Revealing network communities through modularity maximization by a contraction–dilation method

Juan Mei\textsuperscript{1,2}, Sheng He\textsuperscript{1,2}, Guiyang Shi\textsuperscript{1,2}, Zhengxiang Wang\textsuperscript{1,2} and Weijiang Li\textsuperscript{1,2,3}

1 The Key Laboratory of Industrial Biotechnology, Ministry of Education, Jiangnan University, Wuxi 214122, People’s Republic of China
2 School of Biotechnology, Jiangnan University, Wuxi 214122, People’s Republic of China
E-mail: wjlee01@gmail.com

Abstract. Many real-world systems can be described by networks whose structures relate to functional properties. An important way to reveal topology–function correlations is to detect the community structures, which can be well evaluated by graph modularity. By maximizing modularity, large networks can be divided into naturally separated groups. Here, we propose a contraction–dilation algorithm based on single-node-move operations and a perturbation strategy. Tests on artificial and real-world networks show that the algorithm is efficient for discovering community structures with high modularity scores and accuracies at low expenses of both time and memory.

3 Author to whom any correspondence should be addressed.
1. Introduction

Scientifically interesting complex networks often show potential group structures, i.e. a whole network consists of loosely connected dense parts. Such structures usually have strong correlations to functionalities of the networks. For example, relatively independent groups of molecules in a metabolic network might imply functionally independent modules or pathways [1]; a densely linked group of web pages on the World Wide Web might correspond to related topics [2]. Finding loosely tied dense groups of nodes, also known as clusters or communities, is thus an important approach to analyzing complex networks.

Network clustering has been a hot topic in many research fields like sociology, pattern recognition and bioinformatics for decades [3]–[6]. Among various methods proposed for network clustering, the modularity-based approaches have attracted much attention recently. The modularity proposed by Newman and Girvan [7] is an evaluation index of community structures. A large value of modularity corresponds to a good partitioning of the network. In this framework, discovering communities is realized through maximizing the modularity.

It should be noted that there are actually unlimited possible measures to assess network partitioning from different points of view. For example, normalized cut [8] is another popular criterion that is widely used in computer science [9, 10]. No universal, operational standard is available to judge a criterion. All known criteria encounter problems under certain conditions. A recently raised concern on the resolution limit of modularity was discovered by Fortunato and Barthelemy [11], who proved that small tightly connected communities may be theoretically indistinguishable by maximizing modularity. Fortunately, this problem can be solved within the framework of modularity maximization in a hierarchical manner as shown by Arenas et al [12].

The nature of modularity optimization is to compare the modularity values among a large number of partitions of a network. This needs to evaluate the modularity function many times and consequently demands much computational effort. Nonetheless, if a partition is obtained by just changing the community index of a single node from another partition, the difference of modularity between the two community structures can be calculated at very little expense. Moving a single node at a time is thus a very efficient method to drive the system into a local optimum and is often employed as a refinement technique. For example, Ruan and
Zhang [13] adopted this technique in the refining stage of their two-stage procedure. Schuetz and Caflisch [14] also used it to improve the results obtained by the multi-step greedy algorithm. When incorporated in some appropriate global search strategies, however, single-node-move may play the key role in finding partitioning solutions. For example, Guimera and Amaral [15] and later Reichardt and Bornholdt [16] used the simulated annealing (SA) optimization method to avoid the single-node-move procedure getting trapped in local optima.

Although significant advances in community detection methods have been obtained, there are problems relating to efficiency and memory requirements, and new algorithms are still of essential interest to treat very large networks and get better results. For this purpose, we present here a method, called the contraction–dilation (CD) algorithm, which is simple but efficient. Compared with existing algorithms [13, 17, 18], the proposed method can find better solutions with higher modularity in less CPU time.

2. The CD algorithm

According to Newman and Girvan [7], the modularity of a network with \( n \) nodes and \( m \) edges can be written as

\[
Q(c) = \frac{1}{2m} \sum_{i,j=1}^{n} \left( A_{ij} - \frac{d_i d_j}{2m} \right) \delta(c_i, c_j),
\]

where \( A \) is the adjacency matrix of the network, the element \( A_{ij} = 1 \), if an edge is present between nodes \( i \) and \( j \), and 0 otherwise. \( d_i \) is the degree of node \( i \), \( \delta(x, y) = 1 \) if \( x = y \); and 0 otherwise. The community index of node \( i \) is denoted by \( c_i \in \{1, 2, 3, \ldots, c_{\text{max}}\} \) \((i = 1, 2, \ldots, n)\), where \( c_{\text{max}} \) is a predefined maximal number of communities. In practice, the performance of the algorithm is not sensitive to \( c_{\text{max}} \), which can be assigned a large enough value.

The goal of a modularity-based optimization algorithm is to find an integer vector \( c \) that maximizes \( Q \). The main operation of our algorithm is to move a single node, which can be described by a least change operator, denoted by \( T_{i\alpha} \), acting on community index vector \( c \) to yield a new community structure \( c' \). Formally, moving node \( i \) to community \( \alpha \) is represented as

\[
c' = T_{i\alpha}c,
\]

where

\[
c'_j = \begin{cases} 
\alpha, & \text{when } j = i, \\
c_i, & \text{for all other } j.
\end{cases}
\]

The difference of modularity caused by moving a single node can be deduced as

\[
\Delta_{i\alpha}(c) \equiv Q(T_{i\alpha}c) - Q(c) = \frac{d_i(\alpha) - d_i(c_i)}{m} + \frac{d_i[d(c_i) - d(\alpha) - d_i]}{2m^2}, \quad \alpha \neq c_i,
\]

where \( d_i(\alpha) \) is the number of node \( i \)'s connecting partners in community \( \alpha \), and \( d(\alpha) \) the total degree of the nodes in community \( \alpha \). By using (4), the modularity difference caused by a single-node-move can be computed at cost \( O(1) \); this is the reason why our search strategy relies mainly on the single-node-move operations.

Our algorithm starts with all nodes randomly distributed in \( c_{\text{max}} \) communities. To improve this initial community partition, the algorithm repeatedly moves a single node to
the community that maximizes the increment of modularity until no further improvement can be obtained. Because the differences of modularity can be computed very efficiently, the adjustment of nodes’ communities proceeds fast. In this process, nodes gradually aggregate into a smaller number of communities, more and more communities become empty. Since the number of effective (non-empty) communities decreases monotonically, we call this a contraction process.

After a contraction, all nodes are arranged in relatively appropriate communities with a much higher modularity value. This state is a local optimum with respect to the single-node-move operations, i.e. the modularity cannot be increased by changing the community of any single node. To find better community structures, other operations are thus needed to drive the system to escape from the local optimum. We use a simple method that migrates some of the nodes into randomly chosen communities, forcing the number of effective communities to increase suddenly, and the procedure is thus called dilation.

In a dilation process, all nodes change their community indices with a given probability \( p \), called the perturbation rate, which controls the destruction level of the output community structure of the contraction process. When \( p = 0 \), the community structure remains unchanged; when \( p = 1 \), the community structure is totally dissimilar to the output result of the contraction process. For \( p \) values between 0 and 1, a part of the previous community structure is reserved, so the following optimization can hopefully make use of earlier achievements.

The whole procedure applies the contraction and dilation operations iteratively, as shown in the following algorithm:

\[
\text{(Initialization)}
\]

1. Settle all nodes randomly in \( c_{\text{max}} \) communities; record the community structure as \( c_{\text{best}} \) and the \( Q \) value as \( Q_{\text{best}} \)
2. \textbf{repeat} (Contraction)
3. \hspace{1em} \textbf{repeat}
4. \hspace{2em} \textbf{for each} node \( i \)
5. \hspace{3em} move node \( i \) to community \( \beta = \arg \max_\alpha \Delta_{i\alpha} \)
6. \hspace{2em} \textbf{endfor}
7. \hspace{1em} \textbf{until} \( Q \) cannot be increased anymore
8. \hspace{1em} if \( Q > Q_{\text{best}} \), update \( c_{\text{best}} \) and \( Q_{\text{best}} \) (Dilation)
9. \hspace{1em} \textbf{for each} node \( i \)
10. \hspace{2em} with perturbation rate \( p \) move node \( i \) to a randomly chosen community
11. \hspace{2em} \textbf{endfor}
12. \hspace{1em} \textbf{until} NITER times

In the above algorithm, NITER is the given number of iterations of the contraction and dilation processes; \( \arg \max_\alpha \Delta_{i\alpha} \) denotes the value of argument \( \alpha \) that maximizes \( \Delta_{i\alpha} \).

As the contraction and dilation processes proceed, the modularity gradually reaches a high level, further application of the procedures becomes unlikely to find better community structures. The whole procedure should stop when this happens. Experimental observations show that it takes about 30–70 iterations to reach this stage, with usually rather high \( Q \) values.
By end of the contraction and dilation processes, most nodes aggregate in tight groups, but there are still a small proportion of nodes forming small and loosely connected communities. This is usually unfavorable to good partitioning in terms of modularity. To overcome this problem, we try eliminating the unfavorable communities to reduce the intergroup links and increase the modularity. The elimination is realized by a greedy community merging procedure, an effective technique also used by other authors [13, 19]. Numerical experiments show that the final merging indeed raises the modularity values by small, but substantial, differences. Since the community merging usually changes states of a small part of the nodes, it actually acts as a refinement process. For networks with relatively clear community structures, the contraction and dilation procedure alone is enough to achieve satisfactory results. To efficiently identify a good merging, we precompute the change to $Q$ for each potential merging. Specifically, the difference of modularity caused by merging communities $\alpha$ and $\beta$ is given by

$$
\Delta_{\text{merge}}(\alpha, \beta) \equiv \frac{e(\alpha, \beta)}{m} - \frac{d(\alpha)d(\beta)}{2m^2}, \quad \alpha \neq \beta,
$$

where $e(\alpha, \beta)$ is the number of inter-community edges.

3. Results and discussion

To test the performance of our algorithm CD, we applied it to a family of computer-generated networks with heterogeneous community structures and some real-world networks, and compared it with Newman’s spectral with refinement algorithm (SR) [18], Ruan and Zhang’s Qcut algorithm (Qcut) [13], as well as Lehmann and Hansen’s mean field (MF) optimization algorithm [17]. Finally, we tested our algorithm on a class of benchmark graphs, as suggested recently by Lancichinetti et al [20]. We performed all the tests on a computer server with two Intel X5460 CPUs and 16 GB RAM. It should be noted that only one CPU core was used because all the programs are not parallel.

3.1. Computer-generated networks—Zipf block model

Network models with controllable explicit structures are indispensable to test and compare community detection algorithms. The random block model with evenly sized communities has been extensively employed as a benchmark [7, 21]. In such model networks, all intra- and inter-community node pairs are connected with probabilities $p_{\text{in}}$ and $p_{\text{out}}$, respectively. In spite of the simplicity and ease of use, this model is oversimplified.

Considering the inhomogeneity of most realistic networks, we use a new block model with heterogeneous community structures controlled by an extra parameter $\gamma$. Community sizes in this model are designed to obey Zipf’s law [22] with exponent $\gamma$. This can be realized by assigning every node to $\alpha$th community with probability

$$
q_\alpha = \frac{\alpha^{-\gamma}}{\zeta_k(\gamma)}, \quad \alpha = 1, 2, \ldots, k,
$$

where $k$ is the number of designed communities and

$$
\zeta_k(\gamma) = \sum_{\alpha=1}^{k} \alpha^{-\gamma},
$$

which guarantees normalization of the probabilities.
The probability of a node establishing a link with an outside-community partner is set to be $p_{\text{out}}$. The overall average node degree is denoted by $\langle d \rangle$, the link probability of a pair of nodes belonging to the same community $\alpha$ can be written as

$$p_{\text{in}}(\alpha) = \frac{\langle d \rangle - np_{\text{out}}}{n_{\alpha}} + p_{\text{out}},$$

where $n_{\alpha}$ and $n$ are the numbers of nodes in community $\alpha$ and the whole network, respectively. Obviously, the intra-community link densities are different. Define community uncertainty as

$$\chi = \frac{n p_{\text{out}}}{\langle d \rangle}.$$  

It can be seen from (8) that as $\chi$ goes from 0 to 1, the model presents from clear to ambiguous community structures. When $\chi = 1$, the designed communities are totally undetectable because $p_{\text{in}} = p_{\text{out}}$.

Let $Q_{\text{design}}$ be the modularity value corresponding to the model designed community structure. Deduced from (1, 6–9), $Q_{\text{design}}$ for a Zipf block model network turns out to be

$$Q_{\text{design}} = \left(1 - \sum_{\alpha=1}^{k} q_{\alpha}^2\right) (1 - \chi).$$

In this work, we design a family of Zipf block model networks. For each network, we set $n = 3000$, $\langle d \rangle = 50$ and $\gamma = 0.6$. The nodes in each network are divided into 10 communities. So the biggest community has about 670 nodes, whereas the smallest one has about 170 nodes. The community uncertainty $\chi$ is used as the control parameter to change detectability of the community structures. We vary $\chi$ from 0.64 to 0.89, and the two extreme cases correspond to 29 and 40 outside-community links for each node on average, respectively. Link densities of this model with two different $\chi$ values are depicted in figure 1.

Figures 2(a)–(c) show the results of four community detection methods, CD, Qcut, SR and MF applied to the model networks. We see from figure 2(a) that CD uniformly outperforms SR and MF in maximizing $Q$. When $\chi$ is small, CD and Qcut find almost the same $Q$ values. As $\chi$ becomes larger, the community structure becomes ambiguous and CD finds higher $Q$ value than Qcut. It should be noted that the community structure is not significantly clear when $\chi = 0.64$ since a node will have on average more inter- than intra-community links. The $Q$ values found by CD are higher than model designed ones for all $\chi$ values. Since the model community structure is known in advance, the fraction of nodes classified in accordance with the model designed community indices can be computed. As shown in figure 2(b), when $\chi$ is less than 0.78, CD and Qcut successfully detect all communities with almost 100% coincidence. When $\chi$ is larger than 0.78, though CD can find higher $Q$ values than model designed ones, the distribution of nodes into communities is no longer identical to the model designed community structure and thus the coincidence decreases dramatically. This low coincidence is caused by the merging of multiple communities into larger ones, which is in accordance with the predictions of Fortunato and Barthelemy concerning resolution limits of community detection [11]. As shown in figure 2(c), for example, when $\chi = 0.89$, CD finds on average seven out of the ten designed communities.

3.2. Choice of perturbation rate $p$

We use a Zipf block model with $\chi = 0.84$ to investigate the influence of perturbation rate $p$ on overall performance of the algorithm. As seen from figure 3, the average CPU time increases
Figure 1. Link densities of the Zipf block model used in this work. The number of nodes is set to 3000 and the average degree of each node is fixed at 50. All nodes are assigned to 10 communities, whose sizes obey Zipf’s law with exponent $\gamma = 0.6$. The blockwise link densities are determined by the community uncertainty $\chi$. When $\chi$ approaches 1, the community structures become more and more ambiguous. In our numerical experiments, $\chi$ ranges from 0.64 to 0.89 corresponding to relatively clear to ambiguous community structures. The left and right parts show the link densities for the two extreme cases. Gray levels present link densities: the darker the color, the denser the links. Note that when $\chi = 0.89$, the difference between intra- and inter-community link densities are very small so that the communities are very hard to detect.

monotonically when $p$ increases. This is reasonable because larger perturbation rates lead to more nodes being driven to random communities in the dilation procedure and thus result in more effort to arrange them in new suitable locations. However, the longest CPU time does not mean the maximum $Q$ value. With the same iteration times, $Q$ has a peak when $p$ equals 0.6 and the corresponding CPU time is about 3 s. As $p$ becomes larger than 0.6, $Q$ decreases. When $p = 1$, the found modularity value is the worst though it takes longer time to compute. Selecting an appropriate $p$, therefore, can dramatically improve the performance in terms of both the CPU time and $Q$ value. Our numerical experiments show that a choice of $p$ ranging from 0.4 to 0.8 is suitable for most tested cases.

3.3. Real-world networks

In order to further test our algorithm, we chose a set of networks of different sizes. The networks are, the karate club network of Zachary [23], the network of American football games between Division IA colleges [24], a metabolic network of the *C. elegans* [25], a network of email contacts at a university [26], two protein–protein interaction (PPI) networks of *Escherichia coli* and *Saccharomyces cerevisiae* [27], and a coauthorship network of scientists working in condensed matter physics [28].

Performance on real-world networks is concluded in table 1. CD clearly outperforms MF and SR in terms of both quality (higher $Q$ values) and efficiency (less CPU time) for networks
Figure 2. Comparison of the performance of four community detection methods, CD, SR, Qcut, as well as MF on Zipf block model networks. (a) Difference between $Q_{\text{found}}$ and $Q_{\text{design}}$. $Q_{\text{found}}$ and $Q_{\text{design}}$ denote the algorithm found and model designed modularity values, respectively. (b) As the model’s community structure is known in advance, the fraction of nodes classified in accordance with the model designed community indices can be computed, which we call coincidence. (c) The number of communities found by these four algorithms. Each data point is the mean and standard deviation for 100 network realizations.

with more than 100 nodes. Besides, CD has identical performance to Qcut for very small networks, but is faster than Qcut and simultaneously reaches higher $Q$ values for larger networks. MF is sensitive to the number of initial communities, $n_{c0}$. Larger $n_{c0}$ leads to smaller $Q$ values. For instance, when $n_{c0} = 300$, MF takes more than 4 min to reach its largest found modularity, $Q_{\text{max}} = 0.4873$, of the PPI network of S. cerevisiae. Whereas, when $n_{c0} = 20$, it takes $\sim 0.7$ min to reach a higher $Q_{\text{max}} = 0.5459$. This is a shortcoming of MF because in practice it is not trivial to guess a good value of $n_{c0}$. For all tested networks, $n_{c0}$ for MF is tuned to its optimum values. In comparison, other methods do not need to set $n_{c0}$ with much care.

SA is a common global optimization technique that has been used in community detection [15, 16]. The main difference between SA and CD is that SA avoids getting trapped
Figure 3. Test for the parameter perturbation rate $p$. The figure represents the variation of the modularity value found by CD and CPU time with the perturbation rate. The unit of time is seconds. Each data point is the mean and standard deviation of 50 independent runs.

Table 1. Test results of community detection methods on real-world networks.

| Network     | Size $n$ | SR $Q$  | Time (s) | Qcut $Q$  | Time (s) | MF $Q$  | Time (s) | CD $Q$  | Time (s) |
|-------------|----------|---------|----------|-----------|----------|---------|----------|---------|----------|
| Zachary     | 34       | 0.4188  | <1       | 0.4198    | <1       | 0.4198  | <1       | 0.4198  | <1       |
| Football    | 115      | 0.6009  | <1       | 0.6046    | 1        | 0.6006  | <1       | 0.6046  | <1       |
| Met. C. elegans | 453  | 0.4348  | 1        | 0.4334    | 2        | 0.4372  | 1        | 0.4472  | <1       |
| Email       | 1133     | 0.5685  | 11       | 0.5762    | 6        | 0.5469  | 4        | 0.5789  | <1       |
| E. coli PPI | 1850     | 0.3756  | 40       | 0.3940    | 33       | 0.3760  | 16       | 0.4009  | 1        |
| S. cer. PPI | 4969     | 0.5448  | 1020     | 0.5608    | 115      | 0.5459  | 41       | 0.5668  | 4        |
| Collaboration | 27519 | 0.7230a | __b     | 0.7454    | 2780     | __b     | __b     | 0.7535  | 484      |

$^a$ The modularity value found by SR is directly taken from [18].
$^b$ For the collaboration network, the programs for MF and SR ran out of memory and resulted in no output.

in local optima through allowing some unfavorable node movements decreasing the modularity. A large number of unfavorable movements in SA lowers down the search efficiency. Table 2 is the test results of an SA method proposed by Guimera and Amaral [15] with the cooling factor set to 0.8, 0.95 and 0.995, and other parameters fixed as recommended by the authors. It can be seen that SA yields comparably good $Q$ values but takes much longer computational time.

3.4. A class of benchmark networks

Recently, Lancichinetti et al [20] have introduced a model to generate networks with power-law distributed node degrees (with exponent $\gamma$) and community sizes (with exponent $\beta$) to simulate the main feature of a wide range of real-world networks. The ability of an

New Journal of Physics 11 (2009) 043025 (http://www.njp.org/)
Table 2. Test results of SA with different cooling factors (CF).

| Network         | Size n | CF = 0.8 Q  | CF = 0.8 Time (s) | CF = 0.95 Q  | CF = 0.95 Time (s) | CF = 0.995 Q  | CF = 0.995 Time (s) |
|-----------------|--------|-------------|-------------------|-------------|-------------------|-------------|-------------------|
| Zachary         | 34     | 0.4198      | 1                 | 0.4198      | 1                 | 0.4198      | 4                 |
| Football        | 115    | 0.6044      | 2                 | 0.6044      | 4                 | 0.6046      | 46                |
| Met. C. elegans | 453    | 0.4388      | 98                | 0.4448      | 175               | 0.4470      | 1495              |
| Email           | 1133   | 0.5662      | 696               | 0.5773      | 1625              | 0.5779      | 14480             |
| E. coli PPI     | 1850   | 0.3874      | 1817              | 0.3941      | 3924              | —           | —                 |
| S. cer. PPI     | 4969   | 0.5643      | 11145             | —           | —                 | —           | —                 |
| Collaboration   | 27519  | —           | —                 | —           | —                 | —           | —                 |

^a Computation was not finished in 20 000 s.

Figure 4. Test of CD on the benchmark networks proposed by Lancichinetti et al. The number of nodes is set to N = 20 000. The four panels correspond to four pairs for the exponents (γ, β) = (2, 1), (2, 2), (3, 1) and (3, 2). Values of the overall average node degree ⟨k⟩ are set to 20 and 25. Each curve shows the variation of the normalized mutual information with the mixing parameter µ. Each data point is the mean and standard deviation of 10 graph realizations.
algorithm to rediscover the designed community structures is of great interest for the purpose of estimating the performance on real-world networks. We use the software of Lancichinetti et al (http://santo.fortunato.googlepages.com/benchmark.tgz) to test the accuracy of our algorithm in the determination of designed community structures. To evaluate the accuracy, we employ a measure, normalized mutual information, borrowed from information theory, as used in [20, 21], to compute similarity between the model designed partitions and algorithm found ones. The normalized mutual information ranges from 0 to 1, with 1 implying a perfect agreement.

The ambiguity of the community structures in Lancichinetti et al’s model is controlled by a mixing parameter $\mu$, which is defined as the expected proportion of outside links to the total ones of every node. When $\mu > 0.5$, most links of each node connect to outside partners, therefore the community structures are rather ambiguous. From figure 4 we see that even in the ambiguous regions, community structures found by CD strongly agree with the models. On the same models, the methods tested in [20] gave significantly lower agreements.

### 3.5. Influence of NITER

In general, more runs of the contraction and dilation processes result in higher modularity values. However, as the optimization proceeds, better partitioning is gradually harder to find, i.e. the search efficiency decreases. To maximize the efficiency, a tradeoff should be made between effort and gain. In practice, a moderate number of runs of the contraction and dilation processes is usually enough to produce good results, because the deficiency of inadequate optimization can be compensated by the final refinement. This can be seen from figure 5 where we present examples of the influence of different values of NITER on the performance of CD, for the collaboration network [28] and the *S. cerevisiae* protein–protein network [27]. We see from the figure that CD yields high modularity solutions with few iterations of the contraction and dilation processes. Satisfactory modularity values can be reached when NITER is approximately set to 30. For the collaboration network, we tested nine different values of NITER. When NITER equals 5, CD has already found a relatively
Figure 6. The optimal community structures of Zachary’s karate club network at three resolution scales. Nodes belonging to the same community are depicted in the identical colors and shapes. The case of $r = 0$ corresponds to the best community structure of the original network.

higher modularity value, which is larger than the one found by Qcut and SR (table 1). When NITER varies from 30 to 70, the modularity increases slightly and reaches the maximum value. The result for the S. cerevisiae PPI network is similar, and CD needs less runs of contraction and dilation processes to obtain the highest modularity value. Experimental observations show that a choice of NITER ranging from 30 to 70 is suitable for most tested cases.

4. Conclusions

To find communities by maximizing network modularity, we propose a simple and efficient method using a CD strategy. The contraction procedure optimizes community structures based essentially on the single-node-move operation at very low computational expense. This procedure rapidly decreases the number of communities assigned initially and thus is called contraction. To drive the system to escape from local optima, we use the dilation procedure that partially destroys the community structures found by the contraction procedure and increases

New Journal of Physics 11 (2009) 043025 (http://www.njp.org/
the number of communities to a large number. When the contraction and dilation procedures eventually finish, we apply a merging procedure to refine the community structure.

The proposed method outperforms existing algorithms in terms of higher modularity values found and accuracies in less CPU time for computer-generated networks as well as real-world examples. The required memory for CD is approximately $O(n c_{\text{max}})$, which is basically used to store the data $d_i(\alpha)$ (the number of node $i$’s connecting partners in community $\alpha$). The low memory demand is an implicit strength of the algorithm and makes it possible to analyze very large networks. In contrast MF and SR could not be applied to the collaboration network with 27,519 nodes because of the high memory requirements in our tests.

It is also worth noting that our algorithm can be directly applied to weighted networks. Owing to this characteristic, it is easy to use CD in Arenas et al’s approach [12] to cope with the resolution limit problem of modularity-based community detection methods. For example, the original Zachary’s karate club network was detected to have four communities, as shown in figure 6(a). According to Arenas et al’s method, we rescale the network topology by adding self-loops of weight $r$ for every node, i.e. adding $r$ to the diagonal entries of the adjacency matrix. When $r = -2$, the optimal partition has two communities (figure 6(b)). As $r$ increases, the optimal communities become gradually smaller, as shown in figure 6(c) for $r = 2$. The best partitions in terms of modularity for different $r$ values reveal the community structures of the original network at multiple resolution levels.

Acknowledgments

We thank Mark E J Newman, Roger Guimera, Jianhua Ruan and Sune Lehmann for kindly sharing their computer programs. This work was supported by the State High-Tech Development Program of China (no. 2006AA020204).

References

[1] Schuster S, Pfeiffer T, Moldenhauer F, Koch I and Dandekar T 2002 Exploring the pathway structure of metabolism: decomposition into subnetworks and application to Mycoplasma pneumoniae Bioinformatics 18 351–61
[2] Flake G W, Lawrence S, Giles C L and Coetzee F M 2002 Self-organization and identification of Web communities Computer 35 66–71
[3] Wasserman S and Faust K 1994 Social Network Analysis (Cambridge: Cambridge University Press)
[4] Liao T W 2005 Clustering of time series data—a survey Pattern Recognit. 38 1857–74
[5] Brohee S and van Helden J 2006 Evaluation of clustering algorithms for protein–protein interaction networks BMC Bioinformatics 7 488
[6] Filippone M, Camastra F, Masulli F and Rovetta S 2008 A survey of kernel and spectral methods for clustering Pattern Recognit. 41 176–90
[7] Newman M E J and Girvan M 2004 Finding and evaluating community structure in networks Phys. Rev. E 69 026113
[8] Shi J and Malik J 2000 Normalized cuts and image segmentation IEEE Trans. Pattern Anal. Mach. Intell. 22 888–905
[9] Carballido-Gamio J, Belongie S J and Majumdar S 2004 Normalized cuts in 3-D for spinal MRI segmentation IEEE Trans. Med. Imaging 23 36–44
[10] Boykov Y, Veksler O and Zabih R 2001 Fast approximate energy minimization via graph cuts IEEE Trans. Pattern Anal. Mach. Intell. 23 1222–39

New Journal of Physics 11 (2009) 043025 (http://www.njp.org/)
[11] Fortunato S and Barthelemy M 2007 Resolution limit in community detection Proc. Natl Acad. Sci. USA 104 36–41
[12] Arenas A, Fernandez A and Gomez S 2008 Analysis of the structure of complex networks at different resolution levels New J. Phys. 10 053039
[13] Ruan J and Zhang W 2007 Identifying network communities with a high resolution Phys. Rev. E 77 016104
[14] Schuetz P and Caflisch A 2008 Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement Phys. Rev. E 77 046112
[15] Guimerà R and Amaral L A N 2005 Functional cartography of complex metabolic networks Nature 433 895–900
[16] Reichardt J and Bornholdt S 2006 Statistical mechanics of community detection Phys. Rev. E 74 016110
[17] Lehmann S and Hansen L K 2007 Deterministic modularity optimization Eur. Phys. J. B 60 83–8
[18] Newman M E J 2006 Modularity and community structure in networks Proc. Natl Acad. Sci. USA 103 8577–82
[19] Pujol J M, Bejar J and Delgado J 2006 Clustering algorithm for determining community structure in large networks Phys. Rev. E 74 016107
[20] Lancichinetti A, Fortunato S and Radicchi F 2008 Benchmark graphs for testing community detection algorithms Phys. Rev. E 78 046110
[21] Danon L, Duch J, Diaz-Guilera A and Arenas A 2005 Comparing community structure identification J. Stat. Mech. P09008
[22] Zipf G K 1932 Selective Studies and the Principle of Relative Frequency in Language (Cambridge, MA: Harvard University Press)
[23] Zachary W W 1977 An information flow model for conflict and fission in small groups J. Anthropol. Res. 33 452–73
[24] Girvan M and Newman M E J 2002 Community structure in social and biological networks Proc. Natl Acad. Sci. USA 99 7821–6
[25] Duch J and Arenas A 2005 Community detection in complex networks using extremal optimization Phys. Rev. E 72 027104
[26] Guimera R, Danon L, Diaz-Guilera A, Giralt F and Arenas A 2003 Self-similar community structure in a network of human interactions Phys. Rev. E 68 065103
[27] Xenarios I, Rice D W, Salwinski L, Baron M K, Marcotte E M and Eisenberg D 2000 DIP: the database of interacting proteins Nucleic Acids Res. 28 289–91
[28] Newman M E J 2001 The structure of scientific collaboration networks Proc. Natl Acad. Sci. USA 98 404–9