector, respectively. Then, in canonical PSO, a particle adjusts its velocity and position according to the following simple rules:

$$\boldsymbol{V}_i = \boldsymbol{V}_i + c_1 r_1 (\boldsymbol{Pbest}_i - \boldsymbol{X}_i) + c_2 r_2 (\boldsymbol{Gbest} - \boldsymbol{X}_i) \quad (1)$$

$$\boldsymbol{X}_i = \boldsymbol{X}_i + \boldsymbol{V}_i \quad (2)$$

In the above equation, $\boldsymbol{Pbest}_i = \{pbest_i^1, pbest_i^2, \ldots, pbest_i^D\}$ and $\boldsymbol{Gbest} = \{gbest^1, gbest^2, \ldots, gbest^D\}$ are the i-th particle's personal best position and the global best position of the swarm, respectively; the parameters r_1 and r_2 are random numbers between 0 and 1; and c_1 and c_2 are acceleration coefficients termed as cognitive and social components.

Many researchers have worked on improving the swarm's performance in various ways [16]. Shi and Eberhart [39] first introduced an inertia weight ω to the velocity-updating rule as follows:

$$\boldsymbol{V}_i = \omega \boldsymbol{V}_i + c_1 r_1 (\boldsymbol{Pbest}_i - \boldsymbol{X}_i) + c_2 r_2 (\boldsymbol{Gbest} - \boldsymbol{X}_i) \quad (3)$$

The authors argue that a relatively large ω is better for exploration, whereas a small ω enhances exploitation. To balance the global and local search abilities, they suggested a linearly decreasing inertia weight factor in [40]. They also designed fuzzy methods to adjust ω nonlinearly [41]. For more information about the inertia weight, please refer to [54,44].

Canonical PSO was designed for continuous optimization; however, its fast convergence and easy implementation have driven scholars to extend continuous PSO to discrete scenarios. The first trial was the binary PSO (BPSO) algorithm proposed by Kennedy and Eberhart [19], which was based on the binary coding scheme. Several successors of BPSO using different strategies can be found in [31,1]. Although BPSO exhibits good performance for some benchmark instances, the binary coding scheme still limits its application. The direct method of transforming PSO into DPSO utilizes space transformation techniques such as those in [36,37]; however, the mainstream is the particle redefinition DPSO approaches, such as thefuzzy-matrix-based [25], swap-operator-based [27] and crisp-set-based [4] approaches. In [11], we redefined the position and velocity of a particle under the scenario of network topology. Combined with a decomposition strategy and some turbulence operators, a multi-objective discrete PSO algorithm was proposed. Extensive experiments demonstrate the effectiveness of the suggested method.

2.3. Motivation

PSO demonstrates its excellent abilities in solving some optimization problems; however, canonical PSO was originally designed for continuous optimization. Designing a proper PSO framework for discrete contexts and enhancing its abilities of addressing LSGO problems are still challenging. The main contribution of this work is the proposed GDPSO algorithm for mining community structures in large-scale social networks. In GDPSO, all the algorithm components have been carefully designed to fit the discrete large-scale network clustering optimization problem. In our previous work in [11], we suggested a multi-objective DPSO (MODPSO) framework for network clustering. The main difference between GDPSO and MODPSO lies

Please cite this article in press as: Q. Cai et al., Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.09.041

in the design of the particle-status-update principles. MODPSO emphasizes the network topology; therefore, it updates a particle using network linkage information. GDPSO lays more emphasis on the algorithm itself; thus, it updates a particle using a greedy mechanism illustrated in SubSection 3.3. For the single-objective LSGO network problem, experiments demonstrate that our newly designed greedy-mechanism-based update rule is more effective.

3. The proposed algorithm for network clustering

The proposed GDPSO algorithm for social network clustering makes use of the network topology to direct particle status updates. The greedy strategy is designed to guide particles to a promising region. Some small operators, such as heuristic-based initialization and position reordering, are introduced to speed up convergence. This section will describe the proposed algorithm in detail. The whole framework of the proposed GDPSO algorithm for social network clustering is given in Algorithm 1.

Algorithm 1. Framework of the proposed GDPSO algorithm.

Parameters: particle swarm size *popsize*, number of iterations *gmax*, inertia weight ω, learning factors c_1 and c_2;
Input: network adjacency matrix A;
Output: best fitness, community structure of the network;
1: **Step 1)** Initialize the population: initialize position vectors $pop[].X$; initialize velocity vectors $pop[].V = 0$; initialize the *Pbest* vectors $Pbest[].X = pop[].X$;
2: **Step 2)** Evaluate particles' fitness $pop[].fit$;
3: **Step 3)** Update the *Gbest* particle: $Gbest = pop[best].X$;
4: **Step 4)** Set $t = 0$;
5: **Step 5)** Update particle statuses, see SubSection 3.3 for more information;
6: **Step 6)** Resorting particle positions: $resorting(pop[].X)$, see SubSection 3.4 for more information;
7: **Step 7)** Evaluate particles' fitness $pop[].fit$;
8: **Step 8)** Update the *Pbest* particles: if $pop[].fit > Pbest[].fit$, then $Pbest[].X = pop[].X$;
9: **Step 9)** Update the *Gbest* particle: $Gbest = pop[best].X$;
10: **Step 10)** If $t < gmax$, then $t++$ and go to **Step 5)**; otherwise, stop the algorithm and output.

3.1. Fitness function

The determination of an evaluation criterion is an important issue because it directly affects the final results. In this paper, the adopted fitness function is the widely used modularity (normally denoted as Q) first proposed by Newman in [10]. The fitness function can be written as

$$fit(\cdot) = Q = \frac{1}{2m}\sum_{i,j}^{n}\left(A_{ij} - \frac{k_i \cdot k_j}{2m}\right)\delta(i,j)$$

where n and m are the number of nodes and edges of a network, respectively; k_i is the degree of node i and $\delta(i,j) = 1$ if the nodes i and j are in the same group; otherwise, $\delta(i,j) = 0$. Modularity has been shown to be an effective metric in evaluating the goodness of a partition. Normally, we assume that the larger the value of Q is, the better the partition is.

3.2. Particle representation and initialization

To make the proposed GDPSO algorithm feasible for the network clustering problem, we redefine the terms "position" and "velocity" under the discrete scenario.

Definition 1 (*Position*). The position vector represents a partition of a network. The position permutation of the particle i is defined as $\boldsymbol{X}_i = \{x_i^1, x_i^2, \ldots, x_i^n\}$, where $x_i^j \in [1, n]$ is an integer.

In the above definition, x_i^j is called a label identifier, which carries the cluster information. If $x_i^j = x_i^k$, then nodes j and k belong to the same cluster. A graphical illustration of the particle representation is shown in Fig. 2.

It can be observed from Fig. 2 that the above discrete position definition is straightforward and easy to decode and will reduce the computational complexity, especially in the presence of large-scale networks because the dimensions of the fitness function are the same as the number of nodes in the networks.

Definition 2 (*Velocity*). The velocity permutation of particle i is defined as $\boldsymbol{V}_i = \{v_i^1, v_i^2, \ldots, v_i^n\}$, where $v_i^j \in \{0, 1\}$. If $v_i^j = 1$, then the corresponding element x_i^j in the position vector will be changed; otherwise, x_i^j maintains its original state.

Please cite this article in press as: Q. Cai et al., Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.09.041

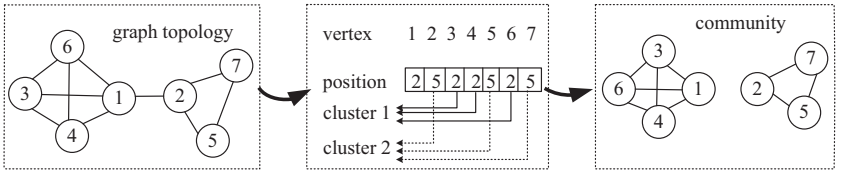


Fig. 2. A schematic illustration of the particle representation.

In the canonical version of PSO, there is a threshold V_{max} that is used to inhibit particles from flying apart because there exists a situation whereby when the speed of a particle is substantial, it will fly out of the boundaries. To define the velocity permutation in the above style, we no longer need this parameter, which is hard to tune.

In the initialization step, the position vectors are initialized using a heuristic method introduced in our previous work in [12]. The velocity vectors are initialized as all-zero vectors. The *pbest* vectors are initialized in the same manner as the position vectors, and the *gbest* vector is set as the best position vector in the original population.

3.3. Particle-status-updating rules

In the proposed GDPSO algorithm, the particle position and velocity vectors have been redefined in a discrete form; thus, the mathematical operators in the canonical version of PSO no longer fit the discrete context. In this paper, the mathematical operators have been redefined as follows:

$$\boldsymbol{V}_i = \omega \boldsymbol{V}_i \oplus (c_1 r_1 \times (\boldsymbol{Pbest}_i \ominus \boldsymbol{X}_i) + c_2 r_2 \times (\boldsymbol{Gbest}_i \ominus \boldsymbol{X}_i)) \tag{4}$$

$$\boldsymbol{X}_i = \boldsymbol{X}_i \circledast \boldsymbol{V}_i \tag{5}$$

In our algorithm, the inertia weight ω and the learning factors c_1 and c_2 are set using typical values of 0.7298, 1.4961 and 1.4961. It can be seen that the above equations have the same format as in canonical PSO but that the key components are different. In the next step, these components will be illustrated in detail.

Definition 3 (*Position* $\ominus$ *Position*). Assume that we are given two position permutations $\boldsymbol{P}_1 = \{p_1^1, p_1^2, \ldots, p_1^n\}$ and $\boldsymbol{P}_2 = \{p_2^1, p_2^2, \ldots, p_2^n\}$. Position $\ominus$ Position is a velocity vector, i.e., $\boldsymbol{P}_1 \ominus \boldsymbol{P}_2 = \boldsymbol{V} = \{v_1, v_2, \ldots, v_n\}$. The element v_i is defined as

$$\begin{cases} v_i = 0 & if \quad p_1^i = p_2^i \\ v_i = 1 & if \quad p_1^i \neq p_2^i \end{cases} \tag{6}$$

The inspiration of the above defined operator comes from two aspects. First, from the perspective of swarm intelligence, a particle will adjust its velocity by learning from its neighbors. The learning process is actually a comparison between the positions; in other words, two position vectors generate a velocity vector. Second, from the viewpoint of graph theory, two position vectors represent two types of network community structures. The defined $\ominus$ operation actually reflects the difference between two network structures.

Definition 4 (*Coefficient* $\times$ *Velocity*). The operator $\times$ is the same as the basic arithmetical multiplication operator. For instance, given a coefficient $c \cdot r = 1.3$ and given a velocity vector $\boldsymbol{V} = \{1, 0, 1, 1, 0\}$, $c \cdot r \times \boldsymbol{V} = \{1.3, 0, 1.3, 1.3, 0\}$.

Definition 5 (*Velocity* $\oplus$ *Velocity*). Velocity $\oplus$ Velocity equals a velocity as well. Given two velocity vectors $\boldsymbol{V}_1 = \{v_1^1, v_1^2, \ldots, v_1^n\}$ and $\boldsymbol{V}_2 = \{v_2^1, v_2^2, \ldots, v_2^n\}$, $\boldsymbol{V}_1 + \boldsymbol{V}_2 = \boldsymbol{V}_3 = \{v_3^1, v_3^2, \ldots, v_3^n\}$. The element v_3^i is defined as

$$\begin{cases} v_3^i = 1 & if \quad v_1^i + v_2^i \geqslant 1 \\ v_3^i = 0 & if \quad v_1^i + v_2^i < 1 \end{cases} \tag{7}$$

The definition of the operator $\oplus$ is straight forward, and the operation is easy to perform. Moreover, it can always make sure that the velocity is binary coded, which is easier for the position to work with.

Definition 6 (*Position* $\circledast$ *Velocity*). The operator $\circledast$ is the key component. A particle will update its position according to a new velocity, i.e., Position $\circledast$ Velocity generates a new position. A good operator $\circledast$ should drive a particle to a promising region. Given a position $\boldsymbol{P}_{old} = \{p_{old}^1, p_{old}^2, \ldots, p_{old}^n\}$ and a velocity $\boldsymbol{V} = \{v_1, v_2, \ldots, v_n\}$, $\boldsymbol{P}_{old} \circledast \boldsymbol{V} = \boldsymbol{P}_{new} = \{p_{new}^1, p_{new}^2, \ldots, p_{new}^n\}$. The element of $\boldsymbol{P}_{new}$ is defined as follows:

$$\begin{cases} p_{new}^i = p_{old}^i \ if \ v_i = 0 \\ p_{new}^i = \arg\max_j \Delta Q(p_{old}^i, j | j \in L_i) \ if \ v_i = 1 \end{cases} \tag{8}$$

where $\boldsymbol{L}_i = \{l_1, l_2, \ldots, l_k\}$ is the set of label identifiers of vertex i's neighbors. The ΔQ is calculated using the following equation:

Please cite this article in press as: Q. Cai et al., Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.09.041

$$\Delta Q(p^i_{old}, j | j \in L_i) = fit(\boldsymbol{P}_{old} | p^i_{old} \leftarrow j) - fit(\boldsymbol{P}_{old}) \quad (9)$$

Eq. (8) can be regarded as a greedy local search strategy because a particle updates its position by choosing the label identifier that can generate the largest fitness increment. In our previous work in [11], we updated the position by choosing the dominated label identifier (identifier owned by the majority of the neighbor vertices of a given node). Through our experiments, we find that the greedy local search update rule defined in Eq. (8) works better for single-objective optimization. For more information, please refer to Section 4.4. A graphical exhibition of the $\circledast$ operation is shown in Fig. 3a. Fig. 3b gives a simple illustration of how a particle updates its status.

In Fig. 3b, X_i represents the current particle's position vector, V_1 and V_2 are intermediate velocity vectors determined using Eq. (6), and V_i is the new velocity calculated using Eq. (7). Under the guidance of the velocity V_i, the current particle updates its current position X_i using Eq. (8), and thus, a new position X_j is obtained.

It can be observed from the above description that (1) the proposed GDPSO algorithm has a concise framework, (2) the newly defined particle representation is direct and easy to decode, and (3) the redefined updating rules are easy to realize. All these merits make this advanced algorithm capable of addressing large-scale networks.

3.4. Particle position reordering

In the proposed algorithm, to avoid unnecessary computing, a particle position reordering operator is designed. Given that $X = \{x_1, x_2, \ldots, x_n\}$ is a position vector of a particle, the particle position reordering operator acts on X and outputs a new vector X'. The operator reorders the elements in X with a starting value of 1. The pseudocode of the reordering operator is given in Algorithm 2.

Algorithm 2. Pseudocode of the reordering operator.

Input: an integer permutation $X = \{x_1, x_2, \ldots, x_n\}$;
Output: reordered permutation $X' = \{x'_1, x'_2, \ldots, x'_n\}$;
1: set $counter = 1$, $X' \leftarrow X$;
2: **for** $i = 1; i \leqslant n; i++$ **do**
3: **if** $x'_i \neq -1$ **then**
4: **for** $j = i + 1; j \leqslant n; j++$ **do**
5: **if** $x'_j = x'_i$ **then**
6: $x_j = counter, x'_j = -1$;
7: **end if**
8: **end for**
9: $x'_i = -1, x_i = counter, counter++$;
10: **end if**
11: **end for**
12: $X' \leftarrow X$;

Fig. 4 gives a graphical illustration of the reordering operator. As illustrated in Fig. 4, *p1* and *p2* are structure-equivalent (they correspond to the same partition). If we do not design the resorting operation, then according to Eq. (6), non-zero velocity vector would be obtained, and according to Eq. (8), a new position would need to be calculated, which would require computing time. However, if we design the operation so that after resorting, *p1* and *p2* are the same, then according to Eq. (6), the obtained velocity will be a zero vector; thus, it would not need to calculate the new position, resulting in reduced computing time.

4. Experimental study

In this section, we test our proposed algorithm on both synthetic networks and on real-world networks. We also compare our algorithm with several state-of-the-art community detection algorithms: GA [32], MOGA [33], LPA [2], CNM [5] and Informap [35]. To enable a fair comparison, in the experiments, the objective function used in the GA is replaced by the modularity, and we use the modularity to choose the ultimate solution from the Pareto front for the MOGA. The population size and the maximum number of iterations for the GDPSO, GA and MOGA are set to 100, the crossover and mutation probabilities for the GA and the MOGA are set to 0.9 and 0.1, respectively. The LPA and Informap are two iterative methods; the maximum number of iteration is set to 5 and 100, respectively. In addition, we also perform a comparison with two canonical data clustering methods: the *k*-means method proposed in [17] and the normalized cut (Ncut) approach proposed in [38]. Our algorithm is coded in C++, and the experiments were performed on a 3.2 GHz Inter(R) Core(TM) i3 CPU 550 machine with 4 GB memory. The operating system is MS Windows XP, and the compiler is VC++ 6.0.

Please cite this article in press as: Q. Cai et al., Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.09.041

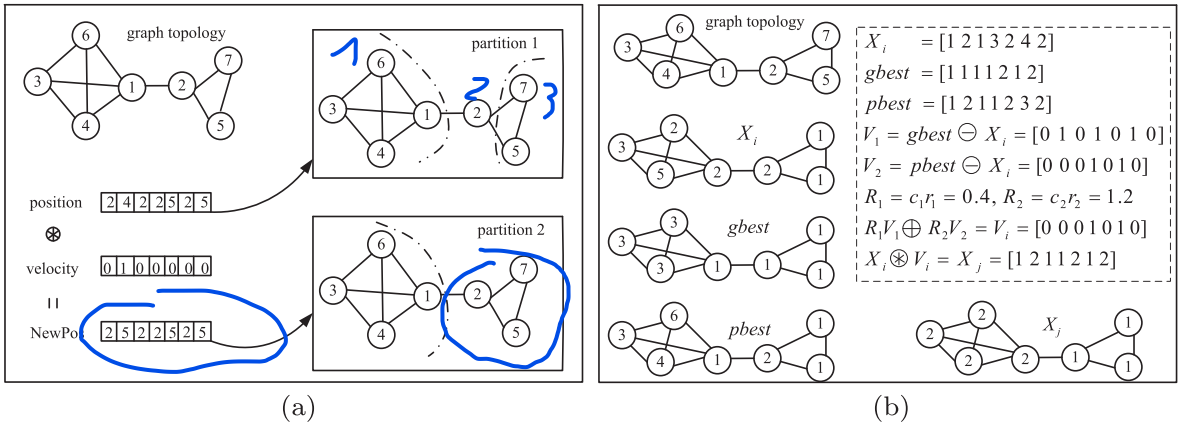


Fig. 3. A schematic illustration of (a) the $\circledast$ operator and (b) the particle-status-updating rules.

4.1. Performance metric

In addition to the modularity index, for the case where the ground truth of a network is known, we adopt the *Normalized Mutual Information* (*NMI*) [7] metric to estimate the similarity between the true community structures and the discovered ones.

Given that A and B are two partitions of a network, C is a confusion matrix. C_{ij} equals the number of nodes shared by community i in partition A and by community j in partition B. Then, *NMI(A,B)* is written as

$$I = \frac{-2\sum_{i=1}^{C_A}\sum_{j=1}^{C_B} C_{ij} log(C_{ij} \cdot n / C_{i.} C_{.j})}{\sum_{i=1}^{C_A} C_{i.} log(C_{i.}/n) + \sum_{j=1}^{C_B} C_{.j} log(C_{.j}/N)} \tag{10}$$

where C_A (or C_B) is the number of clusters in partition A (or B) and $C_{i.}$ (or $C_{.j}$) is the sum of the elements of C in row i (or in column j).

4.2. Results on synthetic networks

We first perform some experiments on the benchmark network proposed by Lancichinetti et al. [21]. The benchmark network is divided into four communities. Each community has 32 nodes. The average degree of each node is 16. Every node shares a fraction $1-\gamma$ of links with the rest of its community and γ with the other nodes of the network. Here, γ is called the mixing parameter. The mixing parameter controls the portion of links within a community and outside of it. We test all the algorithms on 11 computer-generated networks with values of γ ranging from 0.0 to 0.5. Fig. 5 shows the statistical results averaged over 30 runs for different algorithms when the mixing parameter γ increases from 0.0 to 0.5 at intervals of 0.05.

As shown in Fig. 5, when the mixing parameter is no larger than 0.15, all the methods, except the MOGA and k-means methods, can determine the true community structures of the networks ($NMI = 1$). As the mixing parameter increases, the GA, LPA and CNM fail to discover the correct partitions. For $\gamma = 0.4$, our proposed method and Ncut still obtain $NMI = 1$, whereas the other methods cannot. When γ is larger than 0.4, from the curve, it can be observed that the *NMI* values quickly decrease and that our method fails to detect the ground truths. One reason that can account for this is that when γ is larger than 0.4, the community structure of the network is rather vague, that is, we can say that there is no community structure. Moreover, optimizing the modularity has been proven to be NP hard. Therefore, it is very hard for an optimization method to uncover any communities under this scenario.

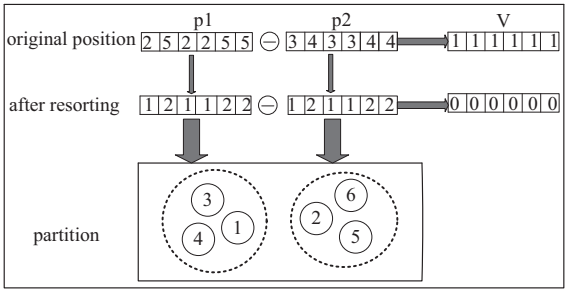


Fig. 4. A schematic illustration of the reordering operation.

Please cite this article in press as: Q. Cai et al., Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.09.041

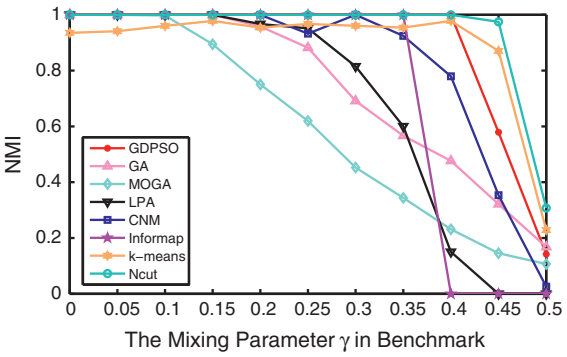


Fig. 5. The experimental results from the benchmark networks.

The experiments on the computer-generated benchmark networks prove that the proposed GDPSO framework for network clustering is feasible. The proposed algorithm is capable of uncovering community structures in social networks. Although from the curves, we may note that Ncut seems to perform the best; however, Ncut needs to specify the clusters in advance, which is very inconvenient.

4.3. Results on real-world networks

In this section, we test the performance of the GDPSO on four small-scale networks and on four large-scale networks. The parameters of each network are listed in Table 1. For each social network, we run each of the Algorithms 30 times.

We first test the algorithms on the four small-scale networks. We also perform a statistical analysis using the Wilcoxon rank sum test. In the right-tailed rank sum test experiments, the significance level is set to 0.05 with the null hypothesis that the two independent samples (the two modularity sets) come from distributions with equal means. Tables 2 and 3 show the experimental results and the hypothesis test p-values.

From the two tables, we can note that our proposed GDPSO algorithm performs remarkably well in terms of the modularity values and the computational time. The small p-values indicate that our proposed algorithm is substantially better than the compared methods. Our proposed GDPSO framework possesses excellent global exploration ability. On the four small-scale networks, our proposed algorithm outperforms the other methods, except for the GA. The GA optimizes the same objective as does the GDPSO. From the experiments, we find that the GA and the GDPSO perform well, but the GDPSO converges faster. In the next step, we will analyze the community structures discovered by the GDPSO on the four small-scale networks.

The Karate network is a network of relations between 34 members of a karate club. Fig. 6 shows the real community structure and the detected structure of the network. It can be observed that our algorithm has discovered four clusters, which are the subdivisions of the real ones. This phenomenon also occurs in the dolphin network. Fig. 7 clearly displays the discovered structure of the dolphin network. Our algorithm divides one of the two clusters in the real structure into four smaller sections. We cannot determine if this division makes sense to the dolphins; from the perspective of optimization, our method simply finds the best objective function value.

The football network represents American football games between Division IA colleges during the regular season in the Fall of 2000. In this network, although our algorithm obtains the largest Q value, through our analysis, we find that several vertices, such as the numbers 29, 43, 37, 81, 60, 91 and 63, are misclassified. This is mainly caused by the nuances in the scheduling of games.

The SFI network represents 271 scientists and their collaborators in residence at the Santa Fe Institute during any part of calendar years 1999 or 2000. Fig. 8 exhibits the discovered communities in the network. From the figure, we notice that the GDPSO splits the network into eight strong communities, with divisions running principally along disciplinary lines. We have reason to believe that the discovered communities are meaningful. The sub-communities that are subdivisions of the original three large-scale groups are centered around the interests of leading members.

Experiments on the four small-scale social networks indicate that the proposed GDPSO algorithm is effective. The algorithm exhibits an outstanding search ability when addressing moderate-scale optimization problems. To further verify its optimization ability, we next test the algorithm on four large-scale networks. Tables 4 and 5 list the statistical results.

The rank sum test results indicate that our proposed algorithm is superior to the GA, MOGA, LPA and k-means algorithm for the four large networks and that it outperforms Informap for the four big networks, except for the e-mail network. The CNM and Informap methods are two deterministic methods. For the four large networks, the CNM seems to perform the best from the angle of modularity, LPA is the fastest, and the k-means algorithm performs poorly. The MOGA, k-means and Ncut methods can hardly handle large-scale networks. For the PGP network, the MOGA cannot provide output after four hours,

Please cite this article in press as: Q. Cai et al., Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.09.041

Table 1
Networks parameters.

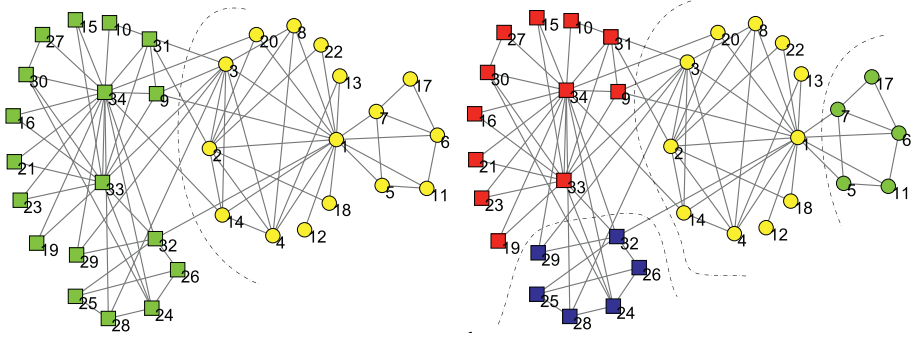
Network	#Node	#Edge	Weighted	Directed	#Clusters	Reference
Karate	34	78	No	No	2	[51]
Dolphin	62	159	No	No	2	[26]
Football	115	613	Yes	No	12	[10]
SFI	118	200	Yes	No	Unknown	[10]
E-mail	1133	5451	No	No	Unknown	[14]
Netscience	1589	2742	Yes	No	Unknown	[30]
Power grid	4941	6594	No	No	Unknown	[48]
PGP	10680	24340	No	No	Unknown	[3]

Table 2
Experimental results on the Karate and dolphin networks. The parameter k used in k-means and Ncut is set to 4 for the Karate network and 5 for the dolphin network.

Network index	Karate					Dolphin				
	Q_{max}	Q_{avg}	T_{avg}	I_{avg}	p-Value	Q_{max}	Q_{avg}	T_{avg}	I_{avg}	p-Value
GDPSO	0.4198	0.4198	2.5800E−2	0.6881	×	0.5285	0.5284	5.8621E−2	0.8935	×
GA	0.4198	0.4198	1.9871E−1	0.6873	0.5000	0.5285	0.5280	2.3868	0.5892	0.1591
MOGA	0.4198	0.4160	1.3975	1	0.0000	0.5085	0.4098	3.1713	0.9442	0.0000
LPA	0.4151	0.3264	3.3133E−3	0.6623	0.0000	0.5258	0.4964	3.8011E−3	0.6194	0.0000
CNM	0.3800	0.3800	5.0700E−2	0.6920	0.0000	0.4950	0.4950	2.2517E−2	0.5730	0.0000
Informap	0.4020	0.4020	2.1331E−1	0.6995	0.0000	0.5247	0.5247	3.7121E−1	0.4662	0.0000
k-Means	0.1429	0.0351	2.6121E−2	0.6059	0.0000	0.4796	0.3787	4.5717E−2	0.4282	0.0000
Ncut	0.4198	0.4198	4.2272E−3	0.6873	0.5000	0.5068	0.5047	8.2015E−3	0.5084	0.0000

Table 3
The experimental results on the football and SFI networks. The parameter k used in k-means and in Ncut is set to 12 for the football network and to 7 for the SFI network.

Network index	Football					SFI			
	Q_{max}	Q_{avg}	T_{avg}	I_{avg}	p-Value	Q_{max}	Q_{avg}	T_{avg}	p-Value
GDPSO	0.6046	0.6041	9.4210E−2	0.8889	×	0.7506	0.7449	4.5217E−2	×
GA	0.5929	0.5829	4.9276	0.8227	0.0000	0.7506	0.7505	6.8536	1.0000
MOGA	0.5280	0.5173	4.1731	0.7883	0.0000	0.7430	0.7323	1.9742	0.0000
LPA	0.6030	0.5848	8.2001E−3	0.8735	0.0000	0.7341	0.7095	5.3731E−3	0.0000
CNM	0.5770	0.5770	1.1292E−1	0.7620	0.0000	0.7335	0.7335	4.5661E−2	0.0000
Informap	0.6005	0.6005	1.2133	0.9242	0.0000	0.7334	0.7334	7.7563E−1	0.0000
k-Means	0.5783	0.5130	5.4805E−2	0.8512	0.0000	0.4376	0.2780	3.4512E−2	0.0000
Ncut	0.6031	0.6007	8.8133E−3	0.9233	0.0000	0.7478	0.7470	9.5000E−3	0.9990



(a) $NMI = 1$ (b) $Q = 0.4198$

Fig. 6. Community structure of the Karate network. (a) Ground truth. (b) Detected structure.

Please cite this article in press as: Q. Cai et al., Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.09.041

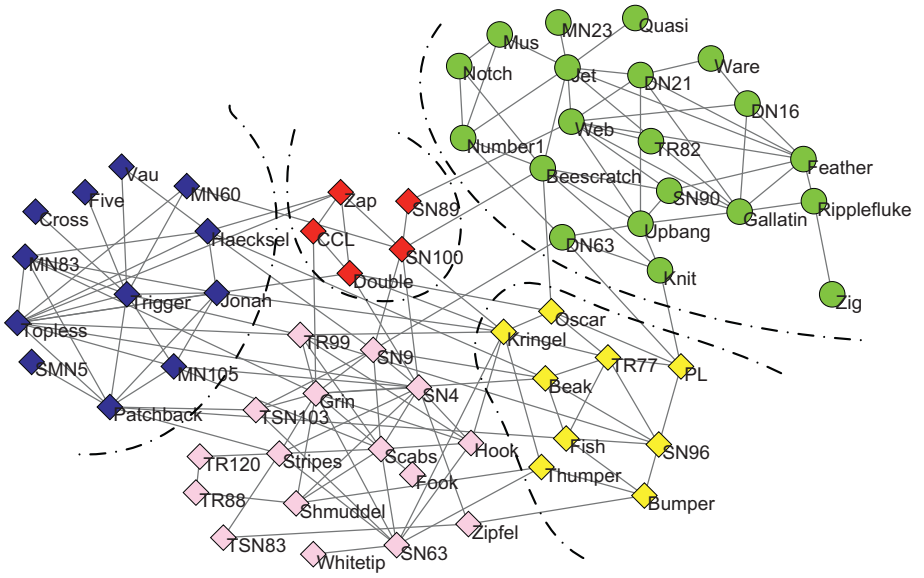


Fig. 7. Our detected community structure of the dolphin network ($Q = 0.5285$).

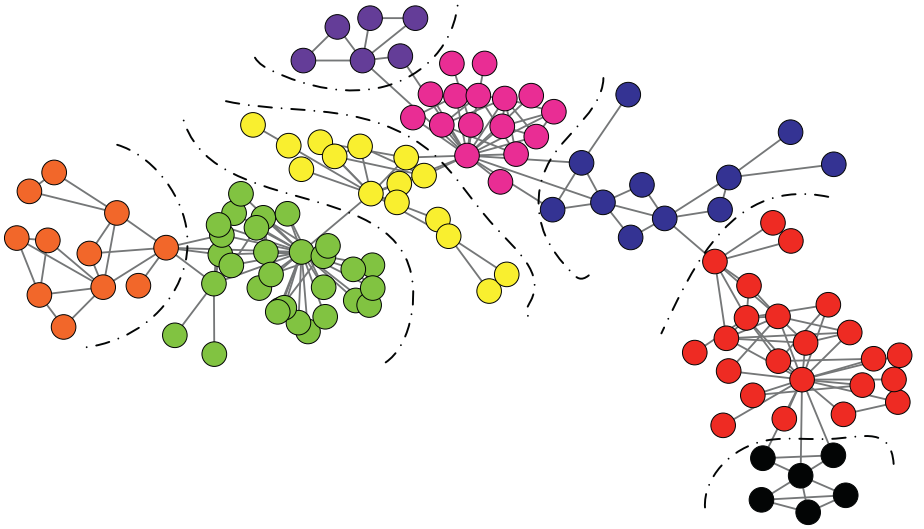


Fig. 8. Community structure of the SFI network ($Q = 0.7506$).

and the *k*-means and Ncut methods run out of computer memory. Through our proposed algorithm, we can obtain good modularity values within a reasonable amount of time. From the tables, it can be observed that our method still falls into local optimum solutions; for example, for the netscience and power grid networks, the modularity values obtained by the GDPSO are smaller than those obtained by the CNM. One reason that can account for this is the designed greedy-mechanism-based particle-position-update principle possibly causing prematurity. In the next subsection, we will further discuss the particle-position-update principle.

4.4. Additional discussion on GDPSO

For the proposed GDPSO algorithm, the population size and the maximum number of iterations affect the performance of the algorithm. Tables 6 and 7 show the influence of these two parameters on the performance of the algorithm.

The two parameters *gmax* and *popsize* are normally set empirically. Small values of the two parameters may not result in convergence, whereas large values will require substantial amounts of computing time. From the above experiments, as shown in Table 6 and in 7, we observe that the impact of the two parameters on the performance of the algorithm is not

Please cite this article in press as: Q. Cai et al., Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.09.041

Table 4
The experimental results for the e-mail and netscience networks. The parameter k used in k-means and Ncut is set to 4 for the e-mail network and to 100 for the netscience network.

Network index	E-mail				Netscience			
	Q_{max}	Q_{avg}	T_{avg}	p-Value	Q_{max}	Q_{avg}	T_{avg}	p-Value
GDPSO	0.5487	0.4783	2.4717E+1	×	0.9540	0.9512	3.5212E+1	×
GA	0.3647	0.3500	3.1015E+2	0.0000	0.9086	0.9003	1.9967E+2	0.0000
MOGA	0.3283	0.3037	5.3716E+2	0.0000	0.8916	0.8810	7.0681E+1	0.0000
LPA	0.2055	0.0070	6.5033E−2	0.0000	0.9266	0.9202	3.9117E−2	0.0000
CNM	0.4985	0.4985	1.5255E+1	1.0000	0.9555	0.9555	2.0414E+1	1.0000
Informap	0.5355	0.5355	2.3200	1.0000	0.9252	0.9252	1.5577	0.0000
k-Means	0.3681	0.3600	5.2580	0.0000	0.6510	0.6359	5.6187E+1	0.0000
Ncut	0.4841	0.4749	4.7000E−2	0.2384	0.9293	0.9268	3.9302E−1	0.0000

Table 5
The experimental results for the power grid and PGP networks. The parameter k used in k-means and Ncut is set to 200 for the power grid network and to 300 for the PGP network.

Network index	Power grid				PGP			
	Q_{max}	Q_{avg}	T_{avg}	p-Value	Q_{max}	Q_{avg}	T_{avg}	p-Value
GDPSO	0.8382	0.8368	4.7818E+2	×	0.8050	0.8013	6.3609E+2	×
GA	0.7161	0.7124	3.4485E+3	0.0000	0.6576	0.6473	3.7798E+4	0.0000
MOGA	0.7035	0.6949	4.9177E+3	0.0000	—	—	—	—
LPA	0.7602	0.7476	1.7373E−1	0.0000	0.7949	0.7845	7.0391E−1	0.0000
CNM	0.9229	0.9229	4.9121E+2	1.0000	0.8481	0.8481	7.8562E+3	1.0000
Informap	0.8140	0.8140	6.3715	0.0000	0.7777	0.7777	1.2313E+1	0.0000
k-Means	Out of memory			×	Out of memory			×
Ncut	0.8875	0.8866	4.4442	1.0000	Out of memory			×

Table 6
The influence of the population size on the performance of the algorithm. The modularity values over 30 independent runs are recorded in this table.

($gmax$ = 100)	$popsize$ = 20		$popsize$ = 60		$popsize$ = 100		$popsize$ = 140		popsize = 180	
	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}
Karate	0.4198	0.4085	0.4198	0.4174	0.4198	0.4198	0.4198	0.4198	0.4198	0.4198
Dolphin	0.5269	0.5265	0.5277	0.5265	0.5285	0.5284	0.5285	0.5284	0.5285	0.5285
Football	0.6046	0.6027	0.6046	0.6035	0.6046	0.6041	0.6046	0.6041	0.6046	0.6043
SFI	0.7484	0.7408	0.7484	0.7439	0.7506	0.7449	0.7506	0.7456	0.7506	0.7453
E-mail	0.5361	0.4162	0.5384	0.4632	0.5487	0.4783	0.5476	0.4819	0.5437	0.4864
Netscience	0.9323	0.9275	0.9535	0.9492	0.9540	0.9512	0.9537	0.9507	0.9547	0.9511
Power grid	0.7685	0.7627	0.8377	0.8311	0.8382	0.8368	0.8383	0.8357	0.8401	0.8372
PGP	0.8028	0.7974	0.8047	0.8002	0.8050	0.8013	0.8047	0.8011	0.8053	0.8017

salient. To obtain a tradeoff between the convergence and the computation time, we set $gmax = popsize = 100$ as the default configuration of the proposed algorithm.

In Section 4, the experiments on small networks demonstrate the effectiveness of the proposed GDPSO algorithm; however, its performance on large networks remains unsatisfactory. We made some effort to determine why the GDPSO obtains unsatisfactory results. Through our analysis, we summarized two key factors. On the one hand, for the network clustering LSGO problem, the diversity preservation is insufficient. However, we have yet to find a better strategy that can both preserve diversity and ensure fast convergence. On the other hand, the designed particle-position-update principle (Eq. (8)) is based on a simple greedy mechanism, which may lead to prematurity.

Eq. (8) is the key component of the algorithm. It updates the label identifier of a vertex with the neighbor identifier that generates the largest increase in the objective function value. From the perspective of graph theory, it is natural to update the vertex identifier with the two different methods shown in Fig. 9.

The two different label-identifier-update principles shown in Fig. 9(b) and (c) make sense from the perspective of sociology. For example, the "maxD" principle is in accordance with the social phenomenon whereby people prefer to learn from or simply imitate the one who is the most attractive and talented among their friends, and the "dominated" principle complies with the social phenomenon that one would like to follow the state that is kept by the majority of his or her friends. Table 8 shows a comparison of the three update principles.

From the table, we can observe that the "maxD" and the "dominated" principles are computationally faster than that of "ΔQ", and the "maxD" principle is the fastest. However, from the modularity index perspective, the "maxD" and the

Table 7
The influence of the maximum iteration number *gmax* on the performance of the algorithm. The modularity values over 30 independent runs are recorded in this table.

(*popsize* = 100)	*gmax* = 50		*gmax* = 100		*gmax* = 150		*gmax* = 200		*gmax* = 250	
	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}	Q_{max}	Q_{avg}
Karate	0.4198	0.4180	0.4198	0.4198	0.4198	0.4198	0.4198	0.4198	0.4198	0.4198
Dolphin	0.5277	0.5267	0.5285	0.5284	0.5285	0.5285	0.5285	0.5285	0.5285	0.5285
Football	0.6046	0.6040	0.6046	0.6041	0.6046	0.6037	0.6046	0.6039	0.6046	0.6043
SFI	0.7487	0.7447	0.7506	0.7449	0.7506	0.7454	0.7506	0.7449	0.7506	0.7441
E-mail	0.5157	0.4731	0.5487	0.4783	0.5237	0.4687	0.5431	0.4864	0.5454	0.4997
Netscience	0.9531	0.9482	0.9540	0.9512	0.9537	0.9507	0.9547	0.9511	0.9507	0.9491
Power grid	0.8365	0.8331	0.8382	0.8368	0.8383	0.8357	0.8401	0.8372	0.8404	0.8380
PGP	0.8041	0.8012	0.8050	0.8013	0.8050	0.7991	0.8053	0.8031	0.8053	0.8008

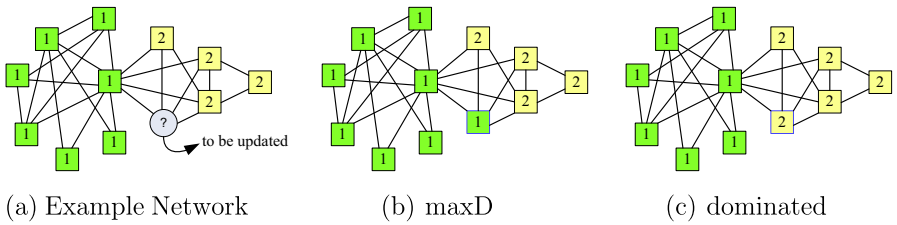


Fig. 9. Different vertex-label-identifier-update principles. (b) Choose the identifier of the neighboring vertex that has the largest degree. (c) Choose the dominated identifier from the neighbor vertices.

Table 8
Comparison of different particle-position-update principles.

Algorithm index	GDPSO (ΔQ)			GDPSO (maxD)				GDPSO (dominated)			
	Q_{max}	Q_{avg}	T_{avg}	Q_{max}	Q_{avg}	T_{avg}	*p*-Value	Q_{max}	Q_{avg}	T_{avg}	*p*-Value
Karate	0.4198	0.4198	2.5800E−2	0.4156	0.4080	1.7724E−3	0.0000	0.4156	0.4089	3.3976E−3	0.0000
Dolphin	0.5285	0.5284	5.8621E−2	0.5268	0.5265	4.4154E−3	0.0000	0.5268	0.5265	2.8781E−2	0.0000
Football	0.6046	0.6041	9.4210E−2	0.6046	0.6038	1.0939E−2	0.3349	0.6046	0.6038	3.6844E−2	0.5000
SFI	0.7506	0.7449	4.5217E−2	0.7484	0.7433	8.7936E−3	0.0153	0.7497	0.7458	2.3563E−1	0.8812
E-mail	0.5487	0.4783	2.4717E+1	0.4941	0.3108	4.3074E−1	0.0000	0.3670	0.2453	1.1583	0.0000
Netscience	0.9540	0.9512	3.5212E+1	0.9522	0.9478	2.2385	0.0000	0.9334	0.9296	1.2988E+1	0.0000
power grid	0.8382	0.8368	4.7818E+2	0.8382	0.8368	4.5868E+1	0.5000	0.8145	0.8116	2.0348E+2	0.0000
PGP	0.8050	0.8013	6.3609E+2	0.8021	0.7977	1.7526E+1	0.0000	0.8145	0.8093	1.0025E+3	1.0000

"dominated" principles all fall into local optima solutions. The rank sum test results indicate that the "ΔQ " mechanism works better than do the other two particle-position-update principles.

5. Concluding remarks

PSO, which has attracted much interest in recent years, has been demonstrated to be an excellent metaheuristic optimization technique. However, two main drawbacks hinder its use in many fields. First, PSO was originally designed for continuous optimization problems, which limits its application in discrete optimization domains. Second, PSO suffers from the curse of dimensionality, i.e., its performance deteriorates quickly as the dimensionality of the search space increases exponentially, which limits its application to LSGO. In this paper, a discrete PSO method designed to discover community structures in social networks was introduced. In the proposed algorithm, we first redefined the particle position and velocity in a discrete form and subsequently redesigned the particle-update rules based on the network topology; consequently, a discrete PSO framework was established. When applying the proposed discrete PSO framework to solve the network clustering problem, because the scale of a real-world social network is especially large most of the time, to alleviate prematurity, a greedy local-search-based mechanism was specially designed for the particle-position-update rule. Experiments on both synthetic and real-world networks demonstrated that the proposed algorithm for network clustering is effective and promising.

Although the proposed discrete PSO algorithm exhibits a good performance, in our experiments, we find that when addressing large-scale networks, if the averaged degrees of the nodes are large, the algorithm's search ability remains insufficient. Moreover, the objective function is non-separable, and in our experiments, we find it to be computationally

Please cite this article in press as: Q. Cai et al., Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.09.041

expensive to evaluate when the scale of the network is especially large. Thus, we actually face an expensive LSGO problem. Our future work will focus on a more in-depth analysis of discrete PSO and its application to expensive LSGO. Such analysis is expected to shed light on how to further improve discrete PSO.

Acknowledgements

The authors wish to thank the editors and anonymous reviewers for their valuable comments and helpful suggestions, which greatly improved the paper's quality. This work was supported by the National Natural Science Foundation of China (Grant Nos. 61273317, 61422209, 61473215), the National Top Youth Talents Program of China, the Specialized Research Fund for the Doctoral Program of Higher Education (Grant No. 20130203110011) and the Fundamental Research Fund for the Central Universities (Grant Nos. K5051202053 and JB142001-8).

References

[1] B. Al-kazemi, C.K. Mohan, Multi-phase discrete particle swarm optimization, in: Information Proceedings with Evolutionary Computation, 2006, pp. 306–326.
[2] J.P. Bagrow, E.M. Bollt, Local method for detecting communities, Phys. Rev. E 72 (October) (2005) 046108.
[3] M. Boguna, R. Pastor-Satorras, A. Díaz-Guilera, A. Arenas, Models of social networks based on social distance attachment, Phys. Rev. E 70 (5) (2004) 056112.
[4] W.N. Chen, J. Zhang, H.S.H. Chung, W.L. Zhong, W.G. Wu, Y.H. Shi, A novel set-based particle swarm optimization method for discrete optimization problems, IEEE Trans. Evolut. Comput. 14 (2010) 278–300.
[5] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (6) (2004) 066111.
[6] M. Clerc, J. Kennedy, The particle swarm – explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evolut. Comput. 6 (1) (2002) 58–73.
[7] L. Danon, A. Díaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification, J. Stat. Mech.: Theory Exp. 2005 (09) (2005) P09008.
[8] S. Datta, C. Giannella, H. Kargupta, *k*-Means clustering over a large, dynamic network, in: Proceedings of the 2006 SIAM International Conference on Data Mining, 2006, pp. 153–164.
[9] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (3) (2010) 75–174.
[10] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Nat. Acad. Sci. USA 99 (12) (2002) 7821–7826.
[11] M. Gong, Q. Cai, X. Chen, L. Ma, Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition, IEEE Trans. Evolut. Comput. 18 (1) (2014) 82–97.
[12] M. Gong, Q. Cai, Y. Li, J. Ma, An improved memetic algorithm for community detection in complex networks, in: Proceedings of 2012 IEEE Congress on Evolutionary Computation, 2012a, pp. 1–8.
[13] M. Gong, L. Ma, Q. Zhang, L. Jiao, Community detection in networks by using multiobjective evolutionary algorithm with decomposition, Physical A 391 (15) (2012) 4050–4060.
[14] R. Guimerà, L. Danon, A. Díaz-Guilera, Self-similar community structure in a network of human interactions, Phys. Rev. E 68 (6) (2003) 065103.
[15] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, R. Tibshirani, The Elements of Statistical Learning, vol. 2, Springer, 2009.
[16] S.-T. Hsieh, T.-Y. Sun, C.-C. Liu, S.-J. Tsai, Efficient population utilization strategy for particle swarm optimizer, IEEE Trans. Syst. Man Cybern. 39 (2) (2009) 444–456.
[17] B.J. Jain, K. Obermayer, Elkan's *k*-means algorithm for graphs, in: Advances in Soft Computing, Springer, 2010, pp. 22–32.
[18] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of 1995 IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.
[19] J. Kennedy, R. Eberhart, A discrete binary version of the particle swarm algorithm, in: Proceedings of 1997 IEEE International Conference on Systems, Man, and Cybernetics, vol. 5, 1997, pp. 4104–4108.
[20] A. Lancichinetti, S. Fortunato, J. Kertész, Detecting the overlapping and hierarchical community structure in complex networks, New J. Phys. 11 (3) (2009) 033015.
[21] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (4) (2008).
[22] D. Lazer, A. Pentland, L. Adamic, S. Aral, A.L. Barabasi, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, T. Jebara, G. King, M. Macy, D. Roy, M. Van Alstyne, Social science. computational social science, Science 323 (5915) (2009) 721–723.
[23] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Trans. Evolut. Comput. 16 (2) (2012) 210–224.
[24] J. Liang, A. Qin, P. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evolut. Comput. 10 (3) (2006) 281–295.
[25] C.J. Liao, C.T. Tseng, P. Luarn, A discrete version of particle swarm optimization for flowshop scheduling problems, Comput. Oper. Res. 34 (10) (2007) 3099–3111.
[26] D. Lusseau, K. Schneider, O.J. Boisseau, P. Haase, E. Slooten, S.M. Dawson, The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations, Behav. Ecol. Sociobiol. 54 (4) (2003) 396–405.
[27] M. Mitrovic, B. Tadic, Spectral and dynamical properties in classes of sparse networks with mesoscopic inhomogeneities, Phys. Rev. E 80 (2009) 026123.
[28] F. Neri, E. Mininno, G. Lacca, Compact particle swarm optimization, Inf. Sci. 239 (2013) 96–121.
[29] M.E.J. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69 (2004) 066133.
[30] M.E.J. Newman, Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E 74 (3) (2006) 036104.
[31] G. Pampara, N. Franken, A. Engelbrecht, Combining particle swarm optimisation with angle modulation to solve binary problems, in: Proceedings of 2005 IEEE Congress on Evolutionary Computation, vol. 1, 2005, pp. 89–96.
[32] C. Pizzuti, GA-Net: a genetic algorithm for community detection in social networks, in: Parallel Problem Solving from Nature (PPSN), vol. 5199, 2008, pp. 1081–1090.
[33] C. Pizzuti, A multiobjective genetic algorithm to find communities in complex networks, IEEE Trans. Evolut. Comput. 16 (3) (2012) 418–430.
[34] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, D. Parisi, Defining and identifying communities in networks, Proc. Nat. Acad. Sci. USA 101 (9) (2004) 2658–2663.
[35] M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Nat. Acad. Sci. USA 105 (4) (2008) 1118–1123.
[36] A. Salman, I. Ahmad, S. Al-Madani, Particle swarm optimization for task assignment problem, Microprocess. Microsyst. 26 (8) (2002) 363–371.
[37] D.Y. Sha, C.Y. Hsu, A hybrid particle swarm optimization for job shop scheduling problem, Comput. Ind. Eng. 51 (4) (2006) 791–808.
[38] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[39] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Proceedings of 1998 IEEE Congress on Evolutionary Computation, 1998, pp. 69–73.
[40] Y. Shi, R. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of 1999 Congress on Evolutionary Computation, vol. 3, 1999, p. 3.

Please cite this article in press as: Q. Cai et al., Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.09.041

[41] Y. Shi, R. Eberhart, Fuzzy adaptive particle swarm optimization, in: Proceedings of 2001 IEEE Congress on Evolutionary Computation, vol. 1, 2001, pp. 101–106.
[42] Y. Shi, H.C. Liu, L. Gao, G.H. Zhang, Cellular particle swarm optimization, Inf. Sci. 181 (20) (2011) 4460–4493.
[43] C.L. Sun, J.C. Zeng, J.S. Pan, S.D. Xue, Y.C. Jin, A new fitness estimation strategy for particle swarm optimization, Inf. Sci. 221 (2013) 355–370.
[44] T.O. Ting, Y. Shi, S. Cheng, S. Lee, Exponential inertia weight for particle swarm optimization, in: Proceedings of 2012 International Conference on Swarm Intelligence, 2012, pp. 83–90.
[45] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evolut. Comput. 8 (3) (2004) 225–239.
[46] F.Y. Wang, D. Zeng, K.M. Carley, W.J. Mao, Social computing: from social informatics to social intelligence, IEEE Intell. Syst. 22 (2) (2007) 79–83.
[47] Y. Wang, B. Li, Two-stage based ensemble optimization for large-scale global optimization, in: Proceedings of 2010 IEEE Congress on Evolutionary Computation, 2010, pp. 1–8.
[48] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440–442.
[49] S. White, P. Smyth, A spectral clustering approach to finding communities in graph, SDM, vol. 5, SIAM, 2005, pp. 76–84.
[50] Z.Y. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Inf. Sci. 178 (15) (2008) 2985–2999.
[51] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. (1977) 452–473.
[52] S. Zhang, R.-S. Wang, X.-S. Zhang, Identification of overlapping community structure in complex networks using fuzzy c-means clustering, Phys. A: Stat. Mech. Appl. 374 (1) (2007) 483–490.
[53] S.Z. Zhao, J.J. Liang, P.N. Suganthan, M.F. Tasgetiren, Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization, in: Proceedings of 2008 IEEE Congress on Evolutionary Computation, 2008, pp. 3845–3852.
[54] Z. Zhou, Y. Shi, Inertia weight adaption in particle swarm optimization algorithm, in: Proceedings of 2011 International Conference on Swarm Intelligence, 2011, pp. 71–79.

Please cite this article in press as: Q. Cai et al., Greedy discrete particle swarm optimization for large-scale social network clustering, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.09.041