Community Detection Based on Graph Dynamical Systems with Asynchronous Runs

Jiamou Liu
School of Computer and Mathematical Sciences
Auckland University of Technology
Auckland, New Zealand
jiamou.liu@aut.ac.nz

Ziheng Wei
School of Computer and Mathematical Sciences
Auckland University of Technology
Auckland, New Zealand
bys7090@aut.ac.nz

***Abstract*—A community in a network is a group of nodes that are densely connected internally but sparsely connected externally. We propose a novel approach for detecting communities in networks based on graph dynamical systems (GDS), which are computation models for networks of interacting entities. We introduce the Propose-Select-Adjust framework – a GDS-based computation model for solving network problems, and demonstrate how this model may be used in community detection. The advantage of this approach is that computation is distributed to each node which asynchronously computes its own solution. This makes the method suitable for decentralised and dynamic networks.**

***Keywords*-Community detection; graph dynamical systems; dynamic networks**

I. Introduction

Graph dynamical systems (GDS) are computation models that capture networks of interacting entities such as biological cells, molecules, and agents, where each entity behaves like a finite-state automaton. Such computation models have been used to simulate complex networks such as traffic networks, spreading of diseases in human interaction networks, and gene annotation in bioinformatics. In this paper, we proposed a GDS-based, distributed approach for *community detection* in social networks. The goal is to develop a method suitable for decentralised and dynamic networks.

Community detection has received a huge amount of attention in recent years due to applications in e-commerce, biology, and political sciences. For example, detecting customer communities for online retailers like Amazon helps to build effective recommendation systems [19]. Detecting scientific communities based on research collaboration networks helps to reveal collaboration patterns [16]. Detecting communities in the protein-protein interaction (PPI) networks help to reveal functional groups that are associated to cancer and metastasis [11]. Other applications appeared in marketing [2], epidemiology [13] and counter-terrorism [23].

Most established methods for community detection are centralised approaches, where the algorithm needs to know in advance the entire network topology [8], [9], [18]. These methods produce highly accurate results. However, such centralized methods prove to be difficult to scale to large decentralised networks where global information is inaccessible. Real networks are expanding at an unprecedented way, e.g. online social networks such as Facebook attract billions of users who interact on a daily base in huge social groups with millions of people. Furthermore, real-life networks are dynamic in the sense that links frequently come and go, and communities evolve continuously. In coping with such dynamic networks, classical approaches such as snapshots analysis may be ineffective, as snapshots may not smoothly track community evolution across different time stamps [3], [17]. All the above pose new challenges and hence call for a novel approach for community detection.

Our contribution. We present GDS as a viable foundation for detecting communities in a decentralised, dynamic network. Here, we view a network as a GDS; each node computes its own community under the assumptions that (a) the node only accesses its local information provided by the neighbourhood and (b) the node functions without synchronisation from a central clock. More specifically we introduce the Propose-Select-Adjust (PSA) framework (Sec. III), which is inspired by the decision making process among a group of people. Each node in the framework repeatedly performs three stages:

(1) Propose: Based on the current information in its neighbourhood, the node makes a proposal to those nodes it hopes to include in its community.
(2) Select: The node selects a received proposal.
(3) Adjust: Based on the selections, the node adjusts its own community, before going back to Propose.

We present the formal definition of a PSA-system, which is targeted not only at community detection, but network problems in general. We then provide an intuitive description of our GDS-based distributed algorithm for community detection (Sec. IV). To evaluate our algorithm, we ran it on several real-life benchmark networks (Sec. V), and measured performance of our algorithm using two quality functions: modularity and performance. Lastly, we apply our algorithm to dynamic networks. As each node in the GDS stores explicitly its own community, the framework naturally reveals community evolution in a changing network.

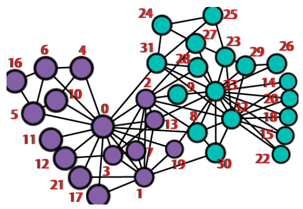

Figure 1. The community structure of an university karate club analyzed by Zachary [25]. The two communities are in two different colours.

II. Preliminaries

Community detection. We regard a network as an undirected graph $G=(V,E)$ where V, E are the sets of nodes and edges, respectively. We assume V is the $\{1,2,\ldots,n\}$ and abuse the notation writing an undirected edge as $(u,v)\in V^2$. For any node $u\in V$, the *neighbourhood* of u, $N(u)$, is the set $\{v\}\cup\{v\in V \mid (u,v)\in E\}$ that contains v and all neighbours of v. For $C\subseteq V$, $N(C)$ denotes $\bigcup_{u\in C} N(u)$.

Girvan and Newman discovered that very often real networks exhibit a special property: nodes in the network can be partitioned into clusters, with high density of edges within each cluster, but low density of edges between these clusters [8]. Any graph with this property is named a *community structure* where each cluster mentioned above is called a *community*. For example, Fig. 1 illustrates the two communities formed in the member network of a karate club in an American university, identified by Zachary [25]. An internal dispute subsequently split the club into two parts, which coincided with Zachary's predication.

Formally, we define a *clustering* of a graph $G=(V,E)$ as a partition $\mathcal{C}=\{C_1,C_2,\ldots,C_k\}$ of V where each *cluster* $C_i\subseteq V$ induces a connected subgraph. A set $C\subseteq V$ is said to be *$\mathcal{C}$-consistent* if it is the union of sets in $\mathcal{C}$.

Let $\mathcal{C}$ be a clustering of a graph $G=(V,E)$, and $C\in\mathcal{C}$ be a cluster. The *intra-cluster density* of C represents the edge connectivity within C; it is defined as

$$\delta_{\mathsf{int}}(C)=\frac{|E\restriction C|}{|C|(|C|-1)/2}, \qquad \text{if } |C|>1$$

and $\delta_{\mathsf{int}}(C)=1$ if $|C|=1$. Let $D_1,D_2,\ldots,D_m$ be all clusters in $\mathcal{C}$ such that $C\cap D_i=\varnothing$ and $D_i\cap N(C)\neq\varnothing$ for all $1\le i\le m$. The *inter-cluster density* of C represents the edge connectivity between C and its neighbouring clusters; it is computed as

$$\delta_{\mathsf{ext}}(C)=\frac{|C|\times|N(C)\setminus C|}{|C|\times(|D_1|+\cdots+|D_m|)}, \quad \text{if } m\ge 1$$

and $\delta_{\mathsf{ext}}(C)=0$ if $m=1$.

As networks vary greatly in the real-world, there has not been a universally accepted formal definition of communities in networks. This paper hence does not aim to provide a formal definition, but instead uses the following intuition: The *community detection problem* aims to compute a clustering $\mathcal{C}$ of a given graph $G=(V,E)$ where each cluster in C has high intra-cluster density but a low inter-cluster density.

Graph Dynamical Systems. GDS are natural models of networks that consist of *cells* linked in undirected graphs. The notion of GDS naturally generalises from *cellular automata*, where cells are normally synchronously updated. Each cell in a GDS functions as a *finite state machine*: at any time, a cell is in one of a finite number of states and may transit to other states based on its current state, and the states of its neighbouring cells (See [14]).

Definition 1: An *graph dynamical system (GDS)* is a tuple $\mathcal{A}=(V,E,Q,(\delta_v)_{v\in V},q_0)$, where

- (V,E) is a graph, and nodes in V are called *cells*;
- Q is a finite set of *states*; $q_0\in Q$ is the *initial state*;
- for each $v\in V$, $\delta_v: Q^{|N(v)|}\to Q$ is the *local transition function* of v.

A *configuration* of $\mathcal{A}$ is a function $c: V\to Q$. A configuration c is called *initial* if every cell in the GDS is at its initial state. Recall that the set of nodes V is $\{1,2,\ldots,n\}$. For every node $v\in V$, denote the cells in $N(v)$ as $\{i_1<i_2<\ldots<i_{k_v}\}$. An *asynchronous run* of $\mathcal{A}$ is a (possibly infinite) sequence of configurations $\rho=c_0,c_1,c_2,c_3,\ldots$ satisfying the following property:

$$\begin{aligned}\forall j\ge 0\forall u\in V:\ & c_{j+1}(u)\neq c_j(u)\\ &\Rightarrow c_{j+1}(u)=\delta_v\left(c_j(i_1),\ldots,c_j(i_{k_v})\right).\end{aligned}\qquad(\star)$$

The run ρ is *initialised* if c_0 is an initial configuration.

The above notion of asynchronous runs is defined in a spirit similar to asynchrnous CA [1], [15]. Such runs differ from both the parallel and sequential GDS as state transitions are nondeterministic in the following sense: at each stage, a node v may either change state according to its local transition, or wait. Hence an GDS has more than one run. In this way, the definition captures the fact that state transitions of cells are asynchronous; there is no central clock synchronising state transitions.

III. The Propose-Select-Adjust Framework

The Propose-Select-Adjust (PSA) framework is used for realising a special type of GDS. It describes a general procedure for implementing a single cell. The framework is inspired by the decision making process among a group of people: Imagine a group of individuals trying to decide on a partitioning of the group, where every member would belong to one and only one subgroup. The constraint is that each individual only sees local information about her own "friendship" – no knowledge is shared among all members. Thus individuals can only make self-centred judgements and decide on the people that she would like to be with. Under this constraint, the following procedures can ensure the group arriving at a collective decision:

1) Propose: Each person individually decides a list of people whom she would like to include in her own community. She then sends an *invitation* to everyone on the list to form a community. Here we implicitly assume that a person makes an invitation to herself.

2) Select: After all invitations are received, the person evaluates the *quality* of each proposed community, selects and accepts the best invitation.

3) Adjust: Once a person accepts an invitation, she then updates her own community according to the accepted proposal. After every individual finishes this step, the whole group would have been divided into a number of communities, and thus a clustering is formed.

The resulting clustering of individuals may not be optimal. In this case, the person should repeat the processes for another iteration. Let $G = (V, E)$ be a graph. In the PSA-

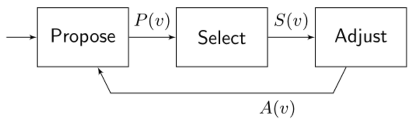

Figure 2. The Propose-Select-Adjust Framework

framework of G, each node $v \in V$ acts as an individual in the above description. Figure 2 illustrates the general stages of a single cell in the PSA-framework. It is important to note that we define the PSA-framework in an abstract sense so that it is not restricted to the community detection problem, but rather, serves as a GDS-based framework for solving network problems in general.

Definition 2: Let $G = (V, E)$ be a finite graph, $v \in V$ and $\Sigma = \{\mathsf{p}, \mathsf{s}, \mathsf{a}\}$. A PSA-*cell* defined on v is a tuple

$$\mathcal{M}_v = (\mathcal{P}, \mathcal{S}, \mathcal{A}, Q, (\delta_{\sigma,v})_{\sigma \in \Sigma}, F_v) \quad \text{where}$$

- $\mathcal{P}, \mathcal{S}, \mathcal{A}$ are finite sets of *proposals*, *selections* and *solutions*, respectively; Q is a finite set of *control states*
- $\delta_{\mathsf{p},v} : (\mathcal{A} \times \mathcal{P} \times Q)^{|N(v)|} \to \mathcal{P} \times Q$ is the *propose function*
- $\delta_{\mathsf{s},v} : (\mathcal{P} \times \mathcal{S} \times Q)^{|N(v)|} \to \mathcal{S} \times Q$ is *select function*
- $\delta_{\mathsf{a},v} : (\mathcal{S} \times \mathcal{A} \times Q)^{|N(v)|} \to \mathcal{A} \times Q$ is *adjust function*
- $F_v : \Sigma \times Q^{|N(v)|} \to Q$ is the *change-step functions*.

A PSA-cell defines a cell in a GDS: the states are the product $\Sigma \times \mathcal{P} \times \mathcal{S} \times \mathcal{A} \times Q$; at any time, a cell v is in a state

$$(\sigma, P(v), S(v), A(v), q) \in \Sigma \times \mathcal{P} \times \mathcal{S} \times \mathcal{A} \times Q$$

where $P(v), S(v), A(v)$ are the current *proposal*, *selection* and *solution* of the cell v, respectively, $q \in Q$ and $\sigma \in \{\mathsf{p}, \mathsf{s}, \mathsf{a}\}$ is called the *step* of v, denoting propose, select and adjust, respectively. The cell v applies the three transition functions $\delta_{\mathsf{p},v}, \delta_{\mathsf{s},v}$ and $\delta_{\mathsf{a},v}$ one-by-one to the appropriate components of its current state:

(i) The cell v starts in step p and applies the propose function $\delta_{\mathsf{p},v}$ to compute a proposal $P(v) \in \mathcal{P}$, according to current solutions and proposals (and control states) of cells in its neighbourhood $N(v)$.

(ii) Then v moves to step s and applies the select function $\delta_{\mathsf{s},v}$ to compute a selection $S(v) \in \mathcal{S}$, according to proposals and selections (and control states) of cells in $N(v)$.

(iii) Then v moves to step a and applies the adjust function $\delta_{\mathsf{a},v}$ to update its candidate solution $A(v)$, according to selections and solutions (and control states) of cells in $N(v)$. The cell then repeats the above cycle.

A PSA-cell v in step σ may change to the next step σ', where $(\sigma, \sigma') \in \{(\mathsf{p}, \mathsf{s}), (\mathsf{s}, \mathsf{a}), (\mathsf{a}, \mathsf{p})\}$, by performing a transition

$$(\sigma, P(v), S(v), A(v), q) \mapsto (\sigma', P(v), S(v), A(v), q')$$

only if $F_v(\sigma, q_1, \ldots, q_k) = q'$ where $q_1, \ldots, q_k$ are the current control states of cells in $N(v)$. In this regard, one obtains a transition function δ_v on $\Sigma \times \mathcal{P} \times \mathcal{S} \times \mathcal{A} \times Q$.

Definition 3: A PSA-*system* is a tuple

$$\Gamma = (V, E, (\mathcal{M}_v)_{v \in V}, \mathcal{P}, \mathcal{S}, \mathcal{A}, Q, p_0, s_0, a_0, q_0)$$

where (V, E) is a finite graph, for each $v \in V$, $\mathcal{M}_v$ is a PSA-cell defined on v with proposal set, selection set, solution set and control states $\mathcal{P}, \mathcal{S}, \mathcal{A}, Q$, respectively, $p_0 \in \mathcal{P}$, $s_0 \in \mathcal{S}$, $a_0 \in \mathcal{A}$ and $q_0 \in Q$.

A PSA-system is essentially a GDS; the initial state of the GDS is $(\mathsf{p}, p_0, s_0, a_0, q_0)$. A *configuration* of the PSA-system is as in a GDS, i.e., a function $c : V \to \Sigma \times \mathcal{P} \times \mathcal{S} \times \mathcal{A} \times Q$. For $v \in V$, we use $A(c, v)$ to denote the solution in the state $c(v)$. A *run* of the PSA-system is a sequence $c_0, c_1, \ldots$ of configurations that satisfies $(\star)$ and for each $v \in \mathcal{V}$ and $\sigma \in \Sigma$, there are infinitely many i's such that $c_i(v)$ is in step σ (this makes sure every cell cycles through propose, select and adjust infinitely often).

A run $\rho = c_0, c_1, \ldots$ is *stablising* if there is $i \geq 0$ such that $\forall j \geq i \forall v \in V : A(c_i, v) = A(c_j, v)$; call the function $\alpha_\rho : v \mapsto A(c_i, v)$ the *limit* of ρ.

Definition 4: A PSA-system is *converging* if all runs are stablising and have the same limit.

We next present a converging PSA-system Γ to solve the community detection problem.

IV. Community Detection with a PSA-System

We first describe the proposal set $\mathcal{P}$ and the solution set $\mathcal{A}$ for Γ. We use the notion of a *tendency tree*, defined with the following intuition: In a social network, an individual u would tend to form a group with another individual v; the node v may either be a neighbour of u or, in case no other node is better suited, the node u itself. In this case we say u *tends to* v. Suppose every node tends to exactly one node. The resulting tendency connections among all nodes will form a directed graph where every node has exactly one outgoing edge. When such a directed graph doesn't contain a cycle (excluding self-loops), it is a forest. A tree in this forest is called a tendency tree.

In the following by a *tree* we mean a tuple (T, f) where T is a set of nodes, and $f : T \to T$ is an *edge function* such that $f(r) = r$ for exactly one node (the root) $r \in T$ and $\exists i > 0 : f^i(u) = r$ for all other nodes $u \neq r$.

Definition 5: A *tendency tree* of a graph (V, E) is a tree (T, f) where $T \subseteq V$ and for any $u \neq v \in T$, $f(u) = v$ implies $(u, v) \in E$.

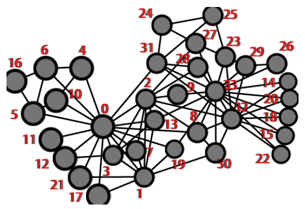

Figure 3. The initial configuration of Zachary's karate club example.

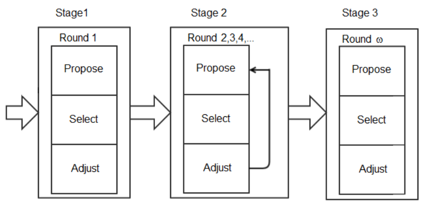

Figure 4. The general computation flow of a cell in the PSA-system Γ.

The proposal set $\mathcal{P}$ and solution set $\mathcal{S}$ of the PSA-system Γ is the set of all tendency trees of (V, E) where the initial proposal and solution are both the empty tree. The selection set $\mathcal{S}$ is $V \cup \{\mathsf{null}\}$ where the initial selection is null.

It remains to describe the control states Q and the transition functions of each node $v \in V$ in Γ. Due to space limitation, we only describe in an intuitive manner the behaviors of cells; from this description the reader will see that the computation can be implemented as a PSA-cell. We also use a running example (Zachary's karate club graph) to illustrate our ideas.

A cell runs a number of *rounds* of the propose-select-adjust cycle, indexed by $1, 2, 3, \ldots$ and ω. The rounds can be divided into three stages: round1, round $i > 1$, and round ω as illustrated in Fig 4; we use the control states in Q to indicate which of these three stages a cell is in. Next we describe informally the runs of a cell v in different stages.

A. Stage 1

Propose. A *k-core* in a graph is an induced subgraph where all nodes have degree at least k . The *core number* of a node v is the largest $\kappa(v)$ such that $N(v)$ contains a $\kappa(v)$-core. For $v \in V$, the *local core* of v is the set $K(v) = \{u \in N(v) \mid |N(u) \cap N(v)| \geq \kappa(v) + 1\}$. In most practical networks, $K(v)$ is a clique-like subgraph; we use $K(v)$ here to approximate the maximal clique that contains v. This is because finding maximal cliques is well-known to be NP-hard, while local cores can be efficiently computed [10]. To make a proposal, the node v sets $P(v)$ to a tendency tree defined on $K(v)$. The proposal for each node in Zachary's karate club is shown in Fig. 5.

Select. The cell v waits for all cells in $N(v)$ to make a proposal. Let $I(v) = \{u \in N(v) \mid v \in K(u)\}$. Note that every cell in $I(v)$ has proposed v to join with itself. To make a selection, v pick the "best" proposal among all $P(u)$ where $u \in I(v)$. For the next definition, recall that a $\mathcal{C}$-consistent set for a clustering $\mathcal{C}$ is a union of clusters in $\mathcal{C}$.

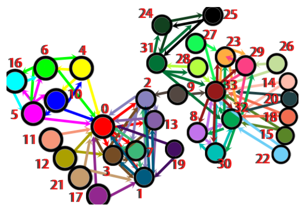

Figure 5. The proposal $P(v)$ made by a node v in Zachary's karate club in Stage 1 consists of nodes in the local core $K(v)$. Every node has a different colour. Self-loops in the tendency trees are omitted. E.g. the tendency tree of 0 contains nodes 0,1,2,3,7.

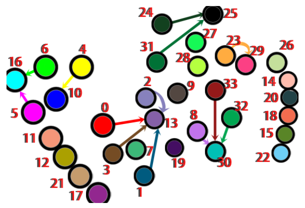

Figure 6. The selection of each node is labeled by an arrow with the same colour as the node. Self-selections are omitted.

Definition 6: Let $\mathcal{C}$ be a clustering of the graph (V, E). We define the *preference relation* $\preceq$ on all $\mathcal{C}$-consistent sets such that $C \prec C'$ if (1) $\delta_{\mathsf{int}}(C) < \delta_{\mathsf{int}}(C')$; or (2) $\delta_{\mathsf{int}}(C) = \delta_{\mathsf{int}}(C')$ and $|C| < |C'|$; or (3) $\delta_{\mathsf{int}}(C) = \delta_{\mathsf{int}}(C')$, $|C| = |C'|$, $\delta_{\mathsf{ext}}(C, \mathcal{C}) > \delta_{\mathsf{ext}}(C', \mathcal{C})$.

Recall that $V = \{1, \ldots, n\}$. The node v sets its selection $S(v)$ as $\min\{u \in I(v) \mid \mu(P(u')) \preceq_{\mathsf{lex}} \mu(P(u))$ for all $u' \in N(v)\}$. The selections of each node in Zachary's karate club is shown in Fig. 6.

Adjust. The cell v waits for all cells in $N(v)$ to make a selection. It then extend its solution $A(v)$ to include all cells who have selected v, i.e., $\{u \in N(v) \mid S(u) = v\}$. After this step all cells would have declared a community $A(v)$; all these communities constitutes a clustering $\mathcal{C}_0$ of the graph. Fig. 7 shows the resulting clustering in Zachary's example.

B. Stage 2

In this stage, each cell aims to optimize the preference of its cluster until no improvement can be achieved. We only present the intuitive ideas:

Propose. Recall that (T, f) is the current tendency forest $A(v)$ of v. The cell v examines all $u \in K(v)$ and compares the current solution $A(v)$ with the union of $A(v)$ and $A(u)$. The new proposal $P(v)$ contains the union of $A(v)$ and $A(u)$ that has the highest preference. This proposal is then propagated to the root of T, who makes a proposal based on proposals of its children and passes it down to all cells in T. In this way, all $v \in T$ will produce the same proposal.

Select. We say that a cluster $C \subseteq \mathcal{C}_0$ *receives* a proposal $P(u) \in \mathcal{P}$, if $P(u)$ contains C. Through propagation of information, the cluster of v examines all proposals it receives and chooses the proposal with the highest preference.

Adjust. For every cluster $C \in \mathcal{C}_0$, we define a *tendency cluster* $\tau(C) \in \mathcal{C}$. There are two cases: 1) every cell $x \in C$

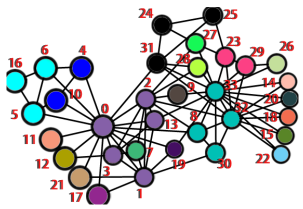
Figure 7. The resulting clustering in Zachary's karate club after the first round. Nodes belong to the same community have the same colour.

selects its parent $f(x)$ in its current tendency tree; in this case the cluster set $\tau(C) = C$. 2) Some cell $x \in C$ selects a node $u \notin C$; in this case the cluster C has decided to join with the community C' that contains u, and we set $\tau(C) = C'$. Note that whenever $\tau(C) \neq C$, $\tau(C)$ always has higher utility vector than C. Hence for every $C \in \mathcal{C}_0$, there is some $j \in \mathbb{N}$ and $D \in \mathcal{C}_{i-1}$ such that $D = \tau^j(C) = \tau^{j+1}(C)$. We call D the *sink* of C.

We define the clustering $\mathcal{C}_1$: $C, C' \in \mathcal{C}_0$ belong to the same cluster in $\mathcal{C}_1$ whenever they have the same sink. Each node v adjusts its solution $A(v)$ to its new cluster in $\mathcal{C}_1$. For this to happen, the tendency tree of $A(v)$ may need to be changed so that it is linked with another cluster in $\mathcal{C}_0$; the resulting tendency tree's root would be the root of its sink.

After all cells in $N(v)$ update their solutions, v then moves back to step p and starts another round. The node v moves on to Stage 3 when no change occurs to $A(v)$ after step a. In Zachary's example, the clustering $\mathcal{C}_0$ and $\mathcal{C}_1$ are the same, so every node directly moves on to Stage 3 after one round in Stage 2.

C. Stage 3 After Stage 2 all cells have computed their clusters which form a clustering $\mathcal{C}_*$. The clusters in $\mathcal{C}_*$ has, in a certain sense, reached *local optimality*: they cannot achieve a higher utility vector if combined with any neighbouring clusters. While they capture the intuitive notion of communities with high intra-cluster density and low inter-cluster density, they are normally too small to reveal any global structure of the network. In real-world networks, communities tends to combine several such optimized clusters (e.g. the two communities in Fig. 1 are formed by combining several clusters in Fig. 7). Hence we use Stage 3 to find such tendency and obtain the final clustering of the network.

Stage 3 is performed similarly as Stage 2. The difference is that a cluster $C \in \mathcal{C}_*$ would send a proposal to every node in its neighbourhood. The clustering $\mathcal{C}_\omega$ is the outcome of the PSA-system and contains all the communities identified in this network. For Zachary's example, we obtained the same clustering as shown in Fig. 1. This shows that the PSA-system correctly detected the communities in Zachary's karate club. Fig. 8 shows tendency of each cluster $C \in \mathcal{C}_*$ in Zackary's example.

V. Experimental Results

We ran PSA-systems on several other real-life networks used as benchmarks for community detection algorithms.

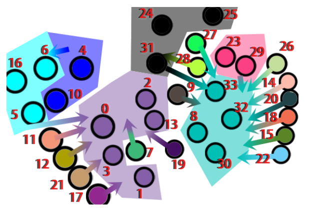
Figure 8. The $\tau(C)$ of each cluster C is shown with an arrow. Self-loops are omitted from the diagram. The two sinks are the cluster of node 0 and the cluster of node 33. Hence the resulting clustering $\mathcal{C}_\omega$ consists of two communities which coincide with Zachary's finding in Fig. 1.

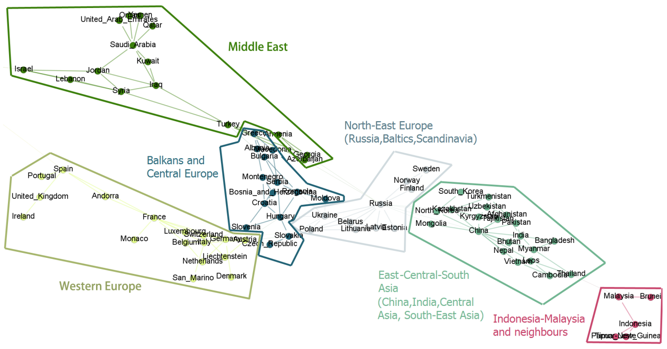
Middle East
North-East Europe (Russia,Baltics,Scandinavia)
Balkans and Central Europe
Western Europe
East-Central-South Asia (China,India,Central Asia, South-East Asia)
Indonesia-Malaysia and neighbours
Figure 9. Running our PSA-system reveals several geographical region, which roughly classified such as South-Eastern Asia (including China), North-Eastern Europe (including Russia), Western Europe, Middle-East, Southern Europe.

Fig. 9 shows the resulting communities in the Euro-Asia country network, where edges indicate land borders between countries. Our algorithm accurately revealed several geographical regions. Fig. 10 shows the resulting communities in the American college football league network, where our algorithm accurately revealed conferences in the league. Fig. 11 shows the resulting communities in the Doubtful sound bottlenose dolphin network, where edges are social

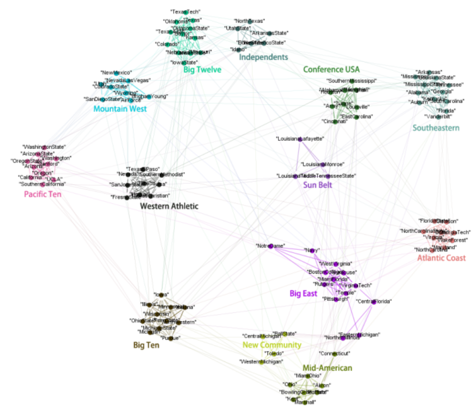

Figure 10. Running our PSA-system reveals several communities in the American college football league, which match the actual conferences to a high precision.

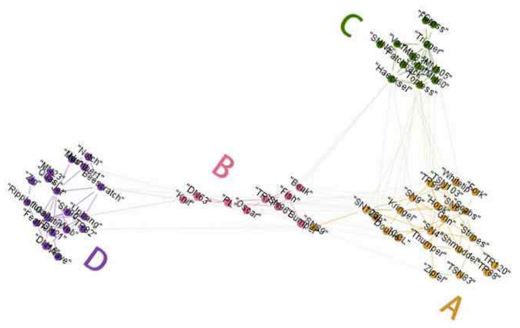

Figure 11. Running our PSA-system reveals four communities in the Doubtful sound bottlenose dolphin social network. Our algorithm revealed four communities which are consistent with the four sub-communities identified by [12].

interactions between dolphins.

We then synthesized networks with 500 nodes based on the relaxed caveman model [22], dividing the nodes into ten group of random sizes. Initially each group forms of a clique. Then we rewire edges in the cliques to another group with probability $p \in [0, 1]$. We ran our algorithm on the synthesized graphs with different values for $p \in [0.05, 0.3]$ and measured the quality of the resulting clusterings using two *quality functions*: For each graph, let $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ be the clustering obtained by the algorithm on graph G.

1) *Performance*: This function counts edges within communities and pairs of nodes that are not linked by edges between different communities (the "correctly interpreted pairs") [7]; it is defined as $\frac{|X \cup Y|}{(|V|-1)|V|}$ where $X = \{(u, v) \in E \mid C(u) = C(v)\}$, $Y = \{(u, v) \notin E \mid C(u) \neq C(v)\}$ and $C(x)$ is the community of x.

2) *Modularity*: This widely used quality function measures the proportion of in-cluster edges taking into account the expected proportion; it is defined as $\mathsf{Mod}(\mathcal{C}) = \sum_{i=1}^{\ell} \left[\frac{|E \upharpoonright C_i|}{|E|} - \frac{d_i^2}{4|E|^2} \right]$ where d_i is the sum of degrees of nodes in C_i [7].

We also ran another well-established modularity-based approximation algorithm for community detection on each graph [5], whose result is compared with outcomes of our algorithm. See Fig. 12.

Dynamic networks. A network is dynamic when it undergoes continuous changes such as adding/deletion of edges. In a PSA-system, a cell continues to loop through the PSA-cycle even when the solution has been stablised. Whenever a change occurs to the neighbourhood of a node v, v would change back to Stage 1 and repropose to its neighbours an updated local core $K(v)$. The change may then propagate to other nodes in v's community, resulting in every node reassigning their solutions. The change is then notified through proposals to the neighbours of the community, who goes back to Stage 2 and re-apply the transition functions to adjust their solutions.

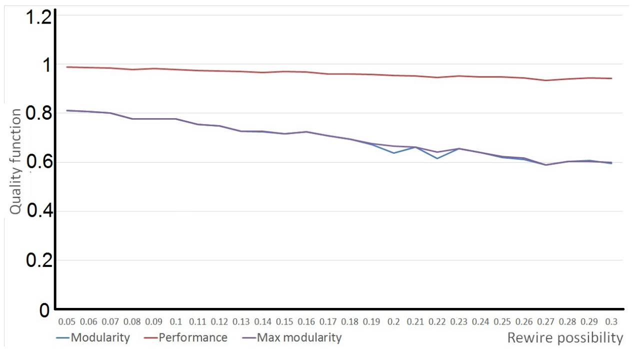

Figure 12. The result shows performance of the identified clusters are close to 1 and modularity almost identical to the maximal modularity [5]. This verifies the validity of our approach.

See the video `http://youtu.be/43cW9CSbg10` that demonstrates changes to communities in a dynamic network. The experiment was carried out on a laptop with Intel Core i7-3630QM CPU 2.4GHz 8.0GB RAM, 128MB JVM Runtime memory. We applied our algorithm on a synthesized 60-node network, which is divided into four clusters with 15 nodes each under the planted 4-model [6]. Firstly we iteratively add random edges with probability 0.9 between nodes in each cluster, and 0.1 between nodes in different clusters. In this phase, a community gradually emerged to enclose all nodes before disintegrate into four clusters. We then add more edges so that all communities merge to one. Then we iteratively remove random edges until an edge appears between nodes from different clusters with probability 0.1. During this phase the community once again disintegrate into four. This experiment demonstrates PSA would serve as a viable framework for detecting community evolution in networks.

VI. Related Works

Conventional methods for community detections include the clique-based agglomerative algorithm in [18], divisive algorithm based on betweenness [8] and the modularity-optimizing algorithm in [9]. All the classical approaches mentioned above are centralised algorithms, which requires a central controller accessing complete information of the network. This means that the algorithms would be difficult to scale to very large, decentralised networks.

Our work differs from these classical works in that our formalism leads to a fully distributed, asynchronous algorithm for community detection. Distributed approaches for community detection have only recently emerged. Rossi et al. proposed an algorithm for clique detection using branch and bound, which distributes multiple copies of a graph among processes [20]. This only finds maximal cliques which do not correspond to communities. Yildiz and Kruege targeted privacy control using communities in Facebook-like online platforms, while we provides a general solution in many different domains [24]. Staudt and Meyerhenke introduced a parallel synchronous computation model by combining machine learning with a synchronous refinement algorithm [21]. Our distributed framework is asynchronous, which means easier to implement and apply to dynamic networks. Lastly we mention Bagnolli et al's recent work using a cellular automata approach for community detection in a similar spirit of our work [4]. The difference is that they use stochastic state transitions which result in a very different mechanism.

VII. Conclusion

We propose a method for community detection that is based on graph dynamical systems that treats each node as a cell. We argue that this formal model offers the following novelties and advantages:

1) Using asynchronous distributed computation enhances efficiency of computation.
2) The algorithm produces results that are adaptable to dynamical updates to the network in run time.

The experimental results demonstrate that this is a viable solution to the community detection problem for decentralised and dynamic networks. As future work we would explore the use of PSA-system in obtaining overlapping communities or detecting social influences in graphs.

References

[1] S. Adachi and J. Lee, Computation by asynchrnously updating cellular automata. Journal of Statistical Physics vol 114, 2004, pp. 261–289.

[2] G. Adomavicius and A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. 17:6, 2005, pp. 734–749.

[3] S. Asur, S. Parthasarathy and D. Ucar, An event-based framework for characterizing the evolutionary behavior of interaction graphs. ACM Trans. Knowl. Disc. from Data vol 3, 2009, pp. 913–921.

[4] F., Bagnoli, E. Massaro and A. Guazzini, Community-Detection Cellular Automata with Local and Long-Range Connectivity, Proc. of ACRI 2012, pp. 204–213, Springer.

[5] V. Blondel, J. Guillaume, R. Lambiotte and E. Lefebvre, Fast unfolding of communities in large networks J. Stat. Mech, P10008, 2008.

[6] A. Condon and R. Karp, Algorithms for Graph Partitioning on the Planted Partition Model, Random Struct. Algor. vol 18 , 2001, pp. 116–140.

[7] S. Fortunato, Community detection in graphs. `CoRR abs/0906.0612`. 2010.

[8] M. Girvan, and M. Newman, Community structure in social and biological networks. Proceedings of the National Academy of Sciences of the USA, 99:12, 2002, pp. 7821–7826.

[9] M. Girvan, and M. Newman, Finding and evaluating community structure in networks. Phys. Rev. E 69, 026113, 2003.

[10] J. Håstad, Clique is hard to approximate within $n^{1-\varepsilon}$, Acta Mathematica vol 182 (1): 1999, pp. 105–142.

[11] P. Jonsson, T. Cavanna, D. Zicha and P. Bates, Cluster analysis of networks generated through homology: automatic identification of important protein communities involved in cancer metastasis. BMC Bioinformatics, 7:2, 2006.

[12] D. Lusseau, and M. Newman, Identifying the role that individual animals play in their social network, Proceedings of the Royal Society of London. Series B: Biological Sciences, 271(6), 2004, pp. 477–481.

[13] L. Meyers, M. Newman, M. Martin and S. Schrag, Applying network theory to epidemics: Control measures for outbreaks of mycoplasma pneumoniae. Emerging Infectious Diseases vol 9(2), 2003, pp.204–210.

[14] H. Mortveit and C. Reidys, An Introduction to Sequential Dynamical Systems, Springer (Universitext), 2007.

[15] C. Nehaniv, Asynchronous automata networks can emulate any synchronous automata network. International Journal of Algebra and Computation, vol 14(5-6), 2004, pp. 719–739.

[16] M. Newman, Coauthorship networks and patterns of scientific collaboration. Proc. Natl. Acad. Sci. USA vol 101, 2004, pp. 5200–5205.

[17] G. Palla, A. Barabási and T. Vicsek, Quantifying social group evolution, Nature, vol 446, 2007, pp. 664–667.

[18] G. Palla, I. Der'enyi, I. Farkas and T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society. Nature, vol 435, 2005, pp. 814–818.

[19] K. Reddy, M. Kitsuregawa, P. Sreekanth and S. Rao, A graph based approach to extract a neighborhood customer community for collaborative filtering, Proc of DNIS 2002. Springer, LNCS 2544, 2002, pp. 188–200.

[20] R. Rossi, D. Gleich and M. Patwary, Parallel maximum clique algorithms with applications to network analysis and storage. `CoRR abs/1302.6256`, 2013.

[21] C. Staudt and H. Meyerhenke, Engineering High-Performance Community Detection Heuristics for Massive Graphs, Proceedings of ICPP 2013, pp. 180–189. IEEE.

[22] D. Watts, Small Worlds, The Dynamics of Networks between Order and Randomness, Princeton University Press, 2003.

[23] C. Weinstein, W. Campbell, B. Delaney and G. O'Leary, Modeling and Detection Techniques for Counter-Terror Social Network Analysis and Intent Recognition, Proc. of IEEE Aerospace conference, 2009, pp.1–16.

[24] H. Yildiz and C. Kruege, Detecting social cliques for automated privacy control in online social networks. Proc. of Pervasive Computing and Communications Workshops, 2012, pp. 353–359. IEEE.

[25] W. Zachary, An information flow model for conflict and fission in small groups, Journal of Anthropoloical Research, vol 33, 1977, pp.452–473.