Soc. Netw. Anal. Min. (2016) 6:77
DOI 10.1007/s13278-016-0390-5

ORIGINAL ARTICLE

Identifying community structures in dynamic networks

Hamidreza Alvari[1] · Alireza Hajibagheri[1] · Gita Sukthankar[1] · Kiran Lakkaraju[2]

Received: 11 November 2015 / Revised: 28 August 2016 / Accepted: 31 August 2016 / Published online: 12 September 2016

Abstract Most real-world social networks are inherently dynamic, composed of communities that are constantly changing in membership. To track these evolving communities, we need dynamic community detection techniques. This article evaluates the performance of a set of game-theoretic approaches for identifying communities in dynamic networks. Our method, D-GT (Dynamic Game-Theoretic community detection), models each network node as a rational agent who periodically plays a community membership game with its neighbors. During game play, nodes seek to maximize their local utility by joining or leaving the communities of network neighbors. The community structure emerges after the game reaches a Nash equilibrium. Compared to the benchmark community detection methods, D-GT more accurately predicts the number of communities and finds community assignments with a higher normalized mutual information, while retaining a good modularity.

Research at University of Central Florida was supported by NSF award IIS-0845159. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the US Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

✉ Gita Sukthankar
gitars@eecs.ucf.edu

Hamidreza Alvari
halvari@eecs.ucf.edu

Alireza Hajibagheri
hajibagheri@eecs.ucf.edu

Kiran Lakkaraju
klakkar@sandia.gov

[1] University of Central Florida, Orlando, FL, USA

[2] Sandia National Labs, Albuquerque, NM, USA

Keywords Community detection · Dynamic social networks · Game-theoretic models

1 Introduction

The natural flux of people's changing social ties and interests generates a dynamic social network. This invisible network can be observed by capturing daily or weekly snapshots of user activities on social media platforms and massively multiplayer online games (MMOGs). It is informative to study changes in the network at the *community level*, as well as the individual level. Communities are emergent groups that are created as people form highly connected subnetworks with their families, co-workers and friends. Often communities are formed by participants with the same goals, interests, or a geographic location. For instance, in MMOGs, network communities may emerge from guilds of players with common economic interests or alliances who share strategic goals. As the network changes, user groups can grow, shrink, or disappear, causing drastic changes in the total number of network communities.

Community detection can help us understand the hidden social structure of the user populations, but the dynamic aspect of networks can pose problems for standard algorithms. Our method uses stochastic optimization to find the best community structure, assuming that the nodes are modeled as rational players who seek to maximize their personal utility while playing a *community membership game* with neighboring nodes. In this game, the active agent decides to join or leave different communities; agents receive benefits from being part of the same community as their network neighbors but are penalized for joining too many communities. The Nash equilibrium of the current

game corresponds to the community structure of the current snapshot. As the network evolves, agents usually find it advantageous to modify their community membership strategy. This article examines the performance of varying the amount of information propagated from prior snapshots. Even in dynamic networks, there are many nodes that retain the same community membership or rejoin their former communities. Thus, propagating information from previous snapshots can provide more favorable initialization conditions for the stochastic optimization procedure.

Much of the power of the D-GT framework [originally introduced in Alvari et al. (2014a)] lies in its potential for customization; however, without guidance, end users can be overwhelmed by myriad choices. Our aim in this article is to present a comprehensive analysis of the algorithm's performance in common scenarios; while retaining the same basic game-theoretic model, we evaluate different variations of our procedure. First, we examine the performance of different utility functions, a similarity-based utility function (Alvari et al. 2011) versus the use of a personalized modularity function (Chen et al. 2010). Then we compare different initialization approaches in which the following information is propagated: (1) no information (D-GTS); (2) the union of the community membership information over all snapshots (D-GT); (3) community membership information from the previous snapshot (D-GTP); and (4) ground truth information for a small seed set (D-GTG). Our game-theoretic model is robust to minor changes in the procedure and most variants outperform the benchmarks.

Our experiments were conducted on networks created from different dynamic processes: internet routers (AS-Oregon Graph, the AS-Internet Routers Graph), shifting organization structure (Enron Email dataset), citation graphs from arXiv (hep-ph) and player interactions in massively multiplayer online games (Travian Messages, Travian Trades). Results were compared against five other methods: LabelRankT, iLCD, OSLOM, InfoMap and Louvain in terms of normalized mutual information (NMI), modularity and the number of detected communities. The next section presents an overview of related work on community detection in dynamic networks. Section 3 describes our problem formulation and our proposed method (Dynamic Game-Theoretic community detection). Experimental results are provided in Sect. 4, before we conclude the article (Sect. 5).

2 Related work

The problem of community detection in static networks has appeared in multiple disciplines including sociology and computer science. This has yielded a diverse set of approaches ranging from traditional network structure-based algorithms (Girvan and Newman 2002; Newman 2006, 2004), optimization techniques (Alvari et al. 2011; Chen et al. 2010), label propagation (Raghavan et al. 2007; Xie and Szymanski 2011, 2012), propinquity (Zhang et al. 2009) and information diffusion (Hajibagheri et al. 2013, 2012). Detecting community structure in dynamic networks, on the other hand, has attracted less research attention due to the complexity of the problem and dearth of good datasets. There are some community detection algorithms originally designed for static networks that continue to perform well in dynamic datasets. For instance, Lancichinetti et al.'s OSLOM (Order Statistics Local Optimization Method) works on single snapshots but also benefits from information from previous network partitions. Like D-GT, OSLOM's optimization procedure can be initialized with the partition from the previous snapshot; it aims to optimize cluster significance with respect to a global null model (Lancichinetti et al. 2011). We use this method as one of our benchmarks, along with two other static community detection algorithms: Louvain (Blondel et al. 2008) and InfoMap (Rosvall and Bergstrom 2008). These algorithms perform well on many static community detection problems and have the benefit of being fast to compute on a single network snapshot.

Other network properties have also been used to perform dynamic community detection; for instance, Hui et al. (2007) proposed a distributed method for community detection in which modularity was used as a measure instead of the objective function. QCA (Quick Community Adaptation) is a modularity-based approach that focuses explicitly on the changes in the network structure, rather than recomputing community structure from scratch at each time step (Nguyen et al. 2014). In this article, we evaluate the use of personal modularity as an alternative gain function to neighborhood similarity.

Some studies have focused on studying the evolution of communities over time. For instance (Hopcroft et al. 2004) identified subsets of nodes, "natural communities," which were stable to small perturbations of the input data. Communities detected in later snapshots were matched to earlier snapshots using the natural community tree structure. Palla et al. (2009) proposed an innovative method for detecting communities in dynamic networks based on the k-clique percolation technique; in their approach, communities are defined as adjacent k-cliques that share $k-1$ nodes.

Machine learning has also been employed to model changes in community structure; for instance, Takaffoli et al. (2014) predict transitions in community structure by learning supervised machine learning classifiers. This requires data on past transitions to train the classifiers, which limits its applicability to certain datasets. Sun et al.

(2007) adopt a data mining approach to detect clusters on time-evolving graphs; community discovery and change detection are performed using the minimum description length (MDL) paradigm.

Rather than independently detecting communities at each snapshot and matching them, another option is to make a local decision to add nodes to existing communities when new edges appear in the network. One of our benchmarks, iLCD (intrinsic longitudinal community detection) (Cazabet et al. 2010), updates the community structure of the network based on time-stamped sets of edges. Nodes are added to communities if its mean number of second neighbors and robust second neighbors exceeds the current average for the community. However, this model is limited to certain types of network changes and cannot handle interconnected pairs of nodes being simultaneously added or the removal of edges.

Optimization can be used to identify minimum cost community assignments in dynamic graphs. FacetNet (Lin et al. 2008) is a framework for analyzing communities in dynamic networks based on an optimization of snapshot costs. It is guaranteed to converge to a local optimal solution; however, its convergence speed is slow, and it needs to be initialized with the number of communities which is usually unknown in practice. Folino and Pizzuti (2014) modeled dynamic community detection as a multi-objective optimization problem. Their approach is parameter free and uses evolutionary clustering to optimize a dual objective function. The first objective selects for highly modular structures at the current time step, and the second minimizes the differences between community structures in the current and previous time steps. D-GT also uses a stochastic optimization procedure, but all of the agents individually optimize their utilities based on local network information.

The Markov Cluster Algorithm (MCL) (Van Dongen 2000) identifies graph clusters by computing the probabilities of random walks through the graph; flow simulations are performed by alternating expansion (matrix squaring) with inflation operations (Hadamard powers). Due to its mathematical simplicity, it is popular for community detection in many domains but is slow and often overestimates the number of communities in the dataset. Regularized MCL (Satuluri and Parthasarathy 2009) is a variant of MCL that prevents overfitting by taking into account neighbor flows. It can also be used within a multi-level framework (Multi-level Regularized MCL) to speed up computation by executing a sequence of coarsening operations on the graph before executing R-MCL.

LabelRankT (Xie et al. 2013) shares some similarities with the MCL techniques described above while improving upon them in several ways; it is a label propagation approach in which the inflation operation is applied to the label distribution matrix rather than to the adjacency matrix. Each node requires only local information during propagation, making it more scalable than MCL and amenable to parallelization. Due to LabelRankT's strong performance and good implementation, it was selected as the best label propagation benchmark for our work.

Our proposed method (D-GT) attempts to simulate the decision-making process of the individuals creating the communities, rather than focusing on statistical correlations between labels of neighboring nodes. We believe that exploiting game theory for dynamic community detection yields more realistic, fine-grained communities since intrinsically game theory is a good representation for expressing the behavior of individuals and strategic interactions among them (Adjeroh and Kandaswamy 2007).

In previous work, we have demonstrated the success of game-theoretic approaches in static community detection across several domains, including detecting guilds in massively multiplayer online games (Alvari et al. 2014b) and predicting trust between users on e-commerce sites (Beigi et al. 2014). Many of these domains featured overlapping communities in static networks (Alvari et al. 2011, 2013); however, in this article, the datasets are dynamic, but not overlapping. D-GT makes a hard assignment at the termination of the stochastic optimization procedure by selecting the community assignment with the highest utility function. In this article, we investigate the performance of different gain functions and initialization procedures, on a variety of evaluation metrics (modularity, NMI, number of detected communities). The strength of the D-GT framework is its versatility; our results show that substantial performance improvements can be achieved by customizing it for the problem at hand.

3 Method

First, we provide the formal definition of the problem and the notations used throughout the paper. Given snapshots $\mathbf{T} = \{T^t \mid \forall t, t = 1, ..., M\}$ of a dynamic network and their corresponding underlying graphs $\mathbf{G}^t = (V^t, E^t)$, with $n^t = |V^t|$ vertices and $m^t = |E^t|$ edges, where $t = 1,...,M$, we aim to detect community structure $\mathbf{C}=\{C^t \mid \forall t, t = 1, ..., M\}$ of the network. Table 1 shows the symbols and definitions used throughout the paper.

We leverage a dynamic agent-based model to detect communities by optimizing each user's utility through a stochastic search process (Alvari et al. 2011). In this article, we evaluate four different variants of the procedure: D-GT (Dynamic Game-Theoretic community detection), D-GTP (D-GT with passing one Previous Snapshot) D-GTS (D-GT with Separate Runs) and D-GTG (D-GT with passing Ground Truth).

Table 1 Definition of symbols

Symbol	Definition
$\mathbf{T}$	Set of snapshots
$\mathbf{C}$	Set of communities
$\mathbf{G}^t$	Graph of tth snapshot with no self-edges
C^t	Community structure of tth snapshot
C_k^t	kth Community in C^t
m^t, n^t	Number of edges and vertices of G^t
$\mathbf{A^t}$	Adjacency matrix of G^t
$\mathbf{S^t}$	Profile of strategies of tth snapshot
s_i^t	ith Agent's strategy in tth snapshot
g_i^t	ith Agent's gain function in tth snapshot
l_i^t	ith Agent's loss function in tth snapshot
u_i^t	ith Agent's utility function in tth snapshot
c_{ij}^t	Similarity between ith and jth agents in the tth Snapshot

3.1 Dynamic Game Theory (D-GT)

In this section, we present the framework for D-GT. We treat the process of community detection as an iterative game performed in a dynamic multi-agent environment in which each node of the underlying graph is a selfish agent who decides to maximize its total utility u_i. Note that hereafter we use the terms *node*, *user* and *agent* interchangeably. The terms *pool*, *list*, or *set* of agents are used to denote the set of nodes maintained during each game.

During the community formation game, each agent assesses whether taking an action (*join*, *switch*, or *leave*) (Table 2) will increase its utility. An agent joins a new community $c^t \subseteq C^t$ by adding its label to s_i^t. It then gains utility u_{Join}^t and its community membership changes:

$$s_i^t \leftarrow s_i^t \cup \{c^t\}. \tag{1}$$

It may leave one of its own communities, say c'^t by removing its label from s_i^t which results in utility u_{Leave}^t and changes the community membership as follows:

$$s_i^t \leftarrow s_i^t / \{c'^t\}. \tag{2}$$

The agent can also simultaneously switch communities:

$$s_i^t \leftarrow s_i^t / \{c'^t\}, \quad s_i^t \leftarrow s_i^t \cup \{c^t\}. \tag{3}$$

Table 2 Definition of possible actions

Action	Definition
Join	Add a new label to s_i^t
Leave	Remove a label from s_i^t
Switch	Remove a label from s_i^t and add a new one
No operation	No action is performed

Even though the switch action is not strictly necessary, having this additional action speeds up the convergence of the stochastic search process.

The new utility $u_i'^t$ for this agent is updated as follows:

$$u_i'^t \leftarrow \max\{u_{\text{Join}}^t, u_{\text{Leave}}^t, u_{\text{Switch}}^t, u_{\text{noOp}}^t\}. \tag{4}$$

To reduce computation time, only communities containing the agent's nearest network neighbors are considered as candidates for the join/switch operation, and we independently identify the best communities for the join and leave operations. If no action improves the agent's utility, it does not change its strategy (*no operation*).

The set of all communities at the tth snapshot is denoted by C^t. We define a strategy profile $S^t = (s_1^t, s_2^t, ..., s_n^t)$ which represents the set of all strategies of all agents, where $s_i^t \subseteq C^t$ denotes the strategy of agent i, i.e., the set of its labels at snapshot t. In our framework, for each snapshot, the best response strategy of an agent i with respect to strategies S_{-i}^t of other agents is calculated as:

$$\arg\max_{s_i^t \subseteq C^t} u_i^t(S_{-i}^t, s_i^t) \tag{5}$$

Agents are selected randomly, without replacement, until all the agents have had the opportunity to play the community formation game in order to guarantee adequate exploration of the strategy search space. The utility function for each agent is calculated by combining the benefit of its community memberships, based on a gain function and subtracting losses incurred:

$$u_i^t(S_{-i}^t, s_i^t) = g_i^t(S_{-i}^t, s_i^t) - l_i^t(S_{-i}^t, s_i^t), \tag{6}$$

We have experimented with two variants on the gain function for agent i.[1] The first gain function is based on similarity between agents:

$$g_i^t(S^t) = \frac{1}{m^t} \sum_{k \in s_i^t} \sum_{j \in C_k^t, j \neq i} c_{ij}^t. \tag{7}$$

Here, we use neighborhood similarity to quantify the structural equivalence between users at time t:

$$c_{ij}^t = \begin{cases} w_{ij}^t (1 - d_i^{\text{in},t} d_j^{\text{out},t} / 2m^t) & A_{ij}^t = 1, w_{ij}^t > = 1 \\ w_{ij}^t / n^t & A_{ij}^t = 0, w_{ij}^t > = 1 \\ d_i^{\text{in},t} d_j^{\text{out},t} / 4m^t & A_{ij}^t = 1, w_{ij}^t = 0 \\ -d_i^{\text{in},t} d_j^{\text{out},t} / 4m^t & A_{ij}^t = 0, w_{ij}^t = 0 \end{cases} \tag{8}$$

w_{ij}^t is defined as the number of common neighbors possessed by nodes i and j, where common neighbors are nodes with direct in-edges from both i and j. $d_i^{\text{in},t}$ and $d_i^{\text{out},t}$

[1] We employ the notation for directed graphs, although it is straightforward to generalize to undirected graphs by ignoring the *in/out* superscripts.

are the in- and out-degrees of node i at snapshot t. m^t and n^t are the number of edges and nodes, respectively, and primarily serve as normalization constants. Note that c_{ij}^t assumes its highest value when two nodes have at least one common neighbor and are also directly connected, i.e., $A_{ij}^t = 1$. Hence, agents playing the community formation game benefit from joining communities containing connected nodes with many common neighbors.

The second gain function measures the personalized modularity of the ith agent:

$$g_i^t(S^t) = \frac{1}{2m^t} \sum_{k \in s_i^t} \sum_{j \in C_k^t, j \neq i} \sum_{k' \in s_j^t} (A_{ij}^t \delta(i,j) - \frac{d_i^{\text{in},t} d_j^{\text{out},t}}{2m^t} |k \cap k'|), \tag{9}$$

where $k \in s_i^t$ and $k' \in s_j^t$ refer to the community labels at snapshot t that agent i and j belong to, respectively; $\delta(i,j)$ is an indicator function that is 1 when i and j are members of the same community. More intuitively, this gain function explains how well the ith agent fits the communities it belongs to, compared to a randomly assigned community.

Similar to what happens in real life, we also consider the loss function l_i^t for each agent, which is linear in the number of labels each agent has. This can be used to model the intrinsic communication overhead belonging to multiple communities and prevents the agents from indiscriminately joining every available community. Therefore, we define the following loss function for agent i:

$$l_i^t(S_{-i}^t, s_i^t) = \frac{|s_i^t|}{m^t}. \tag{10}$$

Here $s_i^t = \{1, 2, ..., k\}$ is the set of labels at snapshot t which agent i belongs to.

In our framework, the best response strategy of the agent i with respect to strategies S_{-i}^t of other agents is calculated by:

$$\arg\max_{s_i'^t \subseteq C^t} g_i^t(S_{-i}^t, s_i'^t) - l_i^t(S_{-i}^t, s_i'^t). \tag{11}$$

The strategy profile S^t forms a pure Nash equilibrium of the community formation game if no agent can unilaterally improve its own utility by changing its strategy:

$$\forall i, s_i'^t \neq s_i^t, \quad u_i^t(S_{-i}^t, s_i'^t) \leq u_i^t(S_{-i}^t, s_i^t). \tag{12}$$

A local equilibrium is reached if all agents play their local optimal strategies:

$$\forall i, \quad s_i'^t \in ls(s_i^t), \quad u_i^t(S_{-i}^t, s_i'^t) \leq u_i^t(S_{-i}^t, s_i^t). \tag{13}$$

Here $ls(s_i^t)$ refers to local strategy space of agent i, which is the set of *possible* label sets it can obtain by performing the actions defined earlier.

3.2 Existence of equilibria

The evolving community structure present in dynamic networks is accounted for by propagating strategies from prior snapshots. D-GT uses the same community formation game described in Alvari et al. (2013). Since strategy propagation only changes the initialization conditions, the same proof of the existence of Nash equilibria in the community formation game applies to D-GT as well; we summarize the argument below.

To see when a certain game has Nash equilibria, recall that potential games are a general class of games that permit pure Nash equilibria (Nisan et al. 2007). For any finite game, there exists a potential function Θ defined on the strategy profile S of the agents that maps this profile to some real values. This function must validate the following condition:

$$\forall i, \ \Theta(S^t) - \Theta(S_{-i}^t, s_i'^t) = u_i^t(S_{-i}^t, s_i'^t) - u_i^t(S^t). \tag{14}$$

Equivalently, if the current strategy profile of the game is S^t and the agent i switches from strategy s_i^t to $s_i'^t$, the potential function exactly mirrors the changes in the agent utility. It is not hard to see that a game has at most one potential function. A game that does possess a potential function is called a potential game. Consequently we have the following theorem:

Theorem 1 *Every potential game has at least one pure Nash equilibrium, namely the strategy profile S that minimizes $\Theta(S)$ (Nisan et al. 2007).*

Proof Let Θ be a potential function for this game and let S be a pure strategy profile minimizing $\Theta(S)$. Consider any action performed by player i that results in a new strategy profile S'. By assumption, $\Theta(S') \geq \Theta(S)$ and by the definition of a potential function, $u_i(S') - u_i(S) = \Theta(S) - \Theta(S')$. Thus, the utility of agent i cannot increase from this move and hence S is stable (Chen et al. 2010). □

Now we provide a sufficient condition to prove our community formation game as a potential game and thus address the existence of the Nash equilibrium. First we have the following definition (Chen et al. 2010):

Definition (*Locally linear function*) A set of functions $\{f_i, 1 \leq i \leq n\}$ is locally linear with locality factor ρ if for every strategy profile S, the following condition holds:

$$\forall i, \quad f_i(S_{-i}, s_i') - f_i(S) = \rho(f(S_{-i}) - f(S)). \tag{15}$$

where $f(.) = \sum_{i\in[n]} f_i(.)$. According to Theorem 2, if we show that our gain and loss functions are locally linear, then we can prove the existence of Nash equilibrium in our framework.

Theorem 2 *Let* $\{g_i, 1 \le i \le n\}$ *and* $\{l_i, 1 \le i \le n\}$ *be the sets of gain and loss functions of a community formation game. If these sets are locally linear functions with linear factors* ρ_G *and* ρ_L*, then the community formation game is a potential game (Nisan et al. 2007).*

Proof We define a potential function as $\Theta(S^t) = \rho_l l(S^t) - \rho_g g(S^t)$ and assume that agent i who changes its strategy from s_i^t to $s_i'^t$. Based on the definitions of locally linear functions and the utility functions $u_i^t(.)$, we have $\Theta(S^t) - \Theta(S_{-i}^t, s_i'^t) = u_i^t(S_{-i}^t, s_i'^t) - u_i^t(S^t)$. Therefore, the community formation game is a potential game (Chen et al. 2010). □

3.3 Algorithm

An overview of the D-GT framework is shown in Algorithm 1. For every snapshot of the network, a set of agents, one representing each node in the graph, is created to play the community formation game. The community structure is initialized either with a set of singleton communities or with communities passed from previous snapshots. During game play, an agent is randomly selected (without replacement) from the pool; it selects an action (join, leave, switch, or no op) by calculating the strategy that yields the highest utility. After the agent plays, the community structure is updated.

The game is played until the number of agents changing their play strategy between permutations falls below the threshold, or the maximum iteration is reached. Empirically, we have discovered that $8n$ is a good iteration limit with a threshold of 5 %. Thus, if there are 1000 nodes in a network snapshot, the community formation game is played until fewer than 50 nodes change strategies or to the maximum of 8000 games. Figure 1 shows an example of the convergence in utility vs. iteration. The outer loop of the algorithm requires iterating over M graph snapshots, and the inner loop requires performing a maximum of $8n$ iterations of the community formation game that, in the worst case, requires considering n community choices. Thus, the overall time complexity of D-GT is $O(Mn^2)$.

D-GT maintains a candidate set of multiple community assignments per agent until the last iteration and then selects the assignment with the highest utility function as the final disjoint partition. In this article, we evaluate several different variants of the procedure:

- *D-GTS* (D-GT with Separate runs): This version does not employ any information from previous runs and hence is equivalent to a static community detection procedure.
- *D-GTP* (D-GT passing one Previous Snapshot): Rather than passing strategy profiles from all previous snapshots, we only initialize the community formation game with the structure from a single previous time slice. For each snapshot T^k, we initialized communities and agents to the existing information from T^{k-1}.
- *D-GTG* (D-GT with passing Ground Truth): D-GTG leverages some ground truth information. A select seed group of ground truth communities with predefined size is used to initialize communities for snapshot T^k; however, we do not pass any discovered community structure to the following snapshots (similar to D-GTS). This variation cannot be used unless some of the agents' community membership is known in advance.

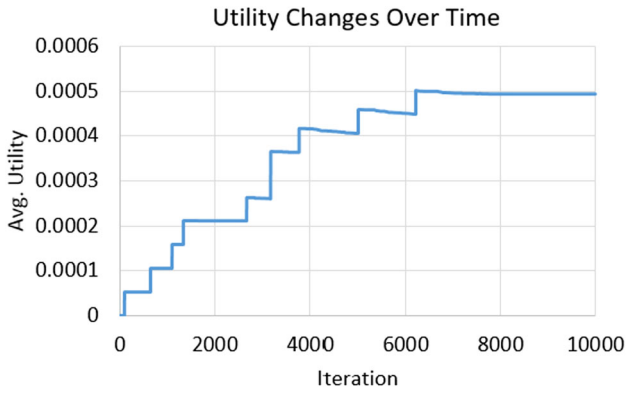

Fig. 1 Change in average utility summed over all nodes vs. iteration for the Travian Trades dataset (one snapshot with 964 nodes). The algorithm converges after 6680 iterations which requires 2.8 s to complete

4 Experimental results

Algorithm 1 D-GT Community Formation Game

Input: Snapshots $\mathbf{T} = \{T^1, T^2, \ldots, T^M\}$
Output: Communities $\mathbf{C} = \{C^1, C^2, \ldots, C^M\}$
for all $T^t \in \mathbf{T}$ **do**
  Initialize $p \leftarrow 0$
  repeat
    Initialize $q \leftarrow 0$
    for all agents in randomized order **do**
      Select best action a using Eqn. 5
      if $a =$ "No operation" **then**
        $q \leftarrow q + 1$
      else
        Update C^t according to a
      end if
    end for
    $p \leftarrow p + 1$
  until $p > 8$ or $q >$ threshold θ
end for

Algorithms were evaluated together on a system with 12G of RAM and Intel CPU 2.53 GHz, and all reported results were averaged over ten repetitions. We compare D-GT with the following community detection baselines:

- *LabelRankT*[2] (Xie et al. 2013). LabelRankT functions according to the generalized LabelRank, in which each node requires only local information during label propagation processing. Several parameters must be set before running the algorithm on the data; we used the best performing values reported in the original paper.
- *iLCD*[3] (Cazabet et al. 2010). iLCD is another well-known community detection approach for dynamic social networks which works by first adding edges and then merging the similar ones. It takes the dynamics of the network into account.
- *OSLOM*[4] (Lancichinetti et al. 2011). The Order Statistics Local Optimization Method (OSLOM) is a versatile community detection algorithm that can handle most types of graph properties including edge directions and weights, overlapping communities, hierarchies and community dynamics. It is based on the local optimization of a fitness function expressing the statistical significance of clusters with respect to random fluctuations.
- *InfoMap*[5] (Rosvall and Bergstrom 2008). InfoMap is a static community detection method that calculates the probability flow of random walks and decomposes the network into modules by compressing a description of the flows. Since this is a static algorithm, we run it separately on each snapshot.
- *Louvain*[6] (Blondel et al. 2008). The Louvain method is a static community detection approach designed to optimize modularity using heuristics. Small communities are found by optimizing modularity locally for all nodes. Then each community is grouped into a single node, and the first step is repeated. We run this algorithm separately on every network snapshot.

4.1 Datasets

To illustrate the strength and effectiveness of our approach, we selected some communication networks from the SNAP[7] graph library as well as two networks (messages and trades) from a well-known multiplayer online game. Statistics for the datasets are provided in Table 3, and description of the datasets is as follows:

AS-Oregon Graph (Leskovec et al. 2005). The dataset contains 9 graphs of autonomous systems (AS) peering information inferred from Oregon route-views between March 31, 2001, and May 26, 2001. These 9 graphs are different snapshots from the data with a minimum of 10,670 and maximum of 11,174 nodes. The number of edges ranges from 21,999 in the snapshot of April 07, 2001, to 23,409 in May 26, 2001. Figure 2 shows the number of edges added and deleted, as well as the number of nodes involved in the changes for the AS-Oregon dataset.

AS-Internet Routers Graph (Leskovec et al. 2005). Similar to AS-Oregon, this is a communication network of who-talks-to-whom from the Border Gateway Protocol logs of routers in the Internet. The dataset contains 733 daily snapshots for 785 days from November 8, 1997, to January 2, 2000. The number of nodes in the largest snapshot is 6477 (with 13,233 edges). Figure 3 illustrates that the structure of the graph can change dramatically at each snapshot.

Enron Email (Sun et al. 2007). The Enron Email network contains e-mail message data from 150 users, mostly senior management of Enron Inc., from January 1999 to July 2002. Each e-mail address is represented by an unique ID in the dataset, and each link corresponds to a message between the sender and the receiver. After a data refinement process, we simulate the network evolution via a series of 12 growing snapshots from January 2000 to December 2000. Enron network changes are shown in Fig. 4.

Travian[8] (Hajibagheri et al. 2015) Travian is a popular browser-based real-time strategy game with more than 5 million users. Players seek to improve their production capacity and construct military units in order to expand their territory through a combination of colonization and conquest. Each game cycle lasts a fixed period (a few months) during which time the players vie to complete construction on one of the Wonders of the World. To do this, they form alliances of up to 60 members under a leader or a leadership team; in this article, these alliances are used as the ground truth for evaluating the community detection procedure.

Travian has an in-game messaging system (IGM) for player communication which was used to create our messages network. Each player can submit a request to trade a specific resource. If another player finds this request interesting, he/she can accept it and the trade will occur; these data were used to build the Trade network. About 70 % of messages are exchanged between users in the same alliance (community), making it more predictive of community structure than the Trades network since only 30 % of edges in this network represent trades occurred

[2] https://sites.google.com/site/communitydetectionslpa.
[3] http://www.cazabetremy.fr/iLCD.html.
[4] http://www.oslom.org/software.htm.
[5] http://www.mapequation.org/code.html.
[6] https://sites.google.com/site/findcommunities.
[7] http://snap.stanford.edu/.
[8] http://ial.eecs.ucf.edu/travian.php.

Table 3 Dataset summary

Data	Oregon	Internet	Enron	Travian (Messages)	Travian (Trades)	Hep-ph
Min # of nodes	10,670	2948	101	1373	964	12,711
Max # of nodes	11,174	6477	137	2100	1336	34,401
Min # of edges	21,999	3386	1432	8511	8080	39,981
Max # of edges	23,409	13,233	5015	19,242	10,221	51,485
# of snapshots	9	733	12	30	30	10

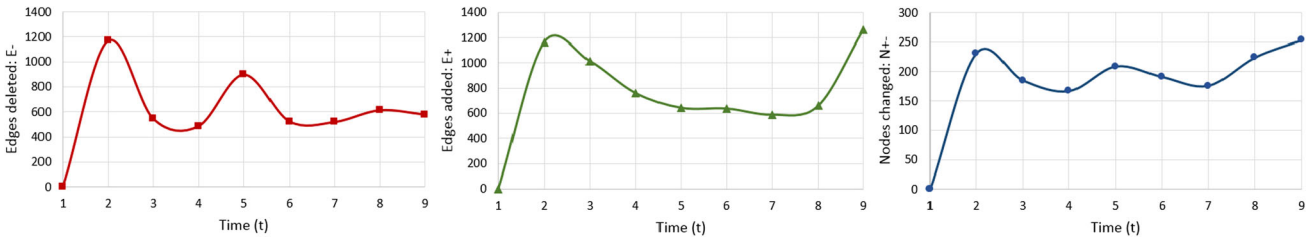

Fig. 2 The structural changes in the AS-Oregon dataset over 9 snapshots including the number of edges deleted (E_-) and added (E_+), as well as the number of nodes involved in changes (N_{+-}). The community detection problem becomes more challenging when there are significant structural changes between snapshots

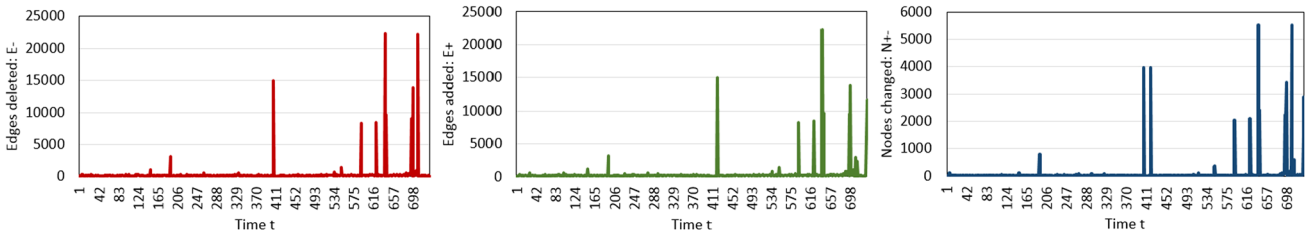

Fig. 3 The structural changes in the AS-Internet dataset over 733 snapshots

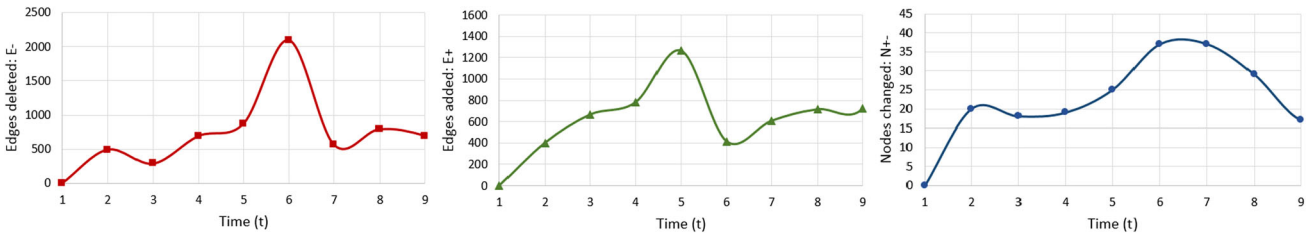

Fig. 4 The structural changes in the Enron dataset over 12 snapshots

between players within the same alliance. The structural changes in both Travian datasets are shown in Figs. 5 and 6.

HEP-PH Citation Graph (Leskovec et al. 2005). The HEPPH (high-energy physics theory) citation graph from the e-print arXiv covers all the citations within a dataset of $n = 29,555$ papers with $e = 352,807$ edges. If a paper i cites paper j, the graph contains a directed edge from i to j. If a paper cites, or is cited by, a paper outside the dataset, the graph does not contain any information about this. These data cover papers in the period from January 1993 to March 2002. There are ten snapshots where each snapshot includes data from twelve months except the last snapshot which only contains edges from the first three months of 2002 (Fig. 7).

4.2 Metrics

We evaluated the performance of all methods using the following metrics.

4.2.1 Normalized mutual information

One way to measure the performance of a community detection algorithm is to determine how similar the partition delivered by the algorithm is to the desired partition,

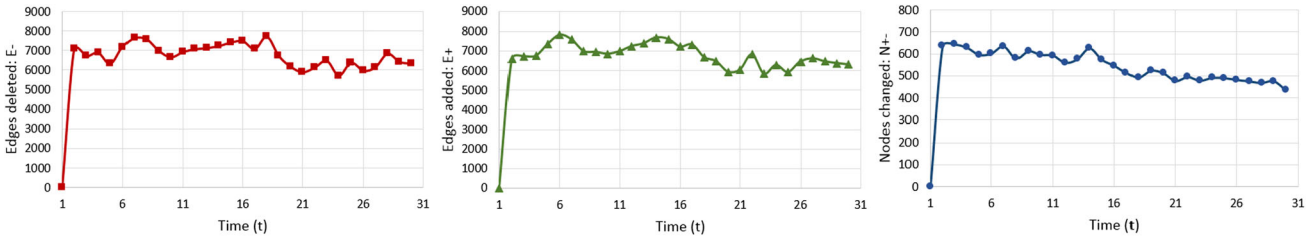

Fig. 5 The structural changes in the Travian Trades dataset over 30 snapshots

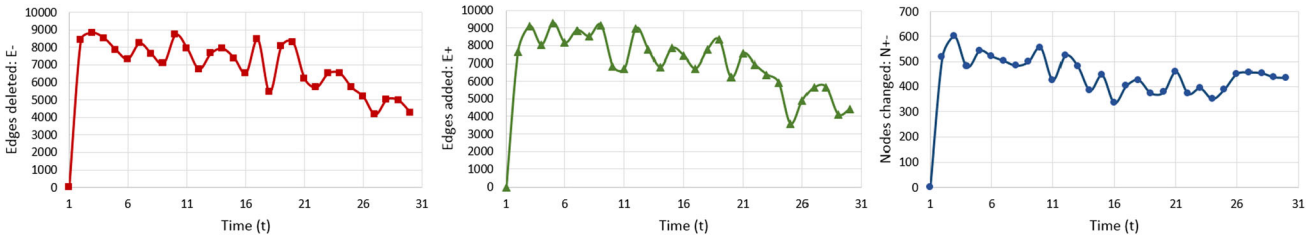

Fig. 6 The structural changes in the Travian messages dataset over 30 snapshots

assuming ground truth information about the community membership exists. Out of several existing measures (Fortunato 2010), we selected the standard version of normalized mutual information (NMI) (Danon et al. 2005), which is computed as follows:

$$\mathbf{I}_{\text{norm}}(X,Y) = \frac{2I(X,Y)}{H(X)+H(Y)}, \tag{16}$$

where $I(X, Y)$ is mutual information between two random variables X and Y (i.e., two community partitions) (MacKay 2003):

$$\mathbf{I}(\mathbf{X},\mathbf{Y}) = \sum_x \sum_y P(x,y)\log\frac{P(x,y)}{P(x)P(y)}, \tag{17}$$

Here $P(x)$ indicates the probability that $X = x$ and joint probability $P(x, y)$ equals to $P(X = x, Y = y)$. $H(X)$ and $H(Y)$ are the entropies of X and Y, respectively. NMI lies in the range [0,1], equaling one when two partitions X and Y are exactly identical and zero when they are totally independent. Code to calculate NMI can be downloaded at: https://sites.google.com/site/andrealancichinetti/software.

4.2.2 Modularity

Modularity measures the difference between the number of intra-community edges for a given community partition versus a random distribution of edges; it is the most popular qualitative measure in detecting communities in social networks. However, it has been shown that modularity has drawbacks and becomes unreliable when networks are too sparse (Fortunato and Barthelemy 2007). It is also useful to examine the number of detected communities in conjunction with modularity to ensure that the algorithm is not being overly aggressive about combining small communities in order to maximize overall network modularity.

Standard modularity $\mathbf{Q}$ is usually defined as follows:

$$\mathbf{Q} = \frac{1}{2m}\sum_{ij}\left[\mathbf{A}_{ij} - \frac{d_i d_j}{2m}\right]\delta_{c_i,c_j} \tag{18}$$

where $\mathbf{A_{ij}}$ is an element of the adjacency matrix, δ_{ij} is the Kronecker delta symbol, and c_i is the label of the community to which vertex i is assigned. However, modularity can be slightly different for directed networks (Leicht and Newman 2008) and can then be reformulated as:

$$\mathbf{Q} = \frac{1}{m}\sum_{ij}\left[\mathbf{A}_{ij} - \frac{d_i^{\text{in}} d_j^{\text{out}}}{m}\right]\delta_{c_i,c_j} \tag{19}$$

where $\mathbf{A_{ij}}$ is defined in the conventional manner to be 1 if there is an edge from i to j and zero otherwise. Here, the probability of the existence of an edge from vertex i to vertex j has the probability $d_i^{\text{in}} d_j^{\text{out}}/m$, where d_i^{in} and d_j^{out} are the in- and out-degrees of the vertices, respectively.

4.3 Evaluation

In this article, we examine four research questions:

1. how does the gain function influence community detection performance?
2. does the initialization procedure affect the performance of dynamic community detection?
3. how does D-GT perform vs. the competitor methods?
4. in cases where the community membership of a small number of the agents is known, can it be used to improve D-GT's performance?

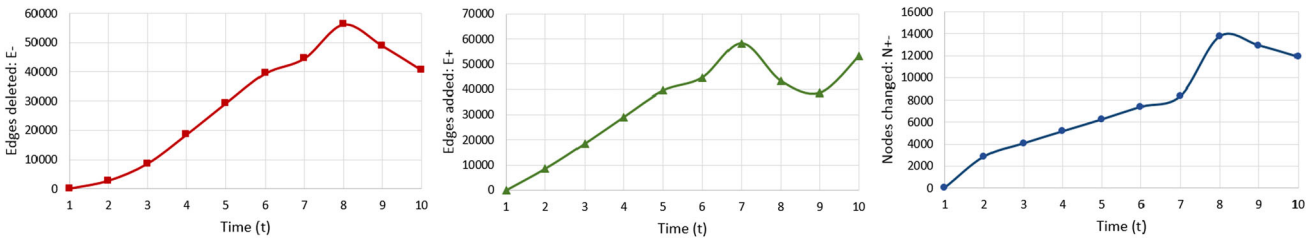

Fig. 7 The structural changes in hep-ph dataset over 10 snapshots

We analyze the performance of the D-GT variants vs. LabelRankT, iLCD, OSLOM, InfoMap and Louvain on the two metrics, normalized mutual information (NMI) and modularity. We hypothesize that D-GT with the modularity gain function will perform well on the modularity evaluation metric, since its stochastic search process essentially optimizes local modularity. Also D-GT's initialization procedure will not necessarily help the modularity optimization process since it can be performed effectively without considering community evolution through time. This suggests that D-GTS is a good candidate for the modularity evaluation metric.

However, in most real-world communities, prior community membership is a good predictor of future membership. Thus, we believe that D-GT is more effective at discovering the real community structure as measured by the normalized mutual information (NMI) evaluation metric. Our previous work (Alvari et al. 2011) has shown that the neighborhood similarity function is a good gain function for recovering the true community structure so we hypothesize that D-GT (similarity) is the best choice for NMI.

Figure 8 shows the average performance of all the D-GT variants with the similarity gain function versus OSLOM, LabelRankT, iLCD, InfoMap and Louvain. Note that it is not possible to calculate the NMI performance on the AS-Internet, AS-Oregon, Enron and hep-ph datasets since we do not have ground truth community structure; for the Travian datasets, we use the alliance membership to calculate the NMI. D-GT (similarity) outperforms all other methods ($p < 0.01$) on this metric.

In D-GT, each node's community membership vector is initialized as the superset of all communities that it has been a member of at any time step. If most of the nodes have remained within the same communities, the correct structure is quickly discovered. In cases where there are cyclic temporal patterns, there is clearly some value in considering earlier community assignments.

As predicted, it outperforms the more myopic D-GTS (that uses no prior information) and D-GTP (one previous snapshot); paired t test comparisons on the two Travian datasets are significant at the $p < 0.01$ level. Figure 9 shows that D-GT (similarity) is more successful than the other D-GT variants at correctly predicting the number of communities, as measured by summed absolute difference between predicted and actual community numbers (lower is better). Note that it is possible to do acceptably well on the NMI metric while still incorrectly estimating the actual number of communities in the dataset. OSLOM also performs well on both metrics (NMI and number of communities).

It is also useful to look at how the number of predicted communities varies between consecutive snapshots. In most cases, the number of communities should remain relatively stable, since the structure of real-world communities rarely changes completely in short period of time. This is definitely true in Travian, where the number of alliances changes relatively slowly. Figure 10 shows the number of predicted communities vs. time on the Travian (Trades) dataset; all of the methods make more consistent predictions over time than LabelRankT.

Table 4 shows the average performance of all the D-GT variants vs. OSLOM, LabelRankT, iLCD, InfoMap and Louvain at optimizing modularity. Bold font shows the absolute best performing algorithm, with italics marking the best performing D-GT variant. All D-GT variants are competitive at optimizing modularity, but none are exceptional. They outperform the other dynamic algorithms (iLCD, LabelRankT) but rarely the static community detection algorithms (Louvain and InfoMap). This is unsurprising since the modularity metric does not intrinsically reward preserving continuity between snapshots. D-GT (similarity) continues to perform well, as does D-GTS (modularity).

In some scenarios, it is plausible that the community membership of a small number of agents is known in advance, and the community detection procedure should leverage this information. For instance, MMOG game alliances often have a leadership council that is publicly known. To handle this problem, we developed a variant (D-GTG: D-GT with passing Ground Truth). Figure 11 shows the performance improvements from increasing the size of the seed groups from 0–20 % of the total number of agents for the Travian (Messages) dataset, and Fig. 12 shows the performance increase for Travian (Trades). Note that extracting community membership information from the

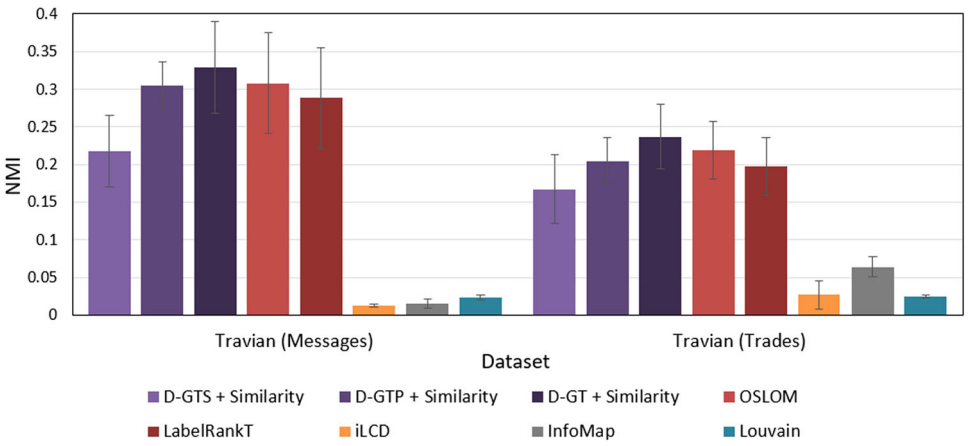

Fig. 8 Normalized mutual information (NMI) evaluation metric on the two Travian datasets with ground truth community membership information; results are averaged over all snapshots. The variants of D-GT are colored in *purple*, LabelRankT (*red*), OSLOM (*Indian red*), iLCD (*yellow*), InfoMap (*gray*) and Louvain (*blue*)

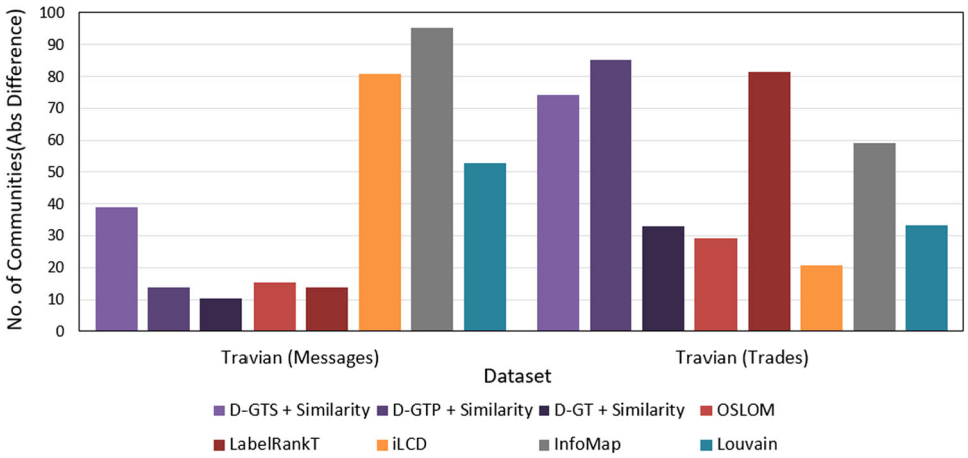

Fig. 9 Absolute difference between the predicted number of communities and the actual number for the two Travian datasets. D-GT (with the similarity gain function) and OSLOM achieve the best performance overall at correctly predicting the number of alliances. (Since this serves a prediction error measurement, lower is better.)

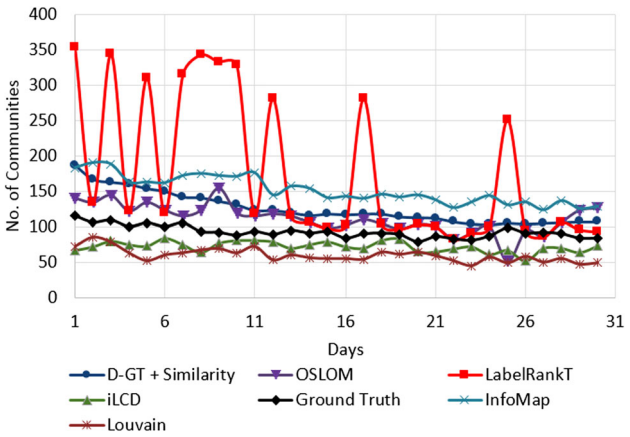

Fig. 10 Number of predicted communities versus time for the Travian (Trades) dataset. LabelRankT's (*red*) predicted number of communities varies drastically between time steps, whereas all other algorithms make more consistent predictions

network structure of Travian (Trades) is a difficult problem because only 30 % of the edges in Travian (Trades) occur between players within the same alliance (community). Also the dataset has a high number of isolated nodes; about 50 % of the nodes do not belong to any alliance.

Table 5 shows the running time of all algorithms on our largest datasets, the AS-Internet dataset (with the highest number of snapshots) and hep-ph (with the highest number of nodes). Overall, the running times provided by all algorithms were quite reasonable, as there are 733 snapshots in the Internet dataset and over 10,000 and 30,000 nodes in Oregon and hep-ph, respectively. InfoMap is the fastest algorithm on two datasets (Internet and hep-ph) and performs almost as well as OSLOM on third one (Oregon). All of the algorithms are sufficiently fast to run on large datasets.

5 Conclusion

This article analyzes the performance of our game-theoretic community detection algorithm, D-GT, on dynamic social networks. These social networks are very common in massively multiplayer online games, such as Travian, where players self-organize into rapidly changing guilds and alliances. We show that D-GT's initialization procedure in combination with the similarity gain function is very effective at recovering the true community structure

Table 4 Modularity evaluation metric on the six datasets; results are averaged over all snapshots

Algorithm/Dataset	Oregon	Internet	Enron	Travian (Messages)	Travian (Trades)	Hep-ph
D-GT + Similarity	0.61 ± 0.02	*0.59 ± 0.02*	**0.50 ± 0.02**	0.45 ± 0.03	*0.43 ± 0.02*	**0.56 ± 0.03**
D-GTS + Similarity	**0.63 ± 0.02**	0.57 ± 0.04	0.48 ± 0.02	0.44 ± 0.02	0.42 ± 0.01	0.55 ± 0.02
D-GTP + Similarity	0.59 ± 0.02	0.58 ± 0.05	0.47 ± 0.03	0.44 ± 0.03	0.41 ± 0.02	0.51 ± 0.04
D-GT + Modularity	0.57 ± 0.03	0.55 ± 0.04	0.47 ± 0.03	0.48 ± 0.04	0.41 ± 0.02	0.55 ± 0.04
D-GTS + Modularity	**0.63 ± 0.02**	0.51 ± 0.02	0.43 ± 0.02	*0.49 ± 0.03*	0.40 ± 0.03	0.54 ± 0.02
D-GTP + Modularity	0.59 ± 0.03	0.48 ± 0.03	0.39 ± 0.02	0.47 ± 0.03	0.42 ± 0.02	0.52 ± 0.04
OSLOM	0.49 ± 0.05	0.52 ± 0.04	0.39 ± 0.03	0.44 ± 0.04	0.32 ± 0.01	0.49 ± 0.01
LabelRankT	0.44 ± 0.01	0.41 ± 0.09	0.34 ± 0.05	0.42 ± 0.04	0.36 ± 0.03	0.43 ± 0.06
iLCD	0.15 ± 0.05	0.11 ± 0.01	0.13 ± 0.05	0.19 ± 0.02	0.09 ± 0.01	0.16 ± 0.07
InfoMap	**0.63 ± 0.04**	**0.61 ± 0.01**	0.49 ± 0.08	0.51 ± 0.02	0.43 ± 0.04	**0.56 ± 0.04**
Louvain	0.59 ± 0.02	0.57 ± 0.04	0.46 ± 0.06	**0.53 ± 0.02**	**0.49 ± 0.02**	0.54 ± 0.07

Bold font shows the absolute best performing algorithm evaluated using the modularity metric, with italics marking the best performing D-GT variant

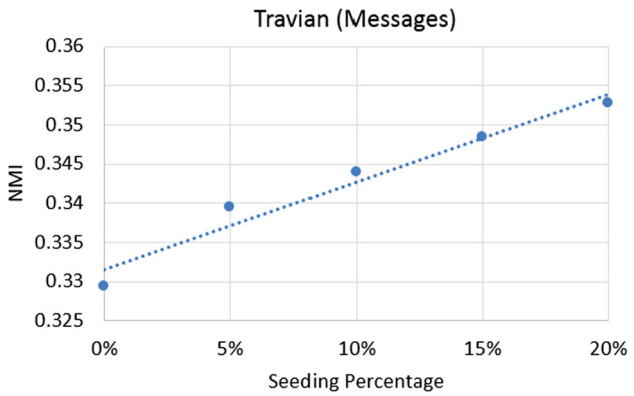

Fig. 11 D-GTG NMI versus seed group size on Travian (Messages)

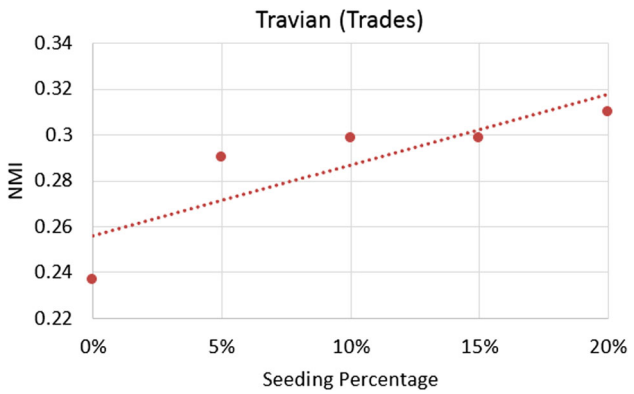

Fig. 12 D-GTG NMI versus seed group size on Travian (Trades)

of the network, both in terms of predicting the number of communities and the players' community membership vectors. It outperforms other dynamic community detection methods including LabelRankT, iLCD and OSLOM. In cases where the communities of a small number of players (e.g., the guild leadership) is known, D-GT can leverage the information to improve the NMI performance. When simply optimizing modularity, considering earlier community membership is less important and static community detection algorithms (InfoMap and Louvain) perform well at this task, but all variants of D-GT offer competitive performance.

Table 5 Running time (sec) of all algorithms on the AS-Internet, AS-Oregon and hep-ph datasets

Algorithm	Internet	Oregon	Hep-ph
D-GT	610	45	1140
LabelRankT	840	90	1250
iLCD	750	80	1000
OSLOM	590	**35**	900
InfoMap	**240**	42	**620**
Louvain	380	60	730

Bold font shows the algorithm with the best (lowest) running time

One nice aspect of D-GT is that it is an easily extensible framework. In future work, we plan to experiment with other utility functions, loss functions and update rules; for instance, Q-learning could be used as the update rule for the agents instead of the community membership game. We also believe that it is feasible to reduce the runtime of D-GT by creating approximate versions of the gain function that can be calculated based on local edge updates.

Acknowledgments We would like to thank Drs. Nitin Agarwal and Rolf T. Wigand at University of Arkansas for providing the Travian dataset.

References

Adjeroh D, Kandaswamy U (2007) Game-theoretic analysis of network community structure. Int J Comput Intell Res 3(4):313–325

Alvari H, Hashemi S, Hamzeh A (2011) Detecting overlapping communities in social networks by game theory and structural equivalence concept. In: Artificial intelligence and computational intelligence, Springer, Berlin, pp 620–630

Alvari H, Hashemi S, Hamzeh A (2013) Discovering overlapping communities in social networks: a novel game-theoretic approach. AI Commun 36(2):161–177

Alvari H, Hajibagheri A, Sukthankar G (2014a) Community detection in dynamic social networks: a game-theoretic approach. In: IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM), pp 101–107

Alvari H, Lakkaraju K, Sukthankar G, Whetzel J (2014) Predicting guild membership in massively multiplayer online games. In: Kennedy W, Agarwal N, Yang S (eds) Social computing, behavioral-cultural modeling and prediction, vol 8393., Lecture notes in computer science Springer, New York, pp 215–222

Beigi G, Jalili M, Alvari H, Sukthankar G (2014) Leveraging community detection for accurate trust prediction. In: ASE international conference on social computing

Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008(10):P10008

Cazabet R, Amblard F, Hanachi C (2010) Detection of overlapping communities in dynamical social networks. In: IEEE international conference on social computing, pp 309–314

Chen W, Liu Z, Sun X, Wang Y (2010) A game-theoretic framework to identify overlapping communities in social networks. Data Min Knowl Discov 21(2):224–240

Danon L, Diaz-Guilera A, Duch J, Arenas A (2005) Comparing community structure identification. J Stat Mech Theory Exp 2005(09):P09008

Folino F, Pizzuti C (2014) An evolutionary multiobjective approach for community discovery in dynamic networks. Knowl Data Eng IEEE Trans 26(8):1838–1852

Fortunato S (2010) Community detection in graphs. Phys Rep 486(3):75–174

Fortunato S, Barthelemy M (2007) Resolution limit in community detection. Proc Natl Acad Sci 104(1):36–41

Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci 99(12):7821–7826

Hajibagheri A, Alvari H, Hamzeh A, Hashemi S (2012) Community detection in social networks using information diffusion. In: IEEE/ACM international conference on advances in social networks analysis and mining, pp 702–703

Hajibagheri A, Hamzeh A, Sukthankar G (2013) Modeling information diffusion and community membership using stochastic optimization. In: IEEE/ACM international conference on advances in social networks analysis and mining, pp 175–182

Hajibagheri A, Lakkaraju K, Sukthankar G, Wigand RT, Agarwal N (2015) Conflict and communication in massively-multiplayer online games. In: Social computing, behavioral-cultural modeling, and prediction, Springer, Berlin, pp 65–74

Hopcroft J, Khan O, Kulis B, Selman B (2004) Tracking evolving communities in large linked networks. Proc Natl Acad Sci 101:5249–5253

Hui P, Yoneki E, Chan SY, Crowcroft J (2007) Distributed community detection in delay tolerant networks. In: Proceedings of ACM/IEEE international workshop on mobility in the evolving internet architecture, p 7

Lancichinetti A, Radicchi F, Ramasco JJ, Fortunato S et al (2011) Finding statistically significant communities in networks. PloS One 6(4):e18961

Leicht EA, Newman ME (2008) Community structure in directed networks. Phys Rev Lett 100(11):118703

Leskovec J, Kleinberg J, Faloutsos C (2005) Graphs over time: densification laws, shrinking diameters and possible explanations. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery in data mining, ACM, pp 177–187

Lin Y-R, Chi Y, Zhu S, Sundaram H, Tseng BL (2008) Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In: Proceeding of the international conference on the world wide web, ACM, pp 685–694

MacKay DJ (2003) Information theory, inference and learning algorithms. Cambridge University Press, Cambridge

Newman ME (2006) Modularity and community structure in networks. Proc Natl Acad Sci 103(23):8577–8582

Newman MEJ (2004) Fast algorithm for detecting community structure in networks. Phys Rev E 69(6):066133

Nguyen NP, Dinh TN, Shen Y, Thai MT (2014) Dynamic social community detection and its applications. PLoS One 9(4):e91431, 04

Nisan N, Roughgarden T, Tardos E, Vazirani VV (eds) (2007) Algorithmic game theory. Cambridge University Press, Cambridge

Palla G, Pollner P, Barabási A-L, Vicsek T (2009) Social group dynamics in networks. In: Adaptive networks, Springer, Berlin, pp 11–38

Raghavan UN, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E 76(3)

Rosvall M, Bergstrom CT (2008) Maps of random walks on complex networks reveal community structure. Proc Natl Acad Sci 105(4):1118–1123

Satuluri V, Parthasarathy S (2009) Scalable graph clustering using stochastic flows: applications to community discovery. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, pp 737–746

Sun J, Faloutsos C, Papadimitriou S, Yu PS (2007) Graphscope: parameter-free mining of large time-evolving graphs. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, pp 687–696

Takaffoli M, Rabbany R, Zaïane OR (2014) Community evolution prediction in dynamic social networks. In: IEEE/ACM international conference on advances in social networks analysis and mining, pp 9–16

Van Dongen S (2000) A cluster algorithm for graphs. Technical report 10, Centrum voor Wiskunde en Informatica

Xie J, Szymanski B (2012) Towards linear time overlapping community detection in social networks. In: Pacific-Asia conference on knowledge discovery and data mining, LNAI, Springer, Berlin

Xie J, Szymanski BK (2011) Community detection using a neighborhood strength driven label propagation algorithm. In: Network science workshop (NSW), IEEE, pp 188–195

Xie J, Chen M, Szymanski BK (2013) LabelrankT: incremental community detection in dynamic networks via label propagation. arXiv preprint arXiv:1305.2006

Zhang Y, Wang J, Wang Y, Zhou L (2009) Parallel community detection on large networks with propinquity dynamics. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, New York, NY, pp 997–1006