# Adaptive clustering algorithm for community detection in complex networks

Zhenqing Ye,[1] Songnian Hu,[1,2] and Jun Yu[1,2,*]
[1]*James D. Watson Institute of Genome Sciences, Zhejiang University, Hangzhou, China*
[2]*CAS Key Laboratory of Genome Sciences and Information, Beijing Institute Genomics, Chinese Academy of Sciences, Beijing, China*
(Received 19 June 2008; published 30 October 2008)

Community structure is common in various real-world networks; methods or algorithms for detecting such communities in complex networks have attracted great attention in recent years. We introduced a different adaptive clustering algorithm capable of extracting modules from complex networks with considerable accuracy and robustness. In this approach, each node in a network acts as an autonomous agent demonstrating flocking behavior where vertices always travel toward their preferable neighboring groups. An optimal modular structure can emerge from a collection of these active nodes during a self-organization process where vertices constantly regroup. In addition, we show that our algorithm appears advantageous over other competing methods (e.g., the Newman-fast algorithm) through intensive evaluation. The applications in three real-world networks demonstrate the superiority of our algorithm to find communities that are parallel with the appropriate organization in reality.

DOI: 10.1103/PhysRevE.78.046115 PACS number(s): 89.75.Hc, 05.10.−a

## INTRODUCTION

In the real world, interactions and connections of our multifacet life and surroundings are dominated by various network relationships from the internet and power grids to metabolic pathways that generate material and energy to fuel our body. These networks can be illustrated as a graph with nodes involving entities and edges, which is a manifestation of functional association or relevance. The study of these complex networks has attracted particular interest in the past decade [1–3]. Statistical analyses of vast datasets from biological, technological, and sociological networks have revealed some common properties in complex networks, such as their small-world character and scale-free degree distributions [4,5].

Recently, the characterization of community (or module) structures in complex networks has received a considerable amount of attentions [6–8], and it is widely believed that large networks in the real world are composed of many sublevele ingredients, such as community structure or module conformation, and that vertices are densely connected within modules while being loosely connected to the rest of the networks. Communities are of interest because they often correspond to behavioral or functional units, such as cell-cycle controls or metabolic pathways in biological networks or social groups within social networks [9,10]. The presence of communities can dramatically alter the behavior of dynamical processes on networks [11] and can be used as bases for reduction or coarse graining of networks for visualization, understanding, and other purposes [12]. Therefore revealing constituents and their relatedness in networks undoubtedly brings richer information for gaining insight into structures and dynamics of many interesting definable systems around us.

Although many questions remain to be addressed, the detection of functional modules is a primary step. A number of mathematical tools and computer algorithms have been recently designed to tackle community detection in complex networks [13–16]. Girvan and Newman [17] have proposed a divisive clustering algorithm that recursively remove edges with the highest betweenness scores that are believed to run between communities, a hierarchical tree is subsequently constructed, and various partitions can be obtained by cutting the tree at different points. However, this method demands too expensive a computational cost that leads to impossible application to large real-world networks. To overcome this problem, Newman [18] suggested an alternative fast agglomerative clustering algorithm based on the definition of modularity (typically represented by a $Q$ function) over possible divisions of a network. For a network with $n$ nodes, this method starts with an initial state where each vertex is a sole member of one community among $n$ communities, and it then repeatedly joins communities together in pairs; the procedure often results in the greatest increase or smallest decrease in $Q$. Similar to the divisive method, a hierarchical tree can be constructed and divided at the point with the highest $Q$ value. This method shows a significant speed gain and moderately improved performance. However, regardless of whether the edge-betweenness-based divisive or the modularity-oriented agglomerative clustering approaches are utilized, one major drawback persists where vertices have no chance to shuffle between modules that have been established in previous steps but are scrambled within modules for further division or combination in the subsequent steps. These traditional clustering methods all have intrinsic problems that hinder the improvement efforts [8,15].

Since a higher quantity of modularity indicates a better quality of module structure for a network, many authors have made great efforts to seek the best topological division by optimizing $Q$ function directly, relying upon various techniques, including simulated annealing [19], external optimization [16], and matrix spectral analysis [14,15]. Unfortunately, exhaustive optimization of modularity requires an impractically large computational effort; these algorithms are generally approximate techniques, more or less suffering

*junyu@big.ac.cn

from heavy computational costs or inaccurate performance [20]. In addition, we need to pay attention to modularity optimization that does not always bring the best clustering in some cases; for instance, it suffers from resolution limit as shown by Fortunato and Barthelemy [21], where small modules tend to merge into bigger ones. However, optimization strategy is still an effective way to detect community structure in complex networks at the present time.

Here, we introduce a different adaptive clustering method (AdClust) that is able to overcome limitations that traditional clustering (divisive or agglomerative) algorithms encounter. It is simply topology oriented, and has no demands for other predefined parameter tuning present in other methods, such as *k*-mean clustering [22]. In our approach, vertices are treated as autonomous agents capable of adapting themselves to appropriate modules according to their local context through a self-organization process, and eventually reach a satisfactory solution in a global scale. Furthermore, we applied our algorithm to three real-world networks to demonstrate successful detection for community structures.

## ALGORITHM

For a given decomposition of an undirected simple network, Newman and Girvan [23] proposed a quantitative modularity to measure the quality of module structure as follows:

$$Q = \sum_i (e_i - a_i^2), \tag{1}$$

where $ei$ is the fraction of edges that connect two nodes inside the module $i$; $ai$ is the fraction of edges that have one or both vertices inside the module $i$. The maximal modularity corresponds to the partition that comprises the most inner edges within modules and the least outer edges between modules. This concept is essential to many algorithms, and also paves the way for our approach where we also pursue the best $Q$ value but assume a different model for the optimization process.

The principle of our algorithm is for each vertex to repeatedly adapt itself to follow the most attractive module according to its surrounding neighbours and local structures until it reaches a satisfactory assignment that is naturally compatible with a better outcome of modularity $Q$. We illustrate the essence of our assumption in Fig. 1. In a network with an initial community structure, each vertex is generally related to two forces. One force ($F_{in}$) comes from the module where a vertex belongs to and is responsible for holding the vertex in its position. The opposite force ($F_{out}$) originates from all other connected modules, pulling the vertex to its potential target modules. A battle among the two forces eventually divides all vertices either to escape for their new destinies or to stay put at their host or source modules. The critical measure that determines the motion of vertices is to pursue a maximal increase of modularity $Q$ within a local context. Based on this assumption, we define our forces as positively proportional to the increase of $Q$ when the vertex merges into a relevant module, and formulate them as follows:

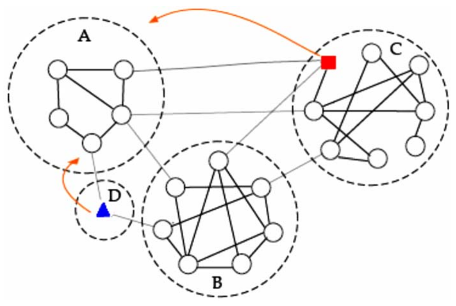

FIG. 1. (Color online) A illustration of adaptive vertex movements according to the force rules. For the square vertex, we can calculate its forces and obtain the results as follows: $F_{out}^{(A)}=0.333 > F_{in}^{(C)}=0.074 > F_{out}^{(B)}=-0.147$, and it should move to module A (arrowed line). Although a similar situation happens to the triangular vertex, we should take care of the influence from the first motion of the square vertex as it may alter the forces of the triangular vertex. Moreover, we note that the motion of the triangular vertex reduces the total number of modules. Such a self-organization process happens to all vertices iteratively until they become stabilized.

$$F_{in}^{(s)} = e_{in}^{(s)} - \frac{k(d_{in}^{(s)} - k)}{2E}, \tag{2}$$

$$F_{out}^{(t)} = \max_{t \in \{ngbs\}} \left\{ e_{out}^{(t)} - \frac{kd_{out}^{(t)}}{2E} \right\}, \tag{3}$$

where $e_{in}^{(s)}$ is the number of edges that connect a vertex within the host module $s$, $d_{in}^{(s)}$ is the total degree of vertices within a source module. Similarly, $e_{out}^{(t)}$ and $d_{out}^{(t)}$ are the counterparts of $e_{in}^{(s)}$ and $d_{in}^{(s)}$ in a target module $t$. We should emphasize that a vertex only belongs to one source module, but may have more than one neighbouring module $\{ngbs\}$. $E$ denotes the sum of edges in a network, and $k$ is the degree of an immediate node we are focusing on. In fact, the terms $F_{in}$ and $F_{out}$ can also be considered as $F$=(actual links) −(expected links given that node $\nu$ is in the respective module). We subsequently calculate the two forces for each vertex allowing them to move to a group with the strongest pulling force.

When all vertices in the network stop moving and become stable, a structure for all modules emerges spontaneously, yielding a stage-specific $Q$ at this time. At the subsequent steps, we calculate the increase of $Q$ for all possible module pairs if they are combined, and merge the pairs with the maximal increase (or minimal decrease) in $Q$ values together. This amalgamation may alter the two forces of related vertices in the merged groups and drive other vertices near the boundaries to move away, and thus affects more vertices, provoking an avalanche of changes in the rest of the nodes, which ultimately spreads to all vertices in the entire network. Therefore we need to constantly compute forces of all vertices and move them according to the force-related rules after every merging event. We repeat such an adaptive procedure and merge modules until all vertices converge into one module. We describe the procedure as the following steps.

(1) Separate vertices into $n$ modules.

(2) Calculate $F_{in}$ and $F_{out}$ for each vertex to make decisions either to move vertices to target modules if $F_{out} > F_{in}$ or to allow them to stay put if $F_{out} \leq F_{in}$.

(3) Repeat step 2 until all vertices are stabilized. As a modular structure emerges, record $Q$ for module partition at this stage.

(4) Calculate increases of $Q$ for all possible module pairs obtained in step 3.

(5) Choose the pairs with the maximal increase (or minimal decrease) of $Q$ for further mergers.

(6) Repeat steps 2–5 until all vertices form a single module.

(7) Choose a stage achieving the maximal $Q$ as the final solution.

In this algorithm, the second routine is an essential step where each vertex plays a role as an autonomous entity demonstrating "flocking behavior" within local environments. Each vertex only circumstantially decides its own fate to move preferably in a direction that leads to an increase of $Q$ without a pre-existing global plan. However, in reality the synchronized movement of multiple vertices appears to follow a global plan and creates a pattern toward an increasing modularity. The process where autonomous entities interact in such a way to create a global order is known as self-organization, as seen in different situations such as for ants to line up while transporting food and for birds to form a flock pattern for traveling long distances [24,25]. Therefore we named our method adaptive clustering algorithm or AdClust, inspired by these self-organization behaviors frequently observed in nature.

Comparing with the Newman-fast algorithm, our method introduces an additional self-organization routine for node traveling before module pairs merge, and this routine will take time $O(n)$. Since there is a maximum of $n-1$ joining operations among module pairs in the worst case scenario, AdClust should be similar to the Newman-fast algorithm in running time $O(n^2)$ on a sparse graph [18], but is theoretically slower as compared to the Newman-fast algorithm due to the additional routines. However, it is worth mentioning that the number of modules after the first self-organization routine is usually dramatically decreased; for instance, the initial 27 519 modules in the physicist network (detailed later) reduced to about 4500 modules after the first self-organization routine. This feature is quite different from the stepwise procedure of the Newman-fast method that reduces modules one at a time from the initial 27 519 modules. Therefore AdClust reduces the number of iterative steps for the mergence of module pairs.

## COMPETITIVE MERITS OF ADCLUST

AdClust is considered an agglomerative approach of hierarchical clustering techniques. Compared with others [13,15,17], we believe that the uniqueness of AdClust is its capability of self-organization where vertices are offered flexibility to move around seeking their best destinations according to their local contexts.

Let us look into an example. Newman [18] has introduced a fast and greedy strategy for modularity maximization. It initially separates each vertex uniquely into its respective module and proceeds by grouping the nodes together to yield a maximal increase of $Q$. By repeating such procedure, a hierarchical tree can be constructed with modules forming from the division of the tree at a maximal $Q$ value. This algorithm improves speed and performs well for a number of real-world large networks. However, as shown in Fig. 2, it fails when linking two modules with nodes of poor connectivity. For example, the two nodes, ⑦ and ⑧, are grouped together initially through the agglomerative procedure due to their maximal increment of $Q$ in early steps, and remain together until the end. This result contradicts the perfect division where the two-module structure gives maximal modularity. In nonadaptive algorithms, initial steps often lead to an unfavorable ending that is not repairable by steps thereafter [8]. Similar situations are also encountered when divisible clustering methods are used [15]. In contrast, our adaptive clustering method ultimately pulls the two nodes apart regardless if they are joined together at the first step or not. We observed that the best conformation with three groups (circled by dotted lines in Fig. 2) can be obtained by the fast-greedy algorithm, but the structure is not stable when our method is applied. When the two nodes (⑦ and ⑧) are grouped together, the inner force on node ⑦ in module B is 0.784, and the outer force from module A is 1.216; the same situation happens to node ⑧. Therefore a differential affinity ($f_{out} > F_{in}$) on the two nodes inevitably drives them apart. Each vertex then migrates toward A and B modules separately, forming a two-module structure.

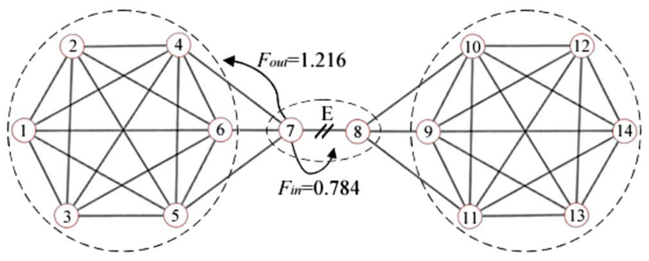

FIG. 2. (Color online) A sample network for a merit illustration of AdClust when competing against Newman-fast algorithm. The two nodes connected through edge $E$ are grouped together first by the agglomerative procedure due to their maximal increase of $Q$, and remain together to form an isolated group ultimately. However, it is clearly seen that $E$ may belong to different groups. In the adaptive clustering algorithm, the two nodes will be separated into their respective groups according to the simple force rules ($F_{out} > F_{in}$), leading to an optimal partition.

In traditional clustering methods, such as the Newman-fast algorithm, one major difficulty is the fact that vertices bundled together into a module in previous steps, and these vertices lose their chance to break away for potential improvement in subsequent steps. Furthermore, an ill-suited combination in early stage may lead to worse outcomes in later steps by iteratively magnifying and accumulating early mistakes, especially when such events are common to real complex networks.

## EVALUATION

One way to test the performance of AdClust is to apply it to *ad hoc* networks with a well-characterized module struc-

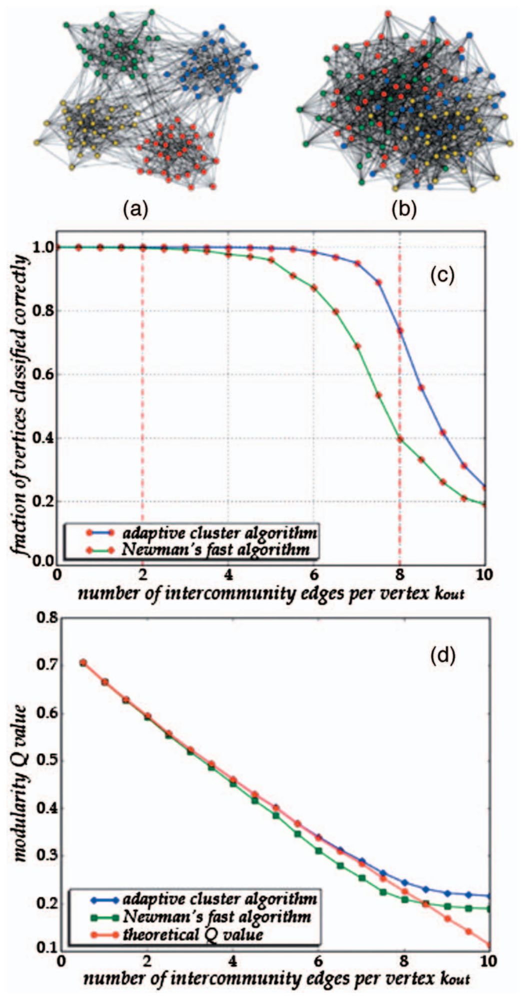

FIG. 3. (Color) Performance tests with artificial networks. In these computer-generated networks, four modules are predisposed according to various parameters described in the text. Each data point is an average over 100 graphs. As $k_{out}$ grows bigger, the boundary of module structure becomes more complicated. (a) When $k_{in}=14$ and $k_{out}=2$. (b) When $k_{in}=8$, $k_{out}=8$. (c) The fraction of nodes correctly classified by the AdClust. Newman-fast algorithm was used as a reference. (d) The average modularity obtained at each point. The theoretical value corresponds to the four predefined community structures.

ture to see if the algorithm is capable of recognizing and extracting the same structure [23]. We generated a series of artificial networks with $n=128$ nodes that are split into four communities; each contains 32 nodes. Edges between two nodes are introduced with different probabilities depending on whether the two nodes belong to the same group ($p_i$) or not ($p_o$): every node has $k_{in}=31p_i$ links on average to its partners in the same community, and $k_{out}=96p_o$ links to the outer world, keeping $k_{in}+k_{out}=16$. While $p_o$ (and therefore $k_{out}$) is varied freely, the value of $p_i$ should be chosen conditionally. As $k_{out}$ increases from zero, the modules become more diffused and harder to identify [Figs. 3(a) and 3(b)].

According to Newman and Girvan [23], we calculated the ratio of correctly classified nodes and the corresponding $Q$ values [Figs. 3(c) and 3(d)], using the Newman-fast algorithm as a reference. AdClust appears better in the detecting module structure than the Newman-fast algorithm; it reaches an accuracy of 74% as boundaries of communities become blurry when $k_{out}$ equals 8, which exceeds the value of 40% in the Newman-fast algorithm. Moreover, AdClust often yields higher $Q$ values than the theoretical predictions for a predefined pattern, suggesting that the modular structures it constructs is closer to the reality, and the predefined pattern would relatively lose its referential significance when the boundary becomes more and more blurry. For instance, a good modular quality of predefined structure may not be authentic when we create random graphs at a growing $k_{out}$ value.

We also would like to point out that AdClust is stochastic in nature since the order to calculate moving forces for vertices and to move them in subsequent steps is not predetermined. We do not enforce any rules to prioritize which vertex to start. To test its robustness, we performed 100 random runs with an artificial network [Fig. 3(b)] and recorded the fraction of times when two nodes are classified in the same group [Fig. 4(a)]. The module structure is clearly revealed, and most node pairs are either always classified or never classified into the same modules. We further calculated relative standard deviation ($RSD=1.68\%$) for $Q$ values for the 100 runs [Fig. 4(b)]. Even the smallest $Q$ from our algorithm is still higher than that of the Newman-fast algorithm (0.234 versus 0.228).

## APPLICATIONS

We applied AdClust to another well-known network analyzed by Zachary—the karaoke club network. This network is widely used as a test case for new methods for complex networks; it consists of 34 members of a karate club as nodes and 78 edges representing friendships among members of the club, which was observed over a period of two years. By chance, a dispute arose and the club eventually split into two smaller groups, centered on the club's administrator and its principle karate teacher [26].

As many authors have tested their approaches to this network for module detection and accuracy evaluation [15,16], we also offered the result based on AdClust, yielding a $Q$ value of 0.4198 (Fig. 5). Not only are the two groups well separated according to the reality but also our algorithm divides them further into four smaller fractions that revealed finer structures. Our algorithm gives rise to more precise answers than other competing algorithms (e.g., $Q=0.4188$ in Refs [16]) even when it works on such a simple yet real network.

A football network has also been used as a showcase to demonstrate some weakness of the Newman-fast algorithm [18], which records the schedule of games among American college football teams in a single season. This network contains 115 nodes and 1114 edges, each node represents a college football team and each edge represents the fact that two teams played together. Because the teams are divided into groups or conferences, with intraconference games being more frequent than interconference games, we have a reasonable idea ahead of time about what communities should be identified. We applied AdClust to this network and obtained

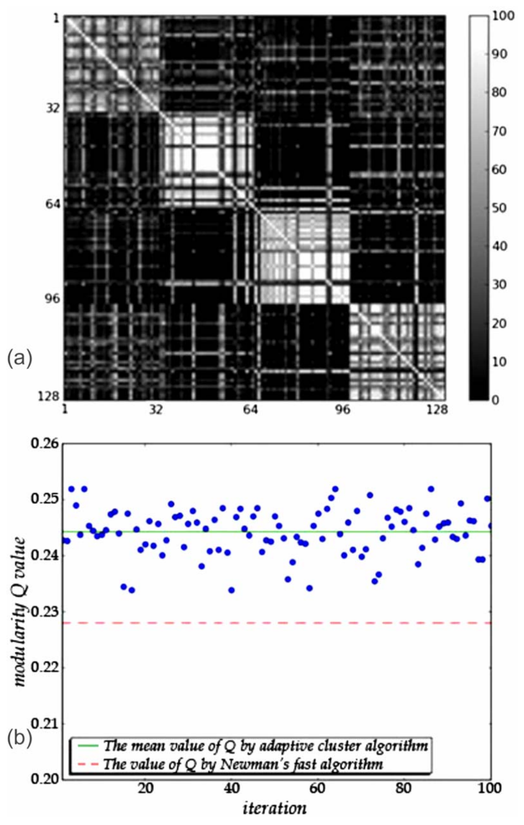

FIG. 4. (Color online) Robustness tests of AdClust. Due to the stochastic feature of AdClust, we performed 100 runs over the network as described in Fig. 3(b) to check for consistency among different partitions. (a) Fractions of timing when two nodes are classified into the same module indicated with a different gray scheme from black (low) to white (high). The modular structure is clearly revealed, and most pairs of nodes are either always classified into the same module (white) or never classified into the same module (black), which suggests that the solution is robust. (b) The modularity value $Q$ for each partition corresponds to each run. The horizontal solid line represents the mean value of total runs, and the dotted line represents the $Q$ value based on the Newman-fast algorithm as a reference.

the modularity $Q=0.605$ that is higher than the value reported, $Q=0.546$ [18]. As shown in Fig. 6, there are ten communities defined by AdClust, whereas the Newman-fast algorithm revealed six communities only [18]. Since the true community structure has been previously known, we can evaluate the two experimental partitions (AdClust and Newman-fast) against the standard classification by the NMI (normalized mutual information) index to measure which partition is closer to the right classification [20]. Our result ($\text{NMI}_{\text{ad}}=0.890$) is significantly better than what is yielded from the Newman-fast algorithm ($\text{NMI}_{\text{fast}}=0.685$).

We further applied Adclust on a large complex network: the collaboration network of scientists working in condensed matter physics. The network contains 27 519 nodes and 116 181 edges and it has been widely used by several authors as a test bed for community-finding algorithms on large networks [14,16]. We obtained $Q=0.761$ with AdClust, greater than the value of $Q=0.679$ arrived at with extreme optimization [16] and the value $Q=0.723$ with spectral partitioning [14]. These two methods are currently considered as good algorithms that yield high $Q$ values.

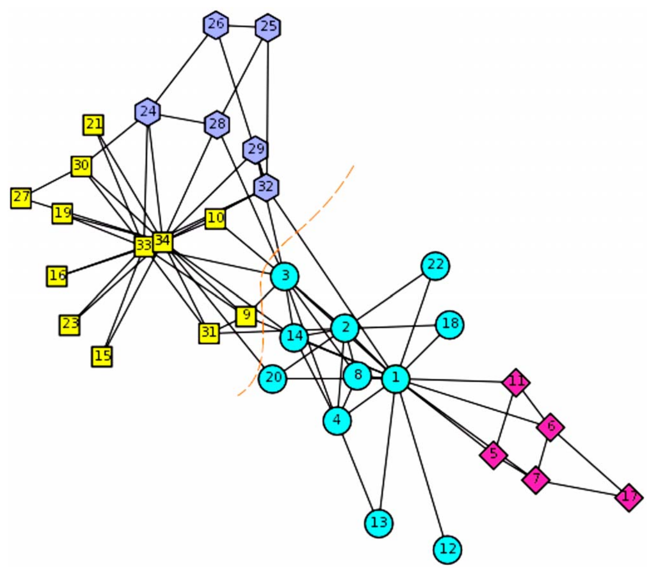

FIG. 5. (Color online) A test case on the Zachary network. The administrator and the instructor are defined as nodes 1 and 33, respectively. AdClust divides this network into four parts as indicated with four different shapes, yielding a $Q$ value of 0.4198. The two groups separated by the dashed line are consistent with disruption among members in reality.

## CONCLUSIONS

In this paper, we proposed an adaptive clustering algorithm for module detection of complex networks. Taking the advantage of adaptive characteristics and self-organization dynamics, AdClust avoids trapping nodes in a transient state of low quality, and gives more satisfactory solutions in real-world applications. In traditional clustering methods, regard-

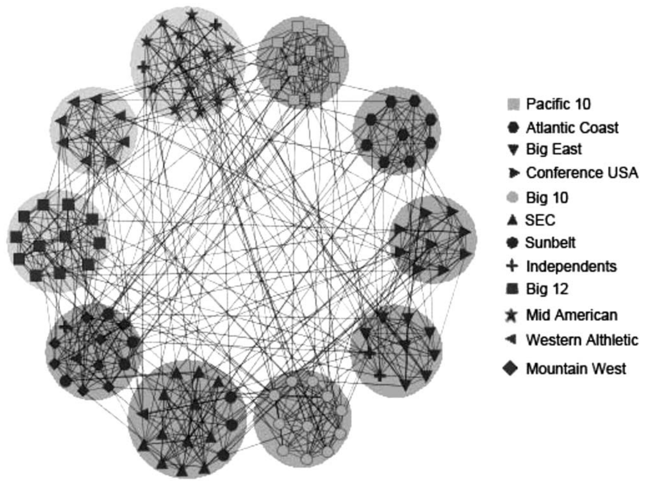

FIG. 6. Clustering for the football network. The different shapes represent 12 conferences in reality, and the ten circles indicate modules found by AdClust. The team members are homologous among most modules.

less if they are divisive or aggregative, a major drawback is the fact that their structural outcomes from early steps are inflexible and do not allow node exchange between established modules in subsequent steps. AdClust provides a mechanism to ensure that each node is capable of shuttling among existing modules, and therefore overcoming limitations occurred in early steps. AdClust keeps each vertex autonomous, allowing the creation of globally optimized patterns. The self-organization process implemented in AdClust enables the vertex to migrate toward its most preferable modules in an adaptive fashion resembling "flocking behaviors" of a living entity and relaxing the limitation of traditional clustering methods, and therefore substantially improving the accuracy of module identifications. Moreover, a series of tests on some predefined artificial networks showed that our adaptive clustering method possesses the capability of gaining better quantity in modularity and accuracy for module detection. Applying AdClust to real-world networks, we showed that the modules found with AdClust exhibit stronger association among members than what were defined by other methods. AdClust provides a plausible solution for clustering problems and is competent to promote analyses on networks from complex systems.

## ACKNOWLEDGMENT

This work was supported by Grant No. 2006CB910400 from the Chinese Ministry of Science and Technology and awarded to J.Y.

[1] S. H. Strogatz, Nature (London) **410**, 268 (2001).
[2] R. Albert and A.-L. Barabási, Rev. Mod. Phys. **74**, 47 (2002).
[3] M. E. J. Newman, SIAM Rev. **45**, 167 (2003).
[4] D. J. Watts and S. H. Strogatz, Nature (London) **393**, 440 (1998).
[5] A. L. Barabási and R. Albert, Science **286**, 509 (1999).
[6] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, Proc. Natl. Acad. Sci. U.S.A. **101**, 2658 (2004).
[7] M. E. J. Newman, Eur. Phys. J. B **38**, 321 (2004).
[8] J. Reichardt and S. Bornholdt, Phys. Rev. E **74**, 016110 (2006).
[9] A. L. Barabási and Z. N. Oltvai, Nat. Rev. Genet. **5**, 101 (2004).
[10] M. C. Gonzalez, H. J. Herrmann, J. Kertesz, and T. Vicsek, Physica A **379**, 307 (2007).
[11] A. Arenas, A. Diaz-Guilera, and C. J. Perez-Vicente, Phys. Rev. Lett. **96**, 114102 (2006).
[12] A. Arenas, J. Duch, A. Fernandez, and S. Gomez, New J. Phys. **9**, 176 (2007).
[13] L. Donetti and M. A. Munoz, J. Stat. Mech.: Theory Exp. (2004), P10012.
[14] M. E. J. Newman, Proc. Natl. Acad. Sci. U.S.A. **103**, 8577 (2006).
[15] M. E. J. Newman, Phys. Rev. E **74**, 036104 (2006).
[16] J. Duch and A. Arenas, Phys. Rev. E **72**, 027104 (2005).
[17] M. Girvan and M. E. J. Newman, Proc. Natl. Acad. Sci. U.S.A. **99**, 7821 (2002).
[18] M. E. J. Newman, Phys. Rev. E **69**, 066133 (2004).
[19] R. Guimera and L. A. N. Amaral, Nature (London) **433**, 895 (2005).
[20] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, J. Stat. Mech.: Theory Exp. (2005), P09008.
[21] S. Fortunato and A. Arenas, Proc. Natl. Acad. Sci. U.S.A. **104**, 36 (2007).
[22] S. White and M. Barthelemy, in *Proceedings of 5th SIAM International Conference on Data Mining*, edited by H. Kargupta, J. Srivastava, C. Kamath, and A. Goodman (Society for Industrial and Applied Mathematics, Philadelphia, 2005), pp. 274–284.
[23] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
[24] T. Gross and B. Blasius, J. R. Soc., Interface **10**, 1 (2007).
[25] G. W. Flake, *The Computational Beauty of Nature* (MIT, Cambridge, MA, 1999).
[26] W. W. Zachary, J. Anthropol. Res. **33**, 452 (1977).