Iterated tabu search for identifying community structure in complex networks

Zhipeng Lü* and Wenqi Huang†
School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
(Received 27 April 2009; revised manuscript received 20 July 2009; published 28 August 2009)

This paper presents an iterated tabu search (denoted by ITS) algorithm for optimizing the modularity of community structure in complex networks. The proposed algorithm follows a general framework composed of two phases: basic optimization and postrefinement. When the basic optimization cannot improve the modularity any more, a postrefinement procedure is employed to further optimize the objective function with a global view. For both these two phases, iterated tabu search algorithm is employed to optimize the objective function. Computational results show the high effectiveness of the proposed ITS algorithm compared with six state-of-the-art algorithms in the literature. In particular, our ITS algorithm improves the previous best known modularity for several small and medium size networks.

DOI: 10.1103/PhysRevE.80.026130 PACS number(s): 89.75.Hc, 05.10.−a

I. INTRODUCTION

Understanding the structure properties of complex networks has attracted a considerable amount of attention from the scientific community in recent years, particularly within physics. Complex networks often consist of a set of vertices connected by edges, where vertices represent entities in the complex system and edges represent connections or relations between them. Examples may include social relationship networks [1], world wide web [2], internet [3], protein-protein interaction network [4], citation and collaboration network [5,6], food web [7], and so on. For various complex networks, a large number of structural features have been analyzed by researchers such as the small world phenomena [8], vertex similarity [9], mixing patterns [10], and betweenness centrality [11].

In the last decade, one of the most popular features that has been the focus of many recent efforts is the community structure or module of complex networks, i.e., the clustering of vertices into groups such that the density of within-group edges is much higher than that of between-group edges. The ability to detect community structure has a large amount of usefulness in many aspects [12]. For example, nodes belonging to the same community may have much more common features than those in different communities, which could be used to simplify the functional analysis of complex networks. Furthermore, community structure may provide insights in understanding some uncharacteristic properties of a complex network system [13,14]. For instance, in the world wide web, community analysis has uncovered thematic clusters [12,13]; in biochemical or neural networks, communities may be functional groups [14] and separating the network into such groups could simplify functional analysis considerably.

Recently, Newman and Girvan proposed a so-called “modularity” measure to evaluate the meaningfulness of a specific division of a complex network into communities [15], which has been widely accepted by the scientific community. High value of modularity represents a good community division, meaning that the real number of within-community edges is much more than expected by random chance. However, the problem of optimizing the modularity of a specific network division has shown to be nondeterministic polynomial (NP) complete [16]. Therefore, it is unfeasible to exhaustively search all possible divisions to find the optimal modularity, even for a small number of vertices. Instead, heuristic algorithms based on metaheuristics have shown to be highly effective.

In recent years, various highly effective heuristic and metaheuristic algorithms have been proposed to optimize the modularity of large networks. According to [17], all these algorithms could be categorized into the following: divisive or link removal, agglomerative or clustering, optimization based, spectral analysis, and others. Newman and Girvan proposed a divisive community identification algorithm that iteratively removes edges with the highest value of edge betweenness [15]. In [18,19], Newman and co-workers proposed a greedy algorithm, iteratively joining a pair of communities that produced the largest gain in modularity until a maximum value of modularity is obtained. Duch and Arenas developed an extremal optimization algorithm, which considered the contribution of each vertex to the total value of modularity [20]. Newman proposed a spectral bisection algorithm by reformulating the modularity in terms of the eigenvectors of a characteristic matrix [12,21]. Schuetz and Caflisch introduced the multistep greedy algorithm for modularity optimization, which is based on the idea of simultaneously merging several pairs of communities [22]. In [23], Danon *et al.* proposed an improved algorithm over [18,19] by considering the inhomogeneities in community sizes. Other algorithms include the L-shell method [24], the hierarchical clustering [25], the simulated annealing [26], the tabu search algorithm [27], and so on. Generally, our iterated tabu search (ITS) algorithm can be classified as an optimization-based divisive algorithm.

This paper presents ITS, a hybrid metaheuristic algorithm integrating a tabu search procedure with an adaptive perturbation operator [from iterated local search (ILS)] for optimizing the modularity of community structure division. In this algorithm, we highlight the balance between intensification and diversification. The proposed ITS algorithm follows

*zhipeng.lui@gmail.com
†wqhuang@hust.edu.cn

a general framework composed of two phases: basic optimization and postrefinement. The basic optimization phase iteratively divides the network community into smaller ones until the modularity function cannot be improved any more. Then the postrefinement phase is employed to further optimize the modularity while maintaining the number of community structures fixed. For both these two phases, an iterated tabu search algorithm is introduced to optimize the objective function. Experiments are presented on the set of seven networks from the literature, showing that the proposed algorithm achieves very competitive results compared with many previous best solutions.

The rest of this paper is organized as follows. Section II describes our iterated tabu search algorithm for the phases of both basic optimization and postrefinement. In Sec. III our computational results are presented and compared with six reference algorithms in the literature, showing the effectiveness of the proposed hybrid metaheuristic algorithm. Finally, Sec. IV concludes the paper.

II. ITERATED TABU SEARCH ALGORITHM

A. Problem definition

Recently, Newman and Girvan proposed a simple quantitative measure of the quality of communities called "modularity," which has become widely accepted by the scientific community [15]. The idea is from the intuition that a network with community structure is different from a random network. Given an unweighted and undirected graph G and suppose the vertices are divided into communities such that vertex u belongs to community $r(u)$ (denoted by $c_{r(u)}$), the modularity is defined as

$$Q = \frac{1}{2m}\sum_{uv}\left[A_{uv} - \frac{d_u d_v}{2m}\right]\delta[r(u),r(v)], \qquad (1)$$

where A is the adjacency matrix of graph G. $A_{uv}=1$ if node u and node v connect with each other, $A_{uv}=0$ otherwise. The δ function $\delta(i,j)$ is equal to 1 if $i=j$ and 0 otherwise. The degree d_u of a vertex u is defined to be $d_u=\Sigma_v A_{uv}$ and $m=\frac{1}{2}\Sigma_{uv}A_{uv}$ is the number of edges in the graph.

One observes that this modularity function can be represented in another way,

$$Q = \sum_i (e_{ii} - a_i^2), \qquad (2)$$

where i runs over all communities and e_{ij} represents the fraction of all links connecting nodes, respectively, in communities c_i and c_j. According to the definition, we have $e_{ij}=\frac{1}{2m}\Sigma_{uv}A_{uv}\delta[r(u),i]\delta[r(v),j]$. Therefore, e_{ii} is the fraction of all links lying in community c_i. a_i is the fraction of links that end in nodes of c_i, i.e., $a_i=\Sigma_j e_{ij}=\frac{1}{2m}\Sigma_u d_u\delta[r(u),i]$.

If all link ends in community c_i are randomly connected, the expected fraction of links lying within community c_i is a_i^2. The fraction of intracommunity edges (e_{ii}) minus the expected value of the same quantity (a_i^2) measures whether a network division indicates a strong community structure. The objective of our algorithm is therefore to find a network division, which maximizes the modularity function Q.

TABLE I. The pseudocode of the iterated tabu search algorithm.

```
1: $s_0 \leftarrow$ initial solution
2: $s' \leftarrow$ tabu search $(s_0)$
3: repeat
4:    $s^* \leftarrow$ perturbation operator $(s')$
5:    $s^{*\prime} \leftarrow$ tabu search $(s^*)$
6:    $s' \leftarrow$ acceptance criterion $(s^{*\prime}, s')$
7: until stop condition met
```

B. General procedure

The main challenges of community identification consist of deciding the number of communities and optimizing the modularity function. Therefore, an effective community discovery algorithm should not only be able to decide by itself the appropriate number of community partitions without prior knowledge but also effectively optimize the modularity function Q as far as possible.

Generally, our algorithm consists of two phases: basic optimization and postrefinement. During the basic optimization phase, we recursively divide each large community into two smaller ones until Q cannot be improved any more. After that, the postrefinement phase is employed to adjust locally the community division while remaining the number of communities unchanged. During these two phases, we use a hybrid metaheuristic algorithm called ITS to optimize the modularity function Q. In the following sections, we first describe the general idea and framework of our iterated tabu search algorithm followed by its adaptation to the two optimization phases.

C. Iterated tabu search algorithm

TS and ILS are two well-known metaheuristics and have proven their effectiveness for solving separately a large number of practical optimization problems [28,29]. In this paper, we consider the possibility of combining them to achieve high-quality solutions for the community structure identification problem.

Tabu search can be used with both long and short CPU times. In general, long CPU time would lead to better results. However, like other local search algorithms, TS also easily falls into local optimum trap even long CPU time is allowed. Therefore, it would be preferred to combine short TS runs with some robust diversification operators to jump out of local optimum trap. Interestingly, ILS provides such diversification mechanisms to guide the search to escape from the current local optimum and move toward new promising regions in the search space [29].

Iterated tabu search algorithm starts with an initial solution and performs tabu search until a local optimum is found. Then, the current local optimum solution is perturbed and another round of TS is performed to the perturbed solution. Finally, an acceptance criterion is used to decide whether the new local optimum solution is accepted as the initial solution for the next run of tabu search. Table I shows the pseudocode of our ITS algorithm.

A fundamental principle of our ITS is to exploit the tradeoff between diversification and intensification. Intensification focuses on optimizing the objective function as far as possible within a limited search region while diversification should be able to drive the search to explore new promising regions of the search space. The intensification of our ITS algorithm is realized by the tabu search procedure. The diversification mechanism—perturbation operator—has two aims: one is to jump out of the local optimum trap just visited; the other is to lead the search procedure to a new promising region.

In order to adapt the above ITS algorithm to the two phases of our community identification algorithm, we just need to define how the initial solution is generated, what is the neighborhood, how the tabu list is designed, what is the stop condition of tabu search, how the perturbation operator is designed, and what is the acceptance criterion.

D. Basic optimization

From a single community, our basic optimization procedure divides the network into two communities. Then each smaller community is further divided into two. This procedure is repeated until the modularity Q cannot be further improved, just as done in [17]. During this recursive division process, we employ the iterated tabu search algorithm to optimize the modularity Q. Therefore, we only need to describe our ITS algorithm for splitting one community into two.

Suppose we divide community c_i into two communities c_j and c_k. Initially, we divide all the vertices belonging to c_i into two random partitions, i.e., each vertex is randomly assigned to c_j or c_k. Then, this initial partition is optimized using our ITS algorithm.

We now focus on the basic search engine of our ITS algorithm—tabu search [28]. It is widely believed that one of the most important features of a local search-based algorithm is the definition of its neighborhood. In a local search procedure, applying a move to a candidate solution leads to a new solution. The neighborhood move of our TS is moving one vertex from a community to another. In order to evaluate this neighborhood move in an efficient way, we use an incremental evaluation technique. The main idea is to maintain in a special data structure the *move value* for each possible move of the current solution. Each time a move is carried out, the elements of this data structure affected by the move are updated accordingly. Let c_j or c_k be two communities and u be a vertex in any of these two communities (say, $u \in c_j$). It is observed that the change to Q incurred by moving vertex u from c_j to c_k can be represented as

$$\Delta Q_{(u,c_j,c_k)} = \frac{d_u^{(k)} - d_u^{(j)}}{m} + \frac{d_u(a_j - a_k)}{m} - \frac{d_u^2}{2m^2}, \tag{3}$$

where $d_u^{(j)}$ (respectively, $d_u^{(k)}$) is the number of edges connecting u and vertices in community c_j (respectively, c_k). m, d_u, a_j, and a_k are the same as in Eqs. (1) and (2).

One observes that after this move is performed, a_j and a_k are updated as follows:

$$a_j' = a_j - \frac{d_u}{2m}, \tag{4}$$

$$a_k' = a_k + \frac{d_u}{2m}. \tag{5}$$

For any vertex $v(v \neq u)$, $d_v^{(j)}$ and $d_v^{(k)}$ are updated as

$$d_v^{(j)\prime} = d_v^{(j)} - A_{uv}, \tag{6}$$

$$d_v^{(k)\prime} = d_v^{(k)} + A_{uv}. \tag{7}$$

For any vertex v in community c_j, we can obtain the updated ΔQ value $\Delta Q'_{(v,c_j,c_k)}$ by substituting Eqs. (4)–(7) into Eq. (3). Therefore, we have

$$\Delta Q'_{(v,c_j,c_k)} = \Delta Q_{(v,c_j,c_k)} - \left(\frac{d_u^2}{m^2} - \frac{2A_{uv}}{m}\right). \tag{8}$$

Similarly, for any vertex v in community c_k, $\Delta Q_{(v,c_k,c_j)}$ is updated as

$$\Delta Q'_{(v,c_k,c_j)} = \begin{cases} -\Delta Q_{(v,c_j,c_k)} & \text{if } v = u \\ \Delta Q_{(v,c_k,c_j)} + \left(\frac{d_u^2}{m^2} - \frac{2A_{uv}}{m}\right) & \text{otherwise.} \end{cases} \tag{9}$$

One easily finds that the term $(\frac{d_u^2}{m^2} - \frac{2A_{uv}}{m})$ in Eqs. (8) and (9) is a constant for a given network. That is to say, it can be precalculated and stored in a data structure. Thus, we only need one addition operation to update each ΔQ value after each move. However, since $d_u^{(j)}$, $d_u^{(k)}$, a_j, and a_k are dynamically updated at each step [Eqs. (4)–(7)], we need four addition and two multiplication operations for directly calculating ΔQ values using Eq. (3). Therefore, the computational efforts of Eqs. (8) and (9) are much fewer than those of Eq. (3). Note that Eq. (3) is just used in the community initialization and we always employ Eqs. (8) and (9) to update ΔQ values during the tabu search procedure. We have to mention that this is not merely a technical trick since at each local search step we need to update ΔQ values of all vertices belonging to the current two communities.

The simple local search algorithm chooses the best move in the current neighborhood at each step until the results do not improve. Compared with the simple local search algorithm, TS introduces a special data structure called *tabu list* to forbid the previously visited solutions to be revisited in order to avoid cycling. In our TS algorithm, when moving one vertex u from one community c_j to another one c_k, this vertex cannot be moved back to the previous community c_j for the next *TabuTenure* iterations. In our present implementation, we have elected to set

$$TabuTenure(u) = C + \text{rand}(10), \tag{10}$$

where C is a constant and rand(10) denotes a randomly generated number from 1 to 10. The constant C is experimentally fixed at 10 in our implementation.

The TS algorithm then restricts consideration to vertices not forbidden by the tabu list and selects a vertex to move that produces the largest ΔQ value (thus improving Q if this value is positive). However, some of those forbidden solutions might be of excellent quality and might not have been visited. To mitigate this problem, an aspiration criterion is applied that permits a move to be selected by overriding its tabu state if it leads to a solution better than the currently known best solution.

This TS process stops when the best solution cannot be improved within a given number α of moves that we call the *improvement cutoff*. In this paper, we experimentally set $\alpha = 2000$ for all tested networks.

When the best solution obtained by the TS algorithm cannot be improved any more, we employ a perturbation operator to reconstruct this local optimum solution. A commonly used perturbation operator is to destruct partially the previous local optimum solution in a *random* way, not necessarily guided by an evaluation function [29]. However, a strong perturbation operator should not only jump out of the local optimum region just visited but also lead the search to move toward new promising areas of the search space. For this purpose, we employ a guided perturbation operator to destruct the reached local optimum solution.

Specifically, when the current TS terminates with the best solution found s', all the vertices of solution s' are ranked in a nonincreasing order according to their ΔQ values. Then, β vertices are selected to be moved from their current community to another, where the vertex of rank i is selected according to the following probability distribution:

$$P(i) = i^{-\phi} / \sum_j j^{-\phi}, \tag{11}$$

where ϕ is a positive real number (empirically set at 2.0) and β denotes the perturbation strength.

It is reasonable that a greater β corresponds to more possibilities of escaping from the current local maximum, while too large β will behave like random start. In this paper, we empirically set $\beta = \hat{n}/5$, where $\hat{n}$ is the total number of vertices in the current two communities.

The rationale behind this strategy is that a vertex having a large value of ΔQ should have priority to move to another community. In sum, this perturbation operator is based on the identification of a set of the critical vertices and moving these vertices to the opposite community.

From this perturbed solution, a new tour of TS is again used to optimize the modularity function. This process is repeated until a certain number of perturbations are performed. We call this procedure the iterated tabu search algorithm.

E. Postrefinement

When the above-mentioned basic optimization phase finishes with the best known solution, a postrefinement procedure is launched to further optimize Q with maintaining the number of communities unchanged, as done in [30]. In this refinement stage, we employ again the above iterated tabu search algorithm to improve Q as much as possible.

In this procedure, the neighborhood move is defined as moving a vertex u from community c_j to c_k denoted by $\langle u, c_j, c_k \rangle$. For a network with n vertices and a community partition with R communities, the size of this neighborhood is bounded by $O(nR)$. At the beginning of the algorithm, ΔQ values are also initialized according to Eq. (3). As demonstrated in the basic optimization procedure, the corresponding ΔQ values are updated as in Eqs. (8) and (9) during the TS procedure. Suppose we perform a neighborhood move $\langle u, c_j, c_k \rangle$. Then, if a neighborhood move involves moving a vertex v in community c_g to c_h and both g and h are unequal to j or k, the ΔQ values of this move $\langle v, c_g, c_h \rangle$ will not be changed and not be necessarily updated. Thus, this technique could save a lot of computational efforts.

Like in the basic optimization procedure, the perturbation operator is employed to perturb the local optimum solution if TS cannot improve the best known results any more. Once again, all the moves are ranked in a nonincreasing order according to their ΔQ values. Then, a certain number of highly ranked moves are randomly chosen to be performed according to Eq. (11). After that, the same TS algorithm is applied to the perturbed solution. This process is repeated until a certain number of perturbations is reached. Finally, the algorithm finishes with the best known solution corresponding to the maximal Q value.

III. COMPUTATIONAL RESULTS

To assess the effectiveness of our proposed ITS algorithm, we carry out experiments on seven networks of different

TABLE II. Computational results of our ITS algorithm.

			Basic optimization				Postrefinement			
Instance	n	m	Q_{best}	t_{best}	Q_{aver}	g_N	Q_{best}	t_{best}	Q_{aver}	g_N
Zachary	34	78	0.4188	0	0.4188	4	0.4198	0	0.4198	4
Jazz	198	2742	0.4422	125	0.4422	4	0.4460	260	0.4453	4
C.Elegans	453	2025	0.4419	247	0.4388	10	0.4513	584	0.4473	10
Email	1133	5451	0.5780	452	0.5648	12	0.5811	875	0.5798	10
Erdös	6927	11850	0.6861	1297	0.6765	120	0.6902	2348	0.6872	108
PGP	10680	24316	0.8418	2639	0.8368	499	0.8427	5168	0.8406	570
Cond-Mat	27519	116181	0.6911	5426	0.6857	341	0.6957	8467	0.6908	404

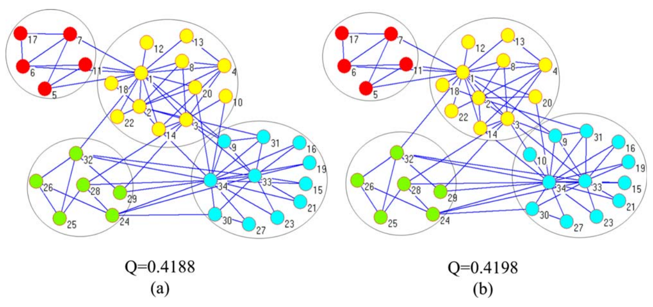

FIG. 1. (Color online) Best network divisions of the Zachary network, respectively, for basic optimization and postrefinement.

sizes from the literature, ranging from 34 to 27 519 vertices, and compare our ITS algorithm with six best performing algorithms in the literature. These networks include a Zachary karate club network (Zachary) [31], a jazz musician collaborations network (Jazz) [32], a metabolic network for the nematode C.Elegans (C.Elegans) [33], a university e-mail network (e-mail) [34], an Erdös collaboration network (Erdös) [35], a trust network of mutual signing of cryptography keys (PGP) [36], and a scientific coauthorship network in condensed-matter physics (Cond-Mat) [37].

Our algorithm is programmed in C and run on a PC running Windows XP with Intel Pentium IV 2.66 GHz CPU and 512 MB RAM. Given the stochastic nature of our ITS algorithm, each problem instance is independently solved ten times with a time limit of 3 CPU h.

Table II gives the computational statistics of our ITS algorithm. Columns 2 and 3, respectively, give the number of vertices (n) and the number of edges (m) of the networks. Columns 4–7 give the results of our basic optimization phase: the best modularity value (Q_{best}), the CPU time (in seconds) to reach the best solution (t_{best}), the average modularity value over ten independent runs (Q_{aver}), and the number of communities (g_N) corresponding to the best community division. Similarly, Columns 8–11 present the results of our algorithm after the postrefinement phase where t_{best} represents the total CPU time to reach the best solution, including that used in the basic optimization. One easily observes that for all the tested networks, our postrefinement procedure consistently improves the results obtained by the basic optimization procedure in terms of both the best and the average modularity values, which shows the effectiveness of our two-phases ITS algorithm.

For example, Fig. 1 demonstrates the best network divisions, respectively, obtained by the phases of basic optimization (Q=0.4188) and postrefinement (Q=0.4198) for the Zachary network. One easily observes that the only difference between these two divisions is the placement of vertex 10. Our experiments show that the network division with Q=0.4198 cannot be obtained by merely using the basic optimization procedure even with much more computational efforts. This might disclose some limitations of the general procedure of recursive partitioning and highlight the importance of the postrefinement phase.

Table II assesses the effectiveness of our ITS algorithm on the seven networks. Now, we turn our attention to the comparison of our ITS algorithm with the most effective algorithms in the literature. Table III gives the computational comparison of our ITS algorithm with six state-of-the-art algorithms, which include Newman's fast (NF) algorithm for community detection [18], the extremal optimization (EO) algorithm developed by Duch and Arenas [20], the clustering algorithm (PBD) by Pujol *et al.* [38], the multistep greedy algorithm (SC) by Schuetz and Caflisch [22], the improved Newman's fast algorithm (DDA) by Danon *et al.* [23], and Newman's spectral (NS) algorithm [12]. In Table III, columns 2 and 3 recall the number of vertices of the network and the best modularity obtained by our ITS algorithm. Columns 4–9 present the best results obtained by these state-of-the-art algorithms and the best result for each instance is indicated in bold.

From Table III, one observes that for five small and medium size networks, our ITS algorithm obtains better modularity than these previous state-of-the-art algorithms. Furthermore, for the two left large networks PGP and Cond-Mat, we have achieved worse results than at least three reference algorithms. It should be noted that at least two algorithms SC

TABLE III. Comparison with the state-of-the-art algorithms in terms of the best modularity.

Instance	n	ITS	NF[a]	EO[b]	PBD[c]	SC[d]	DDA[e]	NS[f]
Zachary	34	0.4198	0.3810	0.4188	0.3937	0.398	0.4087	0.419
Jazz	198	0.4460	0.4379	0.4452		0.4451	0.4409	0.442
C.Elegans	453	0.4513	0.4001	0.4342	0.4164	0.450		0.435
Email	1133	0.5811	0.4796	0.5738		0.575	0.5569	0.572
Erdös	6927	0.6902	0.6723[g]	0.6520[g]	0.6817			
PGP	10680	0.8427	0.7329	0.8459		0.878	0.7462	0.855
Cond-Mat	27519	0.6957	0.6683	0.6790	0.7251	0.748		0.723

[a]Reference [18].
[b]Reference [20].
[c]Reference [38].
[d]Reference [22].
[e]Reference [23].
[f]Reference [12].
[g]The results of NF and EO algorithms on the Erdös network are cited from [38].

and NS can obtain better results than our ITS algorithm for both these two largest networks. In addition, three algorithms EO, PBD, and DDA can achieve better results than our ITS algorithm for at least one of these two large networks, while algorithm NF obtains worse results than our ITS algorithm for both these two networks. In sum, this comparison discloses that our ITS algorithm obtains quite competitive results for these tested networks compared with other state-of-the-art algorithms in the literature.

In this section, we compared our results with those obtained by six reference algorithms in the literature in terms of the best modularity obtained. However, one may wonder whether this comparison is fair since different algorithms use quite different machines and different time limits. In fact, comparing the computing time for different algorithms is a quite complex and difficult issue. However, for indicative purpose, we indicate the CPU time of our ITS algorithm for reaching the best solutions in Table II, from which one observes that the CPU time for most instances are within 1.5 CPU h, although the time limit of our algorithm is set to be 3 CPU h.

Let us give a final comment that all the computational results reported in Table II were obtained without special tuning of the parameters, i.e., all the parameters used in our algorithm are fixed constant [e.g., the *improvement cutoff* of TS α and the selection importance factor ϕ in Eq. (11)] or dynamically and automatically tuned [e.g., the tabu Tenure in Eq. (10) and the perturbation strength β] during the problem solving for all the instances considered here. It is possible that better solutions would be found by using a set of instance-dependent parameters. In fact, we have tried to tune these parameters and observed that the following parameter settings will give satisfying results: $\alpha \in [1000,5000]$, $\phi \in [1.5,3]$, $TabuTenure \in [10,25]$, and $\beta \in [\hat{n}/10,\hat{n}/4]$.

IV. CONCLUSION

Our ITS algorithm for identifying community structure is composed of a basic optimization procedure and a postrefinement procedure. In each procedure, a TS is employed to optimize the modularity while a perturbation operator is used to perturb the local optimum solution when TS cannot improve the results any more. Our ITS algorithm with these strategies achieves a tradeoff between intensification and diversification. In this paper, we also proposed a fast incremental neighborhood evaluation technique, allowing us to save a considerable amount of computational efforts.

In spite of being quite simple in comparison with most top performing algorithms, ITS proves to be highly effective in finding good solutions for the set of seven benchmark instances from the literature, containing from 34 to 27 519 vertices. Compared with six state-of-the-art algorithms, ITS is able to find competitive results for all the tested networks. In particular, it can obtain better results than reported in the literature for five networks with size from 34 to 6927.

However, for the two large networks PGP and Cond-Mat, our ITS algorithm obtains slightly worse results than the previous best known ones. Therefore, our work in progress includes enhancing our heuristic for solving large instances. This work will enable us to report on our approach to even larger networks in future papers.

There are several other directions to extend this work. One immediate possibility is to examine other neighborhoods. ITS and most existing algorithms are based on the simple one vertex move neighborhood. Richer neighborhoods using, for instance, the two vertices move would be worth examining. Furthermore, instead of using the objective function as the unique evaluation measure, other evaluation functions using additional information would likewise be worth exploring. Finally, more advanced adaptive memory strategies from the tabu search provide opportunities for creating further improvements.

ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundation of China (Grant No. 60773194) and the NKBRPC (Grant No. G2004CB318000).

[1] F. Liljeros, C. Edling, L. Amaral, H. Stanley, and Y. Aberg, Nature (London) **411**, 907 (2001).
[2] R. Albert, H. Jeong, and A. L. Barabási, Nature (London) **401**, 130 (1999).
[3] R. Albert, H. Jeong, and A. L. Barabási, Nature (London) **406**, 378 (2000).
[4] H. Jeong, S. Mason, Z. Oltvai, and A. L. Barabási, Nature (London) **411**, 41 (2001).
[5] A. L. Barabási, H. Jeong, E. Ravasz, Z. Néda, A. Schubert, and T. Vicsek, Physica A **311**, 590 (2002).
[6] M. E. J. Newman, Phys. Rev. E **64**, 016131 (2001).
[7] J. A. Dunne, R. J. Williams, and N. D. Martinez, Proc. Natl. Acad. Sci. U.S.A. **99**, 12917 (2002).
[8] J. M. Kleinberg, Nature (London) **406**, 845 (2000).
[9] E. A. Leicht, P. Holme, and M. E. J. Newman, Phys. Rev. E **73**, 026120 (2006).
[10] M. E. J. Newman, Phys. Rev. E **67**, 026126 (2003).
[11] K. I. Goh, E. Oh, B. Kahng, and D. Kim, Phys. Rev. E **67**, 017101 (2003).
[12] M. E. J. Newman, Proc. Natl. Acad. Sci. U.S.A. **103**, 8577 (2006).
[13] V. Spirin and L. A. Mirny, Proc. Natl. Acad. Sci. U.S.A. **100**, 12123 (2003).
[14] D. M. Wilkinson and B. A. Huberman, Proc. Natl. Acad. Sci. U.S.A. **101**, 5241 (2004).
[15] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
[16] U. Brandes, D. Delling, M. Gaertler, R. Göerke, M. Hoefer, Z. Nikoloski, and D. Wagner, in *Proceedings of the 33rd International Workshop Graph-Theoretic Concepts in Computer Science (WG 2007)*, edited by A. Brändstadt, D. Kratsch, and H. Müller, Lecture Notes in Computer Science (Springer-Verlag,

Berlin, 2007), Vol. 4769, p. 121–132.

[17] L. Danon, J. Duch, A. Arenas, and A. Díaz-Guilera, J. Stat. Mech.: Theory Exp. (2005) P09008.

[18] M. E. J. Newman, Phys. Rev. E **69**, 066133 (2004).

[19] A. Clauset, M. E. J. Newman, and C. Moore, Phys. Rev. E **70**, 066111 (2004).

[20] J. Duch and A. Arenas, Phys. Rev. E **72**, 027104 (2005).

[21] M. E. J. Newman, Phys. Rev. E **74**, 036104 (2006).

[22] P. Schuetz and A. Caflisch, Phys. Rev. E **78**, 026112 (2008).

[23] L. Danon, A. Díaz-Guilera, and A. Arenas, J. Stat. Mech.: Theory Exp. (2006) P11010.

[24] J. P. Bagrow and E. M. Bollt, Phys. Rev. E **72**, 046108 (2005).

[25] S. Fortunato, V. Latora, and M. Marchiori, Phys. Rev. E **70**, 056104 (2004).

[26] R. Guimerà, M. Sales-Pardo and L. A. N. Amaral, Phys. Rev. E **70**, 025101(R) (2004).

[27] A. Arenas, A. Fernández, and S. Gómez, New J. Phys. **10**, 053039 (2008).

[28] F. Glover and M. Laguna, *Tabu Search* (Kluwer Academic, Boston, 1997).

[29] H. R. Lourenco, O. Martin, and T. Stützle, *Iterated Local Search*, Handbook of Metaheuristics (Springer-Verlag, Berlin, 2003), pp. 321–353.

[30] J. Ruan and W. Zhang, Phys. Rev. E **77**, 016104 (2008).

[31] W. W. Zachary, J. Anthropol. Res. **33**, 452 (1977).

[32] P. Gleiser and L. Danon, Adv. Complex Syst. **6**, 565 (2003).

[33] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabási, Nature (London) **407**, 651 (2000).

[34] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas. Phys. Rev. E **68**, 065103(R) (2003).

[35] Erdös Number Project, the network contains scientists with Erdös number less than or equal to 2 up to year 2002 (http://www.oakland.edu/enp/thedata.html).

[36] X. Guardiola, R. Guimerà, A. Arenas, A. Díaz-Guilera, D. Streib, and L. Amaral, e-print arXiv:cond-mat/0206240.

[37] M. E. J. Newman, Proc. Natl. Acad. Sci. U.S.A. **98**, 404 (2001).

[38] J. M. Pujol, J. Béjar, and J. Delgado, Phys. Rev. E **74**, 016107 (2006).