Dynamic Community Detection via Evolutionary Clustering

By

Fanzhen Liu

A thesis submitted to Macquarie University

Master of Research

Department of Compuuting

October 2019

MACQUARIE
University
SYDNEY·AUSTRALIA

Typeset in LaTeX 2_ε.

Declaration

The work in this thesis entitled **Dynamic Community Detection via Evolutionary Clustering** is primarily based on our publications [1] and [2] (see List of Publications on page ix), and we developed a system based on these published works and submitted a demo (see [3] on page ix) to a conference. I certify that I was involved in all aspects of these works as principal contributor (e.g., conception, data collection, coding, experiments, analysis and writing). I wish to thank the co-authors who assisted me in these works:

Dr. Jia Wu, Prof. Jian Yang and Dr. Chuan Zhou helped to discuss the feasibility of ideas. Dr. Shan Xue and Prof. Quanzheng Sheng gave suggestions on improving the writing skills. Mr. Fuzhi Luo helped to realise visualisation in the system.

I also certify that the thesis is an original piece of research and is written by myself.

(Signed) _____

Fanzhen Liu

21 October 2019

Dedication

This thesis is dedicated to my parents
for their endless love, support
and encouragement.

Acknowledgements

I would like to gratefully thanks the people who have helped, encouraged, and supported me through the whole year spent on this work.

First of all, I would like to express my sincere appreciation to my supervisor Dr. Jia Wu and my associate supervisor Prof. Jian Yang for their kindness, patience, and detailed guidance. Both of them are good listener and advisor.

Then, I would also like to thank Dr. Chuan Zhou from Academy of Mathematics and Systems Science, Chinese Academy of Sciences, and Dr. Shan Xue for their immediate support and suggestions.

Additionally, I would like to acknowledge the Macquarie University for supporting me with the scholarship provided. Thanks extended to the Department of Computing and the Faculty of Science and Engineering.

Last but not the least, I would like to thank my family. My parents have always been with me. With their love, support, and encouragement, I am able to keep going.

List of Publications

[1] **Fanzhen Liu**, Jia Wu, Shan Xue, Chuan Zhou, and Jian Yang, Quanzheng Sheng, *Detecting the Evolving Community Structure in Dynamic Social Networks*, World Wide Web Journal (**ERA A ranking journal**[1]), accepted on 17 July 2019

[2] **Fanzhen Liu**, Jia Wu, Chuan Zhou, and Jian Yang, *Evolutionary Community Detection in Dynamic Social Networks*, Proceedings of the 29th International Joint Conference on Neural Networks (IJCNN'19) (**CORE A ranking conference**[2]), 1-7 (2019)

[3] **Fanzhen Liu**, Fuzhi Luo, Jia Wu, Chuan Zhou, and Jian Yang, *EDPC: Exploring the Detection Process of Communities in Dynamic Networks*, demo submitted to AAAI'20 (**CORE A* ranking conference**) on 20 September 2019

[1] http://portal.core.edu.au/jnl-ranks/
[2] http://portal.core.edu.au/conf-ranks/

Abstract

Dynamic community detection has been an efficient method to track the evolution of communities in dynamic social networks, which can be treated as an optimisation problem. Since evolutionary clustering (EC) was first proposed to optimise temporal data clustering, many EC-based algorithms have been developed to detect evolving community structure in dynamic social networks. However, there are two main drawbacks with existing EC-based algorithms, which limit the efficiency and effectiveness of dynamic community detection: the classic operators cannot efficiently search for community structures and the general network presentation with an integer vector results in a limited search space. For this study, we first review recent literature regarding dynamic community detection using EC-based methods, and then develop two EC-based algorithms to efficiently detect evolving community structure by designing a migration operator in tandem with genetic operators and adopting a genome matrix-based representation for search space expansion. Compared with state-of-the-art baselines, our algorithms perform better in terms of clustering accuracy and temporal smoothness on both synthetic and real-world networks.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

In the real world, we are surrounded by complex systems, which are composed of different entities and complicated interactions between them. For example, the electric power transmission systems supply our modern technologies with energy [1]. Coherent activities of neurons in our brains help us to comprehend the world [2]. Communications via mobile phones help us to get closer to each other [3]. However, it is not easy to understand network behaviours from these complex systems. As such, network science has emerged as a field to meet the demand to study complex systems where entities are represented as nodes and interactions between entities are represented as edges.

Some structural features of complex networks such as small-world property [4, 5] and power-law degree distribution [6, 7], have drawn significant attention. Another

common structural feature, community structure [8], also helps to understand the inherent functions of these systems. Community structure represents the network partition which naturally divides densely connected nodes into groups, and those groups are called communities. For example, in a protein-protein interaction network, proteins with the same function are inclined to be grouped together [9]. In the collaboration network of the Santa Fe Institute where edges represent co-authorships between researchers in at least one article, researchers are clustered into communities in accord with disciplinary lines [8]. In online social network sites like Facebook, Twitter, Instagram, and LinkedIn, users sharing common friends are likely to form communities [10, 11]. Therefore, community detection can help us to infer the function of proteins, analyse collaborations between researchers, and discover potential friends on social media. In addition to these applications, in marketing, community detection helps to design efficient product recommendation systems through identifying customers with similar interests or purchasing habits [12]. Communities are also used to create data structures so that query tasks can be handled in a timely manner [13, 14].

The temporal dimension of networks conveys highly valuable information such as human mobility [15] and collective behaviours [16], which enables us to better understand the reality [17]. Early studies on community detection initially focused on static networks, such as the Zachary's karate club network [18], the dolphin social network [19], and the American college football network [8]. However, for most networks derived from complex systems, structural features change as nodes/edges are added or removed. For example, changing collaborators for different projects in scientific collaboration networks brings about the change of communities [20]. Similarly, communities of users in mobile subscriber networks evolve due to some events like subscriber churn and handset adaption [21]. Discovering useful evolving patterns of community structures can help to understand the expansion and shrinking of research communities and help operators to identify the subscribers who may leave or join. Therefore, dynamic community detection has become a hot topic in network science [22–27].

The challenges of dynamic community detection are threefold. The first challenge is to detect a high-quality community structure at each time step, which requires that nodes

are densely connected within communities and sparsely connected between communities. The second challenge is to ensure that the detected community structure of each network snapshot is close to the real community structure over time. The third challenge is to take into account the temporal evolution of communities. Faced with these challenges, finding a proper way to deal with dynamic community detection has been a daunting task.

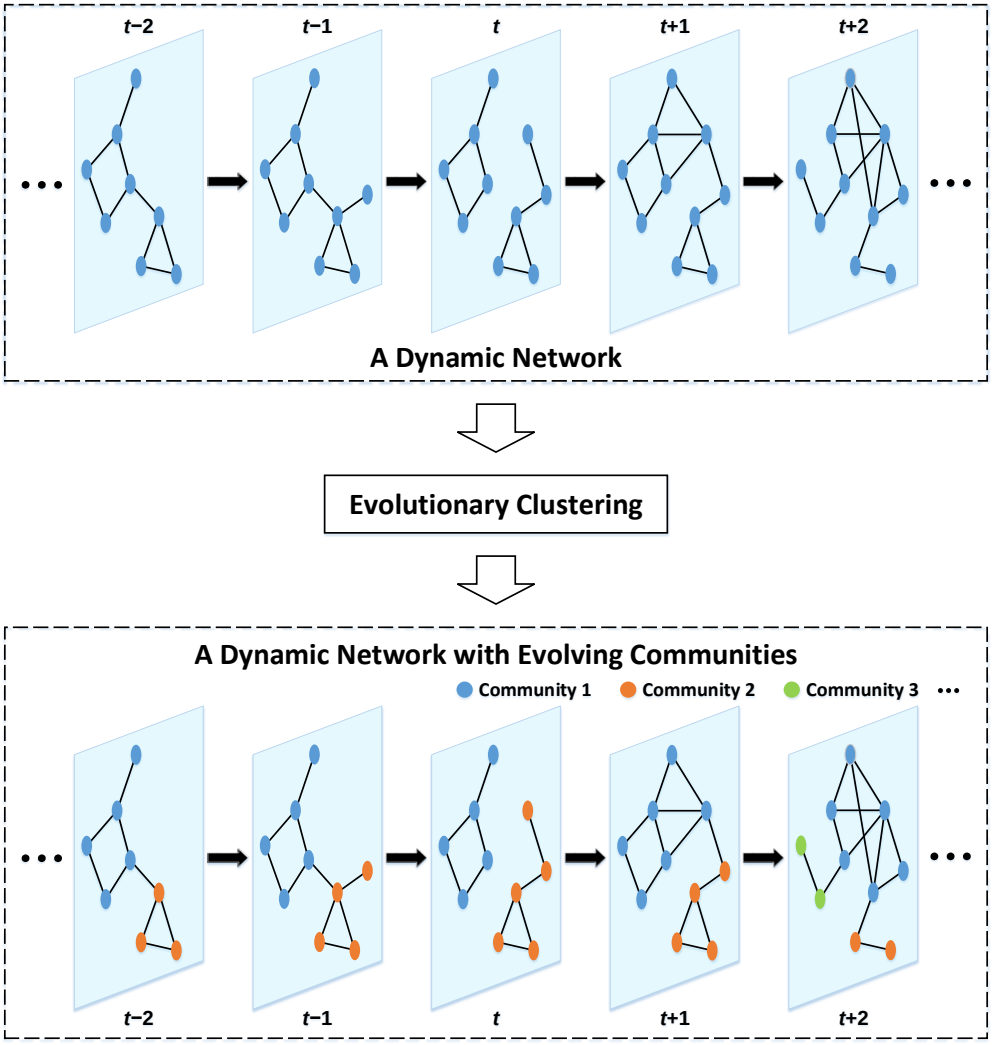


FIGURE 1.1: Dynamic community detection via evolutionary clustering

Evolutionary clustering (EC) [28] was proposed as an efficient framework for clustering temporal data, and it can be used to cluster nodes in dynamic networks into evolving communities (see Figure 1.1). EC emphasises the temporal smoothness of clusters which

suggests no significant change of clustering in a short period. Although EC-based approaches can efficiently deal with dynamic community detection as an optimisation problem, the following two drawbacks limit their effectiveness and efficiency.

- The classic operators such as crossover and mutation, cannot avoid the case that there often exists an inter-community edge between nodes and most of their neighbours. Hence, the overall quality of the new candidate solutions is not satisfactory.
- A popular network presentation based on an integer vector sets every existing edge between nodes within communities to be an intra-community connection. The quality of communities to optimise is evaluated under this setting, which limits the search space.

Thus, we consider developing novel EC-based algorithms to address these two drawbacks and achieve better clustering accuracy and temporal smoothness in dynamic community detection.

1.2 Contributions of the Work

To solve the challenges facing dynamic community detection, we develop new EC-based algorithms to solve dynamic community detection as a multi-objective optimisation problem. The contributions of this thesis include:

- designing a migration operator in tandem with genetic operators to ensure that nodes and most of their neighbours are grouped together, which improves the search efficiency for optimal solutions;
- evaluating the quality of the community structure of each network snapshot directly from genome matrix-based representation to expand the search space;
- applying different methods to generate the initial population;
- conducting a series of experiments on synthetic and real-world datasets to demonstrate the superiority of our proposed methods in terms of clustering accuracy and temporal smoothness.

1.3 Organisation of the Thesis

The rest of this thesis proceeds as follows. Chapter 2 reviews the recent studies on dynamic community detection using EC-based methods. Chapter 3 proposes our EC-based algorithm by integrating a migration operator and applying the genome matrix-based representation. The performance of the proposed algorithm is evaluated and discussed through conducting experiments on synthetic and real-world networks. Based on our proposed algorithm in Chapter 3, Chapter 4 proposes an improved algorithm by adjusting the population generation strategy and employing a more efficient crossover operator. Finally, Chapter 5 presents a conclusion to this thesis and outlines potential directions for future work.

2

Literature Review

Dynamic community detection has been one of efficient methods to study dynamic patterns of networks. The literature on dynamic community detection is reviewed in this chapter. Section 2.1 defines the community structure. Section 2.2 first elaborates on the temporal smoothness framework to uncover the evolving community structure, then introduce widely used metrics for dynamic community detection, and finally review recent work on EC for dynamic community detection.

2.1 What is Community Structure?

There is no universal definition of community [24]. In principle, communities are clusters within which nodes are densely connected and between which nodes are sparsely connected. As a structural feature of complex networks, community structure is the partition

of networks into communities.

According to the internal and external degrees of nodes within communities, communities can be classified into two categories: strong communities and weak communities [29, 30]. The internal degree of a node is the number of edges that connect the node to other nodes within the same community, and the external degree of a node is the number of edges that connect the node to nodes belonging to other communities. A strong community is a subgraph with the internal degree exceeding the external degree for each node. A weak community is a subgraph with the total internal degree of nodes exceeding the total external degree.

Communities can also be divided into two categories, depending on the belonging of nodes. The first category is disjoint communities where each node is allowed to join only one community. For example, communities in football networks are disjoint for a football team is allowed to join only one league [8]. The second category is overlapping communities where a node can belong to multiple communities. For example, communities in social media networks are overlapping because a user can join different groups in social media [31]. Additionally, communities are hierarchical if communities can be further divided into sub-communities [32–34]. This thesis focuses on disjoint community detection in dynamic networks.

The community structures in dynamic networks can be modelled as follows. A dynamic network G is a sequence of network snapshots which can be presented as $G = \{G_1, G_2, ..., G_T\}$. Each snapshot $G_t = (V_t, E_t)$ represents a temporal structure of G at time step t, in which V_t and E_t are a set of nodes and a set of edges at time step t, respectively. Given a network partition $C_t = \{C_t^1, C_t^2, ..., C_t^k\}$, it denotes a community structure of G at time step t, in which each pair of communities from k communities in C_t, $C_t^i \neq \emptyset$ and $C_t^j \neq \emptyset$, do not share any nodes, i.e., $C_t^i \cap C_t^j = \emptyset$.

2.2 Dynamic Community Detection

In this section, we first introduce a popular framework, temporal smoothness, for dynamic community detection. Then, we review recent studies on EC for dynamic community

detection. Finally, we summarise the drawbacks of existing EC-based algorithms.

2.2.1 Temporal Smoothness

The structure of dynamic networks changes over time as nodes join or disappear and edges form or fade, which causes the evolution of communities. To reveal evolving patterns of communities in dynamic networks, a framework known as temporal smoothness [28] has been proposed, which emphasises no great changes of clustering within a short period. Through a linear combination of two conflicting criteria, *snapshot cost* (*SC*) and *temporal cost* (*TC*) [35], a cost function is defined as

$$Cost = \alpha \cdot SC + (1-\alpha) \cdot TC, \tag{2.1}$$

in which *SC* evaluates the quality of a community structure at the current time step and *TC* measures the similarity between community structures at the current and previous time steps, i.e., C_t and C_{t-1}. Additionally, $\alpha \in [0,1]$ is a parameter to adjust the degree of *SC* and *TC*.

2.2.2 Metrics for Dynamic Community Detection

The results of dynamic community detection can be evaluated in different ways, depending on whether the ground truth evolving community structure is known.

Without knowing the ground truth community structure, the quality of the detected community structure is usually measured by quality functions. The modularity [36], usually denoted as Q, is the most widely used quality function that quantify the quality of a community structure. Given a network partition with k communities and m edges, modularity is defined as

$$Q = \sum_{s=1}^{k} \left[\frac{l_s}{m} - \left(\frac{d_s}{2m} \right)^2 \right], \tag{2.2}$$

in which l_s and d_s respectively denote the number of the edges and the sum of the degrees of nodes in the s-th community. Generally, a network partition with a high value of Q

corresponds to a high-quality community structure. Other quality functions not involved in our work are summarized in [37, 38].

The normalised mutual information (NMI) [39] is used to measure the similarity between community structures. Given two partitions of a network, A and B, an element M_{ij} of a confusion matrix M counts the number of nodes shared by the i-th community of partition A and the j-th community of partition B. NMI can be calculated by

$$\mathrm{NMI} = \frac{-2\sum_{i=1}^{k_A}\sum_{j=1}^{k_B} M_{ij}\log\left(M_{ij}\cdot|V|/M_{i.}\cdot M_{.j}\right)}{\sum_{i=1}^{k_A} M_{i.}\log\left(M_{i.}/|V|\right)+\sum_{j=1}^{k_B} M_{.j}\log\left(M_{.j}\cdot|V|\right)}, \tag{2.3}$$

in which k_A and k_B denote the number of communities in partition A and in partition B; $M_{i.}$ and $M_{.j}$ record the sum of the elements in the i-th row of M and the j-th column of M, respectively; $|V|$ counts the number of nodes in a network. The NMI value varies from 0 to 1 and a higher NMI value means a higher similarity between A and B. For dynamic community detection, NMI can be used to calculate the similarity of community structure between two successive time steps.

2.2.3 Evolutionary Clustering for Dynamic Community Detection

Chakrabarti et al. [40] proposed the concept of EC to cluster data over time. They pointed out that the clustering at the current time step should be of a high quality and more consistent with the clustering at the previous time step. On the basis of this idea, they introduced a temporal smoothness framework (see Section 2.2.1), which requires high quality of clustering with no dramatic changes within a short time. Therefore, to discover the evolving community structure in dynamic networks, EC-based methods take into account snapshot quality to measure the clustering quality, and the similarity of community structure between current and previous time steps to measure the temporal smoothness.

Inspired by the idea from [40], several studies [35, 41–43] developed techniques to capture the evolution of communities in dynamic networks. Chi et al. [41] designed evolutionary spectral clustering methods to capture the evolution of communities. They defined a general cost function which consists of *SC* and *TC*. Lin et al. [35] designed an

EC-based algorithm based on the temporal smoothness framework, called FacetNet, to study evolving patterns of dynamic communities. Adopting a modified version of the cost function in [41], FacetNet treated dynamic community detection as an optimisation problem. However, the main limitations of FacetNet includes its requirement to be initialised with the number of communities and its slow convergence. Moreover, Gong et al. [42] proposed a multi-objective immune algorithm with genetic operators and local search to improve the effectiveness and efficiency. Based on one of multi-objective evolutionary algorithms (MOEAs), the non-dominated sorting genetic algorithm known as NSGA-II [44], Folino and Pizzuti put forward a dynamic multi-objective genetic algorithm, known as DYNMOGA, to deal with community detection in dynamic networks as a multi-objective optimisation problem [43]. To lower the computational complexity of MOEAs, a decomposition framework for multi-objective evolutionary algorithms called MOEA/D was developed, which decomposes a multi-objective optimisation problem into several scalar optimisation subproblems and optimises multiple objectives at the same time. Recently, Ma et al. [45] and Gao et al. [46] have applied the frameworks of MOEA/D and temporal smoothness to discover evolving communities in dynamic social networks.

Representation

According to a survey of EC [47], a chromosome representation based on an integer vector, has been adopted for network representation in community detection [48]. To overcome the problem that different chromosomes could correspond to the same network partition as mentioned in [47], a renumbering procedure suggested in [49] was applied to the chromosome representation. The details of this representation are provided here. Given that an individual has only one chromosome, a chromosome is represented as a position vector which implies a network partition. This presentation displays a community ID for each node, and it has been used in recent particle swarm optimisation for dynamic community detection [46] (see Figure 2.1). However, the chromosome representation does not defines the specific partition attribute of each edge, which limits the search space when optimizing a quality function.

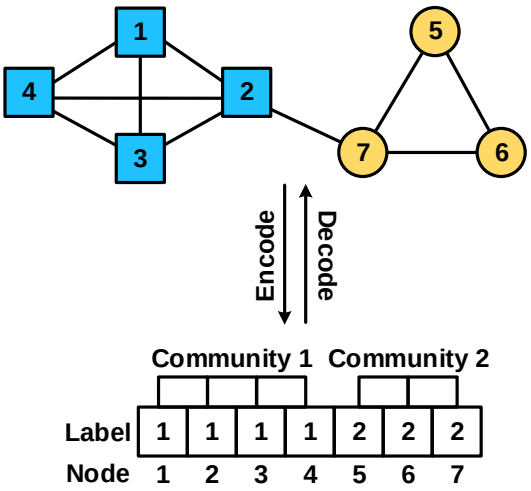


FIGURE 2.1: A chromosome representation based on a integer vector

Also, a network structure can be represented by an adjacent matrix [50]. Each element in the matrix defines the partition attribute of each edge in a network. Because this representation is applied in our algorithms, the details is provided in Section 3.2.

Operators

EC-based algorithms generally employ two genetic operators, i.e., crossover and mutation operators, to generate new candidate solutions during the search process. Under a certain network representation, the efficiency of the two genetic operators depends on not only their design but also corresponding parameters. Generally, the crossover rate and the mutation rate respectively determine the efficiency of crossover and mutation. The selection operator is not always mentioned, but most EC-based algorithms select high-quality solutions to form new populations, which can be regarded as the execution of selection operators.

3

ECD: Evolutionary Community Detection

In this chapter, we propose a novel EC-based algorithm called ECD for dynamic community detection. Section 3.1 provides an overview of ECD, followed by an explanation of the core components of ECD. Section 3.2 elaborates on a genome matrix-based representation. Section 3.3 details the process of population generation based on a bio-inspired model. Section 3.4 presents ECD's operators to generate new candidate solutions. Section 3.5 formulates dynamic community detection as a multi-objective optimisation problem by optimising two objectives introduced in Section 2.2.2. Section 3.6 elaborates on ECD's parameter initialisation that is prepared for decomposition-based multi-objective optimisation. Section 3.7 designs population updating rules for ECD to better search for the optimal solution. Finally, Section 3.8 reports the experiments and evaluates the performance of ECD by analysing and discussing experimental results.

3.1 Framework of ECD

ECD is developed based on the MOEA/D framework to solve dynamic community detection as a multi-objective optimisation problem, and it adopts the *Physarum*-based network model (PNM) to generate the initial population and employs operators to search for the optimal solution under temporal smoothness. An overview of ECD is provided as shown in Figure 3.1.

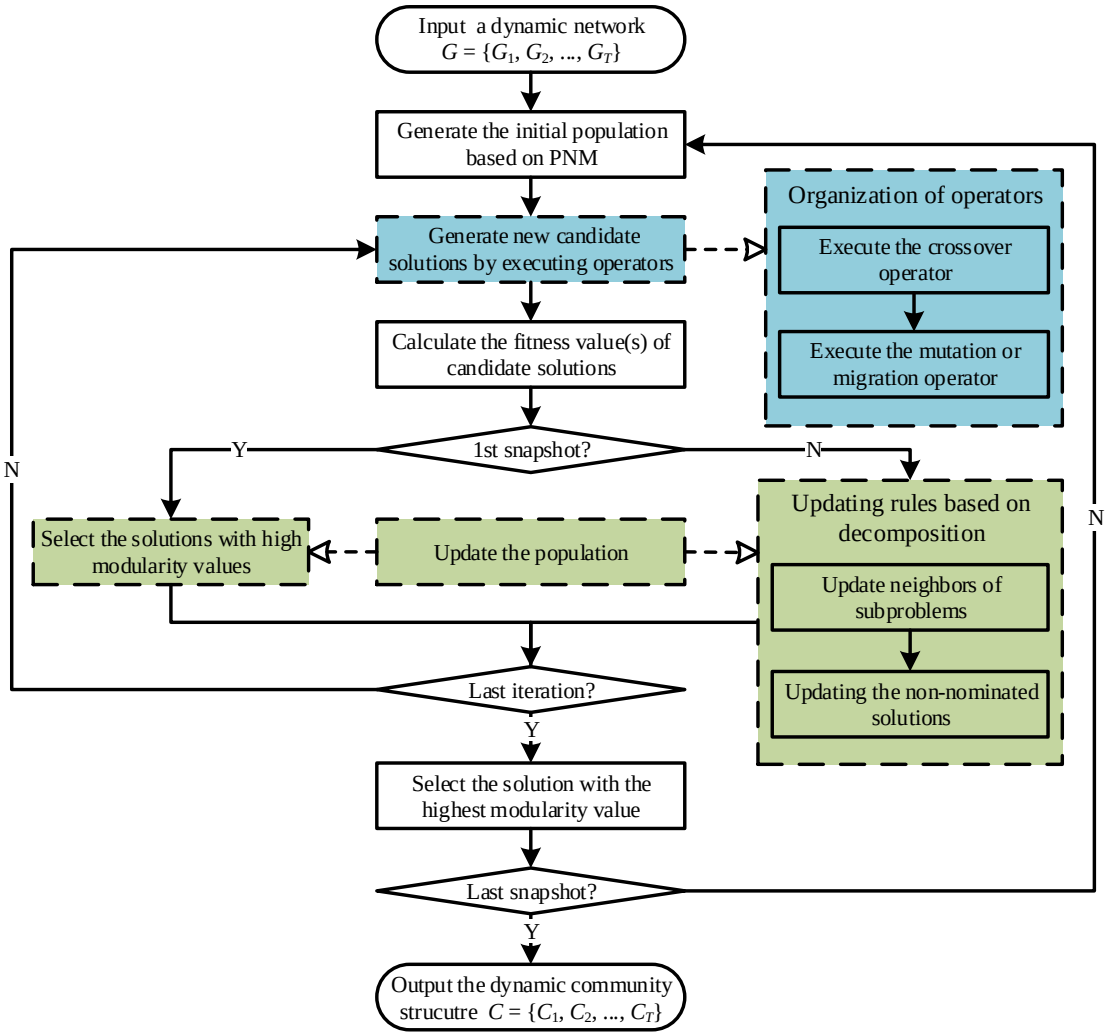


FIGURE 3.1: Framework of ECD

3.2 Genome Matrix-based Representation

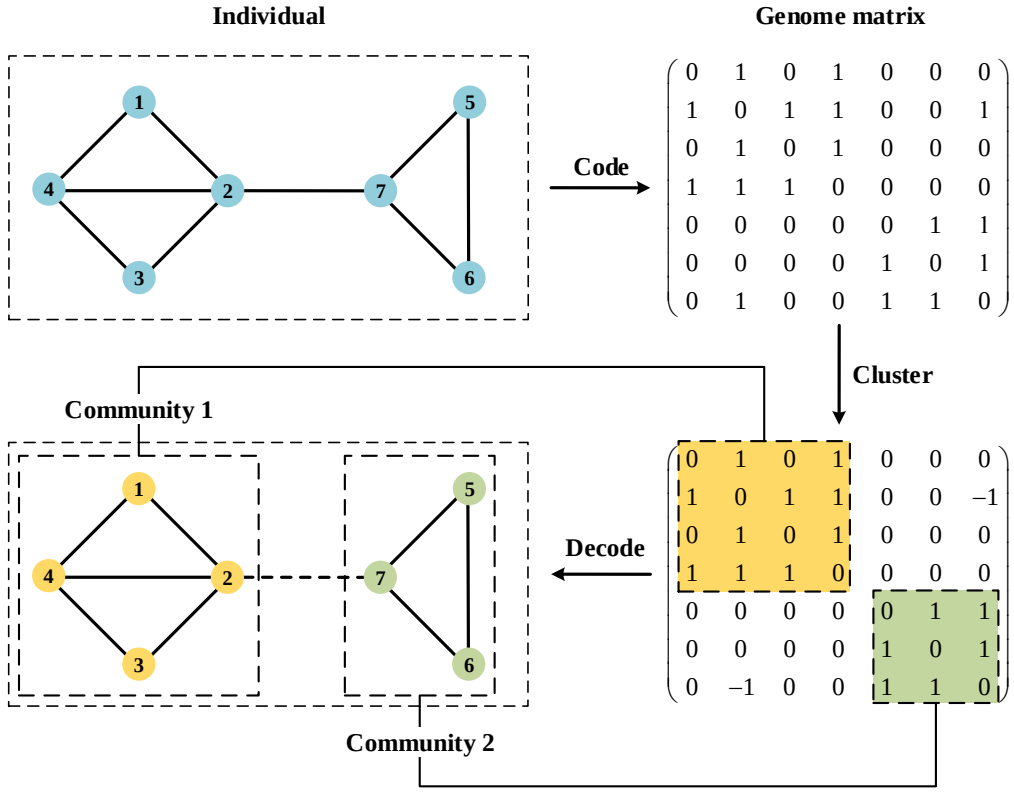


FIGURE 3.2: An illustration of the coding scheme for genome representation. A network is coded as a genome matrix derived from an adjacent matrix whose element is 0 or 1, in which 1 stands for an edge existing between two corresponding nodes and 0 denotes that there is no edge. Following the rule that connected nodes belong to the same community, a community structure emerges through the decoding process after a series of operations on the genome matrix

Different from most EC-based algorithms using a chromosome representation, ECD adopts a genome representation in a coding scheme (see Figure 3.2). An individual representing a candidate solution is coded as an n-order matrix $genome_{n\times n}$, in which $n = |V|$. An element of $genome_{n\times n}$, $genome_{i,j}$, demonstrates the existence of an edge $e_{i,j}$ between v_i and v_j and determines whether $e_{i,j}$ is an inter-community or intra-community connection. In detail, $genome_{i,j} = 1$ suggests that v_i and v_j are intra-connected within a community; $genome_{i,j} =$ -1 means that v_i and v_j are inter-connected; $genome_{i,j} = 0$ shows the nonexistence of an edge connecting them. The coding scheme makes it convenient for

operations like crossover and mutation and it identifies the partition attribute (i.e., inter-community or intra-community) of each edge in a network. Following the decoding rule that the maximum connected components in a network are communities, the community structure of a network can be obtained.

3.3 Population Generation Based on PNM

PNM is a modified version of the *Physarum*-based mathematical model (PM) and its capacity of distinguishing inter-community edges from intra-community edges has been proved [51]. Several community detection algorithms [52–54] have showed an improved performance applying PNM rather than a random way to generate the initial population. Therefore, ECD generates the initial population based on PNM for a better performance. The key mechanism of PNM is a feedback system between the fluxes and conductivities of tubes based on Poiseuille's law. Conductivities of tubes with lager fluxes are reinforced and those with smaller fluxes degenerate. The detailed process of PNM is described below.

First, $D_{i,j}$, $L_{i,j}$ and p_i denotes the conductivity, length of an edge $e_{i,j}$, and the pressure of a node v_i, respectively. The flux of $e_{i,j}$, represented by $Q_{i,j}$, can be calculated as

$$Q_{i,j}^t = \frac{D_{i,j}^{t-1}}{L_{i,j}} \left| p_i^t - p_j^t \right|. \tag{3.1}$$

$$\sum_i \frac{D_{i,j}^{t-1}}{L_{i,j}} |{p^t}_i - {p^t}_j| = \begin{cases} I_0, & \textit{if } v_j \textit{ is an inlet} \\ \dfrac{-I_0}{|V|-1}, & \textit{others} \end{cases}. \tag{3.2}$$

Second, in contrast with only one pair of inlet and outlet being set in a *Physarum* network in PM, PNM selects one node as an inlet and sets the rest as outlets. Under the premise that the sum of the flux in a *Physarum* network is I_0, the flux at an inlet or outlet can be described in Eq. 3.2 based on the law of conservation. Therefore, a set of equations based on Eq. 3.2 are constructed for each node being selected as an inlet once in each iteration. The pressure of each node and the flux in each pipe connecting a pair of nodes at time t are obtained by solving the equation set.

Then, $D^t_{i,j}$ is updated with its value from the previous iteration and $Q^t_{i,j}$:

$$D^t{}_{i,j} = \frac{Q^t_{i,j} + D^{t-1}_{i,j}}{k}. \tag{3.3}$$

At the end of an iteration, the global conductivity D^t is updated by averaging the conductivities:

$$D^t = \frac{1}{|V|}\sum_{i=1}^{|V|} D^t(i). \tag{3.4}$$

in which $D^t(i)$ represents the local conductivity matrix when node v_i is set as the inlet.

Once the final iteration is completed, R edges with the higher conductivities are set as inter-community edges, noting that R is usually a constant between 10 per cent and 20 per cent.

3.4 Operators

3.4.1 Migration Operator

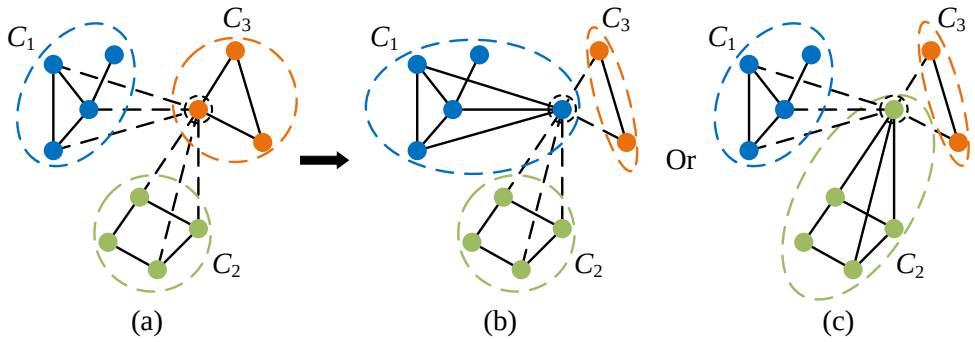


FIGURE 3.3: An illustration of migration using the example of one individual and its possible connections within three communities. The solid lines connecting the nodes represent the intra-community edges and the dotted lines represent the inter-community edges. Nodes of the same colour belong to the same community. The single node circled by dotted line in (a) is a weakly-neighbourhood node in community C_3, because the other two communities C_1 and C_2 hold more of its neighbours, i.e., 3 vs 2. To become a strongly-neighbourhood node, the node would be guided to migrate into either C_1 or C_2. (b) and (c) display two possible results through migration operated on the individual shown in (a)

A migration operator is designed to improve the quality of solutions for community detection. The migration operator emphasises the inter- and intra-connection relationships

between nodes [55]. Figure 3.3 provides an illustration of the migration operator on an individual with two communities.

The migration operator works on the neighbourhood where a neighbourhood consists of a rooted node and its directly connected nodes called neighbours. According to the inter- and intra-connection relationships between nodes and their neighbours, each node v_i can fall into one of three groups as defined in [55]. Specifically, under the condition that v_i belongs to the same community with most of its neighbours, v_i is a strongly-neighbourhood node if there are no other communities holding most of v_i's neighbours; v_i is a neutrally-neighbourhood node if there is at least one other community holding the same number of neighbours as the community to which v_i belongs. In the case that most of the neighbours are clustered into a community different from v_i's, v_i is a weakly-neighbourhood node.

Except for the community to which v_i belongs, the candidate community of the node v_i is a community holding the most of v_i's neighbours. For a neutrally-neighbourhood node, a parameter p_{mi} is introduced to determine whether the node will be migrated into a candidate community that holds the same number of neighbours as the original community. For a weakly-neighbourhood node, it will be directly migrated into a candidate community. If there are multiple candidate communities from which to choose, the weakly-neighbourhood node will be randomly migrated into one of them. For a strongly-neighbourhood node, it will stay in its original community. As a node v_i is migrated, a random number (at least one) of edges between v_i and its neighbours in the new community are changed into intra-community connections, whereas edges connecting v_i and nodes in the original community are all converted to inter-community connections. In practice, with a genome matrix-based representation (see Section 3.2), a node is migrated by changing the values of the corresponding elements in a genome matrix.

3.4.2 Genetic Operators

In ECD, genetic operators cooperate with the migration operator to generate a new population, which include:

- **Crossover:** The crossover operator is executed in a random and uniform way. To form a new population, individuals are randomly selected in pairs to generate new individuals. For a pair of selected individuals x_1 and x_2, a varying number of nodes are randomly selected to be crossover points. Then we exchange the partition attributes (i.e., inter-community or intra-community) of the edges connecting any of the crossover points between the selected individuals. Following the exchange, a new individual x_3 is generated from x_1. The details on crossover operation on the genome matrix-based presentation are presented in Figure 3.4.

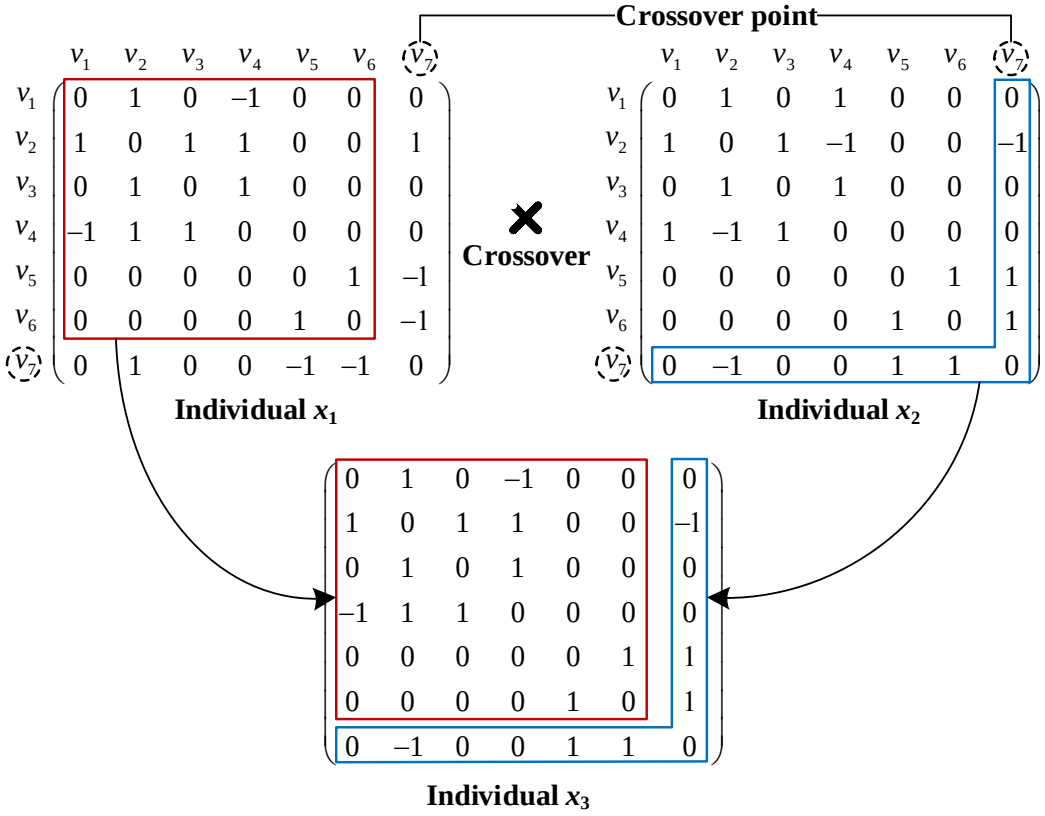


FIGURE 3.4: An illustration of one-way crossover on a pair of individuals adopting the genome matrix-based representation. Individuals x_1 and x_2 are pairwise selected, and v_7 is selected as a crossover point. A new individual x_3 is generated by replacing the elements in the seventh row and the seventh column of x_1's genome matrix with the corresponding elements of x_2's genome matrix

- **Mutation:** The mutation operator generates new individuals by randomly selecting edges and changing their partition attributes. The number of selected edges is equal

to $p_{mu} \cdot m$, in which p_{mu} denotes the mutation rate and m represents the number of edges in a network. In practice, the values of the corresponding elements in a genome matrix are converted from -1 to 1 or from 1 to -1. The detailed mutation operation on the genome matrix-based presentation are provided as shown in Figure 3.5.

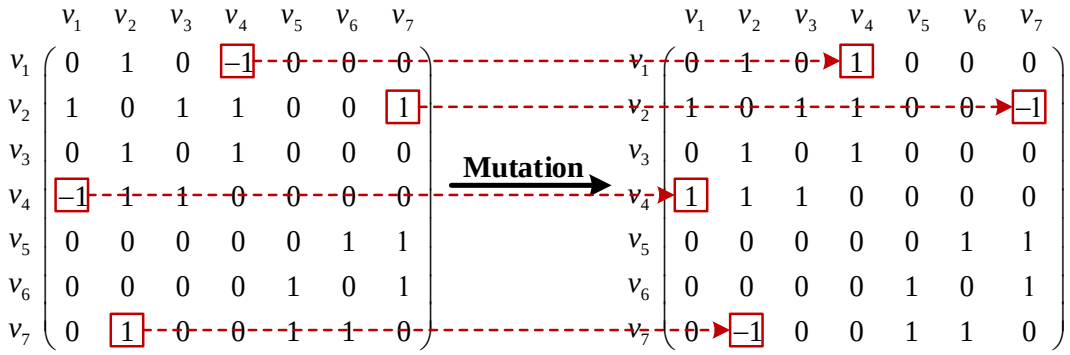


FIGURE 3.5: An illustration of the mutation on an individual adopting the genome matrix-based representation. Two edges, $e_{1,4}$ connecting v_1 and v_4 and $e_{2,7}$ connecting v_2 and v_7, are selected at random. We change the values of corresponding elements $genome_{1,4}$, $genome_{4,1}$, $genome_{2,7}$, and $genome_{7,2}$ of the genome matrix to the opposite

- **Selection:** To avoid the increment of computational cost caused by population expansion, the selection operator eliminates candidate solutions of poor quality and maintains a constant population size for all iterations. In practice, n_c individuals are eliminated based on the evaluation of candidate solutions where n_c represents the number of generated child individuals.

3.4.3 Organisation of Operators

The search process for the optimal solution proceeds as new candidate solutions are generated by executing operators. Figure 3.6 demonstrates how ECD organises the migration operator to work in tandem with crossover and mutation operators by a parameter $p_{mu/mi}$.

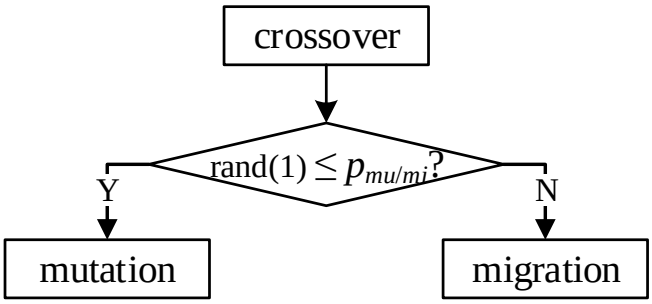


FIGURE 3.6: Operators organised to generate new candidate solutions

3.5 Multi-objective Optimisation

Under the temporal smoothness framework (see Section 2.2.1), recent studies have found that dealing with community detection in dynamic networks as a multi-objective optimisation problem is a powerful method [43], which improves the effectiveness of dynamic community detection. To this end, we apply two metrics for multi-objective optimisation based dynamic community detection to measure *SC* and *TC*, respectively.

In solving the problem of dynamic community detection, two metrics mentioned before (see Section 2.2.2) are going to be maximised at the same time as a multi-objective optimisation. Jointly with modularity to evaluate the quality of a community structure and NMI to measure the similarity of community structure between the current and previous time steps, our multi-objective optimised solution to community detection in dynamic networks is formulated as

$$C_t^* = \begin{cases} \arg\max_{C_t}\{Q(C_t),\ \mathrm{NMI}(C_t, C_{t-1})\},\ t \geq 2 \\ \arg\max_{C_t}\{Q(C_t)\},\ t = 1 \end{cases}. \tag{3.5}$$

At the first time step, i.e., $t = 1$, without a prior community structure to compare, C_t^* is a network partition that represents a community structure with the highest value of modularity. In the following time steps, modularity and NMI are simultaneously optimised with dynamic community detection algorithms.

3.6 Initialisation of ECD

For community detection in dynamic networks, the first network snapshot is partitioned with modularity-based optimisation without prior knowledge of the network partition at the initial step. The snapshots in the following time steps are partitioned by simultaneously optimising modularity and NMI.

Algorithm 1 The initialisation of ECD

1: Initialise a dynamic community structure $C = \{C_1, C_2, ..., C_T\}$ by setting $C_t = \varnothing$ where $t \in \{1, 2, ..., T\}$

2: Initialise a non-dominated solution set $EP = \varnothing$

3: Initialise a weight vector $W = \{w_1, w_2, ..., w_{pop}\}$ where $w_i = \{w_i^1, w_i^2\}$ and $w_i^1 + w_i^2 = 1$

4: Initialise a reference point $Z^* = (z_1^*, z_2^*)$ with $z_1^* = \max(Q(x_i^t))$ and $z_2^* = \max(\text{NMI}(x_i^t, x_i^{t-1}))$ where $x_i \in$ population X

5: Initialise a set of neighbour vectors $B = \{b_1, b_2, ..., b_{pop}\}$ where $b_i = \{b_i^1, b_i^2, ..., b_i^{nn}\}^T$, nn is the neighbour size for each subproblem

After generating the initial population based on PNM (see Section 3.3), for each network snapshot at time step $t \geq 2$, the multi-objective optimisation problem of community detection can be decomposed into pop scalar objective optimisation subproblems [56]. For the i-th subproblem, we generate a group of nn nearest neighbours $b_i = \{b_i^1, b_i^2, ..., b_i^{nn}\}$ based on the Euclidean distances of a weight vector $W = \{w_1, w_2, ..., w_{pop}\}^T$, where nn is the neighbour size of each subproblem; pop is the population size; the weight point $w_i = \{w_i^1, w_i^2\}$ that represents the position of a neighbour satisfies $w_i^1 + w_i^2 = 1$ and $w_i^1 = (i-1)/(pop-i)$. Then, applying the Tchebycheff approach [56], the i-th objective optimisation can be expressed as Eq. 3.6, in which a reference point $Z^* = (z_1^*, z_2^*)$ satisfies that $z_j^* = \{\max f_j(x) | x_i \in X\}$ is the maximum value of the j-th objective function.

$$\begin{aligned} &\max g^{te}(x_i | w_i, Z^*) = \min_{1 \leq j \leq 2} w_i^j \left| f_j(x_i) - z_j^* \right|, \\ &\textit{subject to } x_i \in X. \end{aligned} \tag{3.6}$$

3.7 Updating Rules

The search process for the optimal solution to a multi-objective optimisation problem not merely depends on the related operators but also is closely related to the population updating rules and key parameters during the iteration process. To detect a community structure of a network snapshot at time step $t \geq 2$, ECD updates the population by updating the neighbour b_i ($i \in \{1, 2, ..., pop\}$) for each subproblem based on the Tchebycheff approach. For the j-th neighbour of the i-th subproblem, if $g^{te}(x_i^{'}|w_j, Z^*) < g^{te}(b_i^j|w_j, Z^*)$, then ECD replaces the neighbour b_i^j with $x_i^{'}$, in which $x^{'}$ is a new solution generated by operators. Then those solutions dominated by $x_i \in X$ are cleared up from a non-dominated solution set EP, and x_i is added to EP if not dominated by any solution in EP. Finally, the reference point Z^* is updated by $z_1^* = \max(z_1^*, Q(x_i^t))$ and $z_2^* = \max(z_2^*, \text{NMI}(x_i^t, C_{t-1}))$.

3.8 Experiments

This section reports the experiments conducted to evaluate the performance of ECD. Section 3.8.1 and Section 3.8.2 first give an brief introduction to datasets and baselines, respectively. Then Section 3.8.3 demonstrates how we set the parameters in ECD. Finally, Section 3.8.4 reports and analyses the experimental results.

3.8.1 Datasets

We conduct experiments on six synthetic networks and two real-word networks to evaluate the performance of ECD and state-of-the-art baselines that will be introduced in Section 3.8.2.

The synthetic datasets involved in experiments include the SYN-FIX, SYN-VAR [57], and SYN-EVENT [58] datasets, which are used to generate synthetic dynamic networks with different parameter settings. The two real-world datasets involved in experiments are the Cellphone Calls and the Enron Mail datasets, which are used to generate two real-world dynamic networks. An brief introduction to all datasets and details on generating dynamic networks are given as follows.

SYN-FIX Dataset

The first synthetic dataset, SYN-FIX, consists of 128 nodes which are equally divided into four communities. Each node has an average degree of 16 and shares *zout* edges across communities over snapshots. A large *zout* means a high noise level of a network. The community structure becomes less clear as *zout* increases. To characterise the evolution feature of the community structure, three nodes in each community are randomly selected and added into other communities at each time step. In our work, we set *zout* to 3 and 5 to generated two SYN-FIX networks for 10 time steps, respectively.

SYN-VAR Dataset

The second synthetic dataset, SYN-VAR, initially consists of 256 nodes which are evenly divided into four communities. Contrasted with the SYN-FIX dataset, SYN-VAR datasets involves the attachment and detachment of nodes and the formation and dissolution of communities. Eight nodes per community at the current time step are randomly selected to form a new community for the following time steps. To generate dynamic networks with 10 consecutive snapshots, this change occurs during the first five time steps, and then those selected nodes are returned to their original communities for the next five time steps. Therefore, the number of communities for the 10 time steps is 4, 5, 6, 7, 8, 8, 7, 6, 5, and 4. Additionally, the average degree of each node is equal to half of the size of the community to which the node belongs. At each time step, 16 nodes are randomly deleted and another 16 nodes are refilled to the network. Analogous to the way to generate SYN-FIX networks, we set *zout* to 3 and 5 to generated two SYN-VAR networks for 10 time steps, respectively.

SYN-EVENT Dataset

The last synthetic dataset, SYN-EVENT [58], is designed to simulate different evolution events of dynamic networks. Two events are involved to generate two different dynamic networks, and the rules are given as follows:

- *i. Expansion and Contraction*: 10 per cent of communities are randomly expanded

or contracted by 25 per cent of their size. The new nodes are chosen at random from the other communities for expansion.

- *ii. Merging and Splitting*: from the first time step, 10 per cent of communities are split; 10 per cent of communities are chosen to be pairwise merged.

Based on these rules, two SYN-EVENT networks are generated for 10 time steps containing 250 nodes in each network, with the average degree of 10, the number of communities between 5 and 10, and the mixing parameter of 0.2.

Cellphone Calls Dataset

The Cellphone Calls dataset developed from the VAST 2008 mini challenge 3[1] consists of cell call phone records from the members of the fictitious Paraiso movement. In this dataset, phone calls between 400 cellphones are recorded for a period of ten days in June 2006. To construct a dynamic network, a node represents a cellphone, and an edge represents the occurrence of a phone call between two cellphones. Each network snapshot is constructed based on the records of phone calls made within one day.

Enron Mail Dataset

The other real-world dataset we used to evaluate our proposed algorithm is called Enron Mail[2] from a U.S. enterprise Enron with potential anomalous email communications in 2001. To construct a dynamic network, a node represents a user and an edge represents message sending from a user to another. we split the dataset into 12 snapshots by month, and each snapshot consists of 151 nodes.

3.8.2 Baselines

To evaluate the performance of ECD on dynamic community detection, we conduct a series of experiments on synthetic and real-world networks and compared the results with

[1] http://www.cs.umd.edu/hcil/VASTchallenge08/
[2] http://www.cs.cmu.edu/~enron/

three state-of-the-art algorithms, sE-NMF [59], DYNMOGA [43], and FacetNet [35]. These baselines use different ways to improve the clustering accuracy of dynamic community detection.

- sE-NMF [59] is a semi-supervised evolutionary non-negative matrix factorisation method for detecting dynamic communities, which integrates priori information into the objective function.
- DYNMOGA [43] is a multi-objective genetic algorithm for dynamic community detection which achieves temporal smoothness by maximising snapshot quality and minimising *TC*.
- FacetNet [35] is a framework that analyses communities and their evolution through a robust unified process. It uses *SC* rather than snapshot quality to evaluate the clustering quality.

3.8.3 Parameter Setting

TABLE 3.1: Parameter setting for ECD

Parameter	Value
maxgen	100
pop	100
p_{mu}	0.20
p_{mi}	0.50
$p_{mu/mi}$	0.50
R	0.20

All experiments are repeated five times, and the averaged results are reported in the following section. The parameter setting for ECD is displayed in Table 3.1. *maxgen* represents the maximum number of iterations. *pop* denotes the population size, i.e., the number of individuals or candidate solutions in the population. p_{mu} and p_{mi} represent the

mutation rate and the migration rate, respectively. $p_{mu/mi}$ is a parameter that determines whether to execute the mutation operator or the migration operator (see Section 3.4.3). R is the fraction of edges PNM initialises as inter-community edges (see Section 3.3). Besides, to reduce the computational cost for each subproblem in multi-objective optimisation, we set the neighbour size $nn = 5$ for all synthetic networks and set $nn = 10$ for real-world networks.

3.8.4 Results

The performance of algorithms is evaluated by comparing the similarity between detected and ground truth community structures of corresponding snapshots in all time steps. The similarity is measured by the averaged value of NMI [39] as described in Section 2.2.2.

Results on Synthetic Datasets

TABLE 3.2: NMI values returned by ECD and baselines on the SYN-FIX dataset with $zout = 3$. The best results are highlighted in bold

Time step	1	2	3	4	5
ECD	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
sE-NMF	**1.0000**	**1.0000**	**1.0000**	0.9275	0.9631
DYNMOGA	0.9537	0.9748	**1.0000**	**1.0000**	**1.0000**
FacetNet	0.4862	0.4786	0.4765	0.4710	0.4727
Time step	6	7	8	9	10
ECD	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
sE-NMF	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
DYNMOGA	**1.0000**	**1.0000**	0.9748	0.9748	**1.0000**
FacetNet	0.4739	0.4681	0.4709	0.4691	0.4722

The first set of experiments are carried out on synthetic networks generated from the SYN-FIX and SYN-VAR datasets [57]. As shown in Tables 3.2 and 3.3, ECD shows a

TABLE 3.3: NMI values returned by ECD and baselines on the SYN-VAR dataset with $zout = 3$. The best results are highlighted in bold

Time step	1	2	3	4	5
ECD	**1.0000**	0.9818	**1.0000**	**1.0000**	**0.9450**
sE-NMF	0.9206	0.9518	0.9808	0.9802	0.8755
DYNMOGA	0.9746	**0.9944**	0.9699	0.9939	0.9252
FacetNet	0.5426	0.5567	0.5767	0.5985	0.5662
Time step	6	7	8	9	10
ECD	**0.9450**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
sE-NMF	0.8761	0.9734	0.9764	0.9510	0.9164
DYNMOGA	**0.9450**	0.9825	**1.0000**	0.9935	0.9856
FacetNet	0.5640	0.5929	0.5618	0.5405	0.5092

better overall performance on the SYN-FIX and SYN-VAR datasets with $zout = 3$ than the baselines. ECD consistently holds the highest NMI value of 1 at each time step on the SYN-FIX dataset with $zout = 3$, and also achieves the highest NMI value in most time steps on the SYN-VAR dataset with $zout = 3$, except for a value of 0.9818 which is lower than 0.9944 returned by DYNMOGA at the second time step.

In contrast to the performance on the synthetic datasets with $zout = 3$, ECD does not achieve a best performance among algorithms on the SYN-FIX and SYN-VAR datasets with $zout = 5$. As observed from Table 3.4, ECD only holds the highest NMI value at time step $t = 4$, whereas sE-NMF holds the highest NMI value in most time steps, i.e., $t = 1$, 3, 5, 7, 8, 9, and 10, and DYNMOGA achieves the highest NMI value for time steps $t = 2$ and $t = 6$. Table 3.5 demonstrates that ECD achieves the highest NMI value of 1 at the first two and last two time steps while DYNMOGA achieves the highest NMI value in five successive times steps, i.e., from $t = 4$ to $t = 8$. With $zout = 5$, ECD performs better on the SYN-VAR dataset than on the SYN-FIX dataset. FacetNet performs worst on the SYN-FIX and SYN-VAR datasets no matter *zout* is 3 or 5, because it always returns the

TABLE 3.4: NMI values returned by ECD and baselines on the SYN-FIX dataset with $zout = 5$. The best results are highlighted in bold

Time step	1	2	3	4	5
ECD	0.9143	0.9714	0.8857	**1.0000**	0.9429
sE-NMF	**1.0000**	0.9144	**1.0000**	0.9551	**1.0000**
DYNMOGA	0.9297	**1.0000**	0.9671	0.9748	0.9748
FacetNet	0.4505	0.4478	0.4584	0.4563	0.4552
Time step	6	7	8	9	10
ECD	0.8571	0.9714	0.8489	0.8857	0.8857
sE-NMF	0.9975	**1.0000**	**1.0000**	**0.9314**	**0.9753**
DYNMOGA	**1.0000**	0.9485	0.9794	0.9058	0.8444
FacetNet	0.4517	0.4511	0.4481	0.4427	0.4540

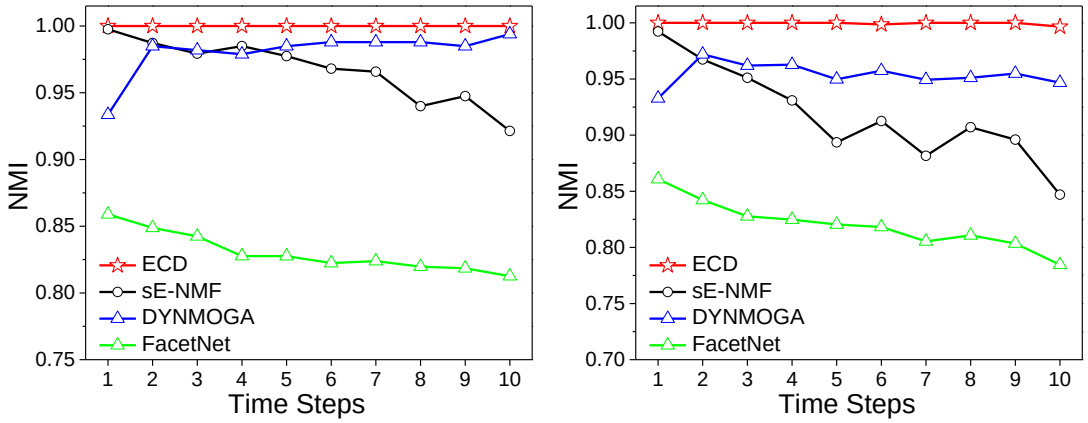
lowest NMI value at each time step.

TABLE 3.5: NMI values returned by ECD and baselines on the SYN-VAR dataset with $zout = 5$. The best results are highlighted in bold

Time step	1	2	3	4	5
ECD	**1.0000**	**1.0000**	0.9507	0.8842	0.8023
sE-NMF	0.9221	0.9280	**0.9804**	0.9550	0.8628
DYNMOGA	0.9126	0.9738	0.9704	**0.9825**	**0.9123**
FacetNet	0.5323	0.5452	0.5674	0.5838	0.5549
Time step	6	7	8	9	10
ECD	0.8160	0.9295	0.9507	**1.0000**	**0.9990**
sE-NMF	0.8737	0.9442	0.9588	0.9309	0.9131
DYNMOGA	**0.8932**	**0.9660**	**0.9720**	0.9935	0.9844
FacetNet	0.5526	0.5773	0.5479	0.5251	0.4954

The second set of experiments are carried out on synthetic networks generated from the SYN-EVENT datasets [58]. Figure 3.7 displays NMI values of in the form of symbol

and line for all four algorithms on the two dynamic networks derived from the SYN-EVENT dataset. As shown in Figure 3.7(a), ECD's consistent NMI value of 1 on the *Expansion and Contraction* network is always the highest for all time steps. Additionally, the NMI value of 1 or very close to 1, returned by ECD on the *Merging and Splitting* network, is also the highest at each time step.



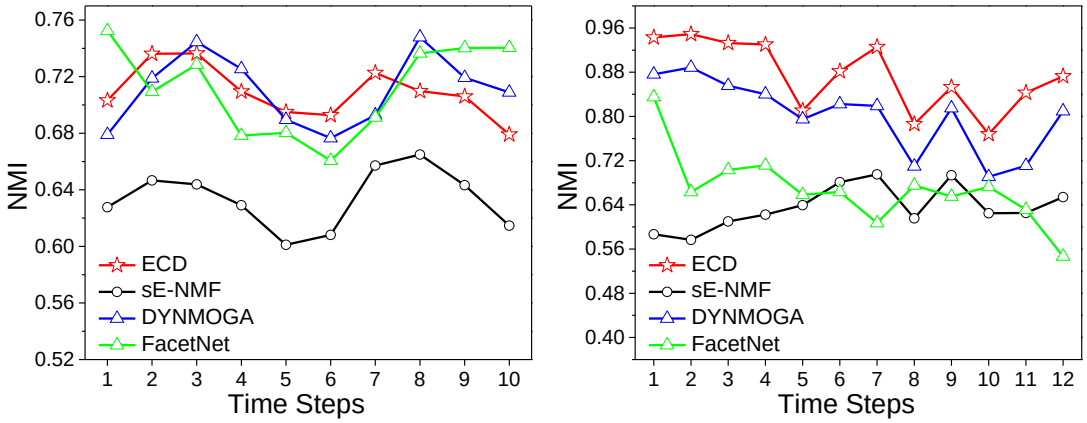
(a) *Expansion and Contraction* (b) *Merging and Splitting*

FIGURE 3.7: Comparison of NMI values among ECD and baselines on the SYN-EVENT dataset: (a) *Expansion and Contraction* and (b) *Merging and Splitting*

Results on Real-world Datasets

The third set of experiments are carried out on two real-world networks. Due to the unknown community structures of the real-world networks, we applied the first step of DYNMOGA which does not require the number of communities as an input parameter, to reveal the community structure of each snapshot as the ground truth.

Figure 3.8 displays the experimental results on the Cellphone Calls network and the Enron Mail network through a comparison of NMI values returned by algorithms at each time step. As Figure 3.8(a) shows, ECD only gets the highest NMI values at time steps t = 2, 5, 6, and 7, which suggests that the evolving community structure detected by ECD is not always the most similar to the ground truth on the Cellphone Calls network. As demonstrated in 3.8(b), ECD has the highest NMI value at each time step, which suggests



(a) Cellphone Calls (b) Enron Mail

FIGURE 3.8: Comparison of NMI values among ECD and baselines on the real-world datasets: (a) Cellphone Calls and (b) Enron Mail

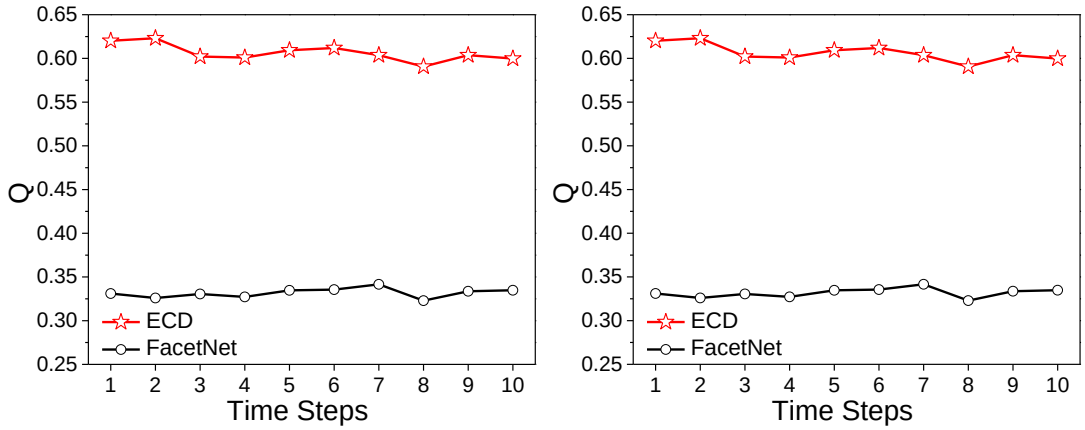
that ECD performs better than the other three baseline algorithms in uncovering the evolving community structure on the Enron Mail network.

3.8.5 Discussion

Comparing NMI values that measures the similarity between detected and ground truth community structures at each time step, our proposed algorithm ECD does not have a satisfactory performance on every dataset.

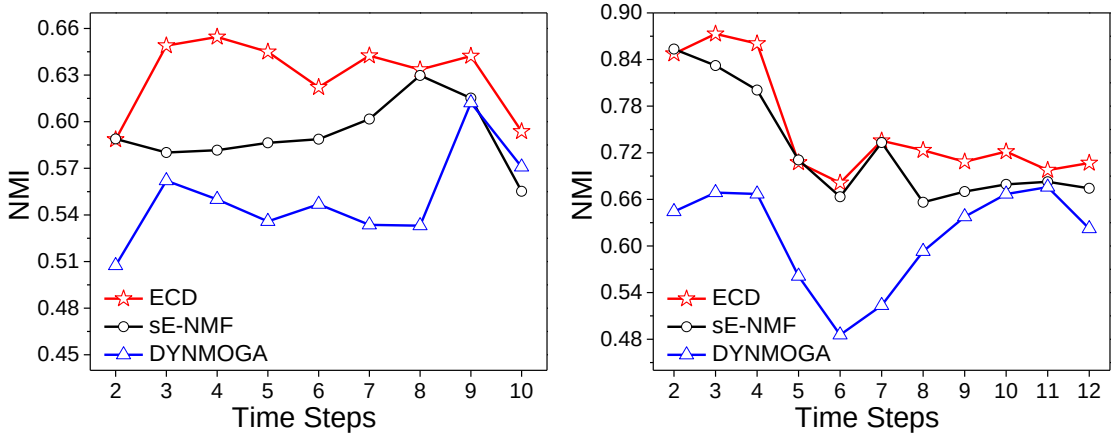
For the SYN-SIX and SYN-VAR datasets, ECD performs the best with $zout = 3$, but it does not always achieve the highest NMI value with $zout = 5$. As described in Section 3.8.1, the community structures in synthetic networks generated from two kinds of datasets become less clear as *zout* increases. so ECD requires improvement to efficiently detect the evolving community structure in networks with high noise levels.

For the real-world datasets, ECD performs much better on the Enron Mail network than any baseline, but does not always detect the community structure most similar to the ground truth on the Cellphone Calls network. Then we evaluate the performance of ECD in terms of temporal smoothness. The temporal smoothness between community



(a) Cellphone Calls (b) Enron Mail

FIGURE 3.9: Comparison of modularity values between ECD and FacetNet on the real-world datasets: (a) Cellphone Calls and (b) Enron Mail



(a) Cellphone Calls (b) Enron Mail

FIGURE 3.10: Comparison of temporal smoothness measured by NMI among ECD, sE-NMF, and DYNMOGA on the real-world datasets: (a) Cellphone Calls and (b) Enron Mail

structures at the current and previous time steps is measured by NMI, which is evaluated from the second time step. As reported in Figure 3.9, FacetNet achieves much lower modularity values than ECD in all time steps, which corresponds to poor quality of community structures detected by FacetNet. Therefore, we compare ECD with two other algorithms on the real-world datasets. The results shown in Figure 3.10 indicate that ECD achieves better temporal smoothness than sE-NMF and DYNMOGA in dynamic community

detection.

3.8.6 Summary

To summarise the experimental results, ECD shows a better overall performance in dynamic community detection. Although ECD performs much better than the baselines on five out of eight networks generated from synthetic and real-world datasets, it still needs to be improved.

4

DECS: Detecting Evolving Community Structure – An Improved ECD

As discussed in Chapter 3.8.5, our proposed algorithm, ECD, does not perform satisfactorily on networks with high noise levels. Therefore, in this chapter, we develop an improved ECD, called DECS, to improve the dynamic community detection results with respect to clustering accuracy and temporal smoothness. Section 4.1 provides an overview of DECS. Section 4.2 details the population generation process by simulating a process of label propagation, and Section 4.3 describes the two-way crossover operator employed by DECS to generate new individuals. Finally, Section 4.4 reports and analyses experimental results to evaluate the performance of DECS.

4.1 Framework of DECS

DECS is developed based on the framework of ECD to improve the clustering accuracy and temporal smoothness. The network representation, selected objectives to optimise, initialisation, operators except for the crossover operator, and updating rules in DECS are the same as those in ECD.

Algorithm 2 DECS: Detecting the Evolving Community Structure

Input: A dynamic network $G = \{G_1, G_2, ..., G_T\}$

Output: A dynamic community structure $C = \{C_1, C_2, ..., C_T\}$

1: **for** time step t from 1 to T **do**
2: Initialise the population $X = \{x_1, x_2, ..., x_{pop}\}$ based on Algorithms 3 and 4
3: **if** $t = 1$ **then**
4: **while** iteration step *iter* does not reach the maximal generation *maxgen* **do**
5: Execute the organised operators described in Section 3.4.3
6: Calculate the value of a single objective function for each individual
7: Rank all individuals in descending order of the objective function value and generate a new population with top *pop* individuals
8: **end while**
9: **else**
10: Initialise the related parameters for problem decomposition by Algorithm 1
11: **while** iteration step *iter* does not reach the maximal generation *maxgen* **do**
12: Execute the organised operators described in Section 3.4.3
13: Calculate the values of multiple objective functions for each individual
14: Generate a new population by following the updating rules described in Section 3.7
15: **end while**
16: **end if**
17: Select the best solution and update C_t
18: **end for**

There are two main differences between DECS and ECD: DECS generates the initial population by applying a label propagation process [60] in networks and adopts a two-way crossover operator [61] to reduce the computational cost.

4.2 Population Generation Based on Label Propagation

Bagrow and Bollt studied label propagation through networks [62]. Label propagation is similar to the process of the spread of an epidemic. Even if two nodes are not directly connected by an edge, they will receive some message, like a viral infection, from each other via a path between them. Nodes receiving the same latest message should be clustered into the same cluster.

Algorithm 3 PGLP: Population Generation Based on Label Propagation

Input: The maximal label propagation iteration $iter_m$

Output: The initialised chromosomes

```
for k from 1 to pop do
    for i from 1 to iter_m do
        for j from 1 to |V| do
            if neighbour size ns_j of v_j more than 1 then
                Assign the label of the community containing most of v_j's neighbours to v_j
            else
                Assign the community label of the neighbour to v_j
            end if
        end for
    end for
end for
```

For community detection, each node v_i in a network is initialised with a unique community label $l_i \in \{1, 2, ..., |V|\}$ which determines which community it belongs to. We assume that a node tends to join the community to which most of its neighbours belong. Driven by label propagation [62], densely connected nodes are evolutionarily clustered

into communities. As the process is iteratively repeated, a network partition with a high clustering quality will be obtained.

Algorithm 4 The transformation from chromosomes into genome matrices

Input: The initialised chromosomes generated by PGLP, an adjacent matrix A

Output: The initialised genome matrices

```
for Each chromosome in the population do
    Set a genome matrix genome = A
    for Each community i do
        for Each node v_j in community i do
            for Each neighbour k of node v_j do
                if neighbour k is not in community i then
                    Set genome_{i,j} = -1
                end if
            end for
        end for
    end for
end for
```

Using the information of neighbours' labels, PGLP simulates the label propagation to generate the initial population based on the chromosome representation, as described in Algorithm 3. To speed up the convergence in our proposed algorithm DECS, after each node v_i is initially assigned with a unique label $l(i) = i$, the label of each node will be updated by assigning the community label that most of its neighbours hold. This operation is repeated $iter_m$ times in Algorithm 3 on each chromosome.

Further, for the successful operation of PGLP on DECS which represents a network as a genome matrix, DECS transforms the chromosome representation into the genome representation as shown in Algorithm 4. We assume that all the edges within communities exist as intra-community connections, while edges across communities exist as inter-community connections.

4.3 Two-way Crossover

To reduce the computational cost, DECS employs a two-way crossover operator on genome matrices, which is inspired by an ideal of crossover operation on chromosomes [61].

The two-way crossover randomly selects individuals in pairs and then randomly selects a varying number of nodes as crossover points. Finally, the inter- and intra-community attributes of the edges connecting to the crossover points are exchanged. The two-way crossover operator which generates two child individuals by exchanging the roles of two parent individuals, is more efficient than the one-way crossover operator employed by ECD.

4.4 Experiments

Following the previous experiments in Section 3.8, we conduct experiments to evaluate DECS's performance on the same synthetic and real-world networks.

4.4.1 Parameter Setting

TABLE 4.1: Parameter setting for DECS

Parameter	Value
maxgen	100
pop	100
p_{mu}	0.20
p_{mi}	0.50
$p_{mu/mi}$	0.50
$iter_m$	5

All experiments are repeated five times, and the averaged results are reported in the following section. The parameter setting for DECS is displayed in Table 4.1. The maximum

number of label propagation iterations $iter_m$ is set to 5, and other parameters are set the same as ECD.

4.4.2 Results

Results on Synthetic Datasets

TABLE 4.2: NMI values returned by DECS and ECD on the SYN-FIX dataset with $zout = 3$. The best results are highlighted in bold

Time step	1	2	3	4	5
DECS	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
ECD	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Time step	6	7	8	9	10
DECS	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
ECD	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**

TABLE 4.3: NMI values returned by DECS and ECD on the SYN-VAR dataset with $zout = 3$. The best results are highlighted in bold

Time step	1	2	3	4	5
DECS	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.9450**
ECD	**1.0000**	0.9818	**1.0000**	**1.0000**	**0.9450**
Time step	6	7	8	9	10
DECS	**0.9450**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
ECD	**0.9450**	**1.0000**	**1.0000**	**1.0000**	**1.0000**

Comparing NMI values returned by DECS and ECD, Tables 4.2 and 4.4 demonstrates that DECS achieves the same averaged NMI value of 1 as ECD in all time steps on the SYN-FIX dataset with $zout = 3$ and the *Expansion and Contraction* network. As shown in

Tables 4.3 and 4.5, DECS performs almost the same as ECD on the SYN-VAR dataset with $zout = 3$ and the *Merging and Splitting* network.

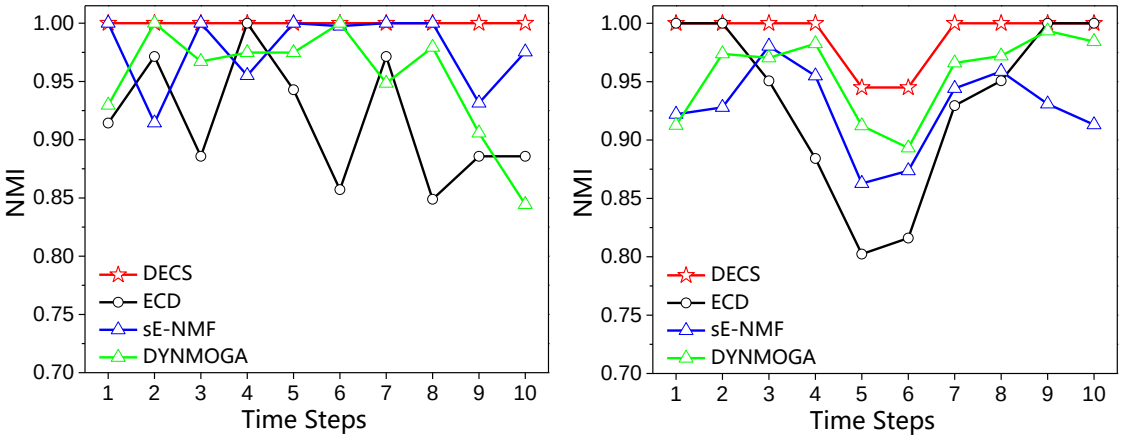
TABLE 4.4: NMI values returned by DECS and ECD on the *Expansion and Contraction* network. The best results are highlighted in bold

Time step	1	2	3	4	5
DECS	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
ECD	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Time step	6	7	8	9	10
DECS	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
ECD	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**

TABLE 4.5: NMI values returned by DECS and ECD on the *Merging and Splitting* network. The best results are highlighted in bold

Time step	1	2	3	4	5
DECS	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
ECD	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Time step	6	7	8	9	10
DECS	**1.0000**	**1.0000**	0.9984	**1.0000**	**1.0000**
ECD	0.9986	**1.0000**	**1.0000**	**1.0000**	0.9966

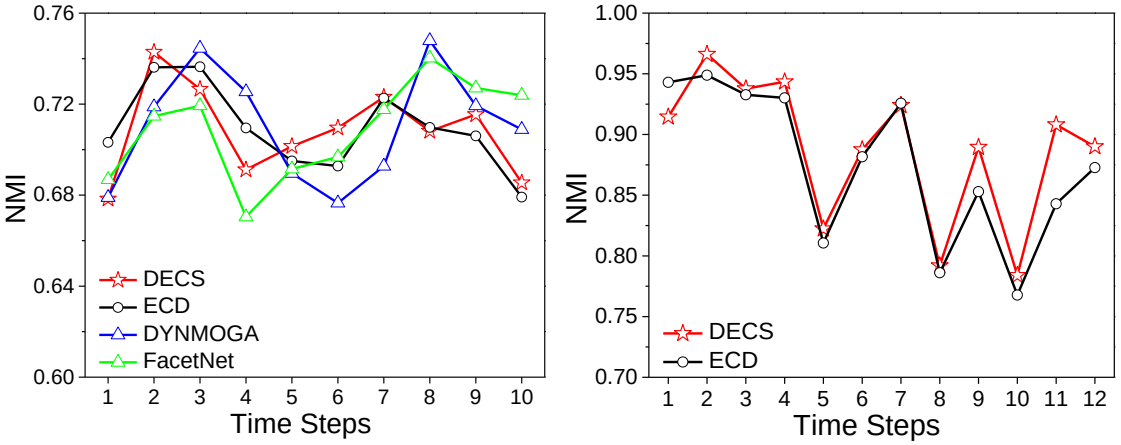
Because FacetNet performs much worse than other algorithms and it appears difficult to determine which algorithm performs best on the SYN-FIX and SYN-VAR datasets with $zout = 5$ (see Tables 3.4 and 3.5), we compare the performance of DECS with ECD, sE-NMF, and DYNMOGA on these two networks. The results returned by DECS on the two datasets with $zout = 5$ are accord with the results on these two datasets with $zout = 3$. As shown in Figures 4.1(a) and 4.1(b), the NMI value returned by DECS is the highest while ECD, DYNMOGA, and FacetNet always obtain lower NMI values at each time step.



(a) SYN-FIX with *zout* = 5 (b) SYN-VAR with *zout* = 5

FIGURE 4.1: Comparison of NMI values among DECS, ECD, and baselines on (a) the SYN-FIX dataset with $zout = 5$ and (b) the SYN-VAR dataset with $zout = 5$

Results on Real-world Datasets

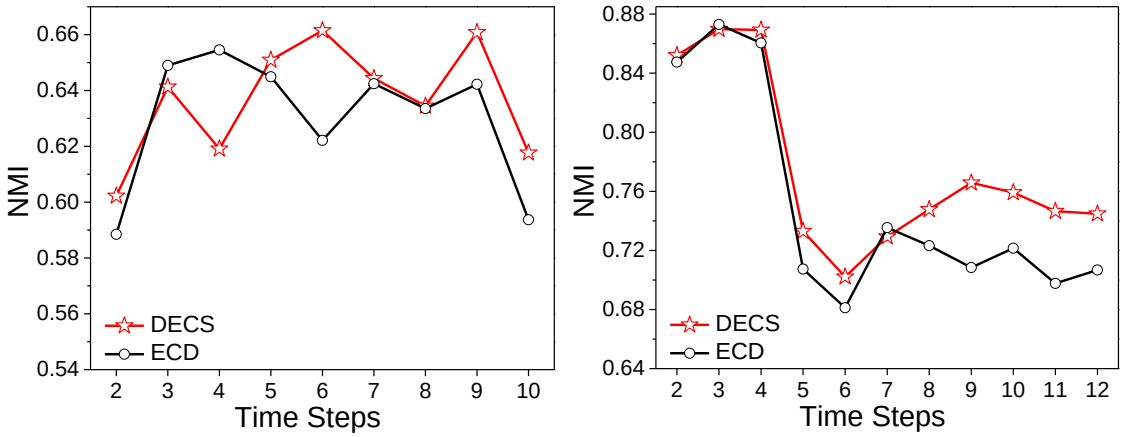


(a) Cellphone Calls (b) Enron Mail

FIGURE 4.2: Comparison of NMI values among DECS, ECD, and baselines on the real-world datasets: (a) Cellphone Calls and (b) Enron Mail

Figure 4.2 reports the experimental results at $nn = 10$ on the two real-world datasets through comparing NMI values returned by algorithms. As demonstrated in Figure 4.2(a), DECS does not achieve the highest NMI value at each time step on the Cellphone Calls dataset. Specifically, DECS achieves the highest NMI value at time steps $t = 2$, 5, 6, and

7, which suggests that the evolving community structure detected by DECS is not always the most similar to the ground truth. When compared to ECD, DECS gets a higher NMI value in 6 out of 10 time steps, i.e., $t = 2$, 5, 6, 7, 9, and 10. Because ECD performs much better than all baselines on the Enron Mail dataset, we compare DECS with ECD on this dataset. As demonstrated in Figure 4.2(b), DECS has a higher NMI value than ECD at each time step except for the first time step.



(a) Cellphone Calls (b) Enron Mail

FIGURE 4.3: Comparison of the temporal smoothness measured by NMI between DECS and ECD on the real-world datasets: (a) Cellphone Calls and (b) Enron Mail

Furthermore, as shown in Figure 4.3, we discuss the performance of DECS on temporal smoothness. Similarly, we directly compare DECS with ECD which achieves a better temporal smoothness than the baselines. Figure 4.3(a) demonstrates that DECS holds a better temporal smoothness from time step $t = 5$ to time step $t = 10$ on the Cellphone Calls dataset. In addition, Figure 4.3(b) shows that DECS holds a better temporal smoothness at every time step except for time steps $t = 3$ and 7. In summary, DECS holds a better temporal smoothness on the real-world datasets.

4.4.3 Summary

The experiments demonstrate that DECS proposed to detect dynamic community structure overcome ECD's shortcoming that ECD performs not well on networks with high noise

levels. Additionally, DECS shows a better performance in terms of clustering accuracy and temporal smoothness, contrasted with ECD and baselines.

5

Conclusion

The operators and network representation that EC-based algorithms use are closely related to the efficiency and effectiveness of dynamic community detection. Following a review of the literature on dynamic community detection via EC, it is clear that state-of-the-art algorithms require further improvements in efficiency and effectiveness. Therefore, we introduce a migration operator that works in tandem with classic genetic operators to exploit the inter- and intra-connection relationships between nodes in networks. In addition, we apply a genome matrix to represent a network snapshot and optimise the modularity directly calculated from the genome matrix.

Then, treating dynamic community detection as a multi-objective optimisation problem, we develop a new EC-based algorithm, ECD, which applies a bio-inspired model for population generation. The experimental results on both synthetic and real-world networks indicate the better overall performance of ECD with respect to clustering accuracy

and temporal smoothness, compared with state-of-the-art baselines.

To further improve the performance of ECD on dynamic community detection in networks with high noise levels, we develop a improved EC-based algorithm, DECS, which is based on the basic framework of ECD. A two-crossover operator used to reduce the computational cost and the PGLP applied to population generation help DECS to better detect the evolving community structure in dynamic networks. The experimental results demonstrate the superiority of DECS in terms of clustering accuracy and temporal smoothness, contrasted with ECD and the baselines.

For future work, we will focus on the scalability of these algorithms to large-scale real-world networks.

List of Symbols

The following list is neither exhaustive nor exclusive, but may be helpful.

EC	evolutionary clustering
SC	snapshot cost
TC	temporal cost
NMI	normalized mutual information
MOEAs	multi-objective evolutionary algorithms
PNM	*Physarum*-based network model
PM	*Physarum*-based mathematical model

References

[1] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. *Classes of small-world networks*. Proceedings of the National Academy of Sciences **97**(21), 11149 (2000). 1

[2] A.-L. Barabási. *Network Science* (Cambridge University Press, 2016). 1

[3] N. Eagle, A. S. Pentland, and D. Lazer. *Inferring friendship network structure by using mobile phone data*. Proceedings of the National Academy of Sciences **106**(36), 15274 (2009). 1

[4] D. J. Watts and S. H. Strogatz. *Collective dynamics of 'small-world' networks*. Nature **393**(6684), 440 (1998). 1

[5] M. E. J. Newman. *Models of the small world*. Journal of Statistical Physics **101**(3), 819 (2000). 1

[6] A.-L. Barabási and R. Albert. *Emergence of scaling in random networks*. Science **286**(5439), 509 (1999). 1

[7] A.-L. Barabási and E. Bonabeau. *Scale-free networks*. Scientific American **288**(5), 60 (2003). 1

[8] M. Girvan and M. E. J. Newman. *Community structure in social and biological networks*. Proceedings of the National Academy of Sciences **99**(12), 7821 (2002). 2, 8

[9] J. Chen and B. Yuan. *Detecting functional modules in the yeast protein-protein interaction network*. Bioinformatics **22**(18), 2283 (2006). 2

[10] A. Traud, E. Kelsic, P. Mucha, and M. Porter. *Comparing community structure to characteristics in online collegiate social networks*. SIAM Review **53**(3), 526 (2011). 2

[11] J. Yang, J. McAuley, and J. Leskovec. *Community detection in networks with node attributes*. In *2013 IEEE 13th International Conference on Data Mining*, pp. 1151–1156 (2013). 2

[12] P. Krishna Reddy, M. Kitsuregawa, P. Sreekanth, and S. Srinivasa Rao. *A graph based approach to extract a neighborhood customer community for collaborative filtering*. In *Proceedings of the Second International Workshop on Databases in Networked Information Systems*, pp. 188–200 (2002). 2

[13] R. Agrawal and H. V. Jagadish. *Algorithms for searching massive graphs*. IEEE Transactions on Knowledge and Data Engineering **6**(2), 225 (1994). 2

[14] A. Y. Wu, M. Garland, and J. Han. *Mining scale-free networks using geodesic clustering*. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 719–724 (ACM, 2004). 2

[15] Y. Zhang, L. Wang, Y.-Q. Zhang, and X. Li. *Towards a temporal network analysis of interactive WiFi users*. EPL (Europhysics Letters) **98**(6), 68002 (2012). 2

[16] C. Gao and J. Liu. *Network-based modeling for characterizing human collective behaviors during extreme events*. IEEE Transactions on Systems, Man, and Cybernetics: Systems **47**(1), 171 (2017). 2

[17] G. Rossetti and R. Cazabet. *Community discovery in dynamic networks: A survey*. ACM Computing Surveys **51**(2), art. no. 35 (2018). 2

[18] W. W. Zachary. *An information flow model for conflict and fission in small groups*. Journal of Anthropological Research **33**(4), 452 (1977). 2

[19] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson. *The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations*. Behavioral Ecology and Sociobiology **54**(4), 396 (2003). 2

[20] M. E. J. Newman. *The structure of scientific collaboration networks*. Proceedings of the National Academy of Sciences **98**(2), 404 (2001). 2

[21] B. Wu, Q. Ye, S. Yang, and B. Wang. *Group CRM: A new telecom CRM framework from social network perspective*. In *Proceedings of the 1st ACM International Workshop on Complex Networks Meet Information & Knowledge Management*, pp. 3–10 (2009). 2

[22] J. Hopcroft, O. Khan, B. Kulis, and B. Selman. *Tracking evolving communities in large linked networks*. Proceedings of the National Academy of Sciences **101**(suppl 1), 5249 (2004). 2

[23] P. Bródka, S. Saganowski, and P. Kazienko. *GED: The method for group evolution discovery in social networks*. Social Network Analysis and Mining **3**(1), 1 (2013).

[24] S. Fortunato. *Community detection in graphs*. Physics Reports **486**(3), 75 (2010). 7

[25] M. Spiliopoulou. *Evolution in Social Networks: A Survey*, pp. 149–175 (Springer US, Boston, MA, 2011).

[26] S. Fortunato and D. Hric. *Community detection in networks: A user guide*. Physics Reports **659**, 1 (2016).

[27] N. Dakiche, F. B.-S. Tayeb, Y. Slimani, and K. Benatchba. *Tracking community evolution in social networks: A survey*. Information Processing Management **56**(3), 1084 (2019). 2

[28] D. Chakrabarti, R. Kumar, and A. Tomkins. *Evolutionary clustering*. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 554–560 (2006). 3, 9

[29] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. *Defining and identifying communities in networks* **101**(9), 2658 (2004). 8

[30] Y. Hu, H. Chen, P. Zhang, M. Li, Z. Di, and Y. Fan. *Comparative definition of community and corresponding identifying algorithm*. Phys. Rev. E **78**, 026121 (2008). 8

[31] X. Wang, L. Tang, H. Gao, and H. Liu. *Discovering overlapping groups in social media*. In *2010 IEEE International Conference on Data Mining*, pp. 569–578 (2010). 8

[32] M. Sales-Pardo, R. Guimerà, A. A. Moreira, and L. A. N. Amaral. *Extracting the hierarchical organization of complex systems*. Proceedings of the National Academy of Sciences **104**(39), 15224 (2007). 8

[33] A. Clauset, C. Moore, and M. E. J. Newman. *Hierarchical structure and the prediction of missing links in networks*. Nature **453**, 98 (2008).

[34] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. *Fast unfolding of communities in large networks*. Journal of Statistical Mechanics: Theory and Experiment **2008**(10), P10008 (2008). 8

[35] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng. *Analyzing communities and their evolutions in dynamic social networks*. ACM Transactions on Knowledge Discovery from Data **3**(2), 8 (2009). 9, 10, 26

[36] M. E. J. Newman. *Modularity and community structure in networks*. Proceedings of the National Academy of Sciences **103**(23), 8577 (2006). 9

[37] J. Yang and J. Leskovec. *Defining and evaluating network communities based on ground-truth*. Knowledge and Information Systems **42**(1), 181 (2015). 10

[38] T. Chakraborty, A. Dalmia, A. Mukherjee, and N. Ganguly. *Metrics for community analysis: A survey*. ACM Computing Surveys **50**(4), 54:1 (2017). 10

[39] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas. *Comparing community structure identification*. Journal of Statistical Mechanics: Theory and Experiment **2005**(09), P09008 (2005). 10, 27

[40] D. Chakrabarti, R. Kumar, and A. Tomkins. *Evolutionary clustering*. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 554–560 (2006). 10

[41] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng. *Evolutionary spectral clustering by incorporating temporal smoothness*. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 153–162 (2007). 10, 11

[42] M.-G. Gong, L.-J. Zhang, J.-J. Ma, and L.-C. Jiao. *Community detection in dynamic social networks based on multiobjective immune algorithm*. Journal of Computer Science and Technology **27**(3), 455 (2012). 11

[43] F. Folino and C. Pizzuti. *An evolutionary multiobjective approach for community discovery in dynamic networks*. IEEE Transactions on Knowledge and Data Engineering **26**(8), 1838 (2014). 10, 11, 21, 26

[44] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. *A fast and elitist multiobjective genetic algorithm: NSGA-II*. IEEE Transactions on Evolutionary Computation **6**(2), 182 (2002). 11

[45] J. Ma, J. Liu, W. Ma, M. Gong, and L. Jiao. *Decomposition-based multiobjective evolutionary algorithm for community detection in dynamic social networks*. The Scientific World Journal **2014**(3), 402345 (2014). 11

[46] C. Gao, Z. Chen, X. Li, Z. Tian, S. Li, and Z. Wang. *Multiobjective discrete particle swarm optimization for community detection in dynamic networks*. EPL (Europhysics Letters) **122**(2), 28001 (2018). 11

[47] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. de Carvalho. *A survey of evolutionary algorithms for clustering*. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **39**(2), 133 (2009). 11

[48] M. Tasgin, A. Herdagdelen, and H. Bingol. *Community detection in complex networks using genetic algorithms*. arXiv preprint arXiv:0711.0491 (2007). 11

[49] E. Falkenauer. *Genetic algorithms and grouping problems* (Wiley New York, 1998). 11

[50] X. Li, C. Gao, and R. Pu. *A community clustering algorithm based on genetic algorithm with novel coding scheme*. In *2014 10th International Conference on Natural Computation (ICNC)*, pp. 486–491 (2014). 12

[51] C. Gao, M. Liang, X. Li, Z. Zhang, Z. Wang, and Z. Zhou. *Network community detection based on the Physarum-inspired computational framework*. IEEE/ACM Transactions on Computational Biology and Bioinformatics **15**(6), 1916 (2018). 16

[52] Y. Lu, M. Liang, C. Gao, Y. Liu, and X. Li. *A bio-inspired genetic algorithm for community mining*. In *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, pp. 673–679 (2016). 16

[53] M. Liang, C. Gao, X. Li, and Z. Zhang. *A physarum-inspired ant colony optimization for community mining*. In *The 21th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 737–749 (2017).

[54] Z. Chen, F. Liu, C. Gao, X. Li, and Z. Zhang. *An enhanced particle swarm optimization based on physarum model for community detection*. In *Proceedings of the 8th International Conference on Swarm Intelligence*, pp. 99–108 (2017). 16

[55] B. A. Attea, W. A. Hariz, and M. F. Abdulhalim. *Improving the performance of evolutionary multi-objective co-clustering models for community detection in complex social networks*. Swarm and Evolutionary Computation **26**, 137 (2016). 18

[56] Q. Zhang and H. Li. *MOEA/D: A multiobjective evolutionary algorithm based on decomposition*. IEEE Transactions on Evolutionary Computation **11**(6), 712 (2007). 22

[57] M.-S. Kim and J. Han. *A particle-and-density based evolutionary clustering method for dynamic networks*. Proc. VLDB Endow. **2**(1), 622 (2009). 23, 27

[58] D. Greene, D. Doyle, and P. Cunningham. *Tracking the evolution of communities in dynamic social networks*. In *2010 International Conference on Advances in Social Networks Analysis and Mining*, pp. 176–183 (2010). 23, 24, 29

[59] X. Ma and D. Dong. *Evolutionary nonnegative matrix factorization algorithms for community detection in dynamic networks*. IEEE Transactions on Knowledge and Data Engineering **29**(5), 1045 (2017). 26

[60] M. Gong, Q. Cai, Y. Li, and J. Ma. *An improved memetic algorithm for community detection in complex networks*. In *2012 IEEE Congress on Evolutionary Computation*, pp. 1–8 (2012). 37

[61] M. Gong, B. Fu, L. Jiao, and H. Du. *Memetic algorithm for community detection in networks*. Physical Review E **84**, 056101 (2011). 37, 39

[62] J. P. Bagrow and E. M. Bollt. *Local method for detecting communities*. Physical Review E **72**, 046108 (2005). 37