TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 10/15 pp706–723
DOI: 10.26599/TST.2020.9010017
Volume 26, Number 5, October 2021

# Game Theoretical Approach for Non-Overlapping Community Detection

Baohua Sun, Richard Al-Bayaty, Qiuyuan Huang, and Dapeng Wu*

**Abstract:** Graph clustering, i.e., partitioning nodes or data points into non-overlapping clusters, can be beneficial in a large varieties of computer vision and machine learning applications. However, main graph clustering schemes, such as spectral clustering, cannot be applied to a large network due to prohibitive computational complexity required. While there exist methods applicable to large networks, these methods do not offer convincing comparisons against known ground truth. For the first time, this work conducts clustering algorithm performance evaluations on large networks (consisting of one million nodes) with ground truth information. Ideas and concepts from game theory are applied towards graph clustering to formulate a new proposed algorithm, Game Theoretical Approach for Clustering (GTAC). This theoretical framework is shown to be a generalization of both the Label Propagation and Louvain methods, offering an additional means of derivation and analysis. GTAC introduces a tuning parameter which allows variable algorithm performance in accordance with application needs. Experimentation shows that these GTAC algorithms offer scalability and tunability towards big data applications.

**Key words:** big data analytics; game theory; clustering; community detection; label propagation

## 1 Introduction

Graph clustering consists of constructing an adjacency matrix from a set of nodes or data points and partitioning them into clusters. The terms "nodes" and "data points" will be used interchangeably throughout this paper, as will the terms "communities" and "clusters". The experimentation and focus of this work are on generalized weighted and directed graphs. These types of graphs contain links between nodes that both carry a weight signifying the strength of the connection and a directionality representing correlation. The directionality does not need (and usually does not) to be characterized by a symmetric adjacency matrix. Often, the graph clustering problem is given without providing the number of clusters. In many practical applications, the graph is considered to be sparse, i.e., the number of links among the nodes is a linear (not quadratic) function of the number of total nodes of the graph. Partitioning occurs such that nodes belonging to the same group are highly connected, while nodes in different groups maintain lesser connectivity by means of optimizing a chosen measure of clustering effectiveness. There are several methods available to address this problem[1], however they all have their own limitations and use cases.

Spectral Clustering (SC)[2] aggregates nodes by virtue of the spectrum of the similarity matrix. The similarity matrix is constructed from the nodes' pairwise distances. SC consists of two parts: (1) perform eigen-decomposition on the similarity matrix to obtain a user-specified number (e.g., $N_c$) of eigenvectors as projections for the nodes into an $N_c$ dimensional subspace, (2) cluster the nodes in the $N_c$ dimensional

• Baohua Sun, Richard Al-Bayaty, Qiuyuan Huang, and Dapeng Wu are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA. E-mail: bsun@ufl.edu; ralbayaty@ufl.edu; idfree@ufl.edu; dpwu@ieee.org.

• An earlier version of this paper has been presented at the International Conference on Big Data Computing and Communications (BIGCOM2019).

* To whom correspondence should be addressed.

Manuscript received: 2019-12-10; revised: 2020-05-24; accepted: 2020-05-25

subspace by using methods such as $k$-means. A drawback to this technique is that it requires the number of clusters as an input. Poorly choosing the number of clusters to provide SC directly impacts the clustering performance. Another drawback of SC is that the eigen-decomposition may be infeasible if the similarity matrix is not symmetric, as the case with many directed graphs. The number of clusters, $N_c$, is application specific and thus a methodology for selecting this parameter should be established prior to using SC. Another drawback of SC is that the eigenvectors for the lower dimensional projection are all given the same priority during the clustering portion of the algorithm. Although Shi and Malik[3] improved the performance of SC by normalizing the similarity matrix using the Normalized cut (Ncut), this method still has the drawbacks of requirement of knowledge about $N_c$ prior to use Ncut and issue with eigen-decomposition for asymmetric similarity matrix.

Modularity Maximization (M-M) aims to detect clusters within a graph by optimizing modularity, which is a measure that can be used to determine the connectivity strength of nodes within a cluster as compared to those outside the cluster[4]. This method is more versatile than SC in that it will detect the number of clusters automatically based on the network structure of the provided graph. M-M faces the resolution limit issue, where multiple small clusters provide greater modularity when combined into a single cluster[5, 6]. This resolution limit prevents outstanding performance of M-M on larger networks containing multiple relatively small groups. Like in SC, M-M also suffers from the ill-conditioned eigen-decomposition problem when using directed graphs that are potentially asymmetric. Scalability is also a concern with this method due to the computational limitations of eigen-decomposition on substantially sized graphs.

In recent years, algorithms have been developed to address the scalability concerns for clustering within increasingly larger graphs[7, 8]. Label Propagation (LP)[9] is a near-linear time algorithm that is easy to implement and applicable to large datasets. However, LP is limited to graphs consisting of unweighted edges between nodes. This algorithm constructs clustering results from the underlying structure in the network without regard to optimizing any measure of a chosen community strength. Another technique aimed at tackling scalability issues is the Louvain method[10]. This method attempts to detect communities using a local modularity criterion aiming at resolving the resolution limit discussed earlier. Additionally, parallel implementations of the Louvain method have also been studied[11]. Although the Louvain method offers a performance improvement for many applications, it still suffers similar limitations as LP. There are also many recent works for detecting communities with game theory[12, 13], as well as work for model prediction and systems[14–16].

Prior work fails to show convincing experimental results on large networks with ground truth information, instead showing evaluations on datasets with either ambiguous labels or none at all. To help address this gap, we take a game theoretical approach to the graph clustering problem on various datasets containing ground truth information ranging from a small number of nodes to that of over a million nodes. Game theory is introduced and applied to community detection in Refs. [17–22], and they concentrate more on explaining the use of modularity as a payoff function in a game, as well as the modification of modularity for optimizing the payoff function. This work establishes a global function which is a measure of clustering performance over the entire graph. To form the foundation of Game Theoretical Approach for Clustering (GTAC), in each iteration we regard some of the nodes on the graph as players in a game and enable them to change their labels in a local fashion such that they optimize the global function. This process is repeated until an equilibrium is achieved for the game. In this paper, we show that LP and the Louvain method can be derived and extended under the framework of GTAC, offering an alternative means of analysis, performance tuning, and scalability.

To the best of our knowledge, there does not exist experimentation or performance evaluations using these methods on large networks whose similarity matrix is both weighted and asymmetric. The barrier of using large datasets is complex, because it requires not only obtaining datasets with high fidelity label data, but also the time, storage, computation, and programming resources of high performance computing machines. For the first time, we fill this gap by using Amazon EC2 web services to perform experiments on graphs containing one million nodes to evaluate the performance of various algorithms benchmarked against known truth.

The main contribution of this paper is three-folded. Firstly, it provides a new framework for clustering in large networks, the concept of degree of modularity is taken as the revenue function of the algorithm, and is generalized on this basis. An overview of the proposed

framework is shown in Fig. 1. Secondly, the advantages and disadvantages of Label Propagation and Louvain methods are analyzed, and the improvement ideas are put forward. These methods are evolved on the proposed GTAC model. Thirdly, variants of the GTAC model are proposed to explain the effects of different node sequences, function definitions, and node initializations on the results.

The structure of this paper is organized as follows. Section 2 presents a brief overview of select aspects of game theory, Section 3 constructs the ties between game theory and the GTAC algorithm, Section 4 derives and explains the proposed GTAC algorithms, Section 5 establishes a hierarchical variant of the GTAC framework, Section 6 outlines the experimental setup, Section 7 details the results and intuition behind the experimentation, and Section 8 concludes this work with highlights and closing remarks.

## 2 Prerequisite: Game Theory

Game theory revolves around studying the complex decision making involved with players in a cooperative or competing engagement[23]. Its foundational applications are in the area of economics, but are also used in various other fields to include biology, artificial intelligence, and computer networking to name a few. In 1928, Von Neumann and Morgenstern[24] formulated the basic concepts of game theory, but it was not until 1944 that they laid the foundation with Oskar Morgenstern in their book Theory of Games and Economic Behavior. To bridge the gap between cooperative and non-cooperative games, Nash[25,26] published two influential papers proposing the Nash Equilibrium. This equilibrium presumes a steady state strategy set for a general class of games where each player's strategy is chosen to be the best considering all other players' strategies at that point in time.

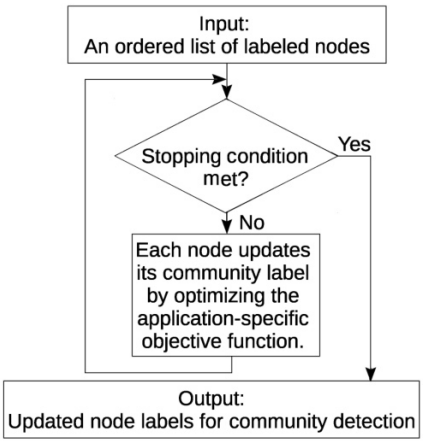



**Fig. 1 Overview of proposed framework.**

There are several essential elements in a game including players, strategy, payoff function, and game outcome/equilibrium. A player, $x$, is given as any decision-making participant in a game, and can be denoted as a set by

$$N = \{x_1, x_2, \ldots, x_n\} \tag{1}$$

Decisions made by the player are called strategies. The strategy set for each player is $S_1, S_2, \ldots, S_n$, respectively. As a result, the strategy set of all the players is given by

$$S = S_1 \times S_2 \times \cdots \times S_n \tag{2}$$

where $\times$ denotes the Cartesian product of sets (i.e., for sets $A$ and $B$, the Cartesian product $A \times B$ is the set of all ordered pairs $(a, b)$ where $a \in A$ and $b \in B$ ). The payoff function, which is also known as a utility function, is a mathematical formulation to calculate the gain and loss of the player strategies. For each player $x_i$, where $i \in N$,

$$u_i : S \to \mathbb{R} \tag{3}$$

denotes the payoff function of player $x_i$. Each game yields a result or outcome. This result is called an equilibrium if it can be considered stable.

Nash Equilibrium is defined as the best strategy that can be made by each player provided that all other players do not change their strategies. If we denote $s^*$ as the strategy resulting in a Nash Equilibrium[17,26,27], we have

$$u_i(s_i, s^*_{-i}) - u_i(s^*) \leqslant 0, \forall s_i \in S_i, \forall i \in N \tag{4}$$

We refer to the strategy profile obtained from $s^*$ by replacing the strategy of player $x_i$ with $s_{i_j}$ as being $(s_{i_j}, s^*_{-i})$, defined as

$$(s_{i_j}, s^*_{-i}) = (s^*_1, s^*_2, \ldots, s^*_{i-1}, s_{i_j}, s^*_{i+1}, \ldots, s^*_n) \tag{5}$$

## 3 Motivation from Game Theory

Here the GTAC scheme is described in detail. A weighted graph, $\boldsymbol{G}$, is defined by $N$ nodes and an $N \times N$ dimensional adjacency matrix, $\boldsymbol{A}$, where each entry $A_{i,j}$ denotes the weight of the link between nodes $i$ and $j$, here $i \in \{1, 2, \ldots, N\}$ and $j \in \{1, 2, \ldots, N\}$. If $\boldsymbol{A}$ is symmetric, this graph is considered to be undirected, otherwise it is a directed graph. The case of the unweighted graph can be seen as a special case of weighted graphs where $A_{i,j} \in \{0, 1\}$ is a binary flag signifying whether or not there is a link between nodes $i$ and $j$.

GTAC constructs the process of clustering, also referred to as community detection, as a game whose objective function is given by

$$F = f(\boldsymbol{A}, \boldsymbol{C}) \tag{6}$$

$$C = \{C_k, k \in \{1, 2, \ldots, K\}\} \tag{7}$$

Each community, $C_k$, is a non-overlapping set of node indices denoting membership to community $k$. The function $f$ depends on the input weight matrix, $\boldsymbol{A}$, and the community detection result shown in Eq. (6). Different applications warrant different constructions of $f$. For fixed $\boldsymbol{A}$, we seek to optimize $F$ by means of varying the community detection result, $\boldsymbol{C}$. A well recognized optimizing criterion for $F$ is that of maximizing community modularity[4, 10].

The search for the optimal solution to many formulations of $f$ is known to be Non-deterministic Polynomial (NP)-hard. The optimization of $F$ is relaxed by reformulating the problem as follows: for a given initial community clustering, improvements to $F$ are made by only modifying the membership of $q$ nodes. For simplicity, this paper uses $q = 1$. Assuming node $x$ currently belongs to community $C_l$, if there exists another community for node $x$, $C_k$ $(k \neq l)$, such that the value given by Eq. (6) improves then node $x$ will change communities to make this improvement. When determining new communities, it is prudent to consider only those of the nodes connected to node $x$. This is equivalent to treating $x$ as an isolated node, meaning that it is not initially assigned to any community, and selecting $C_{\hat{k}}$ such that Eq. (6) is optimized. This step is repeated sequentially for all other nodes in a prescribed order and is iterated until a stopping criterion is met. Reaching this stopping criterion is considered having achieved the Nash Equilibrium of the game.

## 4 GTAC Algorithm

This section details the GTAC algorithm and discusses possible GTAC variations. Algorithm 1 provides the layout for GTAC.

**Algorithm 1 GTAC**

```
Input: An ordered list of labeled nodes
Output: Updated node labels from community detection
while stopping condition not met do
    for each i in ordered list do
        x ← ordered_list(i)                    ▷ iterate list for each i
        k ← arg_k max/min f(A, C)              ▷ optimize f
        labes_list(x) ← k                      ▷ assign new label: k
    end for
end while
return labels_list                             ▷ Community detection result
```

A few remarks should be made regarding GTAC in its basic form.

(1) Different community results may arise from separate label initializations and node orderings.

(2) The function $f$ plays a pivotal role in that it defines "community".

(3) Stopping criteria will influence the community result.

The following sections provide deeper discussion in regards to the essential elements of Algorithm 1.

### 4.1 Initialization and computational order of the nodes

It may seem intuitive that different initializations of the labels or the prioritized ordering of the nodes for computation may yield different community results, however as shown in Ref. [10], the computational ordering of the nodes does not greatly influence the outcome. Thus, in later experiments we utilize the same initialization and computational ordering used by the Louvain method. This method assigns unique labels and identification numbers to nodes upon initialization. The node IDs are then used to form a prioritized, or ordered list.

### 4.2 Different payoff functions for GTAC

There are many candidate payoff functions, $f$, available for application in Step 5 of Algorithm 1. The careful selection of $f$ helps to dictate what defines a community amongst the nodes. The following will highlight the different formulations of $f$, which will subsequently lead to the derivation of different GTAC variants.

#### 4.2.1 GTAC using modularity criteria

Modularity[28] is defined as

$$Q = \frac{1}{2m} \sum_{i,j} \left[ A_{i,j} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{8}$$

where $k_i = \sum_j A_{i,j}$, and $c_i$ is the community to which node $i$ belongs. We use the Kronecker delta function where $\delta(c_i, c_j) = 1$ if $c_i = c_j$, and 0 otherwise. The normalization term uses $m = \frac{1}{2} \sum_i \sum_j A_{i,j}$.

If modularity is chosen as the payoff function, maximizing $F$ can be rewritten as

$$F = \sum_{k=1}^{K} \left[ \sum_{i \in C_k} \sum_{j \in C_k} \left( A_{i,j} - \frac{k_i k_j}{2m} \right) \right] \tag{9}$$

The normalization term $\frac{1}{2m}$ is omitted since it will not influence the optimization solution. Here, $K$ represents the number of total clusters. Using the methodologies provided earlier for GTAC and in Ref. [10], the strategy for node $x$ is given as

$$\hat{k} = \arg\max_k \Delta Q_k = \arg\max_k \left[\frac{\sum_{\text{in}} + 2k_{x,\text{in}}}{2m} - \left(\frac{\sum_{\text{tot}} + k_x}{2m}\right)^2\right] - \left[\frac{\sum_{\text{in}}}{2m} - \left(\frac{\sum_{\text{tot}}}{2m}\right)^2 - \left(\frac{k_x}{2m}\right)^2\right] = \arg\max_k \left[\frac{k_{x,\text{in}}}{m} - \frac{\sum_{\text{tot}} \cdot k_x}{2m^2}\right] \tag{10}$$

where

$$k_{x,\text{in}} = \sum_x \sum_{j \in C_k} A_{x,j} \tag{11}$$

$$\sum_{\text{tot}} = \sum_i \sum_{j \in C_k} A_{i,j} \tag{12}$$

$$k_x = \sum_i \sum A_{i,x} \tag{13}$$

where $\sum_{\text{in}}$ in Eq. (10) is the sum of weights inside the cluster $C_K$. The sum over $x$ in Eq. (11) represents a summation of the $q$ nodes chosen for the game, where in our game we chose $q = 1$. Equation (12) is the total talk to $C_K$. And Eq. (13) is the sum of talk to $x$. $\Delta Q_k$ in Eq. (10) represents the change in $Q_k$ provided a change in player label.

#### 4.2.2 GTAC using unnormalized most friends criteria

In Label Propagation, network structure is the guide by which nodes are driven into communities in lieu of an optimization of community strength by use of a measure[9]. We show that LP can be modeled as a special case of GTAC. If the payoff function is chosen to be the summation of within community talk, where $F$ is given by

$$F = \sum_{k=1}^{K} \left[\sum_{i \in C_k} \sum_{j \in C_k} A_{i,j}\right] \tag{14}$$

then the strategy for node $x$ is resultingly

$$\hat{k} = \arg\max_k \sum_{k=1}^{K} \left(\sum_{i \in C_k,\, i \neq x} \sum_{j \in C_k,\, j \neq x} A_{i,j}\right) + \sum_{i \in C_k, i \neq x} (A_{x,i} + A_{i,x}) \tag{15}$$

The first term in Eq. (15) signifies the $F$ value for the $N-1$ nodes excluding node $x$, and the second term is the change in $F$ value when assigning node $x$ to community $C_k$. The self-loop $A_{x,x}$ is neglected because it should not be allowed to influence the node's strategy in this framework. Since the first term is the same for all $k$, it can be disregarded in the optimization, which reduces to being

$$\hat{k} = \arg\max_k \sum_{i \in C_k,\, i \neq x} (A_{x,i} + A_{i,x}) \tag{16}$$

The formulation in Eq. (16) is called GTAC using Unnormalized Most Friends criteria (GTAC-UMF). This is the strategy group used for node $x$ in both the weighted and unweighted graph scenarios where the graphs may be either directed or undirected. When using GTAC-UMF in cases where the weight matrix $\boldsymbol{A}$ is symmetric, Eq. (16) further reduces to

$$\hat{k} = \arg\max_k \sum_{i \in C_k,\, i \neq x} A_{x,i} \tag{17}$$

Furthermore, if $\boldsymbol{G}$ is an unweighted graph then $\boldsymbol{A}$ will be a binary matrix which admits the "majority vote" strategy group for node $x$ (i.e., join the community to which most of its neighbors belong). This is precisely the strategy used in LP[9].

Further analysis of Eq. (14) reveals limitations of LP. It is trivial to see that maximizing $F$ produces a global optimum when all nodes merge into one conglomerate component, or more simply letting $K = 1$. Because GTAC is a local method, it is difficult for all the nodes in one community to merge into another entirely since the community structure is evolved sequentially across the nodes. It is more likely that the result will become stuck at a local optimum instead of a global, provided that nodes are initialized as having their own independent community. The limitation of Eq. (14) is that it tends to merge communities together when larger amounts of crosstalk is present, as is supported by later experimentation (Section 7) which offers a number of communities less than that of the ground truth.

Additionally, since $\sum_{\text{total}} = \sum_{\text{in}} + \sum_{\text{out}}$ is a constant for a given weight matrix with a fixed cluster arrangement, the optimization on Eq. (14) can be changed from a maximization over $\sum_{\text{in}}$ to a minimization over $\sum_{\text{out}}$ following the same process as that resulting in Eq. (17). $\sum_{\text{out}}$ is defined as the total crosstalk, i.e., the sum of weights across different clusrters.

When comparing Eqs. (9) and (14), we see that the two equations differ in only the subtracted term in Eq. (9), which represents crosstalk. Thus the difference between

the two is that Eq. (9) considers both within and crosstalk while Eq. (14) only considers within talk. With the task of maximizing modularity, the local and global approaches yield different results according to Ref. [1], which points out that modularity-based methods (with the exception of the method by Blondel et al.[10]) have rather poor performance. This poor performance worsens for larger systems with smaller communities due to the well known resolution limit of modularity[1]. It follows that modularity maximization benefits in being implemented as a local method rather than a global method. This reasoning transcends to other techniques as well. As pointed out, maximizing Eq. (14) in a global fashion results in a singular connected component graph, but if performed in a local way (as with the GTAC method), more realistic clusters are formed so long as the crosstalk among communities is not proportionally too large.

#### 4.2.3 GTAC using normalized most friends criteria

The GTAC-UMF algorithm suffers from the "rich become richer" problem, i.e., when a community becomes large it attracts more nodes making the community larger. This problem exists no matter if the attraction is caused by dense connections between nodes or because of the large size of the community via the summation of a large number of small connections being large. The drawback of GTAC-UMF is that it fails to consider the size of the group. To address this drawback, we propose the GTAC using Normalized Most Friends criteria (GTAC-NMF) with the optimization function in Eq. (18). This formulation counter-balances the false attraction caused by large communities using the size of the group as a normalization parameter.

$$F = \sum_{k=1}^{K} \frac{\sum\limits_{i \in C_k} \sum\limits_{j \in C_k} A_{i,j}}{|C_k|^{\alpha}} \tag{18}$$

where $|C_k|$ is the cardinality of $C_k$, and $\alpha$ is a tuning parameter with non-negative real value. It can be seen that GTAC-UMF is a special case of GTAC-NMF when $\alpha = 0$.

Due to the normalization parameter, it is more difficult to derive a simplified strategy for GTAC-NMF than for GTAC-UMF. In order to obtain an optimal strategy, let us compare the gains from the optimization function (Eq. (18)) when node $x$ joins either $C_{k_1}$ or $C_{k_2}$. For simplification, let us refer to these strategies as $k_1$ and $k_2$, respectively. For $k_1$, Eq. (18) becomes

$$F_{k_1} = \left( \sum_{k \neq k_1,\, k \neq k_2} \frac{\sum\limits_{i \in C_k,\, i \neq x} \sum\limits_{j \in C_k,\, j \neq x} A_{i,j}}{|C_k|^{\alpha}} \right) + \frac{\sum_{k_2}}{|C_{k_2}|^{\alpha}} + \frac{\sum_{k_1} + \sum\limits_{i \in C_{k_1}} (A_{x,i} + A_{i,x})}{(|C_{k_1}| + 1)^{\alpha}} \tag{19}$$

$$\sum\nolimits_{k_1} = \sum_{i \in C_{k_1},\, i \neq x} \sum_{j \in C_{k_1},\, j \neq x} A_{i,j} \tag{20}$$

$$\sum\nolimits_{k_2} = \sum_{i \in C_{k_2},\, i \neq x} \sum_{j \in C_{k_2},\, j \neq x} A_{i,j} \tag{21}$$

To obtain the strategy $k_2$, one would simply interchange $k_1$ and $k_2$ variables of Eq. (19). Since the first term in Eq. (19) is the same for both strategies, it may be neglected in the comparison. Electing a better strategy involves determining which provides a larger $F$, which is decided by

$$\hat{k} = \arg\max_{k \in \{k_1, k_2\}} (F_{k_1}, F_{k_2}) \tag{22}$$

We further simplify Eq. (22) by making the approximation:

$$|C_{k_n}|^{\alpha} \approx (|C_{k_n}| + 1)^{\alpha} \tag{23}$$

Using the approximation in Eq. (23) on Eq. (22) provides

$$\hat{k} = \arg\max_{k \in \{k_1, k_2\}} \left( \frac{\sum\limits_{i \in C_{k_1}} (A_{x,i} + A_{i,x})}{|C_{k_1}|^{\alpha}}, \frac{\sum\limits_{i \in C_{k_2}} (A_{x,i} + A_{i,x})}{|C_{k_2}|^{\alpha}} \right) \tag{24}$$

which can further be generalized for pairwise comparisons to make the GTAC-NMF strategy,

$$\hat{k} = \arg\max_{k} \sum_{i \in C_k} \frac{(A_{p,i} + A_{i,p})}{|C_k|^{\alpha}} \tag{25}$$

We see in Eq. (25) that if $\alpha = 0$, GTAC-NMF reduces to GTAC-UMF as shown in Eq. (16). The optimization in Eq. (18) is the generalized form of that in Eq. (14). Also, the solution to Eq. (25) is the generalized form of Eq. (16), provided the approximation in Formula (23).

Now let us examine the influence on community detection by the parameter $\alpha$. An $\alpha$ too large (e.g., larger than 1) makes the algorithm susceptible to crosstalk noise between the clusters. This crosstalk noise will reinforce small group clustering offering a solution consisting of a much larger number of communities than in the ground truth. Setting $\alpha$ too small (e.g., close to 0) will tend the results of GTAC-NMF toward those of GTAC-UMF. The drawback of GTAC-UMF (the resolution limit) is that smaller clusters get absorbed into

the larger ones. In this instance the number of groups will tend to be smaller than that of the ground truth. Per application, a strategic choice of $\alpha$ between 0 and 1 will strike a balance between the two extremes.

### 4.3 Stopping criteria of GTAC

In the GTAC algorithm, and in others, there are a few commonly used stopping criteria, which are enumerated and detailed below. The stopping criterion used in each experimental setup is explained in Section 7.

#### 4.3.1 Modularity-based criteria

This method is used when it is desired for the stopping criteria of the algorithm to correspond to a stagnation in the modularity increase. This criterion is reached when the difference of modularity between consecutive iterations is less than some threshold value.

#### 4.3.2 Bounded iteration

This method is straightforward and is further investigated in Ref. [1]. In their experiments they found that at least 95% of labels were correctly classified by the end of the 5th iteration, irrespective of the quantity of nodes. The key advantages of this stopping criterion are that it is unaffected by label loops in clustering caused by node oscillations[1] and the run time of the algorithm for a given iteration bound is a function of the number of nodes and payoff function operations.

#### 4.3.3 Unbounded iteration

In cases where a fixed number of iterations is not desired, such as when initializations of node labels have a drastic effect on algorithm convergence, unbounded iteration methods are required. Typically, if all node labels do not change between subsequent iterations, the equilibrium of the algorithm is thought to have been achieved. Because of the label oscillation phenomenon as mentioned in Ref. [1], this method should not be used as the sole stopping criteria. Instead, other stopping criteria should be used in tandem, stopping the algorithm whenever either method activates. Two types of unbounded iteration stopping criteria are detailed below.

**(1) Unique labels**. The unique labels stopping criterion will terminate the algorithm when the sets (clusters) of node labels of a previous iteration are the same as those in the current iteration. This method can effectively avoid the oscillation problem in many instances, however it can cause the issue of early stopping if node labels start to exhibit transient oscillatory behavior within a set of unique labels.

**(2) Tabu list**. A tabu list stopping criterion is to detect periodic behaviors in clustering. A tabu list records the clustering results over a finite number of iterations, such as $N_t$ iterations. It terminates when the clustering results are repeated with the previous $N_t$ iterations. Since the tabu list is finite, adding a new entry entails removing an existing entry, usually the oldest. Other means of tabu list updating have been studied in Ref. [29]. To reduce computational complexity, the recording of tabu list does not begin until several iterations have been completed, since it is presumed that convergence of the node labels will not occur within the initial iterations. A benefit of the tabu list is that it also addresses the oscillation problem while also conquering the early stop problem of unique labels, but at the cost of an increased computational and storage complexity.

### 4.4 GTAC algorithm used in the experiments

Here, we list and explain the three GTAC algorithm variants used within our experimentation. For every algorithm variant, the nodes are initialized with unique labels and IDs and subsequently processed in the order of the node IDs.

#### 4.4.1 GTAC-M

This variant uses Eq. (10) as the payoff function and a modularity-based stopping criterion.

#### 4.4.2 GTAC-UMF-UL and GTAC-UMF-TL

These variants use Eq. (16) as the payoff function, the unique labels, and tabu list stopping criteria, respectively.

#### 4.4.3 GTAC-NMF

This variant uses Eq. (25) as the payoff function and the tabu list stopping criteria.

## 5 Hierarchical Game Theoretical Approach to Clustering (HGTAC)

The GTAC algorithm and each variant can be extended to a hierarchical version[30]. This version HGTAC iterates the GTAC algorithms to form a hierarchical tree of community detection results until a stopping criterion for the hierarchy is met. There are two phases in each of these iterations. Phase 1 implements the GTAC algorithm until an equilibrium is achieved, and Phase 2 considers each community obtained in Phase 1 as a distinct node in the hierarchy, treating the total crosstalk between communities as connection weights and the total within talk as self-loops per community.

This hierarchical process iterates until a stopping

criterion is met. Some stopping criteria for the hierarchical process include stopping if the number of resultant communities falls below a provided threshold or if the parameter of the community detection result satisfies a certain bounding requirement. This process is outlined in Algorithm 2.

The first phase applies Algorithm 1 to obtain a community detection result. This result is then passed to the second phase where the nodes in the same community are merged together into what is regarded as a node in the next iteration. Iterations terminate when a stopping criterion is met. The second phase considers the summation of connections between communities as connections between nodes in the new set of hierarchical nodes. HGTAC algorithms have the same naming convention as GTAC.

When using GTAC-M (which uses modularity-based stopping criteria) as the first phase of HGTAC (HGTAC-M), this method becomes that of the Louvain method used in Ref. [10]. Additionally, the HGTAC-UMF is similar to the Louvain method. In our experiments with HGTAC-UMF, where Eq. (6) is maximized, the stopping criterion can not be made similar to that of the Louvain method since Eq. (16) will increase for every community merger, in turn causing Eq. (6) to increase as well. Instead, HGTAC-UMF is stopped when all communities have merged. Experimental results show that the second-to-last level of the hierarchy offers relatively good performance.

## 6 Data Source/Experimental Setup

For community detection, two types of data can be used: truthed and real world. The benefit of using truthed data is that it contains known labels, whereas real world data may not. Experiments in the literature generally utilize synthetic data when it is important to allow for algorithm analysis by means of varying the synthetic data structure and configuration, but tend to use untruthed real world datasets to prove merit in certain application areas.

**Algorithm 2 HGTAC**

1: Input: An ordered list of nodes with labels
2: Output: Node labels from community detection
3: **while** stopping condition not met **do**
4: **Phase 1:** Apply GTAC (Algorithm 1)
5: **Phase 2:** Use Phase 1 results to form new node in hierarchy then update the current set of nodes.
6: **end while**
7: **return** nodes ▷ Community detection result

### 6.1 Truthed data

We use the synthetic data generated from the Lancichinetti-Fortunato-Radicchi (LFR) benchmark network[31] in many of our experiments. These data are widely used for testing community detection algorithms[1, 32–34]. It provides several parameters which allow for tuning of a network, such as the number of nodes, the average node degree, a topology mixing parameter ($\mu_t$), a weight mixing parameter ($\mu_w$), an exponent for controlling the degree sequence ($t_1$), an exponent for controlling the community size distribution ($t_2$), and an exponent for varying the weight distribution ($\beta$). Additionally, a Gaussian data network is utilized in our experimentation; this type of network is also studied in Ref. [35].

### 6.2 Real world data

The value in performing community detection on real world data revolves around helping to better understand the complex and often immense data by means of providing potential topological structure schemes or insight into the network. This insight is valuable for large networks especially when human inspection can be expensive to perform. One type of example of a large network where insight into the topological structure may be desirable is that of online social networks. This work uses data from the social media network Twitter[36].

### 6.3 Performance measure

To evaluate the performance of the various algorithms for community detection, different methodologies are used for the different types of datasets.

#### 6.3.1 Truthed data

For datasets containing ground truth, the Normalized Mutual Information (NMI) is used from Ref. [37]. NMI is a measure used to determine the difference between the ground truth and the community detection results given by an algorithm.

Let us assume that nodes are grouped into $K$ communities by an algorithm. Let $X$ denote the observation of a single node's label output, and $Y$ denote the label's ground truth. For event $X$, the number of nodes in each community is $N_X(k)$, where $k = 1, 2, \ldots, K$ and $\sum_k N_X(k) = N$. The corresponding ground truth event, $Y$, has $N_Y(k_G)$ numbers of nodes per community, where $k_G = 1, 2, \ldots, K_G$ and $\sum_{k_G} N_Y(k_G) = N$. Using

$$P_X(x = k) \approx N_X(k)/N \tag{26}$$

as an approximation, the entropy of the event $X$ is given

as

$$H(X) = \sum_{k=1}^{K} -\frac{N_X(k)}{N} \log\left(\frac{N_X(k)}{N}\right) \tag{27}$$

with $H(Y)$ having the same form. Additionally, if $N_{X,Y}(k, k_G)$ is the number of concurrently labeled nodes in group $k$ for event $X$ and group $k_G$ of event $Y$, the joint entropy is given by

$$H(X,Y) = \sum_{k=1}^{K} \sum_{k_G=1}^{K_G} -\frac{N_{X,Y}(k,k_G)}{N} \log\left(\frac{N_{X,Y}(k,k_G)}{N}\right) \tag{28}$$

This leads to the formulation of NMI:

$$\text{NMI} = \frac{2I(X,Y)}{H(X)+H(Y)} \tag{29}$$

where the mutual information is taken to be

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{30}$$

#### 6.3.2 Real world data

In these experiments and for real world datasets in general, measuring performance can be performed using the value of the nodes' within community talk[38].

### 6.4 Performance measure confidence intervals under isomorphic topologies

In order to avoid bias in the evaluation of the performances caused by various orderings of the nodes, we test the algorithms on each dataset using different isomorphic topologies of the original dataset. These different topologies are also randomly given node labels, which we refer to as randomly permuted datasets. Essentially, each algorithm is run $T$ times using $T$ random permuted datasets.

We evaluate an algorithm's performance by estimating the mean and confidence interval (ci) for the results obtained from each dataset and make comparisons using these estimated parameters. These parameters are assumed to be normally distributed and are used as the statistical results of the experiment.

## 7 Experimental Result

This section first examines the use of synthetic and real data containing truth in experimentation, then untruthed real world data. Experiments are conducted on six types of truthed data and one real world dataset obtained from the Twitter network.

### 7.1 Network data with truth

The following experiments consist of investigating performance on small networks (less than 5000 nodes), medium LFR networks (5000 nodes), large LFR networks (1 million nodes), and the performance of GTAC-NMF under varying $\alpha$ values.

#### 7.1.1 Small networks (less than 5000 nodes)

We use six small datasets with known label data: (1) Karate club, (2) Girvan and Newman (GN) network, (3) LFR easy, (4) LFR hard, (5) LFR harder, and (6) Gaussian network. For each dataset, experiments are conducted using the following GTAC algorithms:

(1) GTAC-UMF-UL (stopping criteria: unique labels);

(2) GTAC-UMF-TL (stopping criteria: tabu list);

(3) GTAC-NMF ($\alpha$ varying from 0.1 to 0.12 and stopping criteria: tabu list);

(4) GTAC-M (using modularity criteria);

(5) HGTAC-UMF-TL (stopping criteria: tabu list);

(6) GTAC-NMF ($\alpha$ varying from 0.1 to 0.12 and stopping criteria: tabu list); and

(7) HGTAC-M (Louvain).

Comparisons are performed against the GTAC algorithms using M-M given in Ref. [4] and Ncut given in Ref. [3]. Since Ncut requires the number of clusters as an input, the ground truth value in each dataset is provided for this particular method. Experimental results for these methods are provided in Table 1, where the results of 100 separate random initializations are given in the form of a mean NMI ($\mu_{\text{NMI}}$) performance with an accompanying 95% confidence interval. Each column contains bold entries which represent the three best performance values including ties. The values range from worst to best in the range [0, 1].

**Karate club.** The karate club dataset given by Ref. [39] is a small, real world dataset with 34 nodes from two communities. This dataset is valuable to use in experimentation due to it being both labeled and real world, and is represented using a weighted, symmetric matrix. Figure 2a shows the two communities distinguished by shape. Group one is represented using downward pointing triangles, and group two as circles. A link between nodes represents a connection whose width signifies the weight of the connection. The 2nd column of Table 1 provides algorithm performances on this dataset. It can be seen that many of the HGTAC and some of the GTAC algorithms are not well suited for this type of dataset. Among the methods, GTAC-UMF-TL and HGTAC-UMF-TL rank the second and third best. The worst performance was achieved by HGTAC-NMF no matter the value of alpha chosen from the range. Notice that Ncut has perfect performance, however this is not surprising since it requires the number of communities initially. M-M consistently clusters the

**Table 1 Performace NMI ($\mu_{\text{NMI}} \pm$ ci) on small datasets.**

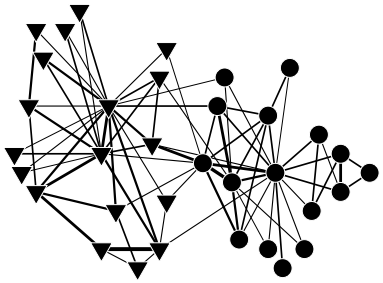
| Algorithm | Karate | GN network | LFR easy | LFR hard | LFR harder | Gaussian |
|---|---|---|---|---|---|---|
| Ncut | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ | $0.98 \pm 0.00$ | N/A |
| M-M | $0.68 \pm 0.00$ | $\mathbf{1.00 \pm 0.00}$ | $0.52 \pm 0.00$ | $0.28 \pm 0.00$ | $0.24 \pm 0.00$ | N/A |
| GTAC-UMF-UL | $0.70 \pm 0.04$ | $0.93 \pm 0.02$ | $\mathbf{1.00 \pm 0.00}$ | $0.97 \pm 0.00$ | $0.91 \pm 0.03$ | $0.43 \pm 0.00$ |
| GTAC-UMF-TL | $\mathbf{0.73 \pm 0.04}$ | $0.93 \pm 0.02$ | $\mathbf{1.00 \pm 0.00}$ | $0.97 \pm 0.00$ | $0.90 \pm 0.04$ | $0.43 \pm 0.00$ |
| GTAC-NMF-0.1 | $0.68 \pm 0.02$ | $0.95 \pm 0.02$ | $\mathbf{1.00 \pm 0.00}$ | $0.99 \pm 0.00$ | $0.97 \pm 0.00$ | $0.43 \pm 0.00$ |
| GTAC-NMF-0.11 | $0.68 \pm 0.02$ | $0.96 \pm 0.02$ | $\mathbf{1.00 \pm 0.00}$ | $0.99 \pm 0.00$ | $0.97 \pm 0.00$ | $0.43 \pm 0.00$ |
| GTAC-NMF-0.12 | $0.68 \pm 0.02$ | $0.96 \pm 0.02$ | $\mathbf{1.00 \pm 0.00}$ | $0.99 \pm 0.00$ | $0.97 \pm 0.00$ | $0.43 \pm 0.00$ |
| GTAC-M | $0.60 \pm 0.01$ | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ | $0.99 \pm 0.00$ | $0.97 \pm 0.00$ | $0.43 \pm 0.00$ |
| HGTAC-UMF-TL | $\mathbf{0.72 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ |
| HGTAC-NMF-0.1 | $0.23 \pm 0.00$ | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ | $0.99 \pm 0.00$ | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ |
| HGTAC-NMF-0.11 | $0.23 \pm 0.00$ | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ | $\mathbf{1.00 \pm 0.00}$ |
| HGTAC-NMF-0.12 | $0.23 \pm 0.00$ | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ | $0.99 \pm 0.00$ | $\mathbf{1.00 \pm 0.00}$ |
| HGTAC-M | $0.70 \pm 0.01$ | $\mathbf{1.00 \pm 0.00}$ | $\mathbf{1.00 \pm 0.00}$ | $0.96 \pm 0.00$ | $0.99 \pm 0.00$ | $0.37 \pm 0.01$ |

34 nodes into 4 groups in each of the 100 random initializations.

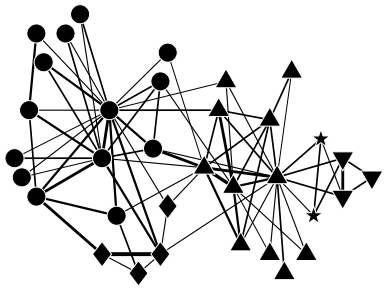
Select community detection results are shown in Fig. 2 to aid in visualizing the performance of the GTAC algorithms on this dataset. Figure 2b represents the community detection result offered by GTAC-NMF with random initializations and $\alpha = 0.1$. Group two has been split into three distinct subgroups, while group one is split into two distinct subgroups. This subdivision of groups one and two unsurprisingly and significantly reduces the NMI. The subdivisions seem to stem from the less dense connections between the groups (crosstalk) than the within subgroup connections. In Fig. 2c, GTAC-UMF-TL correctly clusters all of group one, but group two is separated into two subgroups. This result is much closer to the ground truth, which is supported by an improvement to the NMI value compared to that of Fig. 2b. In comparing Fig. 2b with Fig. 2c, we find that GTAC-NMF has a tendency of splitting groups into smaller divisions, while GTAC-UMF tends to do this to a lesser degree. In some infrequent experiment iterations, GTAC-UMF was seen to cluster all nodes into one community which shows a sensitivity to node label initialization. These phenomena are supported by earlier discussion during the derivation of these two algorithms. Care must be taken when selecting the parameter $\alpha$, since group subdivisions become more frequent by increasing the value. The results of HGTAC-NMF are shown in Fig. 2d. By using the hierarchical variant to GTAC-NMF, we expect to see an improvement to the subdivision problem, however in Fig. 2b we see that adding this hierarchical processing overcompensates by allowing group one to absorb some of the group two nodes. While this over-merging causes HGTAC-NMF to perform poorly on this dataset, later we see that it helps provide better performance than GTAC-NMF on other datasets. The benefit of the hierarchical process is observed to be data dependent.

**GN network.** This experiment aims to test the performance of different GTAC algorithms on networks with communities of equal size. The GN network was proposed by Girvan and Newman[40] and contains four communities consisting of 32 nodes each. We use the LFR network to generate a GN network by setting the number of nodes to 128, average and maximum degrees to 16, minimum and maximum cluster sizes to 32, $\mu_w = \mu_t = 0.1$, and $\beta = 1$. The 3rd column of Table 1 reveals good performance from most of the algorithms on this dataset. HGTAC-UMF-TL, HGTAC-NMF, Ncut, and M-M offer perfect clustering performance with minimal variance, while GTAC-M and HGTAC-M offer nearly perfect results with small variances. The good overall performance on this dataset is likely due to the symmetry and lesser amount of crosstalk of the GN network. While GTAC-UMF and GTAC-NMF do not perform particularly well, their corresponding hierarchical versions have much better performance due to the algorithms' ability to avoid the excessive subdivisions. The observed variance of $\mu_{\text{NMI}}$ from GTAC-M and HGTAC-M signals that the GTAC methods using modularity are more sensitive to initialization than the GTAC methods using most friends criteria.

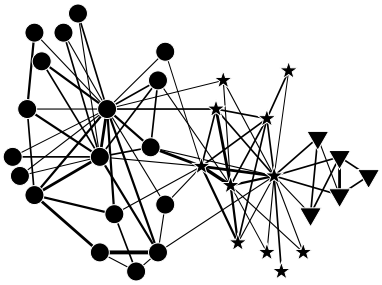
**LFR easy.** The purpose of this experiment is to test the performance of GTAC algorithms on LFR benchmark networks, which offer configurations allowing for easy community detection. We test the network with 1000 nodes, with an average degree equal to 15, maximum



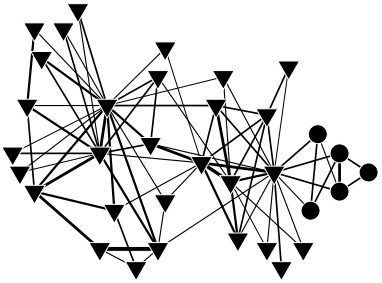
(a) Ground truth



(b) Typical NMI = 0.6731: GTAC-NMF ($\alpha = 0.1$)



(c) Typical NMI = 0.8255: GTAC-UMF-TL



(d) Typical NMI = 0.2260: HGTAC-NMF ($\alpha = 0.1$)

**Fig. 2 Karate data community detection results.**

degree equal to 50, $\mu_t = 0.3$, and $\mu_w = 0.2$. Other parameters are left as the defaults provided in Ref. [40], including $t_1 = 2$, $t_2 = 1$, and $\beta = 1.5$. The experimental results are shown in the 4th column of Table 1. We can see that all of the GTAC algorithms give perfect or near-perfect clustering results. As was the case with the GN network experiments, the non-hierarchical GTAC algorithms appear to be slightly more sensitive to the label initializations. In this experiment, we can see with small network crosstalk that the GTAC algorithms' variance from initialization is not as significant. In comparing other methods we see that Ncut performs nearly optimally with a negligible amount of variance, while M-M performs rather poorly. M-M's lacking performance is caused by merging of the smaller groups, as observed by the resulting 17 groups compared to the actual 28 groups. This over-merging highlights the resolution limit phenomenon of M-M on datasets with many small groups.

**LFR hard.** This experiment shows the effects on clustering from increasing the level of crosstalk within an LFR network. The same configuration is used from the previous experiment with the exceptions of $\mu_w = \mu_t = 0.5$. In changing these parameters, we aim to increase the difficulty of community detection by increasing the amount of crosstalk among communities. The experimental results are shown in the 5th column of Table 1. As compared with the results of the last experiment (the 4th column), there is a performance decrease across the board for the GTAC algorithms with the exception of HGTAC-UMF-TL and HGTAC-NMF with $\alpha = 0.12$ having nearly perfect performance with increased crosstalk. The Ncut method also reveals near perfect results for this experiment, which is slightly better than its result on LFR easy. As expected, the performance of M-M decreases significantly due to the exacerbation of the resolution limit phenomenon caused by increased crosstalk. M-M provides a total of 6 communities when the ground truth should have been 31 communities.

**LFR harder.** This experiment presses the crosstalk examination even farther. The settings are as follows: $\text{degree}_{\text{avg}} = 20$, $\text{degree}_{\text{max}} = 56$, $\mu_t = \mu_w = 0.5$, $\text{clusters}_{\text{min}} = 32$, and $\text{clusters}_{\text{max}} = 100$. The amount of crosstalk is increased by raising the average and maximum degrees. The NMI performance for this experiment is given in the 6th column of Table 1. The results show the performance of all the non-hierarchical GTAC algorithms decreasing yet again, some of the HGTAC algorithms retain their performance (such as HGTAC-UMF-TL and HGTAC-NMF with $\alpha = 0.11, 0.13, 0.15$), and some HGTAC methods with slightly improved performance (such as HGTAC-NMF with $\alpha = 0.1, 0.14$ and HGTAC-M). The performance decreases for HGTAC-NMF with $\alpha = 0.12$ with this dataset, while HGTAC-UMF-TL maintains perfect performance. The performance of Ncut decreases slightly as compared to an average NMI of 1 on previous experiments. The performance of M-M decreases, yielding 6 groups for a ground truth of 17.

**Gaussian network.** A similar method as that in Ref. [35] is used to generate networks of clusters of two dimensional Gaussian data, with the exception of using $k$ Nearest-Neighbor ($k$NN) to aid in creating a sparse network. Real world network data are often subject to sparsity conditions, and thus the Gaussian network is modeled to reflect this property. This network is generated by first creating four groups of nodes from two dimensional Gaussian data with mean values (−0.3, 0), (0, 0), (0.3, 0), and (0.6, 0) and variances of 0.01 in both dimensions. Next, $k$NN is applied to obtain a neighborhood for each point in this two dimensional space (we use $k = 10$). It is important to note that the sparse network obtained from $k$NN will most often create an asymmetric network. The kernel

$$\kappa(p_1, p_2) = \begin{cases} e^{-\frac{1}{2}\mathrm{dist}(p_1,p_2)^2}, & \text{if } p_1 \text{ is a } k\text{NN of } p_2; \\ 0, & \text{otherwise} \end{cases} \tag{31}$$

is used to generate a similarity matrix with $\mathrm{dist}(p_1, p_2)$ being a distance measure between $p_1$ and $p_2$. Euclidean distance is used in our experiments. The rows of the matrix are randomly permuted prior to supplying the matrix as input to the various GTAC algorithms. Performance results are shown in the 7th column of Table 1. We see that HGTAC-UMF-TL and HGTAC-NMF with varying $\alpha$ give ideal results, but all the non-hierarchical GTAC algorithms perform poorly, and HGTAC-M performs the worst. GTAC-UMF and GTAC-NMF tend to over subdivide groups, which is mitigated with the addition of hierarchical processing. GTAC-M and HGTAC-M perform poorly because they are not suitable for asymmetric networks. Furthermore, Ncut and M-M are not feasible to use on this dataset because the similarity matrix is asymmetric.

**Overall.** From the experiments on the above seven types of networks with less than 5000 nodes, we find that HGTAC-UMF gives relatively satisfying results. HGTAC-NMF with varying $\alpha$ also performs well provided the network is not too small (e.g., Karate data). GTAC-UMF and GTAC-NMF with varying $\alpha$ have better performance on small datasets than their hierarchical versions but worse performance on larger datasets. GTAC-M and HGTAC-M perform well on symmetric networks, but their performance decreases significantly for asymmetric networks. For networks with an asymmetric similarity matrix, Ncut and M-M are not capable of being implemented. Ncut performs well with symmetric similarity matrices but requires the number of groups be provided, and there are no performance guarantees when this parameter is not chosen correctly. M-M can suffer drastically from the resolution limit on networks with many small groups.

**Time complexity on small networks.** To help in comparing the various algorithms, the NMI of their run time is listed in Table 2. Ncut and M-M results are not listed for the Gaussian network because of the asymmetry as mentioned previously.

Ncut consumes the least amount of time on the first five datasets due to the fast implementation of eigen-decomposition on symmetric matrices and $k$-means clustering when implemented using MATLAB. We notice that varying the complexity of the LFR network does not necessarily yield an increase in the time complexity for Ncut, however this is not the case for the

**Table 2 Time complexity NMI ($\mu_{\mathrm{NMI}} \pm$ ci) on small datasets.**

(s)

| Algorithm | Karate | GN network | LFR easy | LFR hard | LFR harder | Gaussian |
|---|---|---|---|---|---|---|
| Ncut | **0.0066 ± 0.004 60** | **0.0097 ± 0.0046** | **0.089 ± 0.015** | **0.11 ± 0.011** | **0.063 ± 0.005** | N/A |
| M-M | **0.0390 ± 0.000 29** | 0.9200 ± 0.0180 | 156.35 ± 2.77 | 156.82 ± 0.52 | 193.86 ± 0.49 | N/A |
| GTAC-UMF-UL | 0.0430 ± 0.003 60 | **0.2100 ± 0.0065** | **1.74 ± 0.056** | **2.48 ± 0.072** | **3.17 ± 0.097** | 10.31 ± 0.26 |
| GTAC-UMF-TL | 0.0410 ± 0.002 80 | **0.2700 ± 0.0051** | **2.33 ± 0.089** | 4.39 ± 0.17 | **5.08 ± 0.21** | 10.43 ± 0.59 |
| GTAC-NMF-0.1 | 0.0460 ± 0.003 90 | 0.3200 ± 0.0560 | 3.04 ± 0.80 | 4.98 ± 0.75 | 6.89 ± 2.02 | 10.14 ± 0.29 |
| GTAC-NMF-0.11 | 0.0450 ± 0.002 50 | 0.3200 ± 0.0560 | 3.03 ± 0.79 | 4.95 ± 0.73 | 6.87 ± 1.99 | 10.01 ± 0.26 |
| GTAC-NMF-0.12 | 0.0450 ± 0.002 60 | 0.3200 ± 0.0510 | 3.03 ± 0.80 | 4.97 ± 0.74 | 6.97 ± 2.10 | **9.83 ± 0.23** |
| GTAC-NMF-0.15 | 0.0440 ± 0.001 80 | 0.3100 ± 0.0460 | 3.13 ± 0.89 | 5.00 ± 0.78 | 7.20 ± 2.33 | 9.96 ± 0.27 |
| GTAC-M | **0.0370 ± 0.001 30** | 0.3000 ± 0.0028 | 2.67 ± 0.084 | 4.95 ± 0.16 | 6.82 ± 0.29 | **8.30 ± 0.22** |
| HGTAC-UMF-TL | 0.0550 ± 0.003 20 | 0.3300 ± 0.0073 | 4.48 ± 0.046 | 7.53 ± 0.14 | 10.05 ± 0.19 | 11.05 ± 0.15 |
| HGTAC-NMF-0.1 | 0.0610 ± 0.008 00 | 0.3400 ± 0.0056 | 3.03 ± 0.055 | 5.31 ± 0.083 | 7.14 ± 2.02 | 11.52 ± 0.17 |
| HGTAC-NMF-0.11 | 0.0560 ± 0.000 48 | 0.3400 ± 0.0055 | 3.60 ± 0.057 | **4.34 ± 0.068** | 7.74 ± 1.99 | 11.17 ± 0.18 |
| HGTAC-NMF-0.12 | 0.0560 ± 0.000 50 | 0.3300 ± 0.0055 | 3.69 ± 0.053 | 4.39 ± 0.076 | 8.68 ± 2.10 | 11.32 ± 0.16 |
| HGTAC-M | 0.0460 ± 0.001 60 | 0.3000 ± 0.0072 | 2.93 ± 0.088 | 5.19 ± 0.16 | 7.31 ± 0.29 | 10.67 ± 0.21 |

other algorithms. For the Karate, GN, and LFR datasets, we observe an increased time complexity as the network size increases, which is caused by the dependency of eigen-decomposition on the size of the similarity matrix.

M-M consumes much more time than Ncut. Its run time on the GN network is the largest of the algorithms used. For an LFR easy network size of 1000, the time consumed increases to 156.35 s which is almost one hundred times that of GTAC-UMF-UL. The results have a similar trend for the LFR hard and harder networks. The run time of M-M appears to grow as an exponential function of the size of the network because of the recursive eigen-decomposition computation.

To compare the time complexity differences between the unique label and tabu list stopping criteria, experiments were conducted using each with GTAC-UMF. Except when using the Karate dataset, we found that UL offers faster run time than TL, which is rooted in the fact that UL suffers from the early stopping problem. Examining all the tabulated experiment run time data reveals the time complexity of GTAC-UMF-TL is occasionally double that of GTAC-UMF-UL, however for real world applications it is recommended to use TL because of the early stopping problem of UL.

For GTAC-NMF, the run time does not differ significantly when varying $\alpha$ values on the same datasets. The time complexity increases as a function of the number of nodes and the configuration difficulty of the networks. As compared with GTAC-UMF-TL (a special case of GTAC-NMF when $\alpha = 0$), GTAC-NMF with $\alpha$ varying from 0.1 to 0.15 consumes more time on the first five datasets due to the computation of the group sizes needed by GTAC-NMF. On the Gaussian dataset, more time is needed by GTAC-UMF-TL than GTAC-NMF due to the data normalization step which improves the convergence of the nodes to their community results. Even though GTAC-NMF is slightly more complicated than GTAC-UMF by design, GTAC-NMF can still save time when a dataset requires or has been normalized.

GTAC-M also yields a run time that increases with the complexity of the network. For the GN network and the three LFR networks, GTAC-M consumes more time than GTAC-UMF-TL, but less than GTAC-NMF. GTAC-M is faster than GTAC-NMF on the Karate and Gaussian networks because GTAC-M stops earlier with many smaller groups, i.e., GTAC-M provides 8 groups on the Karate data while the ground truth is 2, and 181 groups on the 4 groups Gaussian network.

On the Karate dataset the time consumed by HGTAC-NMF with varying $\alpha$ (including HGTAC-UMF which is the special case when $\alpha = 0$), shows a slight dependence on $\alpha$. This difference shows that tuning $\alpha$ will not only affect the performance, but also result in varying time complexities.

HGTAC-M runs longer than GTAC-M because of the hierarchical process, however, HGTAC-M spends less time than HGTAC-NMF algorithms on the Karate, GN, LFR easy, and Gaussian network. HGTAC-M consumes more or less time than HGTAC-NMF depending on the value of $\alpha$'s on the LFR hard and harder networks, signifying that tuning the value of $\alpha$ in HGTAC-NMF will also result in varying time complexities.

#### 7.1.2 Medium LFR network (5000 nodes)

This experiment tests the performance of GTAC-UMF-UL, GTAC-UMF-TL, and GTAC-M on medium sized LFR networks with various values of the mixing parameter $\mu_w$. The number of nodes is chosen to be 5000, average degree to be 20, maximum degree to be 50, and $\mu_w = \mu_t$ varying from 0 to 0.8 in increments of 0.05. For LFR networks with $\mu_w = \mu_t > 0.8$, the crosstalk becomes significantly large enough to overshadow the underlying community structure, and for this reason $\mu_w = \mu_t > 0.8$ is neglected. Figure 3 shows the performance NMI result given by GTAC-UMF-TL, GTAC-UMF-UL, and GTAC-M based on 100 random initializations of the LFR network.

With $\mu_w < 0.55$, all three algorithms give near perfect performance, but with $\mu_w > 0.55$ the performance drops sharply. With $\mu_w > 0.7$, the performances of GTAC-UMF-TL and GTAC-UMF have both reached zero, however GTAC-M performs much better in this range. The reason for GTAC-M's success for larger $\mu_w$ is that GTAC-UMF tends to merge groups in the presence of

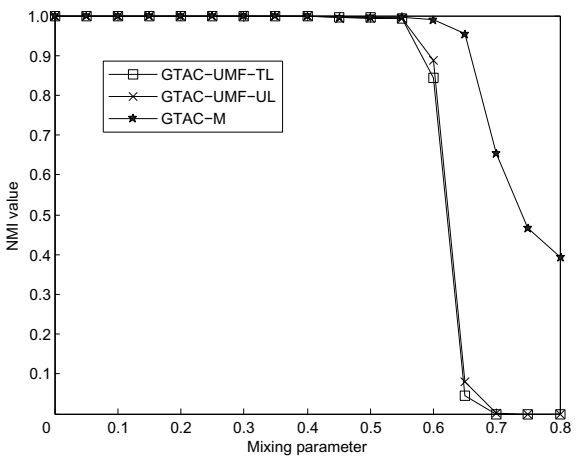



**Fig. 3 Medium LFR network (5000 nodes) performance.**

larger amounts of crosstalk, while GTAC-M is more robust to this noise from the incorporation of both the within talk and crosstalk into its optimizing function.

### 7.1.3 Large LFR network (1 million nodes)

The purpose of this experiment is to compare the performance of GTAC-UMF against the Louvain method when utilized on big networks with ground truth information.

In order to generate this large of an LFR network, Amazon EC2 of type M1 extra large instance[41] was used in order to create a one million-node network using code from Ref. [40]. A high memory cluster eight extra large instance[41] provided by Amazon EC2 was also tested to generate larger networks with greater numbers of nodes, however the LFR code was only capable of generating a maximum of approximately one million nodes. The parameter values used to configure the LFR network were nodes $= 10^6$, $\text{degree}_{\text{avg}} = 20$, yielding approximately 20 million links. Other parameters were set to model[32]: $\text{degree}_{\text{max}} = 2.5 \times \text{degree}_{\text{avg}}$, $\text{cluster}_{\text{max}} = 5 \times \text{cluster}_{\text{min}}$, and the exponents for the degree sequence and community size distribution were $-2$ and $-1$, respectively. These settings were used to generate 17 networks with the same configuration with the exception of varying the mixing parameter $\mu$ from 0 to 0.8 with a step size of 0.05.

We utilized the MapReduce framework on Amazon services to speed up computation. The resulting output provided a matrix of size $2 \times 10^5$ whose first column consisted of the node ID numbers and the second the node labels. A matrix of this size was not able to be processed efficiently in MATLAB, therefore the method of comparison was changed from using a time complexity NMI to using a storage requirement NMI. Matrix manipulation was not used to determine the joint label set for use in calculating the NMI, instead the data was processed in series element-wise.

The results provided by the Louvain method from Ref. [42] was given in *.tree file format, in which the levels of the tree structure detected by each loop of Louvain was separated by a row of zeros. For each row of the tree the node IDs were renamed to represent labels corresponding to the total number of clusters for that level, i.e., the maximal valued node ID of the second level of the tree was not correspondent to the number of nodes belonging to the first level, but rather the number of groups in the first level. In order to calculate the NMI, the node IDs provided in the last level needed to be matched with the node labels from the first level. The number of communities detected by the Louvain method was 12 630 for $\mu_w = 0.4$.

Figure 4 shows the storage NMI results of GTAC-UMF and Louvain on the Large LFR Network. The sharp decrease up to $\mu = 0.05$ highlights the sensitivity of Louvain to crosstalk when dealing with larger networks. As the levels of crosstalk are increased by varying $\mu$ from 0.1 to 0.6, the NMI values of Louvain continues to decrease nearly linearly. With $\mu > 0.7$, the NMI sharply drops again. On the contrary, GTAC-UMF maintains a more stable near perfect result with $0 \leqslant \mu \leqslant 0.6$. The Louvain method's performance is significantly less than that of GTAC-UMF due to the hierarchical processing, but dramatically increases nearing that of GTAC-UMF when only using the first level of community detection offered prior to performing the hierarchical process (which is essentially the GTAC-M algorithm). This helps to identify the hierarchical processing involved in the Louvain method as a drawback. Since maximizing modularity is a global metric, the Louvain method uses a hierarchical process to approach a solution. Understanding that the Louvain method is captured by the GTAC theoretical framework helps to show that this approach can maximize modularity locally, ultimately avoiding local optimum solutions. Unfortunately, increasing modularity during the hierarchical process toward a global optimal may not necessarily offer the best solution since the global optimum may not be achieved by the ground truth. GTAC-UMF does not utilize a hierarchical process, which allows for appropriate community detection results on larger datasets.

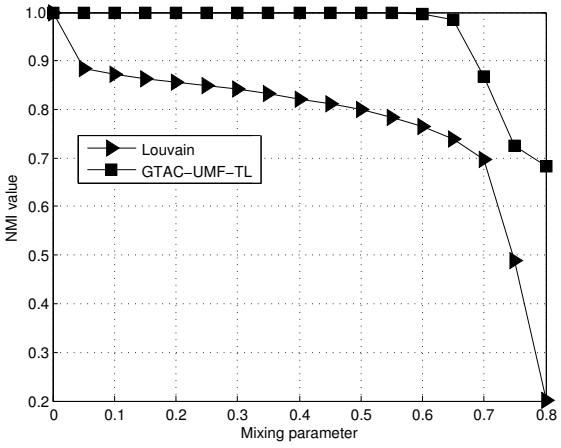



**Fig. 4 Large LFR network (1 million nodes) performance with mixing parameter in range (0 to 0.8): GTAC-UMF vs. 1st level Louvain.**

The results of GTAC-UMF in the large LFR network experiment outperforms those of the medium LFR network experiment for the same $\mu$ value, as can be seen by comparing Figs. 3 and 4. This is due to the average crosstalk between groups decreasing as the number of nodes and clusters increases, making the possibility of the crosstalk being more dominant than the group within talk less likely.

Experimentation using GTAC-M reveals performance similar to that of GTAC-UMF, with negligible drop of NMI from 1 to 0.999 when $\mu$ changes from 0 to 0.5. Comparing GTAC-M and GTAC-UMF, Fig. 5 shows that GTAC-UMF slightly outperforms GTAC-M when $\alpha < 0.55$. GTAC-UMF performs better than GTAC-M and HGTAC-M on larger LFR networks.

Investigations on the effects of $\alpha$ on GTAC-NMF reveal that $\alpha = 0.12$ works particularly well, as shown in Fig. 6. This choice of the parameter offers results for GTAC-NMF which are nearly the same as GTAC-UMF.

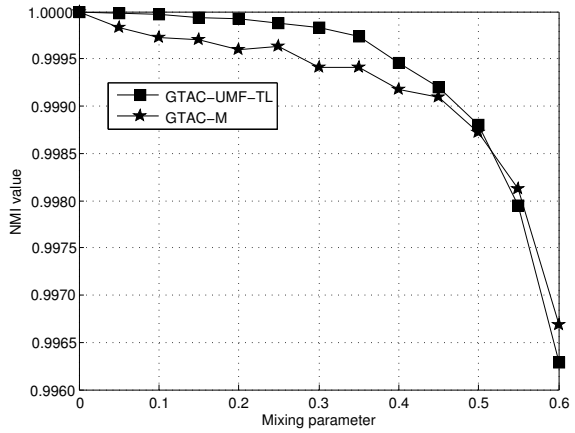



**Fig. 5 Large LFR network (1 million nodes) performance with mixing parameter in range (0 to 0.6): GTAC-UMF vs. GTAC-M.**

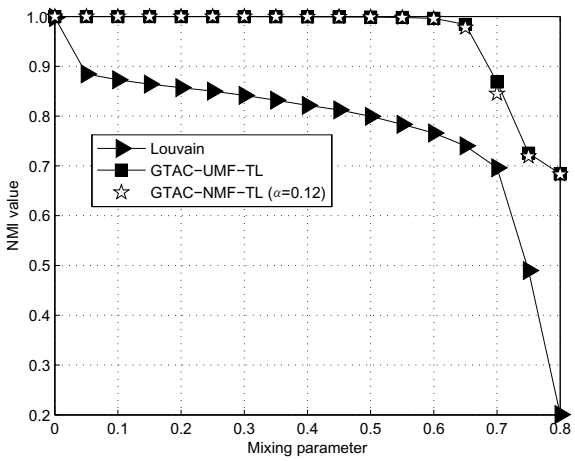
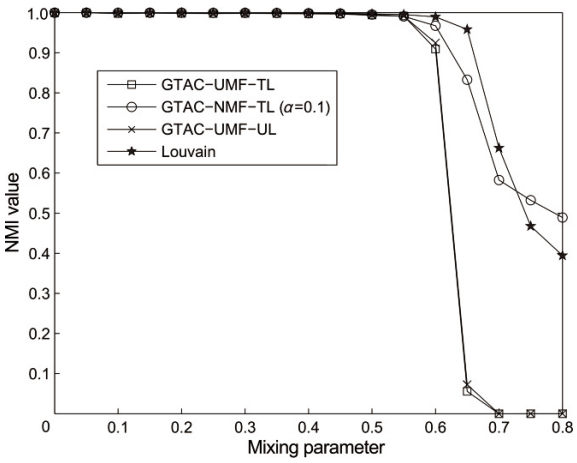



**Fig. 6 Large LFR network performance: GTAC-NMF using $\alpha$ =0.12.**

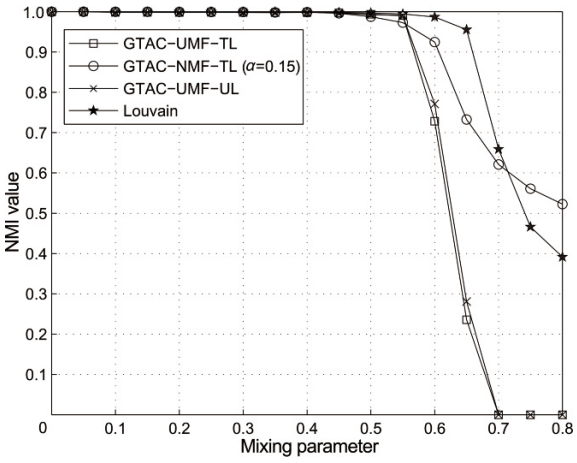
#### 7.1.4 Performance of GTAC-NMF under varying $\alpha$ values

As discussed in Section 4.2.2, GTAC-UMF tends to merge groups together while GTAC-NMF with large $\alpha$ (e.g., $\alpha \geqslant 1$) provides many small groups. It is trivial to view GTAC-UMF as a special case of GTAC-NMF when $\alpha = 0$. In finding the best clustering results, we consider leveraging both algorithms by finding a parameter value for GTAC-NMF in the range $0 \leqslant \alpha \leqslant 1$. The following details the experiments conducted over different values of $\alpha$ on the medium LFR network.

Transitioning away from GTAC-UMF (when $\alpha = 0$), there is an improved performance with parameters $\alpha = 0.1$ and $\alpha = 0.15$, as visualized in Fig. 7. Although the NMI from GTAC-NMF is still obviously below GTAC-M for $0.6 \leqslant \mu_w \leqslant 0.7$, GTAC-M becomes less appealing when $\mu_w \geqslant 0.7$. While there is better performance for $\alpha = 0.15$ than with $\alpha = 0.1$ at larger values of $\mu_w$, it should be noted that there is actually a




(a) $\alpha = 0.1$




(b) $\alpha = 0.15$

**Fig. 7 Performance of GTAC-NMF under varying $\alpha$ values on medium LFR network.**

small gain in performance for smaller values of $\mu_w$ for $\alpha = 0.1$. These phenomena demonstrate the capability of varying the performance of GTAC-NMF algorithms by means of tuning $\alpha$. On using real world data, $\alpha$ can be tuned using training sets prior to the use of GTAC-NMF on subsequent datasets.

While other ranges of $\alpha$ were tested in these experiments, good performance is only achieved by most of the datasets when $\alpha$ is near 0.1. For $\alpha \geqslant 0.2$, the phenomenon of community over-division arises forming many extraneous subgroups.

**Time complexity on large LFR network.** For a one million-node dataset with $\mu = 0.4$ on the LFR network, the time consumed was 46 min when implemented on five M1.large instances consisting of two vCPUs and four ECUs with 7.5 GB memories on EC2[41].

### 7.2 Real world data

This experiment aims to evaluate the performance of different GTAC algorithms on a large, real world dataset from the social network Twitter.

The Twitter network dataset is an unweighted and directed graph consisting of 41.7 million nodes and 1.47 billion links. The data file "*twitter_rv.net*" of size 25 GB was obtained from Ref. [36]. This dataset is formatted such that the first column contains the user and the second column the follower. The data were preprocessed to remove 1.5 million isolated nodes (users) who neither followed nor was followed by any other node, leaving 40.1 million nodes remaining.

After experimentation, Map()/Reduce() was used for performance analysis on EC2. As is the case with most real world datasets, the Twitter network data do not provide ground truth information for clustering, so instead, the within talk percentage was used as a benchmark. The within talk percentage was determined by the ratio of the within talk to the total talk. The total talk was the row count of the dataset (the number of connections from non-isolated nodes), which was 1.47 billion. Counting the number of within talk is performed by comparing the labels of the algorithm results to the dataset node IDs and incrementing the within talk count each time the node IDs belong to the same community label. The total within talk was calculated to be 687 million from the experiment, resulting in a within talk percentage of 46.8%. This reveals that GTAC-UMF can attribute almost half of the links towards being within the $N$ groups and the remainder towards the crosstalk between the $N \times (N-1)$ group pairs. The community detection result given by GTAC-UMF reveals a suitable structure for the communities of the Twitter network. These results show the capability of the GTAC-UMF algorithm as applied to large real world networks with millions of nodes and billions of links.

The Louvain method was attempted on this dataset, however due to memory constraints the code used from Ref. [42] was not able to processed.

**Time complexity on twitter network.** This experiment was performed using 10 M1.xlarge instances consisting of 4 vCPUs and 8 ECUs with 15 GB memory on EC2[41]. The run time for this experiment was 3 hours and 10 minutes.

## 8 Conclusion

Using a game theoretic approach, a novel theoretical framework was developed to perform community detection on various sized datasets both with and without ground truth information. A family of algorithms, game theoretical approach for clustering, was developed with variations according to the payoff functions of particular games. These algorithms focus on providing a scalable means of performing non-overlapping clustering when dealing with potentially unweighted and asymmetric similarity matrices for datasets. The derived GTAC-UMF and GTAC-NMF algorithms are shown to be near-linear time complexity and scalable to larger datasets in our experimentation. For the first time this work provides meaningful performance evaluations on large datasets (consisting of one million nodes) with ground truth information, while other works have focused more on unlabeled real world data. GTAC provides a framework which includes some of current algorithms as special cases, and also provides various cost functions as examples to show how generalized scheme of GTAC to fit into application scenarios. By tuning the parameter $\alpha$ in GTAC-NMF, varying levels of clustering performance can be achieved to satisfy various application needs. Experimental results show the success of these proposed schemes and their potential advantages in the area of big data analytics as compared to other well known algorithms.

### Acknowledgment

The authors would like to thank Dong Wang for his aid in conducting the experiments on the cloud computing platform.

### References

[1] S. Fortunato, Community detection in graphs, *Phys. Rep.*,

vol. 486, nos. 3–5, pp. 75–174, 2010.

[2] U. Von Luxburg, A tutorial on spectral clustering, *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

[3] J. B. Shi and J. Malik, Normalized cuts and image segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

[4] M. E. J. Newman, Modularity and community structure in networks, *Proc. Natl. Acad. Sci. USA*, vol. 103, no. 23, pp. 8577–8582, 2006.

[5] S. Fortunato and M. Barthélemy, Resolution limit in community detection, *Proc. Natl. Acad. Sci. USA*, vol. 104, no. 1, pp. 36–41, 2007.

[6] A. Lancichinetti and S. Fortunato, Limits of modularity maximization in community detection, *Phys. Rev.E*, vol. 84, no. 6, p. 066122, 2011.

[7] M. Rosvall and C. T. Bergstrom, Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems, *PLoS One*, vol. 6, no. 4, p. e18209, 2011.

[8] M. Rosvall and C. T. Bergstrom, Maps of random walks on complex networks reveal community structure, *Proc. Natl. Acad. Sci. USA*, vol. 105, no. 4, pp. 1118–1123, 2008.

[9] U. N. Raghavan, R. Albert, and S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, *Phys. Rev.E*, vol. 76, no. 3, p. 036106, 2007.

[10] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre, Fast unfolding of communities in large networks, *J. Statist. Mech.: Theory Exp.*, vol. 2008, no. 10, p. P10008, 2008.

[11] A. Browet, P. Absil, and P. Van Dooren, Fast community detection using local neighbourhood search, arXiv preprint arXiv: 1308.6276, 2013.

[12] Z. Bu, H. J. Li, C. C. Zhang, J. Cao, A. H. Li, and Y. Shi, Graph K-means based on leader identification, dynamic game, and opinion dynamics, *IEEE Trans. Knowledge Data Eng.*, vol. 32, no. 7, pp. 1348–1361, 2019.

[13] J. Cao, Z. Bu, Y. Y. Wang, H. Yang, J. C. Jiang, and H. J. Li, Detecting prosumer-community groups in smart grids from the multiagent perspective, *IEEE Trans. Syst., Man, Cybernet.: Syst.*, vol. 49, no. 8, pp. 1652–1664, 2019.

[14] H. J. Li, Z. Bu, Z. Wang, and J. Cao, Dynamical clustering in electronic commerce systems via optimization and leadership expansion, *IEEE Trans. Ind. Informatics*, vol. 16, no. 8, pp. 5327–5334, 2019.

[15] H. J. Li, Z. Bu, Z. Wang, J. Cao, and Y. Shi, Enhance the performance of network computation by a tunable weighting strategy, *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 3, pp. 214–223, 2018.

[16] H. J. Li, Q. Wang, S. F. Liu, and J. Hu, Exploring the trust management mechanism in self-organizing complex network based on game theory, *Phys.A: Statist. Mech. Appl.*, vol. 524, p. 123514, 2020.

[17] P. J. McSweeney, K. Mehrotra, and J. C. Oh, A game theoretic framework for community detection, in *Proc. 2012 IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining* (*ASONAM*), Istanbul, Turkey, 2012, pp. 227–234.

[18] R. Narayanam and Y. Narahari, A game theory inspired, decentralized, local information based algorithm for community detection in social graphs, in *Proc. 21$^{st}$ Int. Conf. on Pattern Recognition* (*ICPR*), Tsukuba, Japan, 2012, pp. 1072–1075.

[19] R. I. Lung, A. Gog, and C. Chira, A game theoretic approach to community detection in social networks, in *Nature Inspired Cooperative Strategies for Optimization* (*NICSO 2011*), D. A. Pelta, N. Krasnogor, D. Dumitrescu, C. Chira, and R. Lung, eds. Berlin, Germany: Springer, 2011, pp. 121–131.

[20] W. Chen, Z. M. Liu, X. R. Sun, and Y. J. Wang, A game-theoretic framework to identify overlapping communities in social networks, *Data Min. Knowl. Disc.*, vol. 21, no. 2, pp. 224–240, 2010.

[21] W. Chen, Z. M. Liu, X. R. Sun, and Y. J. Wang, Community detection in social networks through community formation games, in *Proc. Twenty-Second Int. Joint Conf. on Artificial Intelligence-Volume Volume Three*, Barcelona, Spain, 2011, pp. 2576–2581.

[22] Y. Narahari and R. Narayanam, Game theoretic models for social network analysis, in *Proc. 20$^{th}$ Int. Conf. on World Wide Web*, Hyderabad, India, 2011, pp. 291–292.

[23] M. J. Osborne, *An Introduction to Game Theory*. New York, NY, USA: Oxford University Press, 2004.

[24] J. Von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior* (*Commemorative Edition*). Princeton, NJ, USA: Princeton University Press, 2007.

[25] J. Nash, Equilibrium points in n-person games, *Proc. Natl. Acad. Sci. USA*, vol. 36, no. 1, pp. 48–49, 1950.

[26] J. Nash, Non-cooperative games, *Ann. Math.*, vol. 54, no. 2, pp. 286–295, 1951.

[27] R. D. McKelvey and A. McLennan, Computation of equilibria in finite games, *Handb. Comput. Econ.*, vol. 1, pp. 87–142, 1996.

[28] M. E. J. Newman, Analysis of weighted networks, *Phys. Rev.E*, vol. 70, no. 5, p. 056131, 2004.

[29] K. Altisen, S. Devismes, A. Gerbaud, and P. Lafourcade, Analysis of random walks using tabu lists, in *Proc. 19$^{th}$ Int. Colloquium on Structural Information and Communication Complexity*, Reykjavik, Iceland, 2012, pp. 254–266.

[30] G. Karypis, E. H. Han, and V. Kumar, Chameleon: Hierarchical clustering using dynamic modeling, *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[31] A. Lancichinetti, S. Fortunato, and F. Radicchi, Benchmark graphs for testing community detection algorithms, *Phys. Rev.E*, vol. 78, no. 4, p. 046110, 2008.

[32] A. Lancichinetti and S. Fortunato, Community detection algorithms: A comparative analysis, *Phys. Rev.E*, vol. 80, no. 5, p. 056117, 2009.

[33] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, Finding statistically significant communities in networks, *PLoS One*, vol. 6, no. 4, p. e18961, 2011.

[34] S. Gregory, Finding overlapping communities in networks by label propagation, *New J. Phys.*, vol. 12, no. 10, p. 103018, 2010.

[35] B. H. Sun and D. P. Wu, Self-organizing-queue based clustering, *IEEE Signal Process. Lett.*, vol. 19, no. 12, pp. 902–905, 2012.

[36] H. Kwak, C. Lee, H. Park, and S. Moon, What is twitter, a social network or a news media? in *Proc. 19$^{th}$ Int. Conf. on World Wide Web*, Raleigh, NC, USA, 2010, pp. 591–600.

[37] L. Danon, A. J. Duch, and A. Arenas, Comparing community structure identification, *J. Statist. Mech.: Theory Exp.*, vol. 2005, no. 9, pp. 1–10, 2005.

[38] J. J. Whang, X. Sui, and I. S. Dhillon, Scalable and memory-efficient clustering of large-scale social networks, in *Proc. IEEE 12$^{th}$ Int. Conf. on Data Mining* (*ICDM*), Brussels, Belgium, 2012, pp. 705–714.

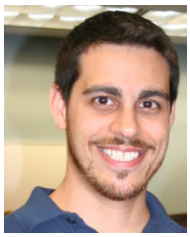
[39] W. W. Zachary, An information flow model for conflict and fission in small groups, *J. Anthropol. Res.*, vol. 33, no. 4, pp. 452–473, 1977.

[40] A. Lancichinetti, LFR benchmark code, https://sites.google.com/site/andrealancichinetti/files, 2010.

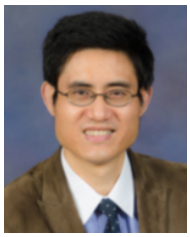
[41] Amazon EC2 instance types, http://aws.amazon.com/en/ec2/instance-types/, 2013.

[42] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, Louvain method: Finding communities in large networks, https://sites.google.com/site/findcommunities/, 2008.

**Baohua Sun** received the BE degree from Beijing University of Posts and Telecommunications, Beijing, China in 2006, the ME degree in electrical engineering from Tsinghua University, Beijing, China in 2010, and the PhD degree in electrical and computer engineering from University of Florida, Gainesville, US in 2013. He is now the director of AI Research at Gyrfalcon Technology, Milpitas, US. His current research interests include Natural Language Processing (NLP), two-dimensional word embedding, and applying deep learning to Tabular data Machine Learning (TML).



**Richard Al-Bayaty** is pursuing PhD degree at Department of Electrical and Computer Engineering, University of Florida. His current research interest includes community detection and machine learning.



**Dapeng Wu** received the PhD degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA in 2003. He is a professor at the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA. His research interests are in the areas of networking, communications, signal processing, computer vision, machine learning, smart grid, and information and network security. He received University of Florida Term Professorship Award in 2017, University of Florida Research Foundation Professorship Award in 2009, AFOSR Young Investigator Program (YIP) Award in 2009, ONR Young Investigator Program (YIP) Award in 2008, NSF CAREER Award in 2007, the IEEE Circuits and Systems for Video Technology (CSVT) Transactions Best Paper Award for Year 2001, and the Best Paper Awards in IEEE GLOBECOM 2011 and International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine) 2006. Currently, he serves as the editor in chief of *IEEE Transactions on Network Science and Engineering*. He was the founding editor-in-chief of *Advances in Multimedia* between 2006 and 2008, and an associate editor for *IEEE Transactions on Communications*, *IEEE Transactions on Signal and Information Processing over Networks*, *IEEE Signal Processing Magazine*, *IEEE Transactions on Circuits and Systems for Video Technology*, *IEEE Transactions on Wireless Communications*, and *IEEE Transactions on Vehicular Technology*. He is also a guest-editor for *IEEE Journal on Selected Areas in Communications* (*J-SAC*), Special Issue on Cross-layer Optimized Wireless Multimedia Communications, and Special Issue on Airborne Communication Networks. He has served as Technical Program Committee (TPC) chair for IEEE INFOCOM 2012, and TPC chair for IEEE International Conference on Communications (ICC 2008), Signal Processing for Communications Symposium, and as a member of executive committee and/or technical program committee of over 100 conferences. He was elected as a distinguished lecturer by IEEE Vehicular Technology Society in 2016. He has served as the chair for the Award Committee, and the chair of Mobile and wireless multimedia Interest Group (MobIG), Technical Committee on Multimedia Communications, and IEEE Communications Society. He was an elected member of Multimedia Signal Processing Technical Committee, IEEE Signal Processing Society from 2009 to 2012. He is the fellow of IEEE.

**Qiuyuan Huang** received the PhD degree from University of Florida in 2017. She is a senior researcher in the deep learning group at Microsoft Research, Redmond, WA, USA. Her current research interests are focused on the deep learning and natural language processing areas; topics include neural-symbolic for reasoning, self-supervised learning, and multi-modal intelligence.