Alexandria Engineering Journal 112 (2025) 177–208

Contents lists available at ScienceDirect

Alexandria Engineering Journal

journal homepage: www.elsevier.com/locate/aej

Original article

Bird flock effect-based dynamic community detection: Unravelling network patterns over time

Siti Haryanti Hairol Anuar [a,*], Zuraida Abal Abas [b], Iskandar Waini [c], Mohd Fariduddin Mukhtar [d], Zejun Sun [e], Eko Arip Winanto [f], Norhazwani Mohd Yunos [b]

[a] Fakulti Teknologi Dan Kejuruteraan Elektronik Dan Komputer (FTKEK), Universiti Teknikal Malaysia Melaka (UTeM), Hang Tuah Jaya, Durian Tunggal, Melaka 76100, Malaysia
[b] Fakulti Teknologi Maklumat Dan Komunikasi (FTMK), Universiti Teknikal Malaysia Melaka (UTeM), Hang Tuah Jaya, Durian Tunggal, Melaka 76100, Malaysia
[c] Fakulti Teknologi Dan Kejuruteraan Industri Dan Pembuatan (FTKIP), Universiti Teknikal Malaysia Melaka (UTeM), Hang Tuah Jaya, Durian Tunggal, Melaka 76100, Malaysia
[d] Fakulti Teknologi Dan Kejuruteraan Mekanikal (FTKM), Universiti Teknikal Malaysia Melaka (UTeM), Hang Tuah Jaya, Durian Tunggal, Melaka 76100, Malaysia
[e] School of Information Engineering, Pingdingshan University, Henan 467000, China
[f] Computer Engineering, Dinamika Bangsa University, Jambi 36138, Indonesia

ARTICLE INFO

Keywords:
Network structure
Dynamic community detection
Bird flock effect
Similarity measure

ABSTRACT

Community structure is essential for topological analysis, function study, and pattern detection in complex networks. As establishing community structure in a dynamic network is difficult, it gives a unique perspective in many interdisciplinary fields. Many researchers have explored the challenging technique that requires parameter specification and optimization for quality result. This study proposed an eco-system conceptual framework based on bird flock effect. Relying on the natural law of rule, we designed a dynamic community detection named DCDBFE. The design of algorithm was based on the three basic rules of bird flock: separation, alignment, and cohesion phase. Then, we provide an explanation of similarity measure used between vertices to identify the modules attraction. DCDBFE employs an incremental community detection approach to repeatedly detect communities in each network snapshot or time step. The contributions are obtained for high quality community detected, free-parameter and well stability. To test its performance, extensive experiments were conducted on both synthetic and real-world networks. The outcomes demonstrate that our approach can effectively find satisfaction from each time step by comparison with the other well-known algorithms.

1. Introduction

Over the past 20 years, network analysis has evolved into an indispensable tool for interpreting complex systems across diverse domains, from sociology to biology. In sociology, network analysis helps uncover social structures, relationships, and behavioral patterns [1], while in biology, it reveals interactions between genes, proteins, and ecosystems [2]. Data mining leverages networks to detect clusters and correlations in large datasets [3], and intelligent computing applies graph theory to optimize algorithms for decision-making [4]. Social networks provide insights into community formation, influence spread, and collective behavior [5], whereas communication networks improve message routing and efficiency across channels [6]. In healthcare, network models are used to track disease spread and improve patient care coordination [7], and wireless sensor networks ensure reliable data transmission in environmental monitoring [8]. Telecommunication relies on network topologies to manage traffic and prevent congestion [9], while transportation networks optimize routes and logistics [10]. Economics employs network models to understand market behavior and financial dependencies [11], and e-commerce uses them to enhance recommendation systems and buyer-seller interactions [12].

Market segmentation identifies consumer clusters based on shared traits, and manufacturing utilizes network structures to streamline production processes and enhance supply chain connectivity [13]. Nature-inspired networks model ecosystem dynamics and species interactions [14], and in agriculture and aquaculture, network analysis optimizes resource use and tracks food production systems [15]. Supply chain management relies on network theory to enhance efficiency,

* Corresponding author.
E-mail address: sitiharyanti@utem.edu.my (S.H.H. Anuar).

https://doi.org/10.1016/j.aej.2024.10.097
Received 13 November 2023; Received in revised form 23 October 2024; Accepted 24 October 2024
Available online 4 November 2024

mitigate risks, and ensure sustainability [16]. The interdisciplinary nature of real-world networks presents both opportunities and challenges, particularly when attempting to capture dynamic interactions within these systems [17]. As networks grow and change over time, understanding the evolution of communities within them becomes crucial for uncovering hidden patterns and relationships [18]. However, despite the advances in dynamic community detection, one of the most persistent problems is the instability of these algorithms, which often struggle to maintain consistent accuracy when dealing with networks that experience rapid or unpredictable changes [19].

The problem of instability in dynamic community detection is further complicated by the scale and complexity of the networks involved. The real-world networks often involve massive datasets with constantly shifting structures [20]. Existing algorithms, while effective in certain contexts, frequently fail to keep up with the rapid pace of change in these dynamic environments. They may produce results that are inconsistent or highly sensitive to minor fluctuations, making it difficult to trust the detected community structures. This issue poses a significant barrier to some practical application of dynamic community detection, where the ability to detect stable, evolving communities is essential for optimizing performance and decision-making [21].

Addressing this challenge requires the development of more adaptive and resilient algorithms that can maintain stability even in highly dynamic settings. Many current methods struggle to balance accuracy with computational efficiency, especially when applied to large-scale networks across diverse sectors [22–25]. The instability in dynamic community detection not only limits our understanding of the network's evolution but also hampers the ability to make informed decisions based on the detected patterns. Generally, dynamic community detection techniques can be categorized into two types, independent and dependent detection [26]. However, each type has its strengths and weaknesses depending on the nature of the network and the availability of data, but the overarching challenge remains to maintain a balance between temporal stability and structural adaptability.

This study addresses the dynamic community detection problem through the perspective of connectedness and influences, inspired by bird flock effect. Leveraging these properties, we present a dynamic community detection algorithm based on bird flock effect (DCDBFE) to reveal the community structure within networks. The networks are modeled based on ecological systems to imitate the formation process of bird flock and facilitate the analysis of their temporal evolution. DCDBFE is an unsupervised approach, as it does not require labeled data [27,28]. In recent years, unsupervised approaches have gained popularity for their effectiveness in detecting unknown events and addressing real-world challenges. By simulating the behaviour of the bird flock effect, the proposed dynamic community detection algorithm offers several advantages. The main contributions of this work are summarized as follows:

1) **Novelty**: The DCDBFE algorithm leverages the bird flock effect and is based on natural ecological principles to identify accurate modules or communities.
2) **High quality**: The DCDBFE algorithm through the experiment shows the most efficient and accurate value quality of community detection compared to others. Proven in Figs. 6–28.
3) **Parameter free**: The DCDBFE algorithm does not need initial parameter to set.
4) **Stability**: The DCDBFE algorithm shows that the result of quality community detection is very good and consistent. Proven in Figs. 6–28.
5) **Extensive experiments** are carried out on six groups of synthetic networks and six real-world networks to verify the performance of the proposed algorithm.

The following is the structure of the paper. Section 1 presents an introduction, and Section 2 elaborate the related works. Section 3 describes the basic idea, nomenclature used in this study and methodology of DCDBFE in depth. Section 4 present the datasets, evaluation metrics and the performance evaluation of experiment. Section 5 is the results and discussions of this work. Finally, Section 6 is the conclusion. This ends the article with elaborating the limitation or challenges encountered during the extension testing on network datasets and suggests future enhancements to this work.

2. Related work

Dynamic community detection in complex networks has garnered increasing attention recently, emerging as a significant area of study due to its potential to enhance our understanding of how social phenomena evolve over time. Numerous studies have explored the temporal evolution of communities using various approaches [26]. The development of this field has been acknowledged through several comprehensive review papers, which highlight the preferred methodologies based on previous research focused on tracking community detection and evolution in temporal networks.

The existing classification schemes in this field originated from the ideas of Bansal, Bhowmick and Paymal, [29], as well as Hartmann, Kappes, and Wagner, [30], who identified two primary approaches; offline and online clustering. This classification has evolved with contributions from Aynaud et al. [31], Rossetti and Cazabet, [32], and Dakiche et al. [20], who introduced additional perspectives. Aynaud et al., 2019 proposed three categories for detecting communities: two-stage approaches, evolutionary clustering, and coupling graphs. Rossetti and Cazabet identified three themes in dynamic community detection: instant optimal or two-stage approaches, temporal trade-off, and cross-time or incremental approaches. Furthermore, Dakiche et al. [20] classified studies on tracking community evolution in dynamic social networks into four main approaches; independent community detection and matching, dependent community detection, simultaneous community detection on all snapshots, and dynamic community detection on temporal networks.

This paper follows the thematic classifications proposed by Elishi et al., who suggested two main approaches: independent and dependent community detection. Independent community detection, also known as two-stage clustering, treats each network snapshot independently and applies static community detection methods to dynamic cases [33–42]. In this approach, the network's progression is divided into multiple time steps, with communities identified at each step. Each snapshot detects communities independently, which are then matched with those from the previous step based on similarity measures. Common similarity measures used for this matching process include Jaccard, Sorensen, and Salton indices, among others. The advantage of this method is that it can be applied directly to the dataset without requiring modifications. However, a significant drawback is that the results of dynamic community detection can be unstable. This instability may lead to inconsistencies in the detected communities across consecutive time steps, as the approach does not consider prior information.

On the other hand, dependent community detection eliminates the need to match communities, streamlining the community identification process [43–62]. This approach detects communities at a given time (present) based on those identified in a previous period (prior). This method can be further categorized into an evolutionary approach is typically aligned with the cost function and incremental approaches is associated with the direct method.

The evolutionary approach, which considers temporal dependencies between snapshots, is associated with cost function methods and focuses on maintaining smooth transitions in community structures over time by optimizing single or multiple objectives. This approach accounts for the influence of historical community structures on the current one.

For example, FacetNet is a well-known technique of evolutionary approach that employs a hybrid method combining non-negative matrix decomposition and cost function optimization for dynamic community

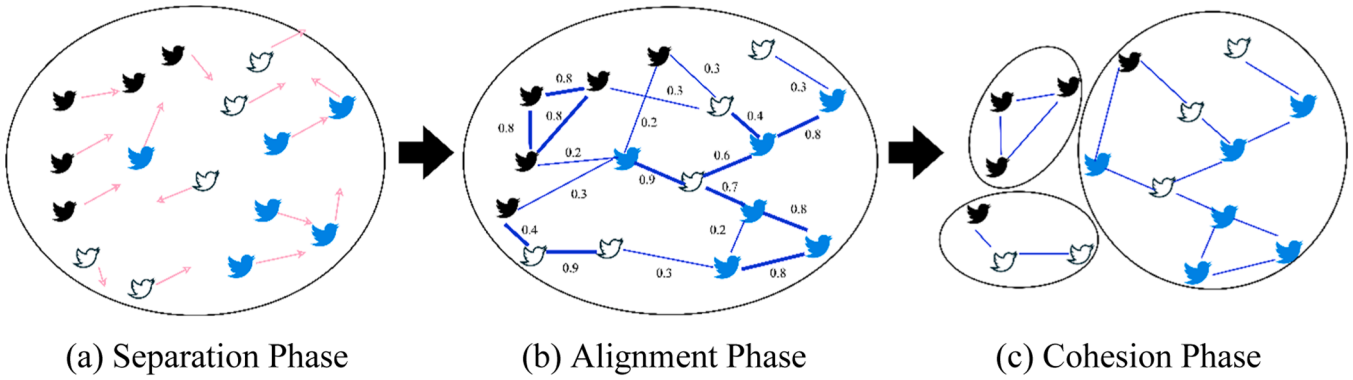


Fig. 1. The formation process of bird flock effect model.

discovery iteratively [63]. Another example is DYNMOGA, which utilizes a multi-objective optimization genetic algorithm and requires manual parameter specification to identify the optimal partitions for dynamic community detection [64]. The goal is to maximize cluster accuracy with respect to current data while minimizing clustering drift between time steps. This method also requires user input for either snapshot or temporal quality preferences. S. Ranjkesh et al. proposed a robust memetic algorithm called RDNMA_NET adapts to changes in network structures over time by combining evolutionary algorithms and local search strategies to maintain robust and accurate community detection across evolving networks [43]. H. Ma et al. proposed HoKT (Higher-Order Knowledge Transfer) which is a multi-objective genetic algorithm that enhances optimization by leveraging knowledge from higher-order patterns and relationships within and across evolving solutions, enabling more effective search and convergence in dynamic and complex problem spaces [44]. D. Li et al. proposed jLMDC (Joint Learning Model for Dynamic Community detection) which is a novel approach for detecting communities in temporal networks that simultaneously performs feature extraction and clustering in a unified optimization framework, capturing both structural and temporal patterns more effectively while reducing computational complexity compared to traditional methods [55].

However, this approach has several significant drawbacks. These optimization methods are time-consuming because they typically optimize objectives iteratively, requiring the complete regrouping of each snapshot even when changes are minimal [64]. Additionally, many optimization algorithms involve parameters that are difficult to set and produce nondeterministic results, often requiring multiple runs to achieve the best outcome. Consequently, these optimization methods are not well-suited for large-scale or rapidly evolving networks.

To address the drawbacks of the previous approach, the incremental approach is associated with the direct method and uses global community discovery to identify structural changes in each network time step incrementally. This direct method reduces time complexity by utilizing information from the previous time step to detect communities in the current step, allowing for rapid adaptation to changes in the network [65]. The incremental approach continuously updates community structures as new nodes or edges appear, directly utilizing information from the previous time step to detect communities in the current time step.

For instance, IncNSA identifies dynamic communities by examining vertex variations. It analyzes only the community affiliations of a subset of vertices, specifically those that are new or active from the previous snapshot, as the network evolves [66]. Similarly, DCDID employs an information dynamics model to simulate node communication and leverages batch processing to incrementally detect communities in dynamic networks, improving detection by filtering out unmodified subgraphs [67]. A related process is seen in DCDME, which is based on the sociological concept known as the Matthew effect, where "the rich get richer and the poor get poorer" forms the foundation of the algorithm [68]. However, the DCDME algorithm may not be suitable for dynamic networks with a low clustering coefficient, as this could indicate that the network changes too rapidly for stable communities to form or that the network's structure is decentralized. As a result, DCDME might struggle with variability and complexity, potentially leading to less stable community detection.

Furthermore, many incremental approaches suffer from diminishing results over time due to the accumulation of incorrect historical information, despite their contributions to efficiency. To address these challenges, our objective is to propose a stable and parameter-free incremental dynamic community detection algorithm that consistently identifies high-quality communities. In this paper, we demonstrate the embedding of the behavior of the natural law of the bird flock effect in the dynamic community detection algorithm.

3. Methodology

Dynamic community detection algorithms in complex networks have been significantly influenced by natural phenomena, drawing on the self-organizing and self-adapting behaviours observed in nature to create more effective methods for identifying community structures [69]. A notable example is the application of swarm intelligence principles, inspired by the collective behaviour of animals like bees, swarms, birds, fish, and insects [70]. In natural systems, individuals operate by adhering to straightforward behavioural rules, such as aligning their speed with that of their neighbours, maintaining proximity, and avoiding collisions [71]. These seemingly simple rules guide each individual's actions, yet collectively, they result in highly complex, adaptive, and coordinated group behaviour [72]. Remarkably, this sophisticated group dynamics emerges without any centralized control, highlighting the power of simple interactions in producing intricate patterns of organization [73].

This study was interested in exploring the intersection of mathematics, and biological phenomena, particularly on bird flocking behaviour. This fascinating area of study, known as bio-inspired computing, involves deriving mathematical models from natural systems and applying them to solve complex problems. The unique aspects of bird flocking were used to inspire a community detection concept, where the three fundamental rules of the bird flock effect were integrated into the graph analysis process to identify modules or communities in each stage of a continuously dynamic network. In static networks, community detection algorithms commonly rely on basic metrics like centrality and node similarity [74]. However, when dealing with dynamic networks, these metrics must be reformulated and adapted to account for the changes over time [75]. This adaptation ensures that the community detection process remains accurate and effective as the network evolves. In this section, we provide a detailed explanation of the DCDBFE algorithm, including its basic idea, relevant

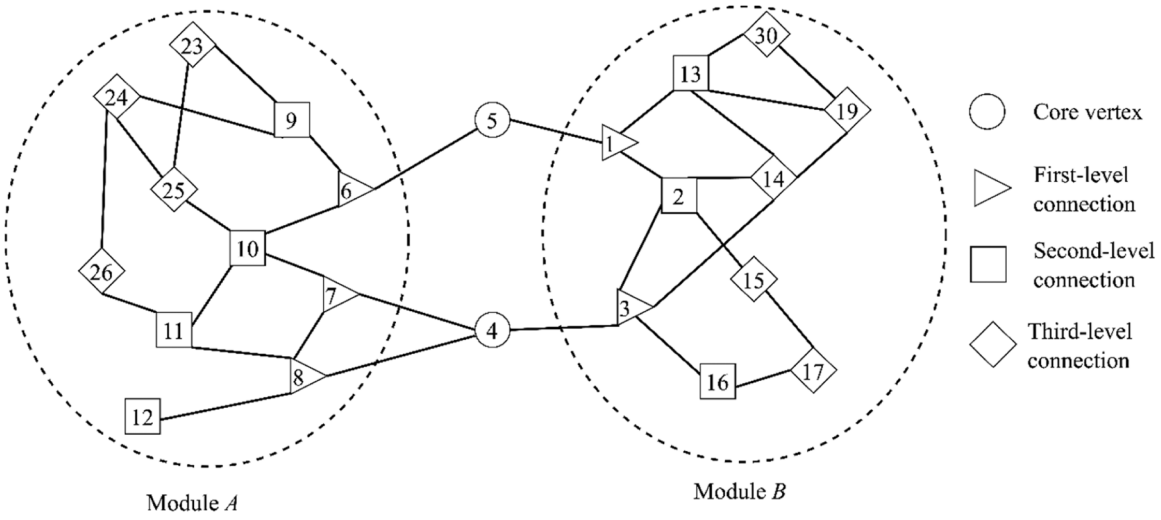


Fig. 2. A toy network of two distinct modules, *A* and *B*.

definitions, model and framework of DCDBFE.

3.1. Basic idea

In the context of network dynamics and community detection, an ecosystem can be viewed as a complex network of interactions, where the bird flock effect model draws inspiration from the coordinated movements of birds in a flock. This model leverages specific characteristics of bird flocking, such as self-organization, collective decision-making, and adaptive behavior, to represent how entities in a network interact and influence each other [76]. From an ecosystem perspective, bird flocks exhibit emergent behavior where individual birds follow simple rules based on their local interactions, leading to complex global patterns. This is analogous to dynamic community detection in networks, where communities emerge based on the interactions between vertices rather than predefined structures. This bio-inspired approach enhances the efficiency and accuracy of community detection in evolving networks [77].

In the physical world, people are attracted by activities according to their interests, a phenomenon similar to the bird flock effect in the sky. The concept behind the formation process of bird flock effect, where the algorithm is designed is based on three basic rules from Craig Reynolds in 1987 [78,79]. Fig. 1 illustrates the formation process of bird flock effect model through three fundamental rules: Separation, Alignment, and Cohesion. Each rule corresponds to a phase in the dynamic behaviour of entities within a network, and together, they help in understanding how communities form and evolve.

In the Separation phase, depicted in Fig. 1**(a)**, each bird (or network entity) maintains a certain distance from its neighbors to prevent overcrowding. This is visually represented by the arrows pointing away from one another, ensuring that the entities do not cluster too closely at this initial stage. The concept here is similar to individuals in a social network who begin by exploring their surroundings independently, without forming any close ties. For example, imagine a group of people at a conference where each person starts by moving around independently, avoiding clustering too early, thus exploring different topics or discussions without being influenced by others.

During the Alignment phase, depicted in Fig. 1**(b)**, the entities begin to align themselves with others based on commonalities or shared behaviors. This phase is illustrated by the birds adjusting their positions to align with their neighbors, which is quantified by the numbers on the lines (e.g., 0.2, 0.4, 0.8, 0.9, etc.). These numbers represent the degree of alignment or similarity between entities. For instance, in a workplace setting, colleagues may begin to work more closely together based on shared projects or goals, aligning their efforts to be more cohesive as a team.

Finally, in the Cohesion phase shown in Fig. 1**(c)**, the entities, which were once separated, come together to form tightly knit clusters or communities. The birds are now grouped into distinct clusters. The connections within each group are stronger, indicating a high level of interaction and mutual influence. For example, in an online community, this would be the stage where users with similar interests form distinct groups, such as forums or chat groups, where they engage more deeply with one another, sharing ideas and collaborating closely.

3.2. Relevant definitions

Let a graph, $G = (V, E)$ be the given network, which are undirected and unweighted graph with the set of vertices, V and the set of edges, E. We formulated several symbols used in the proposed algorithm based on the definitions. Fig. 2 illustrates a toy network to explain each definition in detail. This network consists of 23 vertices, divided into two distinct modules, *A* and *B*. The vertices are organized into three levels based on their connectivity. The vertices within each module are represented by different shapes, indicating their varying levels of connectivity and their roles in bridging the two modules. This layered structure of level connection—with circle as core vertex, triangles representing the highest connectivity and influence, squares indicating significant internal connections, and diamonds showing peripheral connections—visually explains how different nodes contribute to the network's overall organization and module attractiveness.

Definition 1. : Resource allocation.

Resource allocation is a similarity measures introduced by Zhou et al. in 2009, as represented in Eq. (1) as follows [80].

$$simRA_{uv} = \sum_{i \in CN_{uv}} \frac{1}{D_i} \tag{1}$$

where $simRA_{uv}$ is a similarity measure between two vertices u and v in the network, CN_{uv} represents the common neighbors between vertices u and v, $\sum$is the summation symbol indicating the summation over all the inverse of the degree of each common neighbor, and D_i denotes the degree of vertex i, which is the number of edges connected to i.

The edge attraction between vertices reflects their similarity, calculated using the

$simRA_{uv}$measure, which prioritizes giving resources to high-degree vertices, leading to accurate but less diverse spreading. Initially, each vertex has its own position based on similarity, and as the bird flock

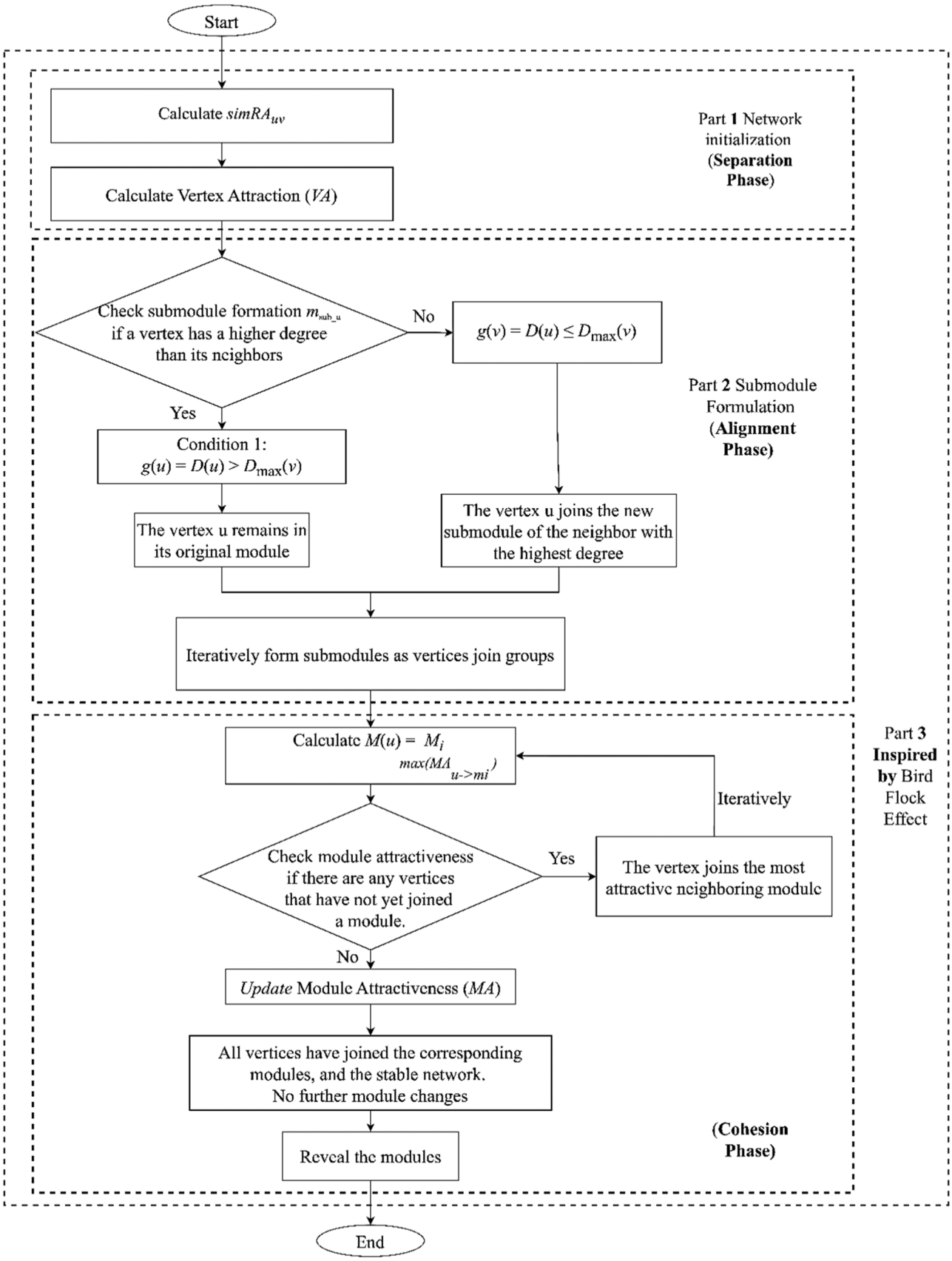


Fig. 3. Flowchart of community detection inspired by bird flock effect.

spreads during the alignment phase, modules are naturally revealed by these similarities. This approach is particularly effective in network analysis, as it shows that neighbors with fewer connections significantly impact vertex similarity, making it a valuable method for uncovering structural patterns. Despite being simple, local methods like $simRA_{uv}$ are preferred for their efficiency and effectiveness in comparative studies.

From Fig. 2, we can calculate the $simRA_{uv}$ as in Eq. (1) between each vertex u and v in the graph. For example, $simRA_{4,7}$ is the similarity between vertex 4 and 7. The process begins by identifying the neighbors of both vertices. Vertex 4 is connected to vertex 3, vertex 7, and vertex 8, while vertex 7 is connected to vertex 10, vertex 8, and vertex 4. The next step is to find the common neighbors between these two vertices, which involves identifying the vertices that both shares. In this case, the common neighbor between vertex 4 and vertex 7 is vertex 8. Once the common neighbor is identified, the $simRA_{4,7}$ is calculated by summing the inverse of the degrees of the common neighbors. The degree of a vertex is the number of edges connected to it. Vertex 8, the common neighbor, has a degree of 4, meaning it is connected to four other vertices in the graph. Using the Eq. (1), the inverse of vertex 8's degree is calculated as $simRA_{4,7} = \sum_{D_8 \in CN_{4,7}} \frac{1}{D_8} = \frac{1}{4} = 0.25$. This score reflects the structural similarity between vertex 4 and vertex 7 based on their shared neighbor, and the degree of that neighbor.

A high similarity score implies that two vertices have strong structural cohesion, which is crucial for forming stable communities. This same similarity calculation applies uniformly across all vertex pairs in the graph, ensuring a consistent metric for determining attraction between vertices. In dynamic community detection, these scores serve as a key factor in module attraction, continuously influencing how vertices are grouped and how communities evolve over time. The evolving similarity between vertices allows the algorithm to dynamically adjust community membership, reflecting the current state of the network.

Definition 2. : Vertex attractiveness

In a graph, $G = (V,E)$, the attractiveness of vertex v to u is defined as in Eq. (2).

$$VA_{v \to u} = simRA_{uv} * D_u \tag{2}$$

where VA is a vertex attractiveness and it is the product of the $simRA_{uv}$similarity measure between vertex $u$and v, and D_u the degree of vertex u. This suggests that the vertex exerts a local influence, and a higher value indicates that the vertex is more likely to attract or move towards a similar vertex.

Similarly, just as the first bird in a flock can attract others to join and fly together, we introduce a new concept called module attractiveness. This concept explains how a module can attract a vertex based on how well that vertex is connected within the module. The stronger the connections a vertex has within a module, the more attractive that module becomes to the vertex.

Definition 3. : Module attractiveness

In a graph, $G = (V,E)$, the concept of module attractiveness (MA), as defined in Eq. (3).

$$MA_{v \to A} = \underbrace{D_A(u)^2}_{First-Level\ Connection} + \underbrace{\sum_{v \in N(u), v \in A} D_A(v)}_{Second-Level\ Connection} + \underbrace{\sum_{w \in N(v), w \in A} D_A(w)}_{Third-Level\ Connection} \tag{3}$$

where $MA_{v \to A}$ represents the module attractiveness of a vertex v to a module A by considering the three components: (i) $D_A(u)$ represents the degree of vertex u in the first-level connection, indicating its direct connections of core vertex within the module. (ii) $D_A(v)$ represents the internal degree of vertex v in module A at the second-level connection. According to the connection between vertices and distinct modules, the $MA_{v \to A}$ can be divided into two conditions: one is $D_A(v)$ equality, and the other is $D_A(v)$ inequality. (iii) $D_A(w)$ represents the internal degree of vertex w is a neighbor of the neighbors of u in module A at the third-level connection.

We can clarify Eq. (3) for calculating the module attractiveness follows from first, second and third-level of connectivity in the modules. There are two conditions**. (i) First condition, when a vertex is equal within two modules at the first-level connection.** For example, for vertex 5 in Fig. 2, $D_A(5) = D_B(5) = 1$, it is not easy to move which module, A or B, are more attractive to vertex 5. Consequently, the impact of indirect neighbors appears in vertex 5 must be attentively considered. In module A, vertex 6 is connected with vertex 5, and the degree of vertex 6 is 2, so $D_A(6) = 2$. However, the problem is internal degree at second-level connection is still equal $D_A(6) = D_B(1) = 2$. Then, we consider the third-level of connection, where vertex 9 and 10 are connected with vertex 6 in module A and the degree of vertex 9 is 2 and the degree of vertex 10 is 3, total is 5, $D_A(9) + D_A(10) = 2 + 3 = 5$. So according to Eq. (3), the attraction of module A to vertex 5 can be calculated as $MA_{5 \to A} = D_A(5)^2 + [D_A(6)] + [D_A(9) + D_A(10)] = 1^2 + 2 + 5 = 8$. In module B, since the degree of vertex 1 is 2, equal to module A, we consider the third-level connection, where vertex 13 and 2 are connected with vertex 1 in module B and the degree of vertex 13 is 3 and the degree of vertex 2 is 3, total is 6, $D_B(13) + D_B(2) = 3 + 3 = 6$. So according to Eq. (3), the attraction of vertex 5 to module A can be calculated as $MA_{5 \to A} = D_A(5)^2 + [D_A(6)] + [D_A(9) + D_A(10)] = 1^2 + 2 + 5 = 8$ and the attraction of vertex 5 to module B can be calculated as $MA_{5 \to B} = D_B(5)^2 + [D_B(1)] + [D_B(13) + D_B(2)] = 1^2 + 2 + 6 = 9$. Because module B is more attractive to Vertex 5, Vertex 5 is more likely to join module B.

Second condition, (ii) when a vertex is inequal within two modules at the first-level connection. For example, for vertex 4 in Fig. 2, the degree of vertex 4 in module A is $D_A(4) = 2$, while the degree of vertex 4 in module B is $D_B(4) = 1$. From this value, module A is more attractive to vertex 4, then vertex 4 is more likely to join module A. To clarify using Eq. (3), the module attraction of vertex 4 to module A is $MA_{4 \to A} = D_A(4)^2 + [D_A(7) + D_A(8)] + [D_A(10) + D_A(11) + D_A(12)] = 2^2 + 3 + 5 = 12$, while the module attraction of vertex 4 to module B is$MA_{4 \to B} = D_B(4)^2 + [D_B(3)] + [D_B(2) + D_B(14) + D_B(16)] = 1^2 + 3 + 5 = 9$.

In conclusion, the module attractiveness analysis for vertex 5 reveals that it has a stronger connection to module B, with a total attractiveness score of 9, compared to module A, which has a score of 8. This indicates that vertex 5 is more closely integrated within module B, with higher connectivity across all three layers, suggesting that it plays a more central role in module B's structure. Meanwhile, the module attractiveness analysis for vertex 4 reveals that it has a stronger connection to module A, with a total attractiveness score of 12, compared to module B, which has a score of 9. The calculation considers both direct connections and indirect connections through subsequent layers, demonstrating that vertex 5's and 4's influence.

3.3. Bird flock effect model

After introducing some symbols and definitions, we started building the bird flock effect model. The flowchart of community detection inspired by bird flock effect contains three parts: the network initialization, submodules formulation, and inspired by bird flock effect, as shown in Fig. 3.

The explanation of flowchart of community detection inspired by bird flock effect is in below:

Part 1: **Network Initialization**. The process begins with the network initialization, where the similarity measure ($simRA_{uv}$) between vertices is calculated, providing a foundation for understanding the relationships within the network. This is followed by calculating the vertex attraction, which determines the potential of a vertex to either remain within its current submodule or join a new one. This part corresponds to the separation phase in the bird flock effect, where individual vertices (analogous to birds) begin to separate from each other based on their

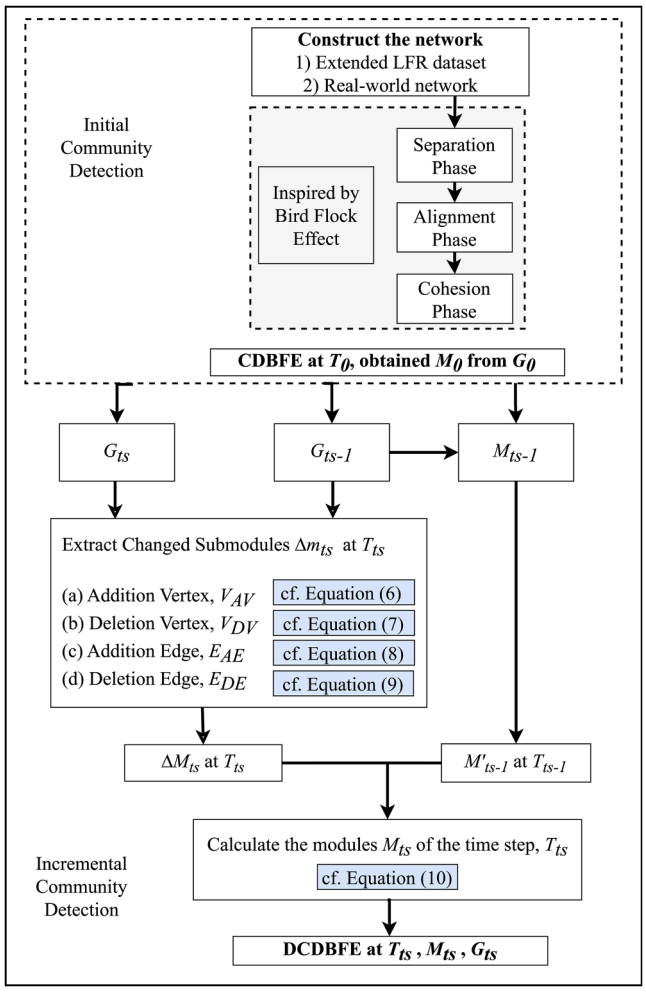


Fig. 4. Framework of DCDBFE.

unique characteristics using degree of vertices.

Part 2: **Submodules Formation**. The flow then transitions into submodule formulation, which is associated with the alignment phase. Bird flocking in the sky is not dependent and each bird has their own characteristics and types. Therefore, the differences among the birds show the attractiveness of themselves to others and we call vertex attractiveness (cf. Eq. (2)). The more attraction the birds have, the other birds will join the module or group automatically. In simple terms, every vertex selects its neighbours with the strongest attraction according to two conditions as in Eq. (4):

$$m_{sub\_u} = \begin{cases} g(u), & D(u) > D_{\max}(v), v \in N(u) \\ \underset{\max(MA_{(u\to v)})}{g(v)}, & D(u) \le D_{\max}(v), v \in N(u) \end{cases} \tag{4}$$

where $g(u)$ represents the original module to which vertex u belongs, $d(u)$ is the degree of vertex u, and m_{sub_u} denotes as the submodule. If vertex u has more attraction than its neighbors, then it has more degree and vertex u will remain in the original module, $m(u)$. Similarly, if neighbor vertex v has more degree, then vertex u will join the module in which vertex v belongs $g(v)$. During this phase, the algorithm checks whether each vertex has a higher degree of connectivity than its neighbors iteratively. If a vertex's degree is higher, it remains in its original module, but if it is lower, the vertex joins the submodule of the neighboring vertex with the highest degree. This decision-making process mirrors the behavior of birds aligning themselves within a flock based on the direction and movement of those closest to them.

Part 3: **Inspired by Bird flock effect**. After obtaining the submodules, an increasing number of vertices are attracted to different submodules according to Eq. (5).

$$M(u) = \underset{\max\left(MA_{u\to m_i}\right)}{M_i} \tag{5}$$

where $M(u)$ is the module that vertex u will join, M_i is the neighborhood module of vertex u is evaluating, and $MA_{u\to m_i}$ denotes the maximum module attractiveness that vertex u experiences toward any of the available submodules m_i. The structure and attractiveness of submodules may change when a vertex joins it by variations of modules formation. The update process culminates by iteratively refining and stabilizing the network, ensuring all vertices are grouped into the most attractive neighboring modules, resulting in a cohesive and stable network structure. The bird flock effect metaphorically unites these phases, illustrating the dynamic adjustment and final alignment of all vertices into reveal the modules.

This process continues if there are vertices in the network that have not yet joined a module. Over multiple cycles, each vertex adjusts its attractiveness based on the evolving network topology. As vertices join different modules, the network structure gradually stabilizes, reaching a point where vertices no longer change their module assignments. This stabilization ensures that the community structure is cohesive and well-defined, akin to a flock of birds moving together in harmony once their positions within the group are settled.

3.4. DCDBFE framework

We used the bird flock effect to embed in dynamic community detecting algorithm. Here, we provide the step by step to understand the specific iterative approach employed by DCDBFE for detecting communities in each network snapshot. Fig. 4 illustrates the whole framework of proposed algorithm. The process begins with constructing the network using either an extended LFR dataset or real-world networks. The proposed dynamic community detection approach consists of two main components: initial and incremental community detection, each designed to identify and adapt the community structure within evolving networks.

The detailed of framework of DCDBFE algorithm:

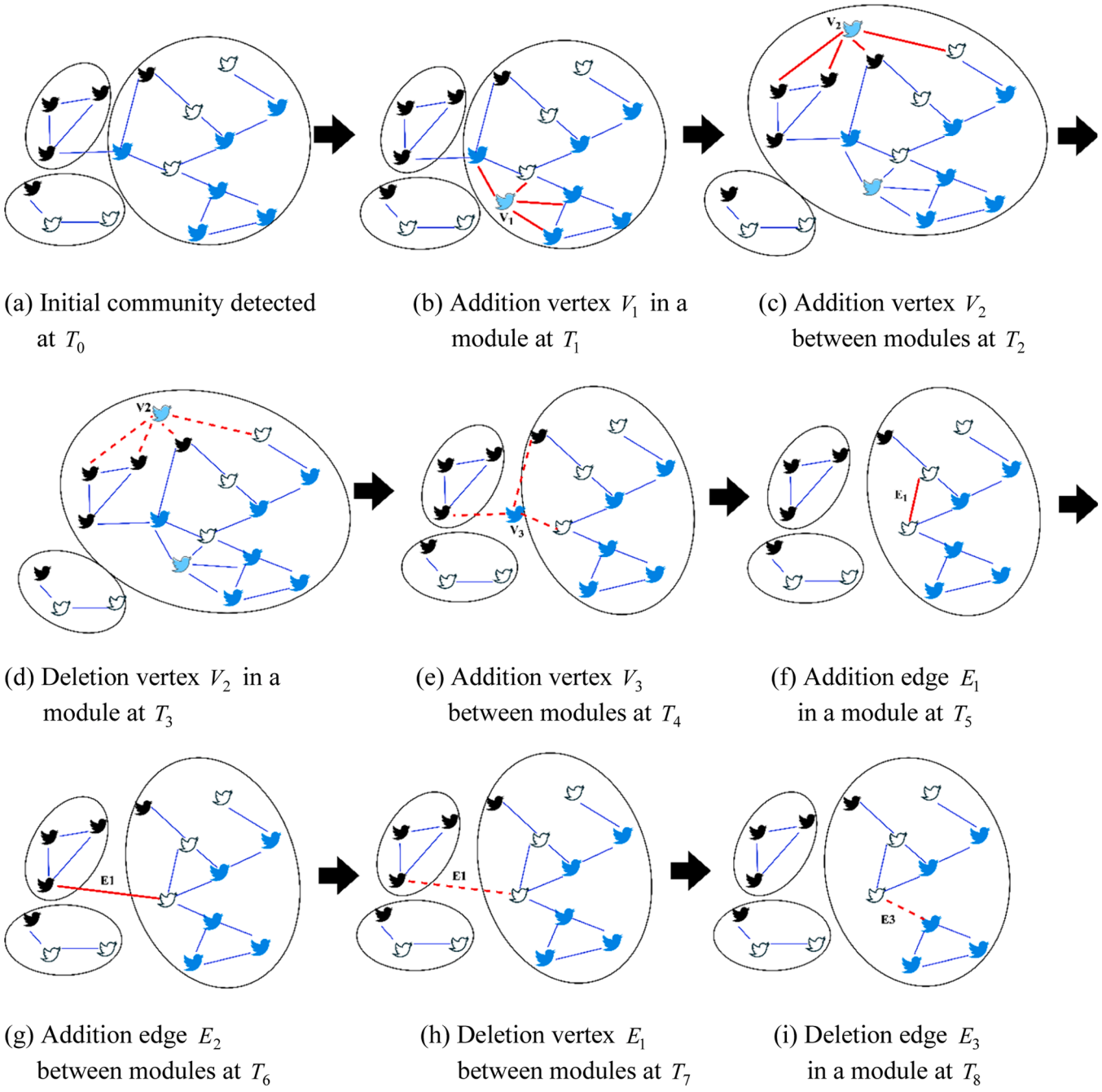
Initial community detection. This initial community detection step is labelled as CDBFE. This component is guided by a model inspired by the bird flock effect, which involves three key phases: Separation, Alignment, and Cohesion. This approach obtained initial modules M_0 at time step T_0 from the whole network G_0.

By studying the model, researchers can now guess how these modules will react to changes in their surroundings. According to the three phases of bird flock effect model mentioned earlier, Fig. 5 demonstrates the process of incremental community detection based on the bird flock effect from time steps T_1 until T_8. This figure illustrates how communities evolve over time within a network as vertices, V and edges, E are added or removed, thereby affecting the overall community structure.

Incremental community detection. This incremental community detection step, labeled as DCDBFE, will repeat the process of community detection at each time step after the initial community detection. The detailed process of incremental community detection is provided below:

(1) **Extract Changed Submodules,** Δm_{ts}. In contrast to a static network, the vertices and edges in a dynamic network can vary over time, leading to ongoing changes in the community structure across different time steps. These changes include adding or deleting vertices and edges, which are handled by specific algorithms.

(a) **Addition Vertex Event,** V_{AV}. When a new vertex and its associated edges are added to the current time step, G_{ts}, rather than the previous time step, G_{ts-1} it is referred to as a vertex addition



(a) Initial community detected at T_0

(b) Addition vertex V_1 in a module at T_1

(c) Addition vertex V_2 between modules at T_2

(d) Deletion vertex V_2 in a module at T_3

(e) Addition vertex V_3 between modules at T_4

(f) Addition edge E_1 in a module at T_5

(g) Addition edge E_2 between modules at T_6

(h) Deletion vertex E_1 between modules at T_7

(i) Deletion edge E_3 in a module at T_8

Fig. 5. Demonstration of the process of incremental community detection based on bird flock effect from time steps T_1 until T_8.

event. Let's denote the addition vertex as V_{AV}. We define this formally in Eq. (6):

$$V_{AV} = \{u | u \in V_{ts}, u \notin V_{ts-1}\} \tag{6}$$

where V_{ts} and V_{ts-1} denote the vertices in networks G_{ts} and G_{ts-1}, respectively. The addition of new vertices can lead to changes in the community structure within the network, and two possible scenarios may arise: (i) the addition of a vertex within a single module, and (ii) the addition of a vertex between two modules. Fig. 5(b) demonstrates when a vertex V_1 is introduced within a module at T_1, the community structure remains unchanged. The inclusion of additional vertices within a module enhances interconnectivity, consequently increasing the connection density. However, the overall number of modules remains unchanged, and the newly inserted vertex must be partitioned within the community. Quick Community Adaptation (QCA) provides theoretical proof for this [81]. On the other hand, Fig. 5(c) shows the addition of a vertex V_2 between two modules at T_2 has the potential to induce modifications in the community structure. In this scenario, it is necessary to document the newly added vertex along with its corresponding edges and the modules it connects to. These elements are then incorporated into the submodules, denoted as Δm_{ts}.

Deletion Vertex Event, V_{DV}. When a vertex and its associated edges are eliminated from the current time step G_{ts}, rather than the previous time step G_{ts-1}, it is referred to as a deletion vertex event. Deletion vertex is represented by V_{DV} as shown in Eq. (7):

$$V_{DV} = \{u | u \notin V_{ts}, u \in V_{ts-1}\} \tag{7}$$

where V_{ts} and V_{ts-1} denote the vertices in networks G_{ts} and G_{ts-1}, respectively. Fig. 5(d) and Fig. 5(e) show a deletion edge within single module and between module, respectively. We observe that deletion vertex of vertex V_2 within a module at T_3 and vertex V_3 between modules at T_4 leads to changes in the community structure. When deleting a vertex, its associated edges must also be removed. The connected modules are then added to the submodules Δm_{ts}.

Addition Edge Event, E_{AE}. Similarly, adding new edges to the current time step G_{ts}, as opposed to the previous time step G_{ts-1} is referred to as an edge addition event. This is defined formally in Eq. (8):

$$E_{AE} = \{e | e \in E_{ts}, e \notin E_{ts-1}\} \tag{8}$$

where E_{ts} and E_{ts-1} denote the edges of networks G_{ts} and G_{ts-1}, respectively. Adding edge E_1 inside a module at T_5 will not change the structure of the community; instead, it will increase the density of edges within the module as shown in Fig. 5(f). However, Fig. 5(g) show changes in the community structure may result from the addition of edges E_1 between modules at T_6. Therefore, there is no need to re-detect the module for newly added edges.

Deletion Edge Event, E_{DE}. An edge deletion event occurs when an edge is eliminated from the previous time step G_{ts-1}, and instead appears in the current time step G_{ts}. The deleted edges are represented by E_{DE} as shown in Eq. (9):

$$E_{DE} = \{e | e \notin E_{ts}, e \in E_{ts-1}\} \tag{9}$$

where E_{ts} and E_{ts-1} denote the edges of networks G_{ts} and G_{ts-1}, respectively. A community's structure may change as a result of edge deletion. However, because there are typically few connections between modules, eliminating the edges E_1 between them at T_7 often has little effect on their structure. In contrast, Fig. 5(i) shows that removing edge E_3 within a module at T_8 might cause a split or division. Consequently, it is necessary to compute the deleted edges and the related modules in the submodules Δm_{ts}.

(2) **Compute Changed Modules ΔM_{ts}.** By adding and deleting vertices and edges from the network, we may compute the submodules Δm_{ts}, which may change the community structure. Next, we use the bird flock effect model to find modules in the submodules to complete Algorithm 1 (CDBFE).

(3) **Compute Unchanged Modules, M'_{ts-1}.** The modules M_{ts-1} can be obtained at time steps G_{ts-1}. Additionally, the modules whose structures are most likely to change, as indicated by the changed submodules Δm_{ts}, can be computed. By computing the difference between the two sets, the unchanged modules M'_{ts-1} at current time step G_{ts} can be obtained.

(4) **Compute all Modules, M_{ts} of network G_{ts}.** Modules in time step, M_{ts} consist of unchanged modules M'_{ts-1} and changed modules ΔM_{ts}. Let M_{ts} represent the modules of network G_{ts} at time step, ts, which is defined in Eq. (10):

$$M_{ts} = M'_{ts-1} + \Delta M_{ts} \tag{10}$$

In this study, dynamic community detection based on bird flock effect model uses repeated event processing such as adding and removing vertices and edges, to produce an accurate and efficient community. This approach is supported by some researchers with a good understanding of community detection [66–68,82].

3.5. DCDBFE algorithm

In this section, we introduce the DCDBFE algorithm, an extension of the CDBFE algorithm, which is proposed to find communities in dynamic networks. While the CDBFE algorithm is used for static networks, we have improved and modified it to be embedded in a dynamic setting. The DCDBFE consists of three steps: (1) Initialization community detection, (2) Change submodules, and (3) Dynamic community detection. The pseudo code is presented as Algorithm 16, and we will explain the steps involved in these algorithms.

Initialization Community Detection. The bird flock effect model is applied to simulate the network's behaviour in detecting communities, drawing inspiration from how birds flock together based on their positions and attractions to one another. Initially, each vertex is treated as a separate module, with its degree representing its resources. Submodules are then formed by drawing vertices towards their neighbouring vertices, as described in Eq. (4). The bird flock effect is further simulated through an iterative process, following the procedures outlined in Eqs. (3) and (5). This iterative approach is crucial in Phase 3, where module attractiveness is calculated using Eq. (3). The process involves evaluating each vertex's attraction to its neighbours, extending up to the third layer. Eq. (5) models the structure and attractiveness of submodules for each vertex. The network's module structure reaches stability when all vertices have joined different modules, and no further changes occur.

If the community structure of the network is known at advance, ground truth measure is used [27]. The most popular measurement for evaluating the quality of community detected uses the normal mutual index (*NMI*) [83] and adjusted random index (*ARI*) [84] as indicator. Due to topologically driven influences, the network eventually converges as it changes over time, allowing us to obtain the optimal module partition.

For example, flock control is distributed to each bird by having them follow their neighbors' actions. The behavior of separation requires birds to avoid their neighbors, with the force of separation increasing as they get closer. Due to this separation rule, each bird in the flock creates space around itself. More space gives a bird greater maneuverability, reducing the risk of collisions. A balanced flock is achieved when all separation forces equal zero, meaning each bird is sufficiently distanced from its neighbors. Birds should coordinate their direction based on the similarity value, with greater attraction balancing the flock as birds maintain the same direction, which also maintains separation and cohesion forces. Consequently, we employed the CDBFE algorithm to determine the original network's community structure at a given time step, T_0. Algorithm 1 provides the pseudocode for the CDBFE algorithm.

Algorithm 1. CDBFE

Input: $G(t) = (V, E)$

Output: Final set of modules, $M(v)$

1://**Initialize communities of each vertex in changed submodule**
2: **for** each vertex v in V **do**
3: $M(v) = N(v)$ the number vertex of v
4: **end for**
5://**Compute the submodules**
6: **for** each vertex v in V do
7: **for** each vertex u in $N(v)$ **do**
8: compute the $simRA_{uv}$ **similarity measure** using Eq. (1)
9: compute the **Vertex Attractiveness** using Eq. (2)
10: Imax = 0
16: **while** Flag do
11: compute submodules using Eq. (4)
12: **end for**
13://**Simulate the bird flock effect**
14: Flag = TRUE
15: NMImax = 0
16: **while** Flag do
17: **for** each node v in V **do**
18: **for** each node u in $N(v)$ **do**
19: compute the **Module Attractiveness** using Eq. (3)
20: **end for**
21: simulate the bird flock effect using Eq. (5)
22: **end for**
23: S = *NMI* (C; true clusters)
24: **if** S > *NMI* max **then**
25: *NMI* max = S
25: **else**
26: Flag = FALSE
27: **end if**
28: **end while**
29://**return modules** $M(v)$ **in changed submodule**

- **Changed Submodules.** Considering actions that might lead to changes in the community structure, we divided network events into four categories—addition vertex, deletion vertex, addition edge, and deletion edge. Algorithms 2–5 show the detailed process that each event returns a changeable submodule, ΔM_{ts}.

Algorithm 2. Addition Vertex Event.

Input: V_{AV}, G_{ts}, M_{ts-1}
Output: Changed submodules, Δm_{ts}

for each vertex $v \in V_{AV}$ **do**
  if $N(v)$ in the same module $M_{N(v)}$ **then**
    $M_{N(v)} \leftarrow v$
  else
    $\Delta m \leftarrow v$
    for each vertex $u \in N(v)$ **do**
      $\Delta m_{ts} \leftarrow M(u)$
    end for
  end if
end for

Algorithm 3. Deletion Vertex Event.

Input: V_{DV}, G_{ts}, M_{ts-1}
Output: Changed submodules, Δm_{ts}

for each vertex $v \in V_{DV}$ **do**
  for each vertex $u \in N(v)$ **do**
    $\Delta m_{ts} \leftarrow M(u)$
  end if
end for

Algorithm 4. Addition Edge Event

Input: E_{AE}, M_{ts-1}
Output: Changed submodules, Δm_{ts}

for each edge $e \in E_{AE}$ **do**
  if $M(u) \neq M(v)$ **then**
    $\Delta m_{ts} \leftarrow M(u)$
    $\Delta m_{ts} \leftarrow M(v)$
  end if
end for

Algorithm 5. Deletion Edge Event

Input: E_{DE}, M_{ts-1}
Output: Changed submodules, Δm_{ts}

1: **for** each edge $e \in E_{DE}$ **do**
2: **if** $M(u) = M(v)$ **then**
3: $\Delta m_{ts} \leftarrow M(u)$
4: **end if**
5: **end for**

- **Dynamic Community Detection.** Over time, we identified the modules by constructing upon the obtained changed submodules using the bird flock effect model. The DCDBFE algorithm outlined in Algorithm 6 starts with an initial community detection at time step T_0. In Step 1, the algorithm receives the dynamic network, represented as a series of snapshots $DynG_t = \{G_0, G_1, \ldots, G_{ts}\}$, as input. Step 2 involves applying the CDBFE (Algorithm 1) to the initial network snapshot G_0, which results in the first set of modules M_0. This set M_0 serves as the baseline for further dynamic community detection in subsequent time steps. The dynamic phase of the algorithm begins at Step 3, where a loop iterates through each time steps starting from $ts = 1$.

Within each iteration, Step 5 computes the sets of vertices and edges that have been added or deleted at the current time step, denoted as V_{AV}, V_{DV}, E_{AE}, E_{DE}. In Steps 6 and 7, the algorithm updates the community structure by adding and deleting vertices using the respective Eqs. (6 and 7). Steps 8 and 9 involve updating the edges in the community structure based on the added and deleted edges using Eq. (8 and 9). In Step 10, the algorithm calculates the modules that remain unchanged from the previous time step. Step 11 then applies the *CDBFE* function to the modified network snapshot ΔG_{ts}, producing an updated community structure M_{ts}. Finally, Step 12 computes the complete community structure for the current time step using Eq. 10, and the loop continues until all time steps are processed, concluding in Step 13.

Algorithm 6. DCDBFE.

Input: $DynG_t = \{G_0, G_1, \ldots, G_{ts}\}$
Output: $DynM_t = \{M_0, M_1, \ldots, M_{ts}\}$

1: //Initial community detection at time step, T_0
2: $M_0 = CDBFE(G_0)$ # using Algorithm 1 (CDBFE)
3: //Dynamic community detection at time step, ts
4: **for** $ts = 1$ to **do**
5: Compute V_{AV}, V_{DV}, E_{AE}, E_{DE}
6: $\Delta m_{ts} \leftarrow$ Addition Vertex $(V_{AV}, G_{ts}, M_{ts-1})$ using Eq. (6)
7: $\Delta m_{ts} \leftarrow$ Deletion Vertex $(V_{DV}, G_{ts-1}, M_{ts-1})$ using Eq. (7)
8: $\Delta m_{ts} \leftarrow$ Addition Edge (E_{AE}, M_{ts-1}) using Eq. (8)
9: $\Delta m_{ts} \leftarrow$ Deletion Edge (E_{DE}, M_{ts-1}) using Eq. (9)
10: Compute the unchanged modules, M'_{ts-1}
11: $M_{ts} \leftarrow CDBFE(\Delta G_{ts})$
12: Compute M_{ts} using Eq. (10)
13: **end for**

4. Experiments

In this study, we evaluated the performance of our dynamic community detection algorithm, DCDBFE, on both synthetic and real-world networks commonly used in the study of community detection over time. The results were compared with those of several popular dynamic community detection methods, including FaceNet, QCA, DCDID, DYNMOGA, IncNSA, and DCDME. A brief description of five of these methods is provided below.

FacetNet (2009) is a dynamic community detection technique that employs non-negative matrix decomposition and cost function optimization, using Kullback-Leibler (KL) divergence to minimize snapshot and temporal costs in evolutionary clustering. However, its effectiveness is limited by the need for parameter settings, such as specifying the number of communities in the network [85].

QCA (2011) is a modularity optimization method designed for dynamic online social networks, adapting to structural changes to identify new communities efficiently with minimal computational resources. QCA may overlook certain structural properties of communities and struggle with scalability in extremely large networks, leading to increased complexity and longer processing times [81].

DYNMOGA (2014) utilizes a multi-objective genetic algorithm to detect communities in dynamic networks, aiming to maximize cluster accuracy while minimizing drift between time steps. Despite its effectiveness, DYNMOGA requires manual parameter tuning and can face challenges with high spatial and temporal complexity, especially in networks with rapidly changing dynamics [64].

DCDID (2019) is an incremental approach that uses an information dynamics model and batch processing to incrementally detect communities in dynamic networks, improving accuracy by filtering unmodified subgraphs. However, it faces challenges with parameter tuning, high time complexity, and reduced detection accuracy as the network's complexity increases, leading to higher computational costs over time [67].

IncNSA (2020) is an incremental approach that identifies dynamic communities by analyzing node variations, focusing on new or active nodes from previous snapshots to detect high-quality community structures. While it excels in identifying community structures, it may not efficiently address all node affiliations, as it primarily targets newborn or active nodes, potentially overlooking others in the network [66].

DCDME (2022) is an incremental approach that based on the Matthew effect, designed to detect community structures in dynamic networks by incrementally processing each network snapshot. While it performs well without requiring settings and scales effectively, DCDME may struggle with networks that have a low clustering coefficient, leading to challenges in reliably detecting communities in rapidly changing or decentralized structures [68].

Table 1
Description of parameters of the extended LFR.

Symbol	Definition
n	Number of vertices in each time step
S	Number of time steps in the dynamic network
μ	Mixing parameter
k	Average degree of the dynamic network
p	Probability of vertex switching between two adjacent time steps
e	Number of modules expands in every time step
c	Number of modules contracts in every time step
b	Number of modules births in every time step
d	Number of modules deaths in every time step
m	Number of modules mergers in every time step
s	Number of modules splits in every time step

Table 3
The real-world network features.

Network	Feature				
	Nodes (n)	Edges (m)	Average number of degrees ($\bar{k}$)	Average clustering coefficient ($\overline{CC}$)	Number of Time Step (S)
WC	88	537	11.4	0.38	8
HS2011	123	1271	20	0.51	7
HS2012	175	1629	18	0.43	8
PS	239	6146	50.8	0.52	9
NCC	107	375	4.3	0.49	17
	180	567			
CC	107	375	4.3	0.49	17
	166	543			

4.1. Data description

This section provides two types of datasets: synthetic and real-world networks.

Synthetic networks

Synthetic network is a generated network which the ground-truth communities are known and provided by Lancichinetti-Fortunato-Radicchi (LFR) in 2008. Then, in 2010, Greene at al. made an improvement to suit for dynamic network called extended LFR [86]. This extended LFR model allows the generation of networks with specific parameters, such as the average degree of vertices d, and the mixing parameter $0 < \mu < 1$ [27], providing a controlled environment for testing. The ability to quickly generate these networks, due to their linear complexity in construction, makes them ideal for analyzing how various hyperparameters—such as resolution parameters, time decay factors, and community merging thresholds—affect the accuracy and performance of community detection methods. By leveraging synthetic networks, researchers can systematically optimize hyperparameters to enhance the effectiveness of dynamic community detection in real-world scenarios. Table 1 provides detailed descriptions of the parameters of the extended LFR model. The details of synthetic network used in this study are shown in Table 2.

The description of each extended LFR network is provided below [67,68]:

1. **Expansion and contraction networks**: This is the process of adjusting the size of a community or the number of vertices and edges inside a community in each time step. The process is also known as the network grows or shrinks, respectively. It affects the overall network size and connectivity. In this study, the variation of

Table 2
Synthetic network used in experiment.

No	Extended LFR Network	Fix Parameter, $n = 1000$, $S = c_{\min} = 20$								
		e	c	b	d	m	s	μ	k	p
1	**Effect of Expansion and Contraction**	5, 20, 40	5, 20, 40	-	-	-	-	0.1	5	0.1
2	**Effect of Birth and Death**	-	-	2,8,16	2, 8	-	-	0.1	5	0.1
3	**Effect of merger and Split**	-	-	-	-	5, 20, 40	5, 20, 40	0.1	5	0.1
4	**Effect of Mixing Parameter**	-	-	-	-	-	-	0.2, 0.4	5	0.1
5	**Effect of Community Density**	-	-	-	-	-	-	-	5, 15, 25	0.1
6	**Effect of Vertex Switching**	-	-	-	-	-	-	0.1	5	0.1, 0.4, 0.8

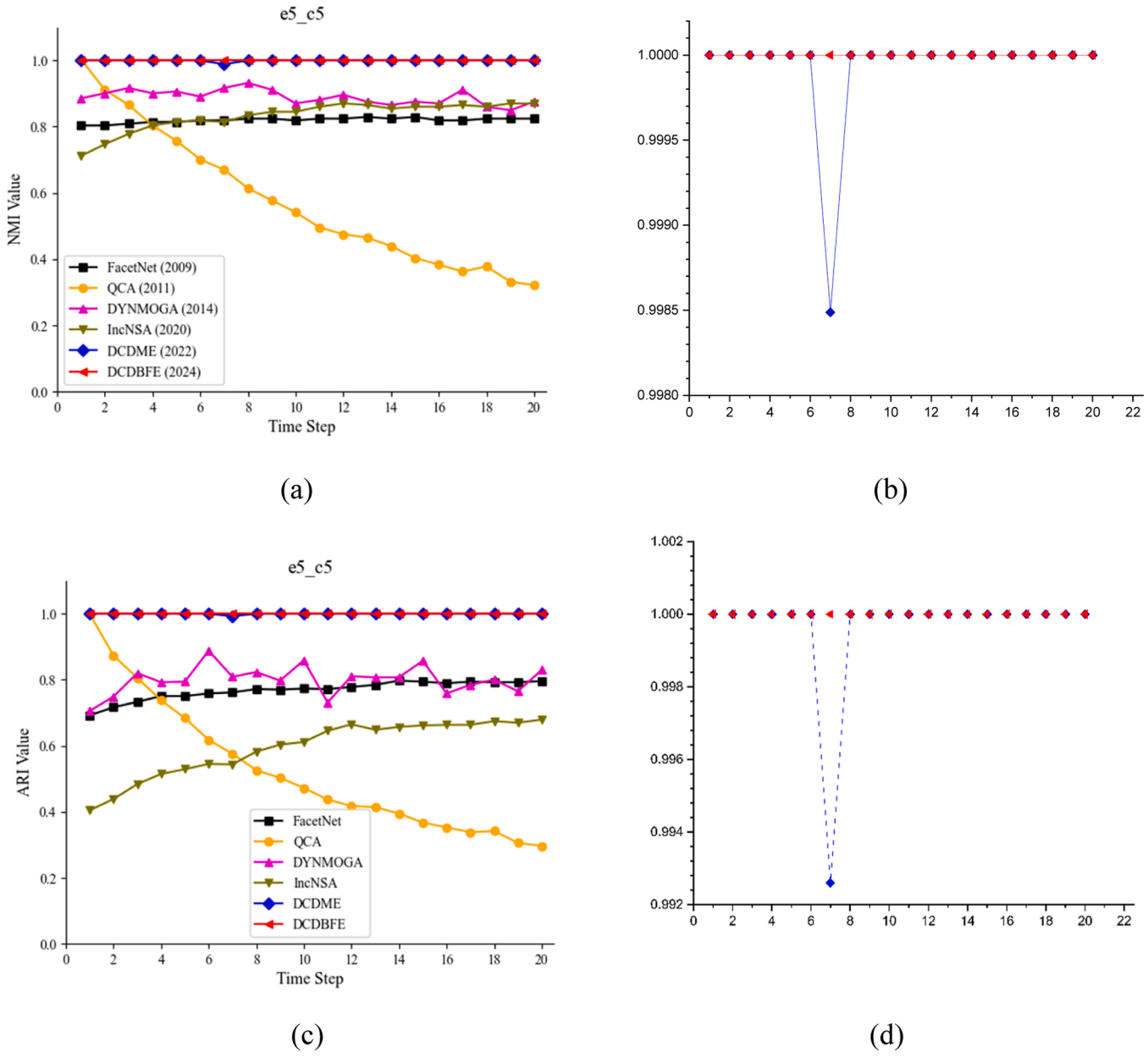


Fig. 6. Comparison of NMI and ARI indicators with 5 events of expansion and 5 events of contraction.

expansion, $e = 5, 20, 40$ and the variation of contraction, $c = 5, 20, 40$.

2. **Birth and death networks**: This is the process of introducing new nodes into the network or removal or deactivation of existing nodes from the network. It does not necessarily imply the addition of new edges unless explicitly stated. It affects the network composition. In this study, the variation of number of communities birth, $b = 2, 8, 16$ and the variation of number of communities death, $d = 2, 8$. Extended LFR benchmark could not generate the time path when community birth and death both reached 16, therefore no $d = 16$ variety.
3. **Merger and split networks**: This is the process of combining multiple clusters or components into one or dividing a cluster or component into multiple parts. It affects the internal structure and organization of clusters. In this study, the variation number of community merger, $m = 5, 20, 40$, and the variation number of community splits, $s = 5, 20, 40$.
4. **Mixing parameter networks**: This is the process of controlling the degree of interconnectedness between vertices within a community (k^{intra}) versus vertices between different communities (k^{inter}). It is used to assess the strength and clarity of community structures within a network. Lower mixing parameters (μ≈0) indicate stronger and more distinct communities, while higher mixing parameters (μ≈1) suggest more inter-community connections and less distinct community boundaries. The mixing parameter, μ can be defined as; $\mu = \frac{k_i^{\text{inter}}}{k_i}$ where k is the degree of vertex i. Alternatively, it can be expressed in terms of the intra-community connections as: $\mu = 1 - \frac{k_i^{\text{intra}}}{k_i}$. In this study, the variation of mixing parameter value is $\mu = 0.2, 0.6$. Community detection becomes harder as mixing parameters increases.
5. **Community density networks**: This is the process of measuring the cohesiveness and distinctiveness of community structures. It is about intra- and inter-community edge densities, focused on understanding

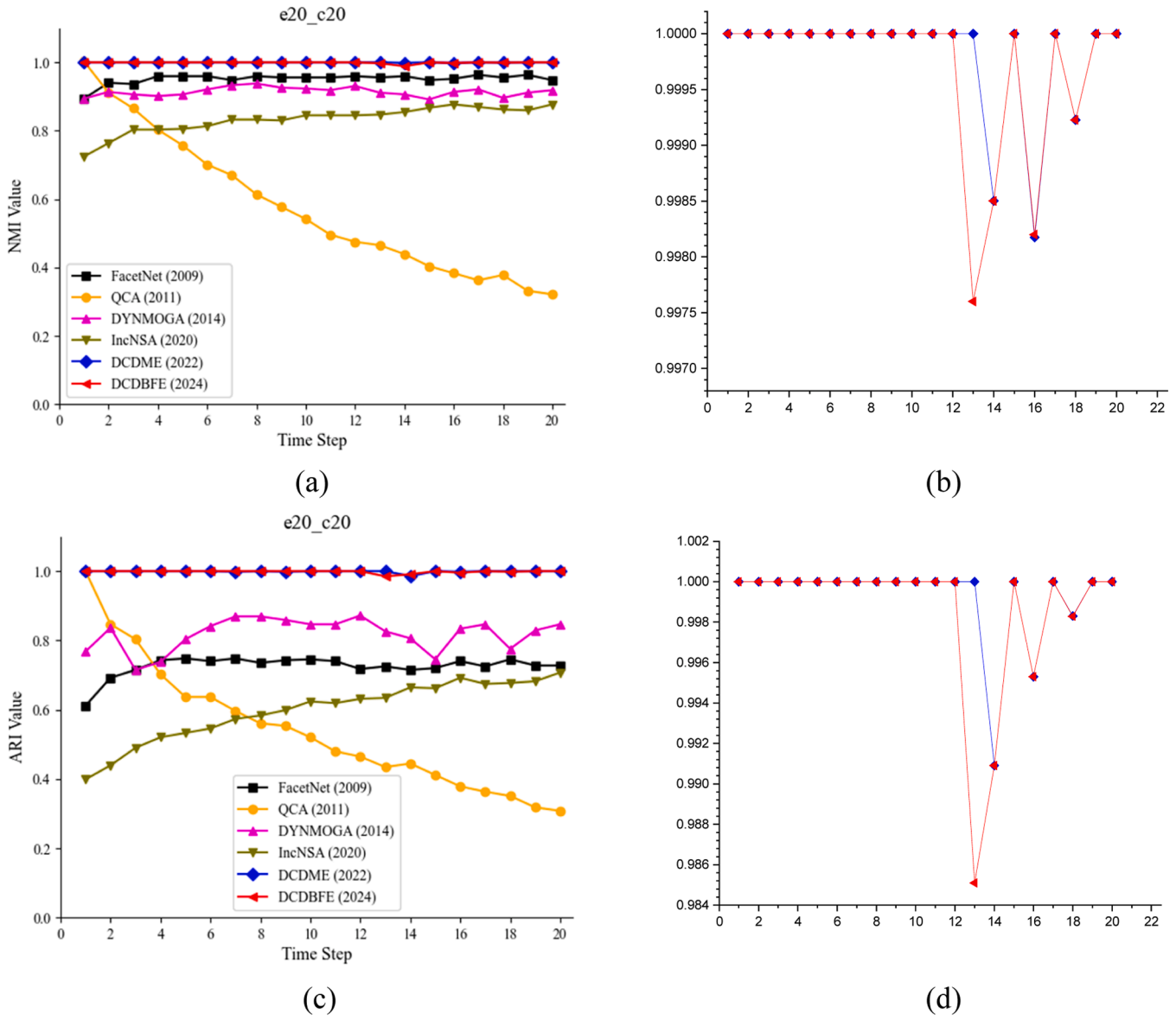


Fig. 7. Comparison of NMI and ARI indicators with 20 events of expansion and 20 events of contraction.

community strength and interactions. In this study, the variation of the average degree, $k = 5, 15, 25$.

6. **Vertex switching networks**: This is the process of altering the network structure while maintaining certain constraints like degree distribution. It is an edge rewiring between existing nodes, randomizing or optimizing specific network properties. In this study, the variation of probability, $p = 0.1, 0.4, 0.8$

Real-world networks

We also conducted an experiment on four real-world networks, with the details provided in Table 3.

The real-world datasets use in sociology. Fundamentally, empirical networks are unknown ground truths. The description of each social network is provided below:

1. **Primary School dynamic networks**: A student-teacher social network and the data set contains a time series network of contacts between children and teachers. Each child or teacher represents a vertex, and the contact between them represents an edge.
2. **Workplace Contact dynamic networks**: A social network of contacts between people in a French office building. The time series has five departments as real communities and recorded contact between people at intervals is 20 seconds.
3. **High School 2011 dynamic networks**: A social network of connections formed between student by three classes of a high school in Marseille, France. The time series is in December 2011. HS2011 has 7 paths or time slices and each of it has three communities. In HSD11 dynamic network, one vertex represents a student, and one edge indicates that there is a connection between the students.
4. **High School 2012 dynamic networks**: Similar to HS2011, a social network of connections formed between student by five classes of a high school in Marseille, France. The time series is in November 2012. HS2012 has 7 paths or time slices and each of it has five communities. The other information is consistent with HSD11.
5. **Cumulative Co-authorship Network (CC)**: This dataset represents a collaboration network derived from a citation network. The version used in this study has been curated and modified based on reference Chakraborty, T. et al., 2014. In this dynamic network, each node represents an author, while edges denote coauthorship relationships. The CC network accumulates the changes of nodes and edges over time, reflecting ongoing collaborations.

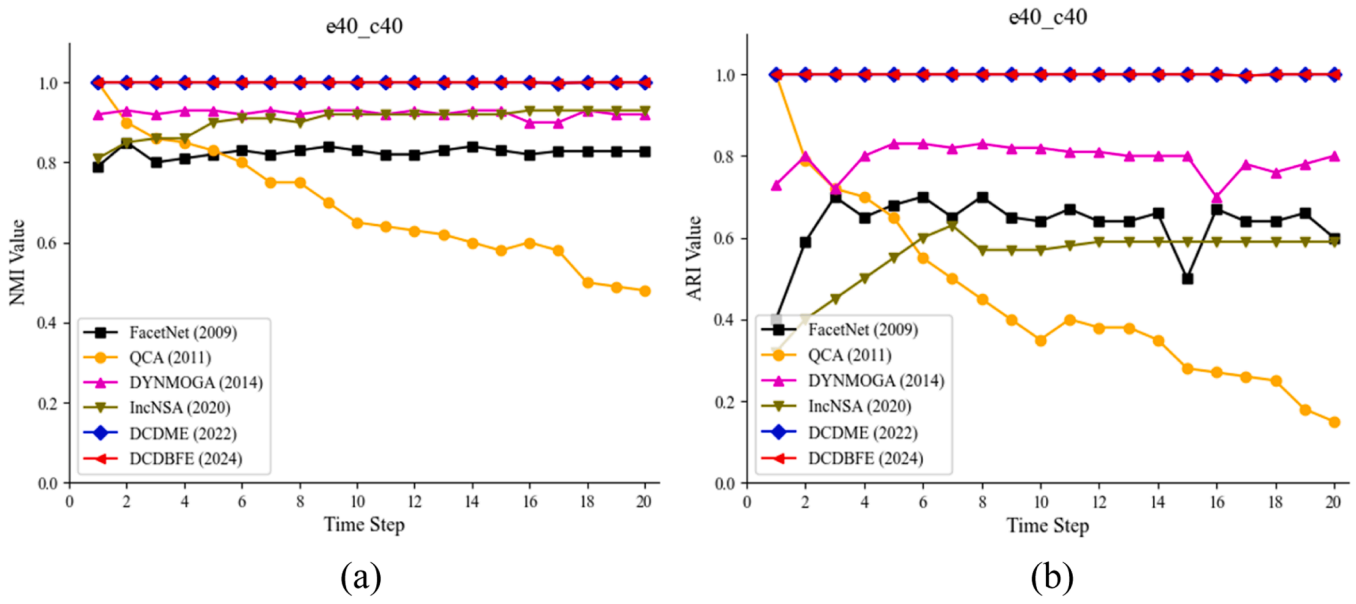


Fig. 8. Comparison of NMI and ARI indicators with 40 events of expansion and 40 events of contraction.

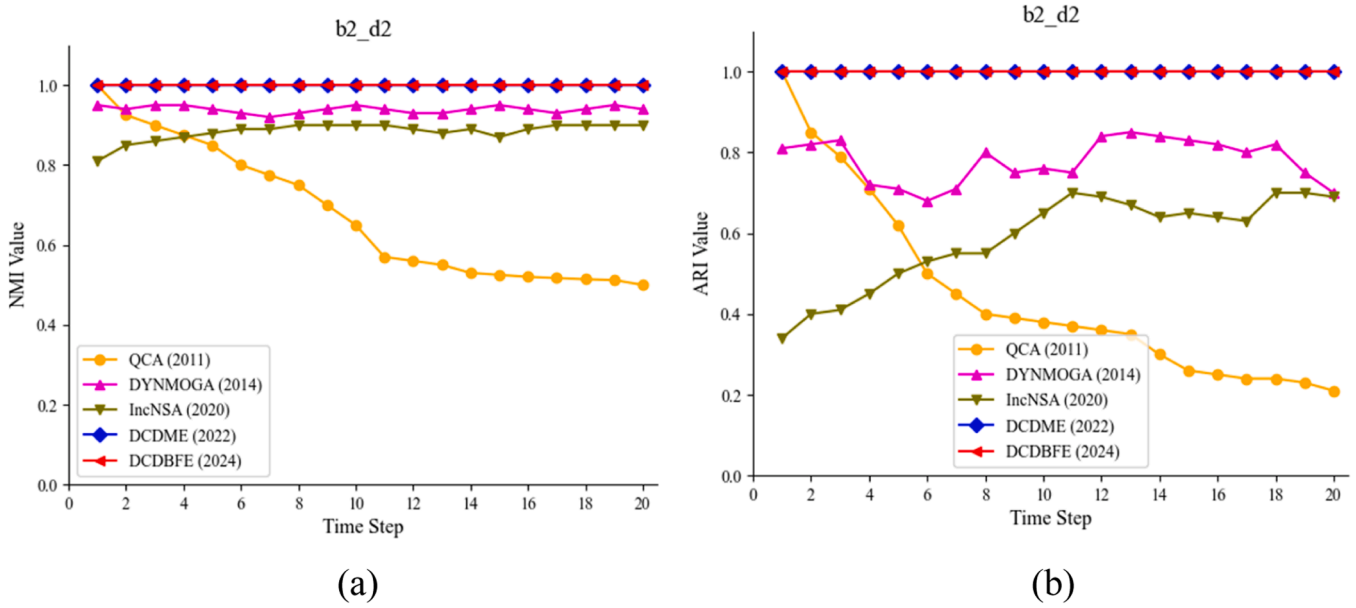


Fig. 9. Comparison of NMI and ARI indicators with 2 events of birth and 2 events of death.

6. **Noncumulative Co-authorship Network (NCC)**: Similar to the CC network, this dataset also represents a collaboration network. However, unlike the CC network, the NCC dynamic network does not accumulate changes. In this case, even if two authors coauthor multiple papers, only a single link exists between them, regardless of how many collaborations occur.

4.2. Evaluation metrics

To evaluate the effectiveness and community structure correctness of different methodologies, two commonly criteria; normalized mutual index (NMI) [83], and adjusted random index (ARI) [87], were used. Results improved with higher NMI, and ARI values [88].

A well-known information theory function of NMI is represented as in Eq. (11) as follows:

$$\mathrm{NMI} = \frac{2\mathrm{I}(\mathrm{U};\mathrm{V})}{\mathrm{H}(\mathrm{U}) + \mathrm{H}(\mathrm{V})} \tag{11}$$

where U and V are the partitions of communities, I(U;V) denotes the mutual information of random variables U and V, and H(U) represents the entropy of U. NMI represents the degree of dependence between two partitions. When the ground truth is known, the NMI can be used to compare the partitions given by proposed algorithms to the ground truth and non-overlapping community identification. NMI computations calculate the similarity (mutual information) between network groups, which indicates a clustering method's robustness and homogeneity. The value of NMI ranges from 0 to 1, where 0 represents that the detected communities are completely independent of the real communities, and 1 denotes a perfect match with the ground truth. A higher NMI score indicates a better alignment between the detected communities and the actual structure of the network.

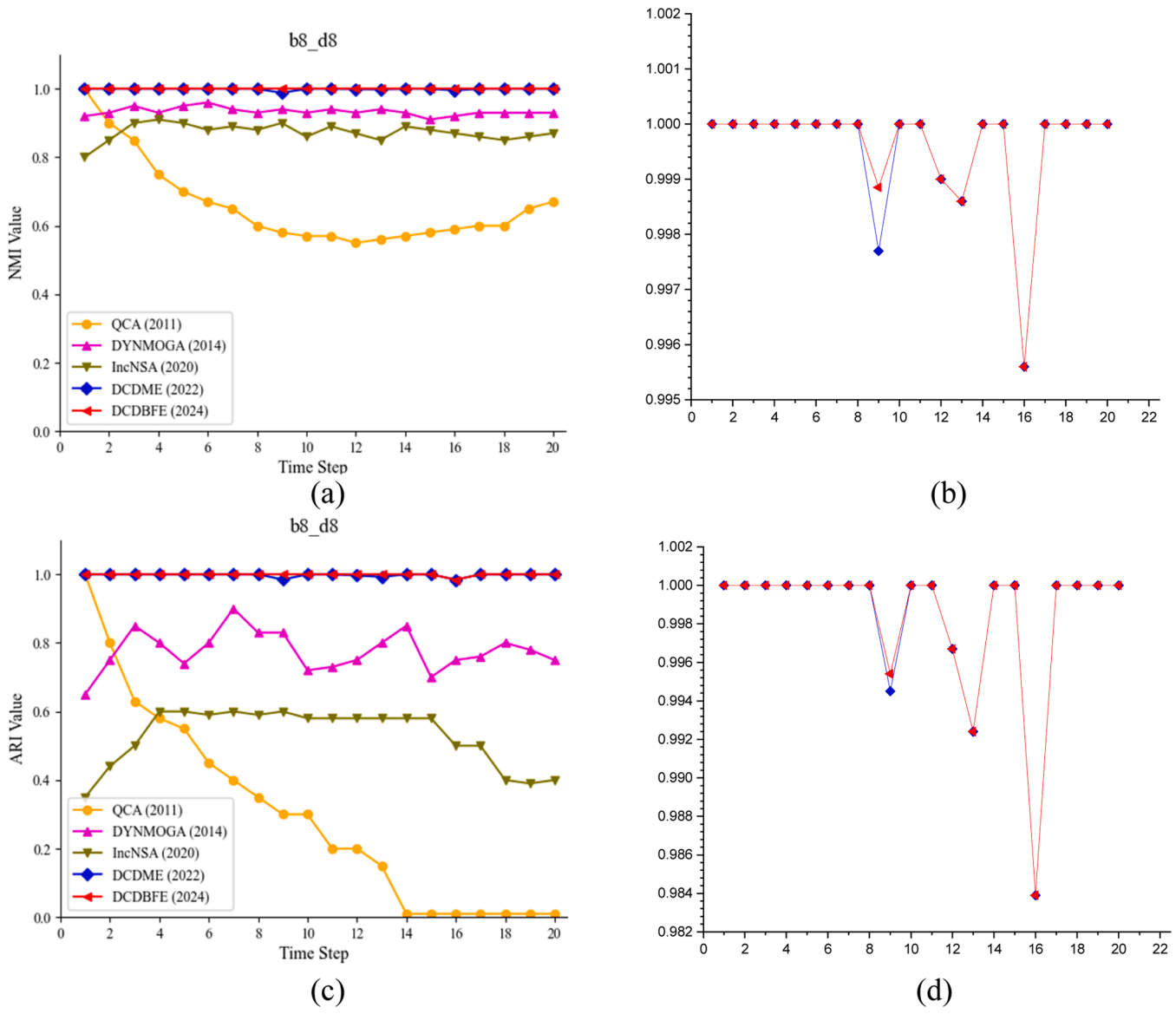


Fig. 10. Comparison of NMI and ARI indicators with 8 events of birth and 8 events of death.

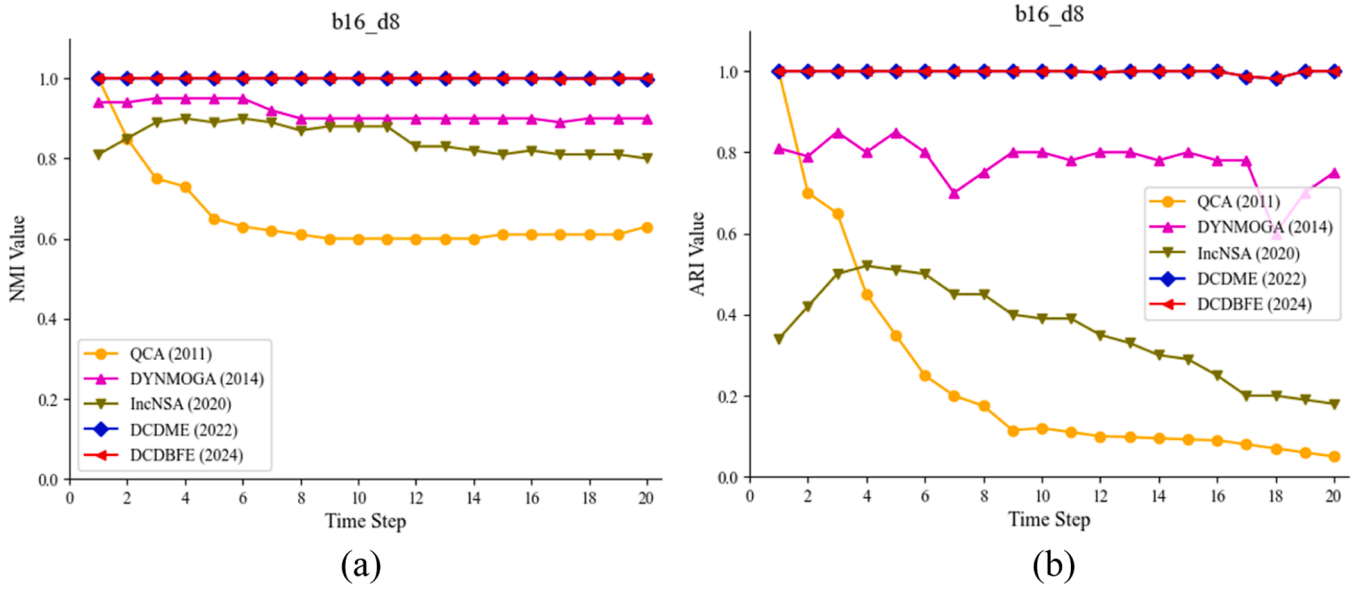


Fig. 11. Comparison of NMI and ARI indicators with 16 events of birth and death 8 events of death.

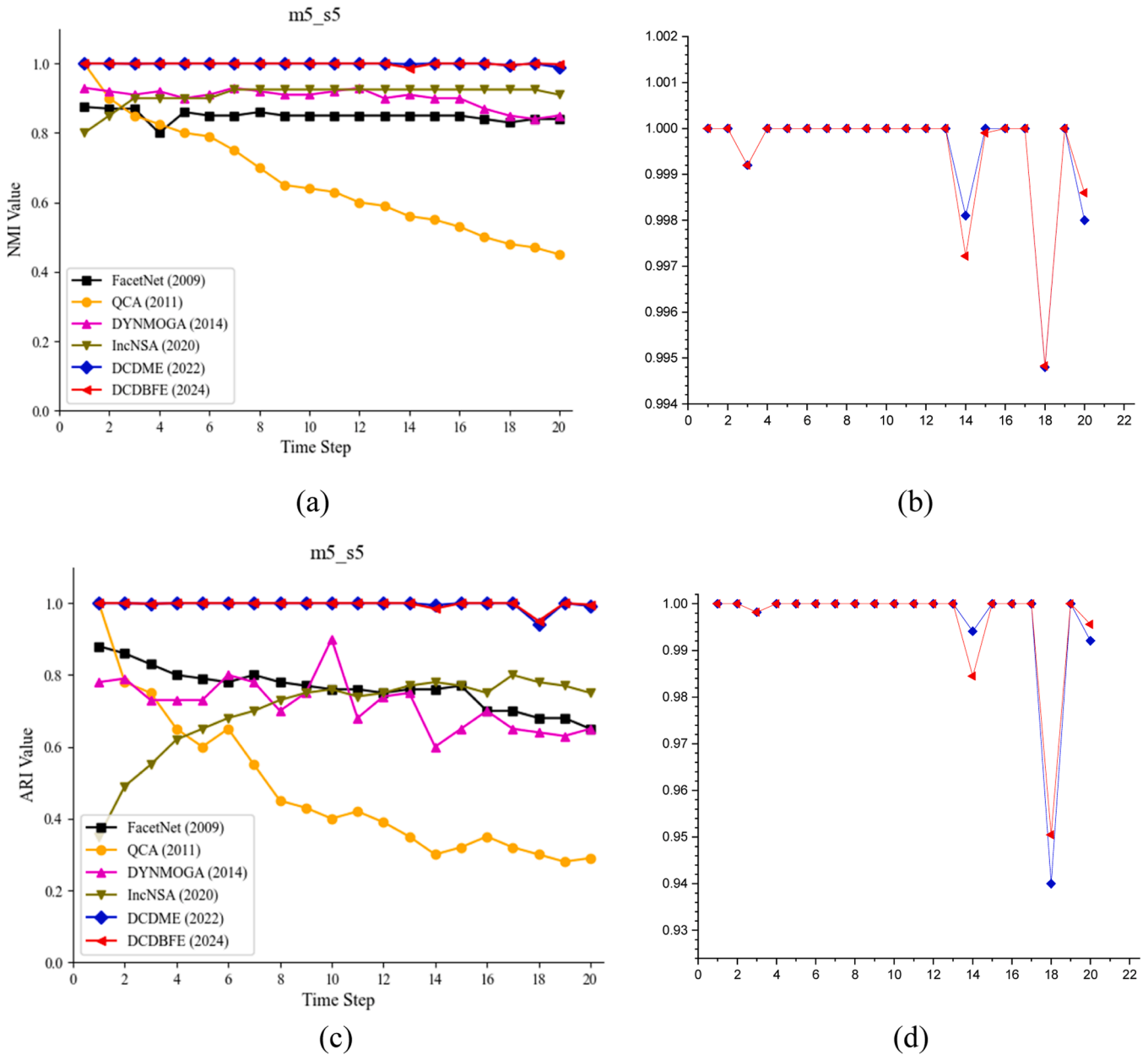


Fig. 12. Comparison of NMI and ARI indicators with 5 events of merger and 5 events of split.

ARI is another function to evaluate the similarity between two communities. The definition of ARI is explained as in Eq. (12) as follows:

$$\mathrm{ARI} = \frac{\mathrm{RI} - \mathrm{RI}_{\mathrm{expected}}}{\mathrm{RI}_{\mathrm{max}} + \mathrm{RI}_{\mathrm{expected}}} \tag{12}$$

where R is the random and I is the index. RI denotes the similarity of two partitions, which includes all pairs of samples. Next, it calculates the numbers of pairs that are assigned to the same or different partitions in the predicted and true partitions [89]. More details can be referred to [87]. It quantifies the accuracy of community detection results, with a higher ARI indicating better performance.

4.3. Performance evaluation

In this section, we execute all the algorithms on a device with installed python running on an Intel Core i7-7700 CPU @ 2.8 GHz and 24 GB of RAM. In these experiments, we employed the parameters of comparison algorithms to the default values suggested by the authors. The codes of QCA, FacetNet, DYNMOGA, IncNSA, DCDME, and DCDBFE are available at GitHub (https://github.com/sitiharyanti/DCDBFE_2024). For each dynamic network, the average result of each algorithm was obtained by averaging 20 independent runs. We conducted experiments on both well-studied synthetic and real-world networks to validate our approaches.

5. Results and discussion

This section presents the results and analysis on the development of dynamic community detection based on bird flock effect (DCDBFE). The performance of the proposed method is compared to various community detection methods in synthetic benchmark and real-world networks.

5.1. Results on synthetics networks

In the actual world, community expansion and contraction, birth and death, merger and splitting, and vertex switching across communities

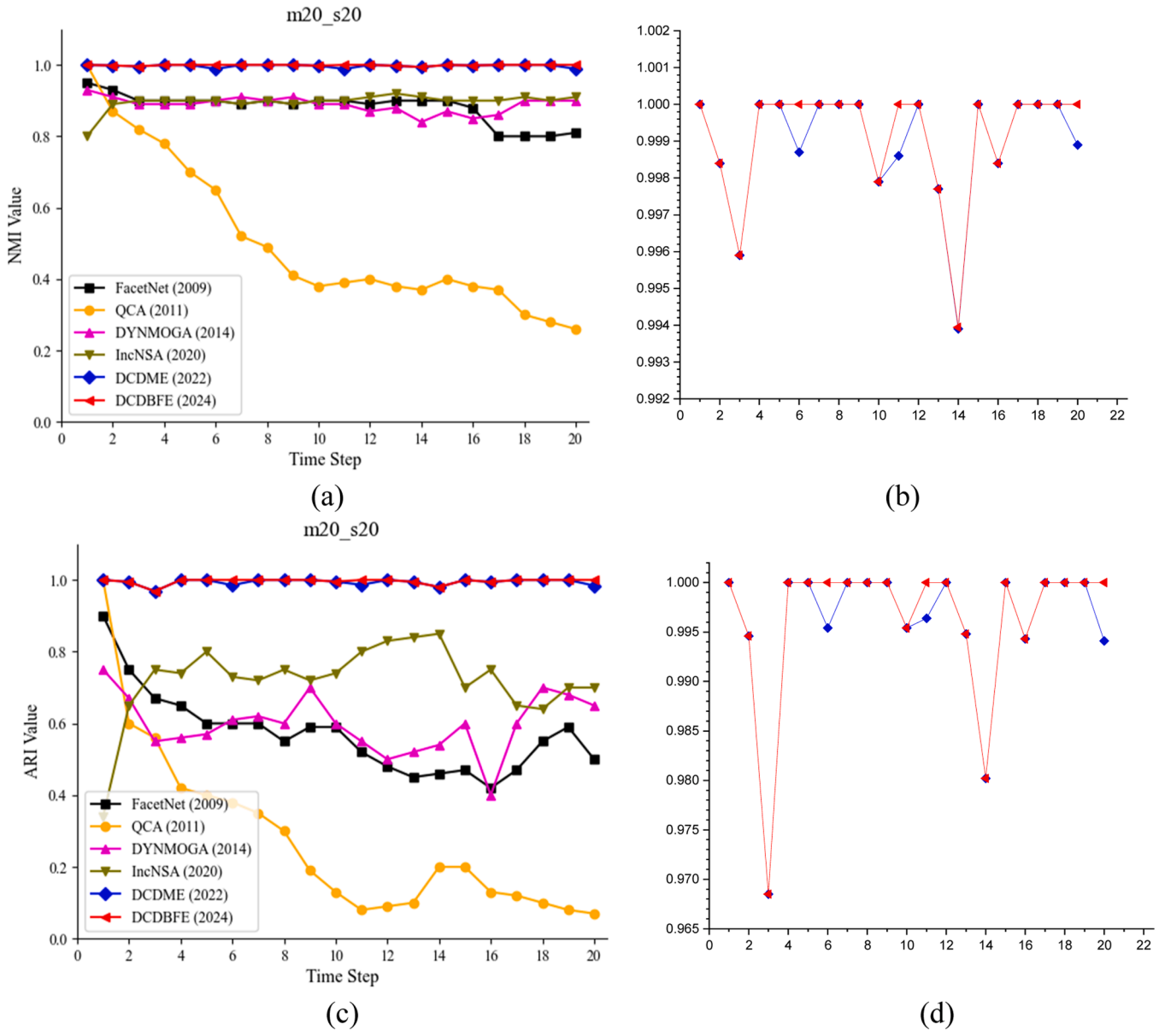


Fig. 13. Comparison of NMI and ARI indicators with 20 events of merger and 20 events of split.

occur in dynamic networks. It is now standard practice in the field to test community detection methods on data that have been generated [90]. To simulate community structure fluctuation, we investigated six effects in extended LFR networks. Without loss of generality, Table 3 lists all extended LFR network parameters. Figs. 6–22 show the result of comparison value of NMI and ARI obtained from these six types of effect.

1) **Effect of expansion and contraction**

We set the parameters to $k = 5$, $\mu = 0.1$, and $p = 0.1$, while varying the number of expanding and contracting communities from 5 to 40 to further assess the performance of each algorithm on community expansion and contraction events. The effectiveness of performance algorithms on 5 events of expansion and 5 events of contraction with the comparison value of NMI and ARI measurements is shown in Fig. 6(a)-(d).

In Fig. 6(a), the NMI values of DCDBFE consistently maintain the highest NMI, indicating excellent clustering quality with a slight dip to approximately 0.999 at the 7th time step. In contrast, QCA shows a significant decline, starting around 0.85 and dropping steadily to approximately 0.3 by the end of the time steps. The other methods exhibit moderate fluctuations. Fig. 6(b) is a zoom-in view of Fig. 6(a), highlighting the stability of DCDBFE. DCDME also shows stable performance with high NMI values, close to 1.0, but experiences a slight drop to about 0.998 at the 7th time step.

Fig. 6(c) illustrates the ARI values, which mirror the NMI trends. DCDBFE maintains high ARI values, demonstrating robust (high) community detected consistency, while QCA's ARI declines considerably over time. Fig. 6(d) focuses on high ARI values, further emphasizing DCDBFE's stability, with a slight decrease to around 0.998 at the 7th time step. Overall, DCDBFE outperforms other methods in both NMI and ARI, maintaining superior clustering quality and consistency. In contrast, QCA exhibits the most significant performance decline, with its lowest NMI reaching 0.3 and its ARI dropping to 0.2.

Furthermore, Fig. 7(a) demonstrates the NMI performance for six algorithms over 20-time steps for 20 events of expansion and 20 events of contraction. The DCDBFE algorithm consistently achieved an NMI of 1.0, indicating perfect clustering performance. DCDME also maintains high NMI values with minor fluctuations. In contrast, algorithms like QCA and IncNSA exhibited a noticeable decline in NMI as time

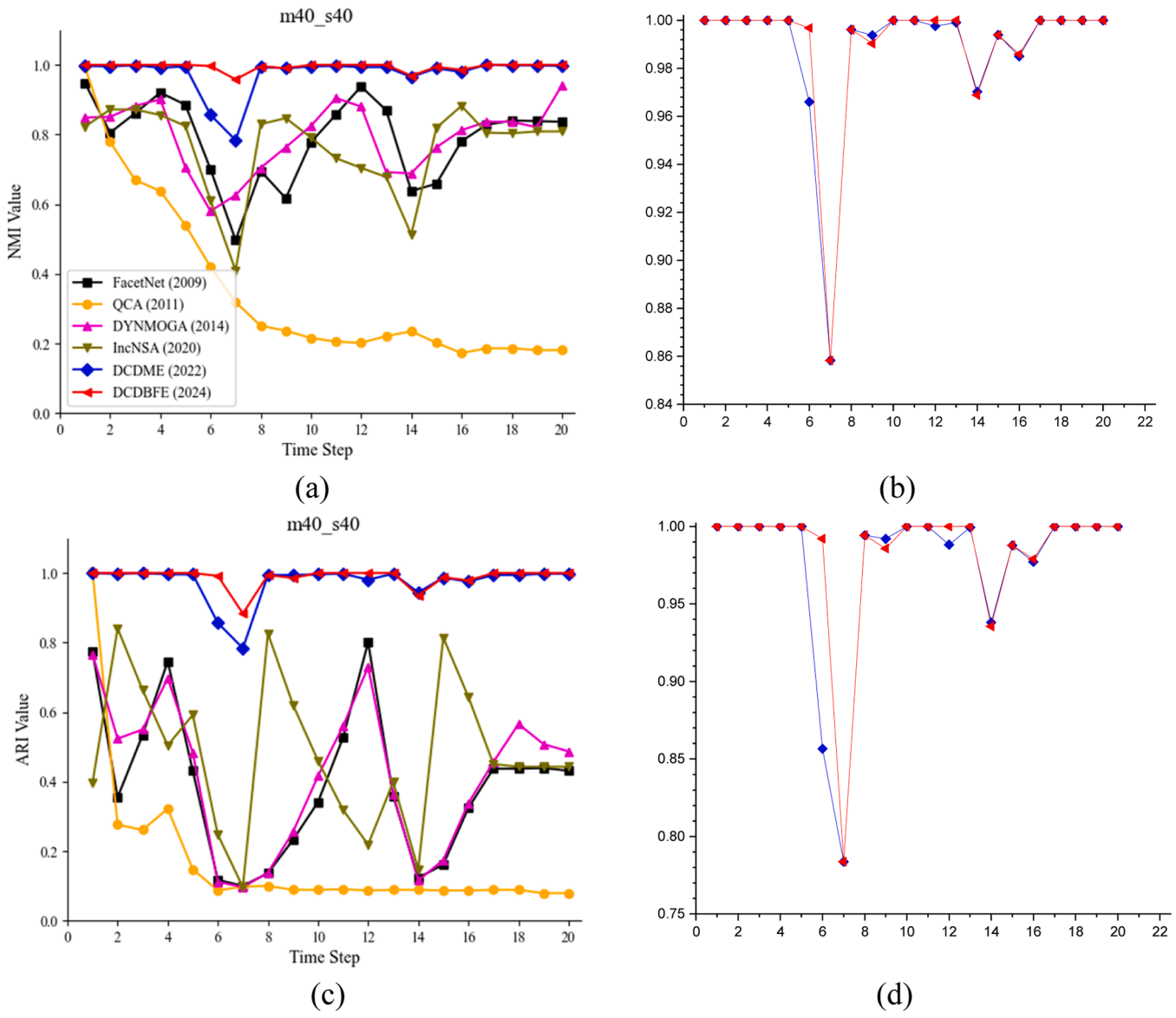


Fig. 14. Comparison of NMI and ARI indicators with 40 events of merger and 40 events of split.

progressed, with IncNSA showing a sharp drop after the midpoint. DYNMOGA demonstrated moderate performance with some fluctuations, while FacetNet remained relatively stable. Fig. 7(c) displays the ARI for the same algorithms and conditions, corroborating the trends observed in the NMI analysis. DCDBFE and DCDME maintained the highest ARI values, indicating superior clustering agreement over time. The ARI performance of QCA and IncNSA showed more variability, with IncNSA experiencing a significant decline, similar to its NMI trend. DYNMOGA and FacetNet showed moderate ARI values, consistent with their NMI performance. Fig. 7(b) and Fig. 7(d) provide a zoomed-in view of the best-performing algorithms (DCDBFE and DCDME), confirming their near-perfect community detected consistency with minimal deviations. This detailed comparison highlights the stability of DCDBFE in dynamic community detection tasks, significantly outperforming other algorithms.

Fig. 8(a) shows that DCDBFE algorithm consistently maintains an NMI of 1.0 throughout all time steps, indicating perfect clustering performance and stability despite the dynamic changes. DCDME also performed well with high NMI values, although it experienced minor fluctuations. On the other hand, QCA exhibited significant decline in NMI, indicating that it struggled to maintain accurate clustering as the network evolves. DYNMOGA and IncNSA showed relatively stable but lower NMI values compared to DCDBFE and DCDME, suggesting moderate performance in handling dynamic changes. FacetNet maintained a stable but moderate NMI performance throughout the time steps.

Fig. 8(b) displays the ARI values for the same algorithms and conditions, which reflect the degree of agreement between the detected modules and the true module labels over time. Similar to the NMI results, DCDBFE achieved the highest ARI values, indicating superior clustering consistency and accuracy. DCDME also showed high ARI values with minor variability, reinforcing its stability in dynamic environments. QCA again showed a significant drop in ARI, highlighting its difficulty in maintaining reliable clustering as the network undergoes expansion and contraction. DYNMOGA exhibited better stability in ARI compared to NMI, yet still fell short of the top-performing algorithms. IncNSA and FacetNet displayed moderate ARI values, ranging from 0.3 to 0.6, indicating average performance. Overall, the figure emphasizes the exceptional stability and performance of DCDBFE and DCDME in dynamic network clustering, significantly outperforming older algorithms like QCA, which struggle to adapt to dynamic changes.

2) **Effect of Birth and Death**

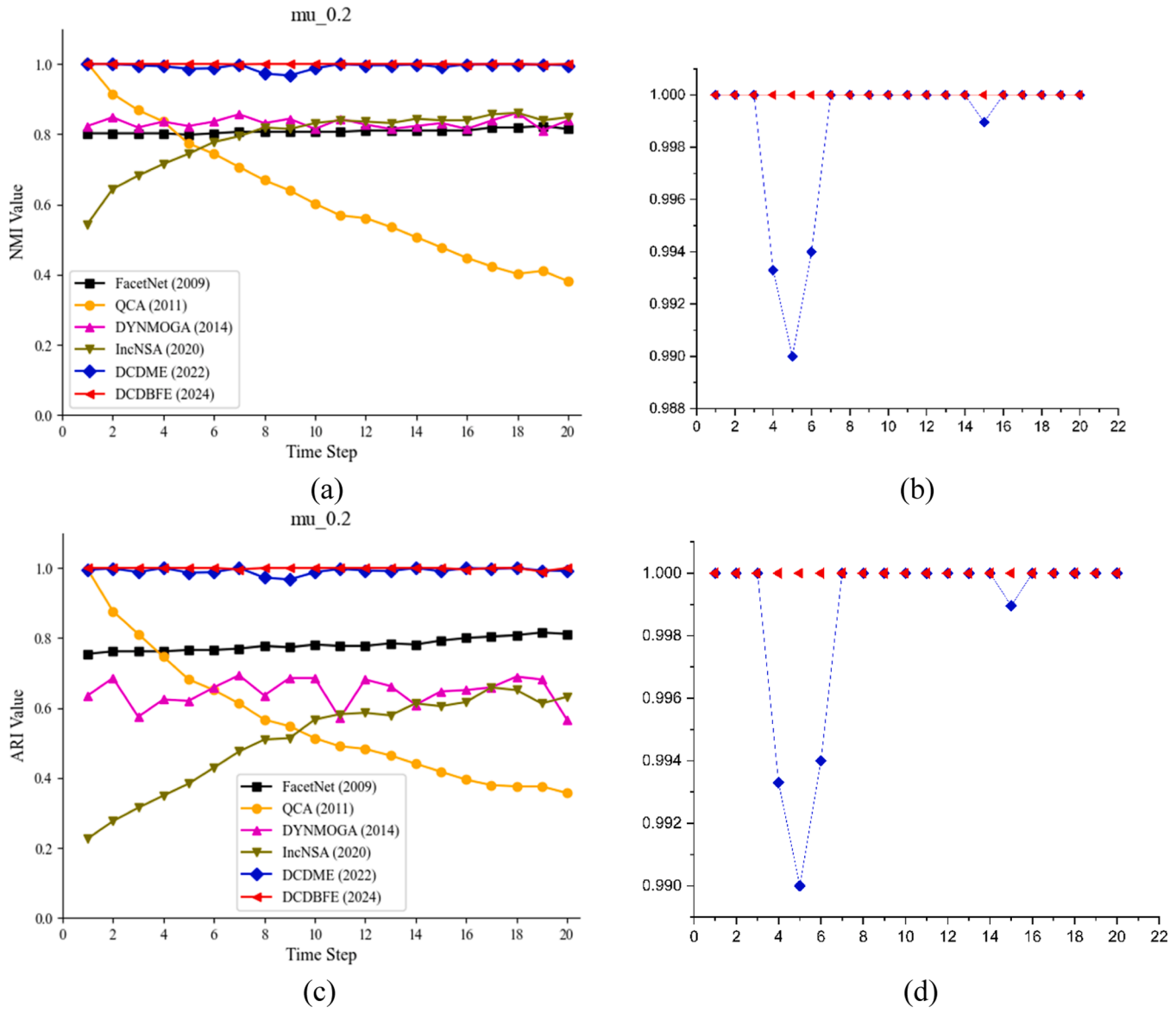


Fig. 15. Comparison of NMI and ARI indicators with mixing parameter $\mu = 0.2$.

We set the parameters to $k = 5$, $\mu = 0.1$, and $p = 0.1$, while varying the number of birth and death communities from 5 to 40 to further assess the performance of each algorithm on community birth and death events. Because the dynamic LFR model could not generate a dynamic network when the number of birth and death communities both reached 16, we changed the number of birth communities from 2 to 16 and varied the number of death communities from 2 to 8 as shown in Figs. 9–11. The NMI and ARI indicators show how the DCDME and DCDBFE algorithms perform under different scenarios of community birth and death events (birth2 death2, birth8 death8, and birth16 death8). Both algorithms exhibited slight changes in NMI and ARI values as the number of birth and death events increased. This slight fluctuation suggests that while these algorithms are stable in detecting and preserving community structures, the introduction of new communities (birth events) and the dissolution of existing ones (death events) introduce additional complexity that challenges the algorithms' ability to maintain perfect stability. The AA (Adamic-Adar) and RA (Resource Allocation) similarity measures used in these algorithms are effective at capturing local structural information, but as the frequency of these dynamic events increases, the network's evolving nature makes it harder to consistently maintain the same community structure, leading to minor variations in the NMI and ARI values. These small changes reflect the inherent difficulty in tracking community births and deaths, even with stable algorithms like DCDME and DCDBFE.

Fig. 10(a) shows a small improvement (0.15 %) when birth and death events were 8 at 9th time path. DCDBFE consistently achieved an NMI of 1.0, demonstrating perfect clustering accuracy throughout the dynamic changes. DCDME also performed well, maintaining high NMI values with minor dips at certain points. IncNSA and DYNMOGA exhibited moderate NMI values with noticeable fluctuations, indicating a moderate level of adaptability to the birth and death events. However, QCA showed a significant decline in NMI, reflecting its poor performance in maintaining accurate clustering under these dynamic conditions.

Fig. 10(c) shows the ARI values for the same algorithms, providing a measure of agreement between the detected clusters and the true cluster labels over time. Similar to the NMI results, DCDBFE maintained the highest ARI values throughout the time paths, confirming its superior clustering consistency and accuracy. DCDME also showed high ARI values, although it experienced slight drops at specific time points. IncNSA and DYNMOGA exhibited fluctuating ARI values, indicating their moderate performance in dynamic environments. QCA again

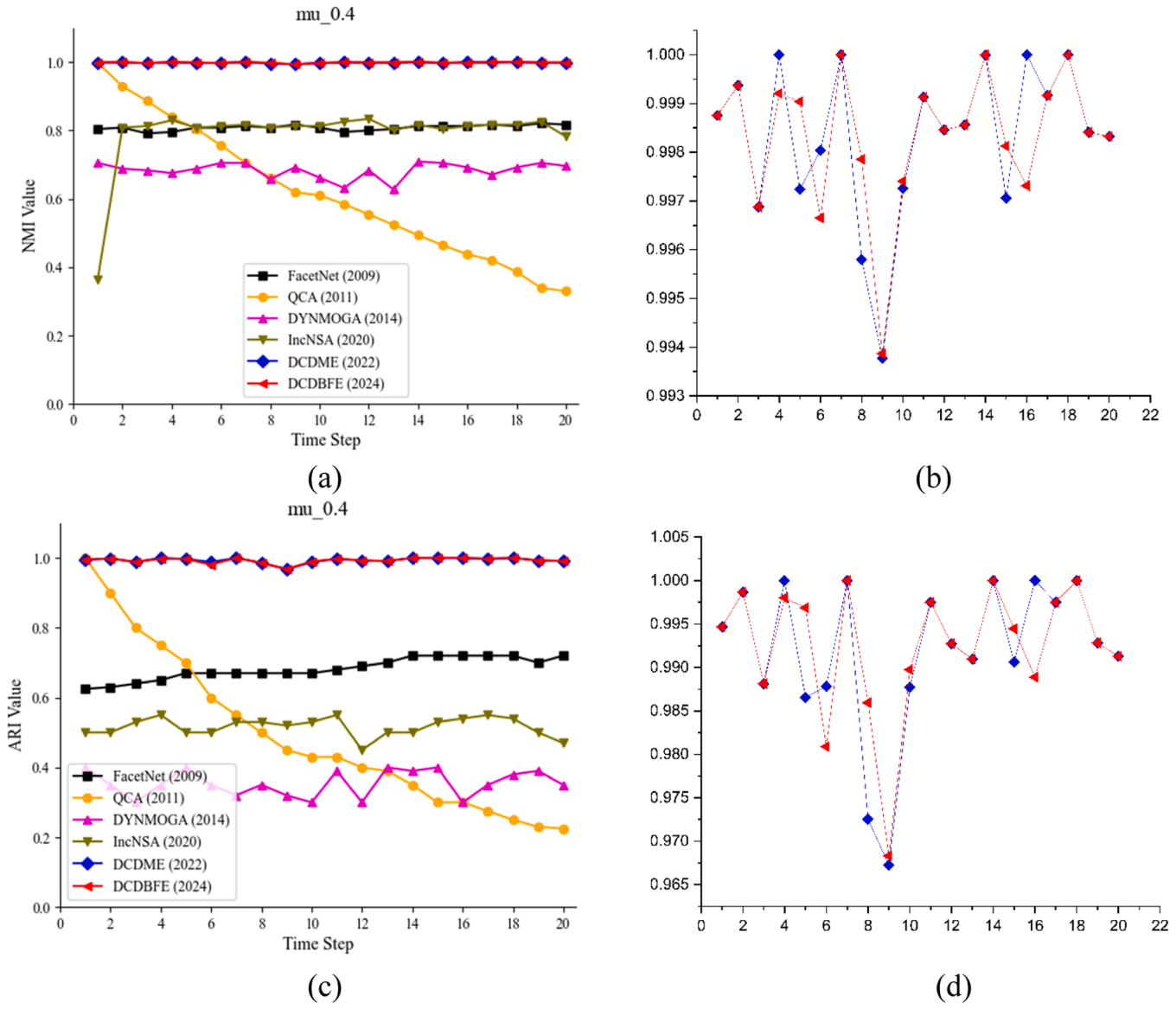


Fig. 16. Comparison of NMI and ARI indicators with mixing parameter $\mu = 0.4$.

showed a significant decline in ARI, highlighting its inability to maintain reliable clustering as the network undergoes birth and death events. Fig. 10(b) and Fig. 10(d) provide a zoomed-in view of the best-performing algorithms (DCDBFE and DCDME), showing that despite minor fluctuations, these algorithms maintain near-perfect clustering accuracy. Overall, the figure emphasizes the robustness and superior performance of DCDBFE and DCDME in handling dynamic network conditions, outperforming older algorithms like QCA, which struggle significantly with the birth and death events.

Fig. 11(a) shows that DCDBFE and DCDME maintain near-perfect NMI values throughout the time steps, indicating their superior adaptability to the dynamic changes of the community. In contrast, DYNMOGA shows a steady decline in performance, and QCA exhibits the weakest performance, with its NMI values sharply decreasing early on. Fig. 11(b) presents similar trends in ARI, where DCDBFE and DCDME maintain almost perfect ARI scores, while DYNMOGA and QCA struggle significantly as the death events increase. The inability to generate all 16 death events could be due to the limitations of certain algorithms in managing large-scale node or community removal, which leads to performance degradation, especially in older algorithms like DYNMOGA and QCA. The best performance from DCDBFE and DCDME can be attributed to their advanced methodologies, which allow them to handle both birth and death events efficiently, maintaining high accuracy despite the structural changes within the network.

3) Effect of merger and Split

In Figs. 12–14, the NMI and ARI indicators illustrate how the DCDME and DCDBFE algorithms respond to different numbers of community merger and split events (merger5 split5, merger20 split20, and merger40 split40). For these algorithms, there are slight changes in the NMI and ARI values as the number of merger and split events increases. This suggests that while these algorithms are stable in tracking community structures, the increasing complexity of the network dynamics—due to more frequent mergers and splits—introduces minor challenges in maintaining perfectly stable community detection. The use of AA and RA similarity measures in these algorithms helps in capturing local structural details effectively, even as communities merge and split, but as the number of such events increased, the algorithms faced more difficulty in consistently preserving the exact community structures, leading to slight fluctuations in the NMI and ARI values. These minor changes indicate that while the algorithms are generally reliable, the

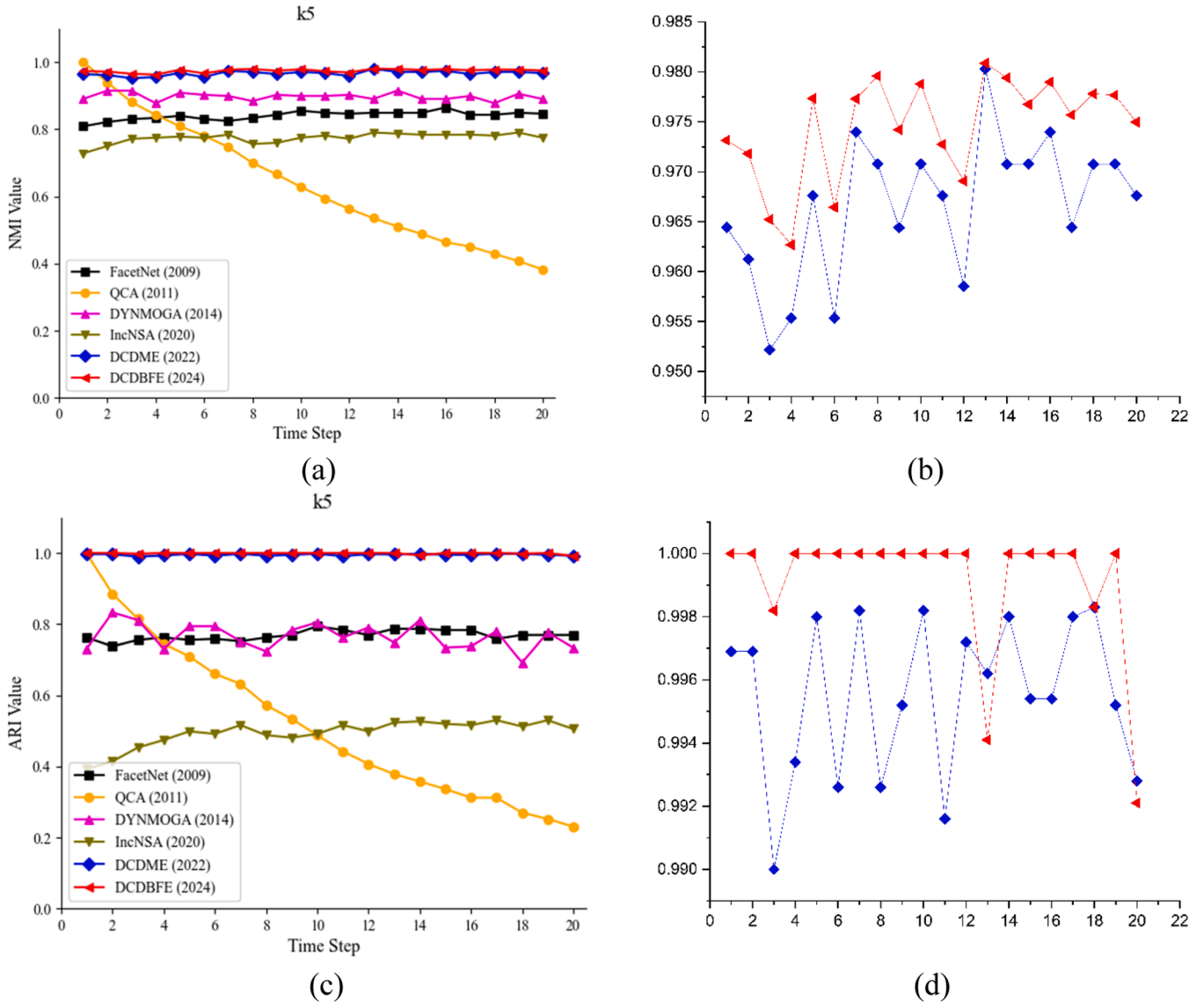


Fig. 17. Comparison of NMI and ARI indicators with average degree $k = 5$.

increased frequency of structural changes in the network introduces complexity that slightly impacts their performance.

Fig. 12(a) shows DCDBFE and DCDME consistently perform at the highest level, maintaining NMI values close to 1.0 throughout all time steps, signifying excellent adaptability to dynamic community merging and splitting. However, DYNMOGA shows a steady decline in NMI values after the 5th time step, highlighting its difficulty in handling such events. Fig. 12(b) zooms in on DCDBFE and DCDME, showing minor fluctuations, particularly at time steps 14 and 18, where both algorithms briefly dip below 0.998 but quickly recover.

In Fig. 12(c), ARI values reflect similar trends, with DCDBFE and DCDME maintaining close to perfect performance, while DYNMOGA and QCA exhibit significant performance degradation over time. IncNSA performs moderately well, with more variability. Fig. 12(d) highlights specific dips for DCDBFE and DCDME during the same time steps, especially at time steps 14 and 18, where both algorithms briefly drop to around 0.95 but regain their near-perfect ARI afterward. These fluctuations in both NMI and ARI likely result from the complexity introduced by community splits and merges, which challenge even the most advanced algorithms, though DCDBFE and DCDME show the best overall stability and performance.

In Fig. 13(a), DCDBFE and DCDME consistently maintain near-perfect NMI values, while DYNMOGA steadily declines after the 5th time step, reflecting its difficulty in handling complex community changes. IncNSA and FacetNet demonstrate moderate performance, with more variability. Fig. 13(b) focuses on DCDBFE and DCDME, showing minor fluctuations around time steps 8, 12, and 16, where the NMI slightly dips but remains above 0.995.

Fig. 13(c) highlights the ARI performance, where DCDBFE and DCDME continue to dominate with ARI values consistently near 1.0, showcasing their robustness in accurately detecting community structures despite 20 merger and 20 split events. In contrast, DYNMOGA experiences a sharp decline in ARI after the 5th time step, while QCA and FacetNet show moderate but fluctuating performance. IncNSA performs relatively better than the older algorithms but still struggles to match the consistency of DCDBFE and DCDME. Fig. 13(d) further zooms in on the ARI fluctuations of DCDBFE and DCDME, showing small dips at time steps 8, 12, and 16, where the ARI momentarily falls to around 0.98 before quickly recovering. These fluctuations can be attributed to the complexity introduced by the high number of merging and splitting events, but overall, DCDBFE and DCDME demonstrate superior adaptability, maintaining high accuracy throughout the process.

Fig. 14(a) shows that DCDBFE and DCDME maintain near-perfect NMI values close to 1.0 throughout all time steps, while the

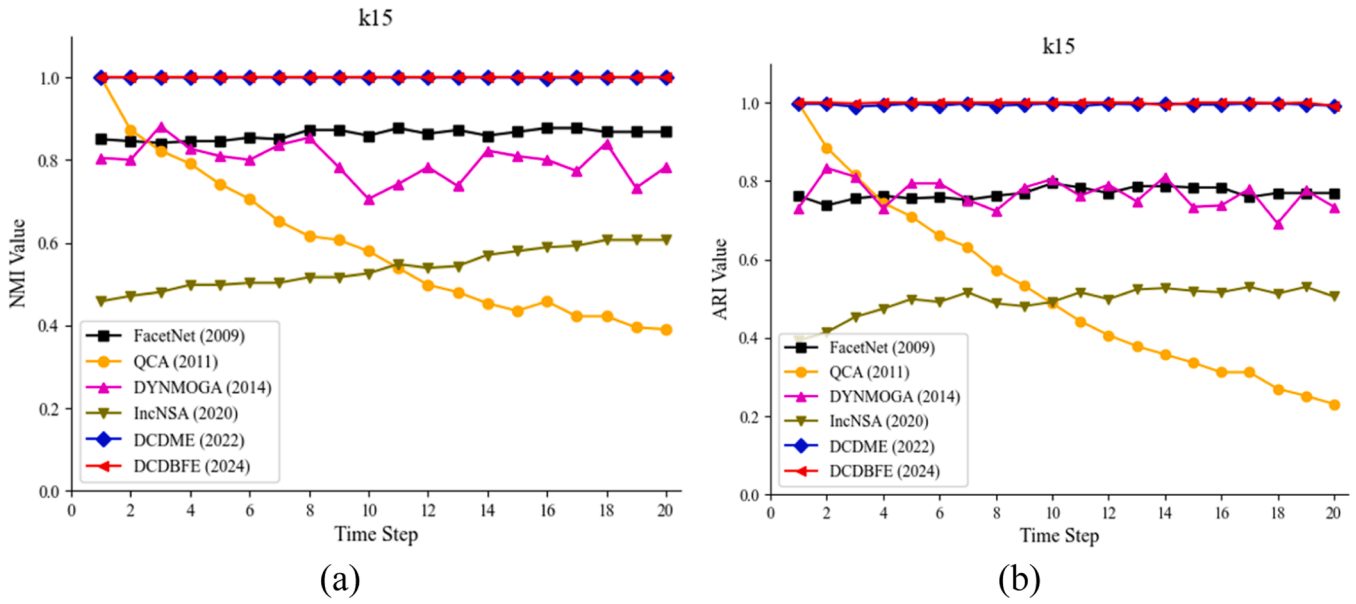


Fig. 18. Comparison of NMI and ARI indicators with average degree $k = 15$.

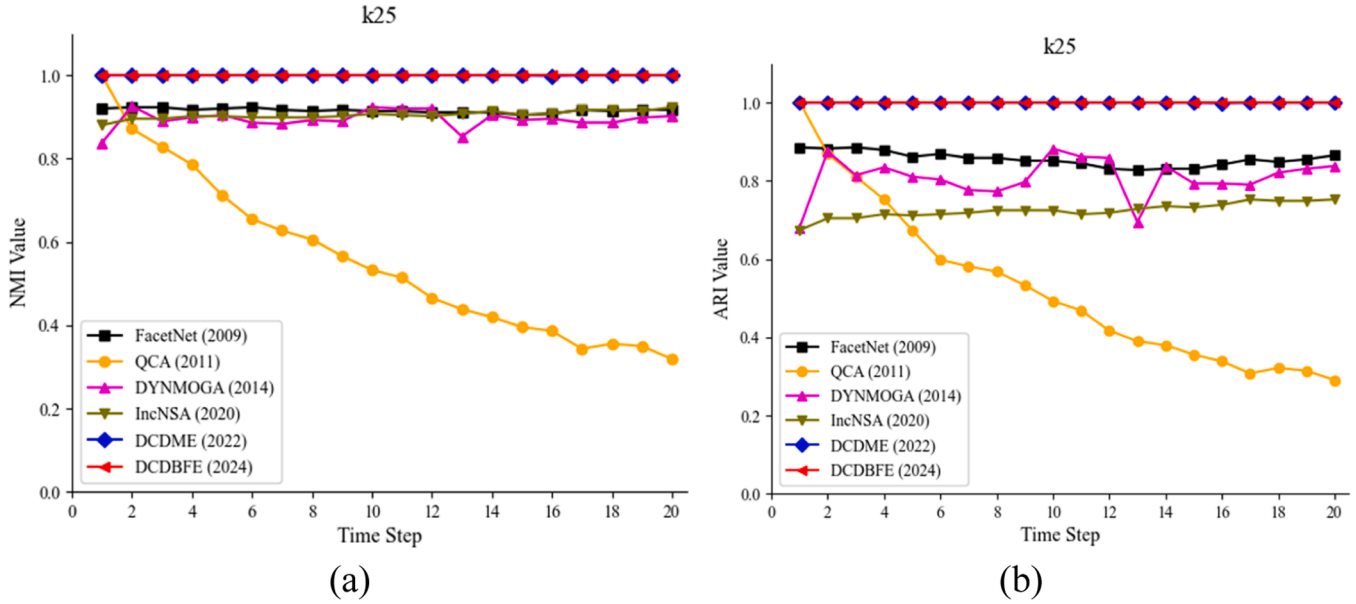


Fig. 19. Comparison of NMI and ARI indicators with average degree $k = 25$.

performance of DYNMOGA drops sharply after the 5th time step and continues to decline. FacetNet and IncNSA show moderate performance, but their NMI values fluctuate significantly, particularly around the 5th and 10th time steps. In Fig. 14(b), DCDBFE and DCDME show minor dips in NMI at the 6th, 12th, and 14th time steps, but these fluctuations are brief, and both algorithms quickly recover. Fig. 14(c) highlights similar patterns in ARI, where DCDBFE and DCDME again dominate with high values, while DYNMOGA struggles significantly, dropping to very low ARI values after the 5th time step. QCA and FacetNet exhibit significant variability in ARI values, indicating their challenges in handling complex dynamic changes. Fig. 14(d) zooms in on the fluctuations of DCDBFE and DCDME, particularly during the 6th, 12th, and 14th time steps, where ARI values momentarily drop to around 0.8 but quickly recover to near-perfect levels. Despite these minor fluctuations, DCDBFE and DCDME demonstrate superior adaptability and stability in managing high volumes of merger and split events, solidifying their position as top performers in dynamic community detection.

4) Effect of mixing parameter

In Fig. 15 and Fig. 16, the NMI and ARI indicators illustrate the performance of the algorithms at different mixing parameters ($\mu = 0.2$ and $\mu = 0.4$). For both the DCDME and DCDBFE algorithms, there were slight changes in the NMI and ARI values as the mixing parameter increased from 0.2 to 0.4, but overall, they remained relatively stable. This stability suggests that these algorithms effectively manage the overlap between communities, likely due to their use of AA and RA similarity measures, which are adept at capturing the local structural information even as communities become more mixed. However, when the mixing parameter increased further to $\mu = 0.8$, the results became very unstable, indicating that the algorithms struggled to maintain accurate community detection in highly mixed networks. At higher mixing

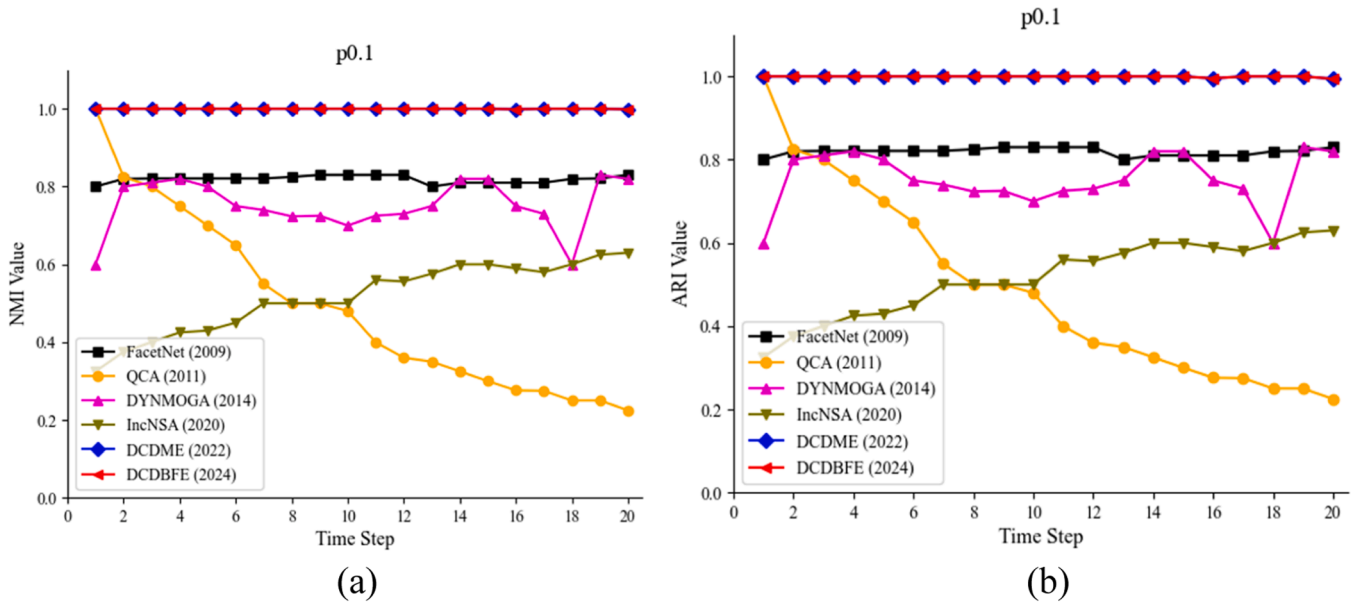


Fig. 20. Comparison of NMI and ARI indicators with probability $p = 0.1$.

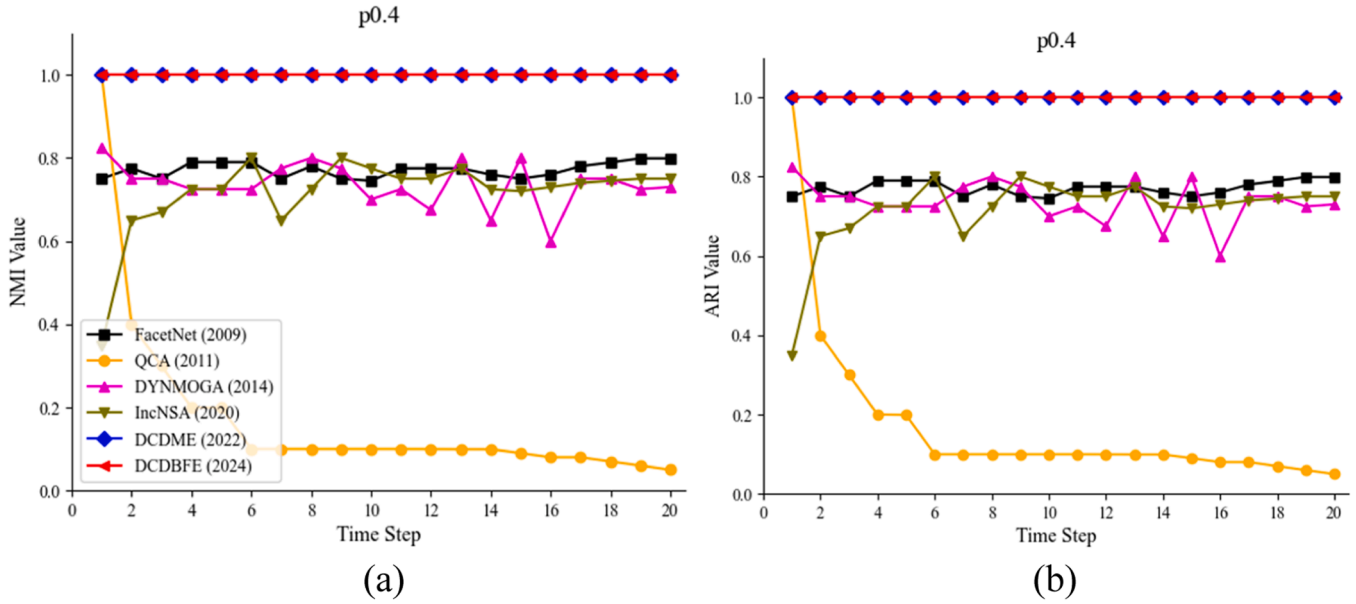


Fig. 21. Comparison of NMI and ARI indicators with probability $p = 0.4$.

parameters, the distinction between communities became more blurred, making it difficult for the similarity measures to discern the underlying structure, leading to significant fluctuations in NMI and ARI values. This instability reflects the challenge of accurately detecting communities in networks where nodes share connections with many different communities.

Fig. 15 compares the performance of six different algorithms in terms of NMI and ARI for dynamic community detection, with a mixing parameter $\mu = 0.2$. Fig. 15(a) and Fig. 15(c) present the variation of NMI and ARI values over 20 time steps, while Fig. 15(b) and Fig. 15(d) focus on a detailed comparison of DCDME and DCDBFE, showing their exceptional stability. In Fig. 15(a), DCDBFE and DCDME achieve near-perfect NMI values, maintaining around 1.0 throughout the time steps. By contrast, DYNMOGA experiences a significant decline from approximately 0.8 at the 5th time step to about 0.45 at the 15th. QCA and IncNSA exhibit moderate performance, with FacetNet staying relatively stable but lower than the top performers. In Fig. 15(c), the ARI values show a similar pattern: DCDBFE and DCDME remain close to 1.0, while DYNMOGA drops sharply. QCA and IncNSA maintain moderate levels, and FacetNet is more consistent. The consistently high values for DCDBFE and DCDME indicate their stability in handling dynamic changes, especially with low noise, while the significant drop in DYNMOGA suggests its lower adaptability to evolving community structures.

In Fig. 16(a) and Fig. 16(c), DCDBFE and DCDME consistently achieve near-perfect NMI and ARI values, showing exceptional stability across all time steps. Fig. 16(b) and Fig. 16(d) provide a zoomed-in view of the best-performing algorithms of DCDME and DCDBFE, showing their exceptional stability. In contrast, DYNMOGA steadily declines in both metrics after the initial time steps. At the 4th and 5th time steps, DCDBFE and DCDME maintain NMI and ARI values close to 1.0, reflecting superior performance. However, at the 6th and 8th time steps, there are slight dips, especially in NMI, where DCDBFE drops to around

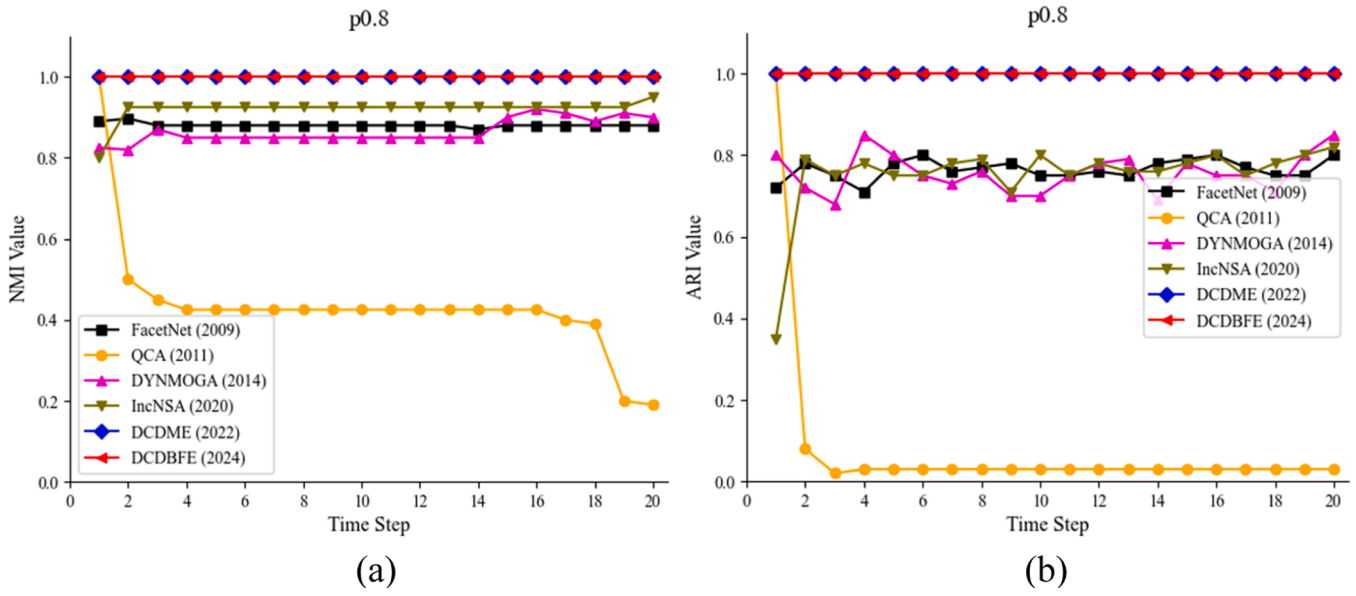


Fig. 22. Comparison of NMI and ARI indicators with probability $p = 0.8$.

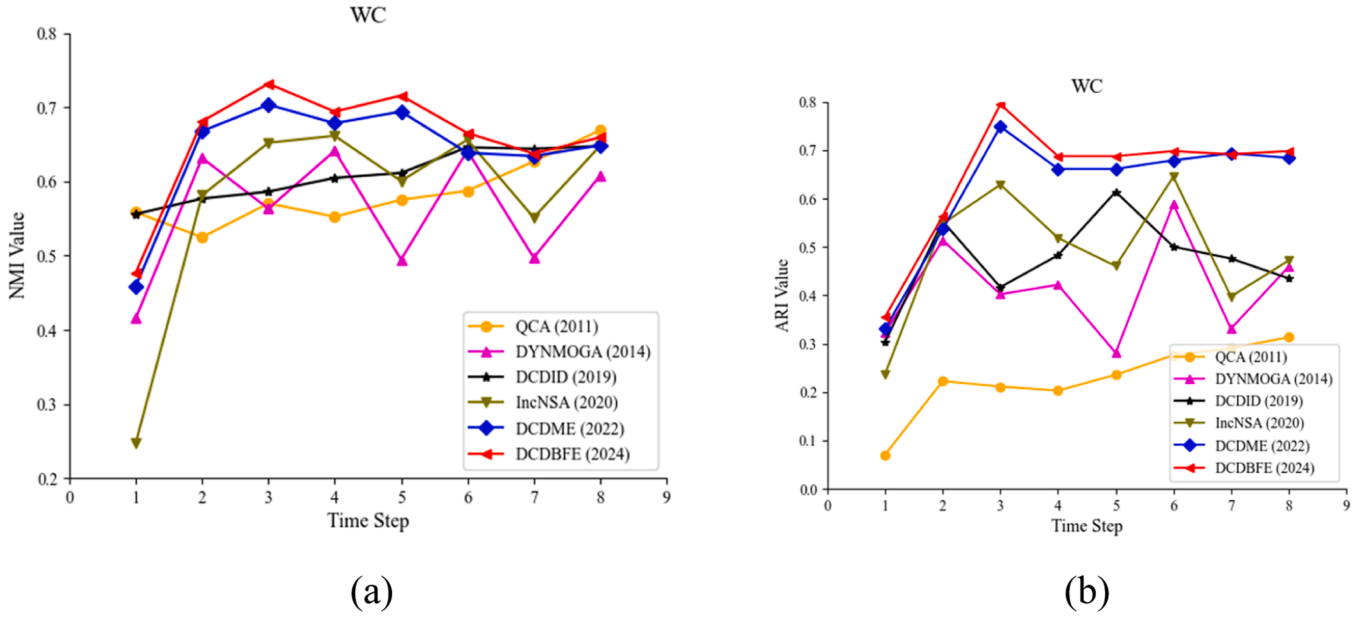


Fig. 23. Comparison methods on workplace contact in terms of NMI and ARI.

0.998. These fluctuations may be attributed to changes in the data structure, where community overlaps or dynamic adjustments temporarily challenge the algorithms' ability to detect clusters consistently. At the 15th and 16th time steps, both algorithms exhibit small fluctuations, but their values remain above 0.995, while DYNMOGA continues its sharp decline in ARI, falling to around 0.5 by the 15th time step. These results underscore the strong resilience of DCDBFE and DCDME in managing complex and dynamic data scenarios despite temporary fluctuations due to structural changes in the network.

5) **Effect of community density**

In Figs. 17–19, the NMI and ARI indicators compare the performance of various algorithms across different time steps and average degrees (k = 5, 15, and 25). For the DCDME and DCDBFE algorithms, the NMI and ARI values remained stable and consistent when the average degree was higher (k = 15 and k = 25), indicating that these algorithms effectively preserve the community structure in denser networks. The lack of change in these metrics suggests that the AA and RA similarity measures employed by these algorithms are particularly well-suited to handling networks with higher connectivity, where they can accurately capture the local structural information. However, when the average degree was lower (k = 5), there were some fluctuations in the NMI and ARI values, likely because sparser networks present fewer connections, making it more challenging for these similarity measures to maintain consistent community detection. In sparse networks, the lower number of edges reduces the effectiveness of AA and RA in capturing the necessary structural cues, leading to more noticeable changes in the community structure as detected by these algorithms.

Fig. 17(a) and Fig. 17(b) display the NMI values, while Fig. 17(c) and Fig. 17(d) show the ARI values. The algorithms compared are FacetNet, QCA, DYNMOGA, IncSNA, DCDME, and DCDBFE. In Fig. 17(a) and

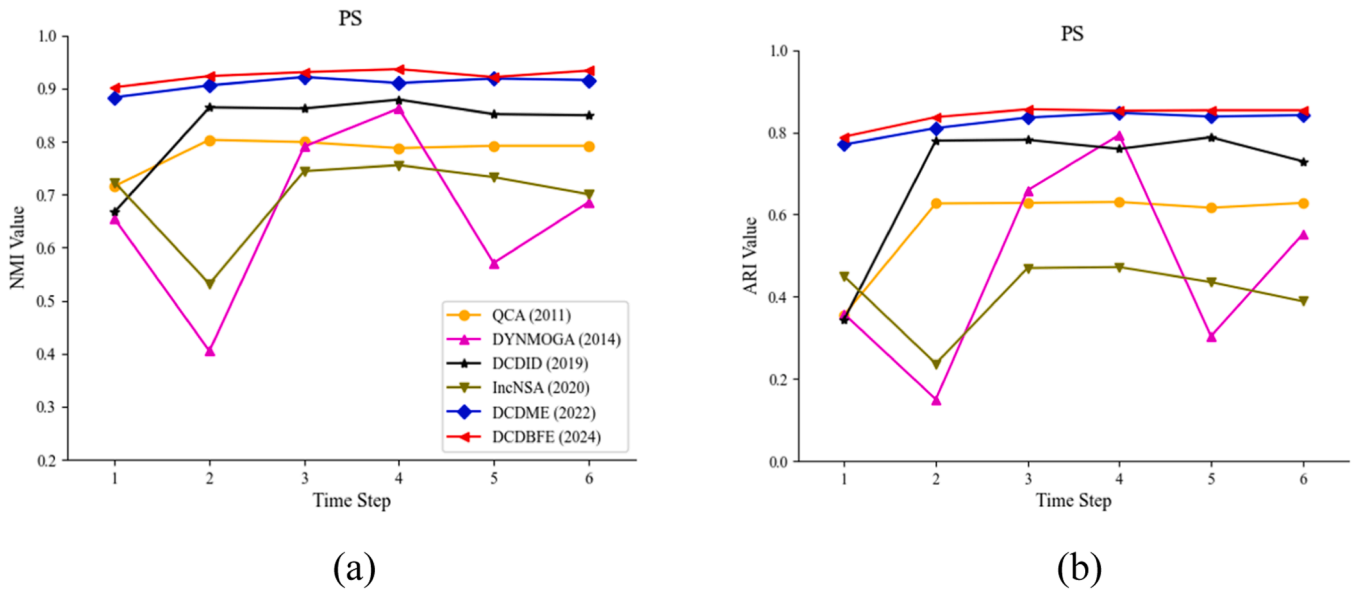


Fig. 24. Comparison methods on primary school in terms of NMI and ARI.

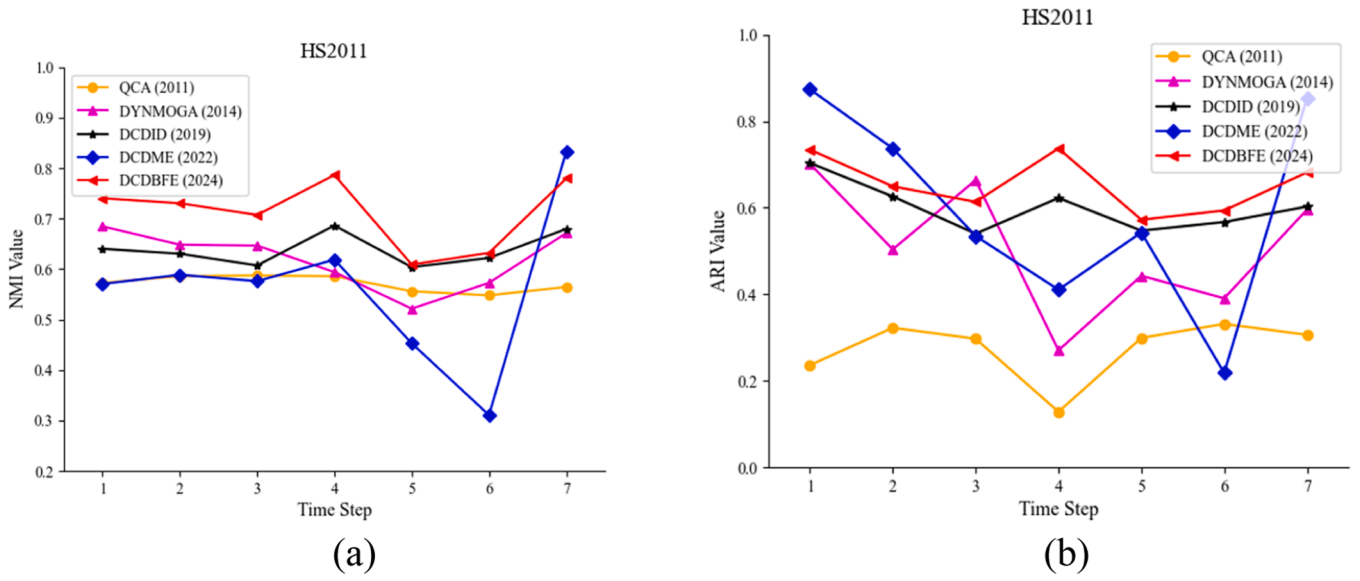


Fig. 25. Comparison methods on high school in 2011 in terms of NMI and ARI.

Fig. 17(c), which focus on the performance over time, DCDME and DCDBFE maintain higher NMI and ARI scores compared to the other algorithms. IncSNA shows a notable decrease in both NMI and ARI values as the time steps increase. Fig. 17(b) and Fig. 17(d) show variations in NMI and ARI across time steps for DCDBFE and DCDME, highlighting the slight fluctuations in the performance of these two algorithms, with DCDBFE consistently achieving higher stability and accuracy.

Fig. 18 and Fig. 19 compare the performance of clustering algorithms based on NMI and ARI over time for two different average degrees, $k = 15$ and $k = 25$. In both figures, the algorithms DCDME and DCDBFE demonstrate consistently high and stable performance across time steps for both NMI and ARI, regardless of the increase in the average degree. In contrast, IncSNA shows a steep decline in both metrics, particularly after the initial time steps, indicating its poor performance in maintaining accurate clustering over time as the network evolves. The other algorithms, such as FacetNet, QCA, and DYNMOGA, show moderate and relatively stable performance but are consistently outperformed by DCDME and DCDBFE. This suggests that DCDBFE is the most stable and accurate algorithm, offering superior clustering stability as measured by both NMI and ARI, particularly as the average degree k increases, highlighting its effectiveness in more complex networks.

6) Effect of vertex switching

We varied the value of p from 0.1 to 0.8 and fixed the parameters k = 5, and $\mu = 0.1$. Figs. 20–22 shows the performance of comparison algorithms with different p on NMI and ARI metrics and we only visualized the results of community detection when p is 0.1, 0.4, and 0.8 across different time steps. Notably, both the DCDME and DCDBFE algorithms maintained consistent NMI and ARI values throughout the time steps, unlike the other algorithms, which showed significant fluctuations. This consistency indicates that DCDME and DCDBFE are highly robust and stable in preserving the quality of the detected communities over time, regardless of the probability parameter. The reason why is because the stability of these algorithms can be attributed to their effective use of AA

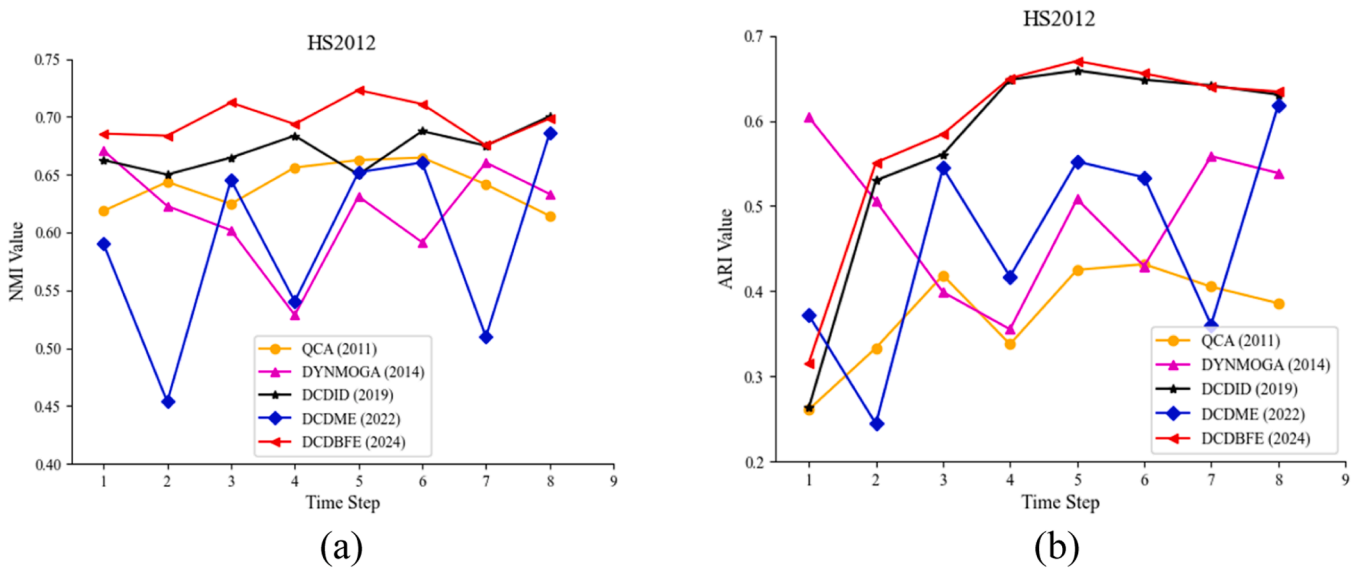


Fig. 26. Comparison methods on high school in 2012 in terms of NMI and ARI.

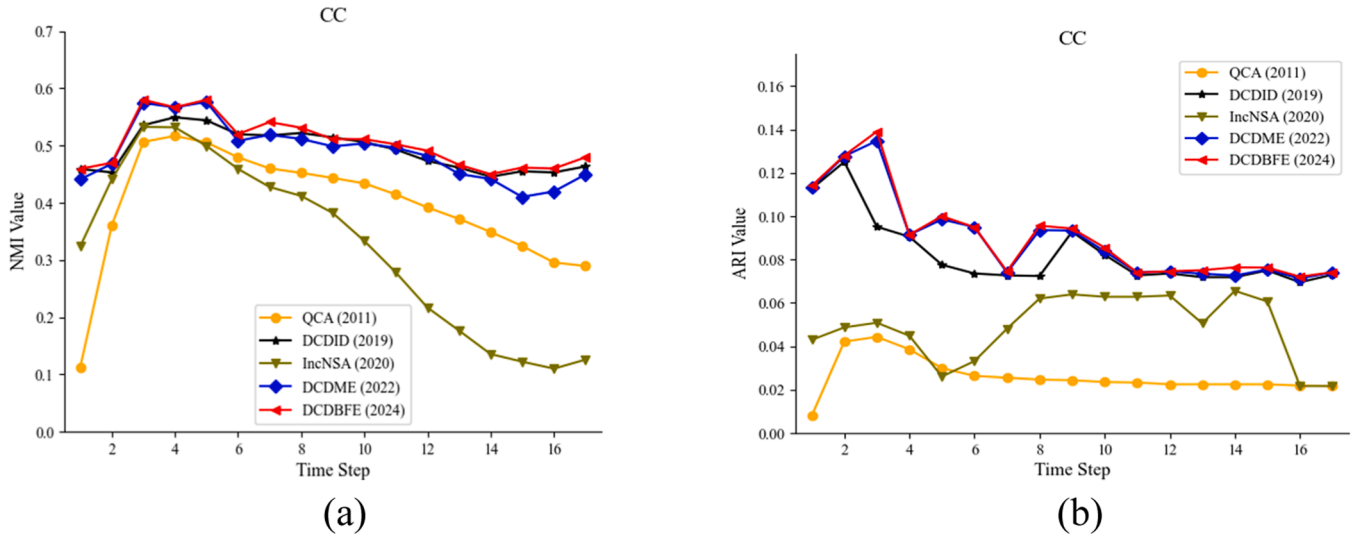


Fig. 27. Comparison methods on cummulative coauthor in term of NMI and ARI.

and RA similarity measures, which robustly capture the local structural information in dynamic networks, ensuring consistent community detection as the network evolves. Even as the probability of vertex switching increased from 0.1 to 0.8, potentially causing more drastic changes in the network structure, DCDME and DCDBFE continued to accurately track the underlying community structure, reflected in the unchanging NMI and ARI values across different time steps.

The other algorithms—QCA, FacetNet, DYNMOGA, and IncNSA—showed varying degrees of sensitivity to changes in the network's structure, as reflected in the fluctuations of their NMI and ARI values across different time paths and probabilities. QCA exhibited the most significant decline in performance, particularly as the probability of vertex switching increased, indicating its vulnerability to network changes. FacetNet, while more stable than QCA, still showed some variability, suggesting that it struggled to maintain consistent community detection as the network evolves. DYNMOGA and IncNSA displayed moderate fluctuations, with DYNMOGA generally maintaining higher NMI and ARI values than QCA and FacetNet, yet still showing sensitivity to increasing vertex switching probabilities. IncNSA, while more stable at higher probabilities, also exhibited some instability, particularly at lower probabilities. These fluctuations indicate that these algorithms are less stable in capturing and maintaining community structure in dynamic networks, particularly as the network undergoes more significant changes.

5.2. Results on real-world networks

Now we tested the performance of DCDBFE using four real-world networks that have received significant attention from many researchers [66–68,81,91]. The details of these six real-world networks are explained in Section 4.1. In the experiments, we used NMI and ARI to evaluate the quality of the extracted community structures due to the absence of ground-truth community structures. The comparison of algorithms on the six real-world dynamic networks is shown in Figs. 23–28, illustrating the overall performance of each approach.

Fig. 23 shows a comparative analysis of different algorithms on dynamic workplace contact networks, evaluating their performance using NMI and ARI over 8th time steps. The results indicated that DCDBFE and DCDME significantly outperformed older algorithms like QCA, DYNMOGA, IncNSA, and DCDID in terms of both NMI and ARI. DCDBFE

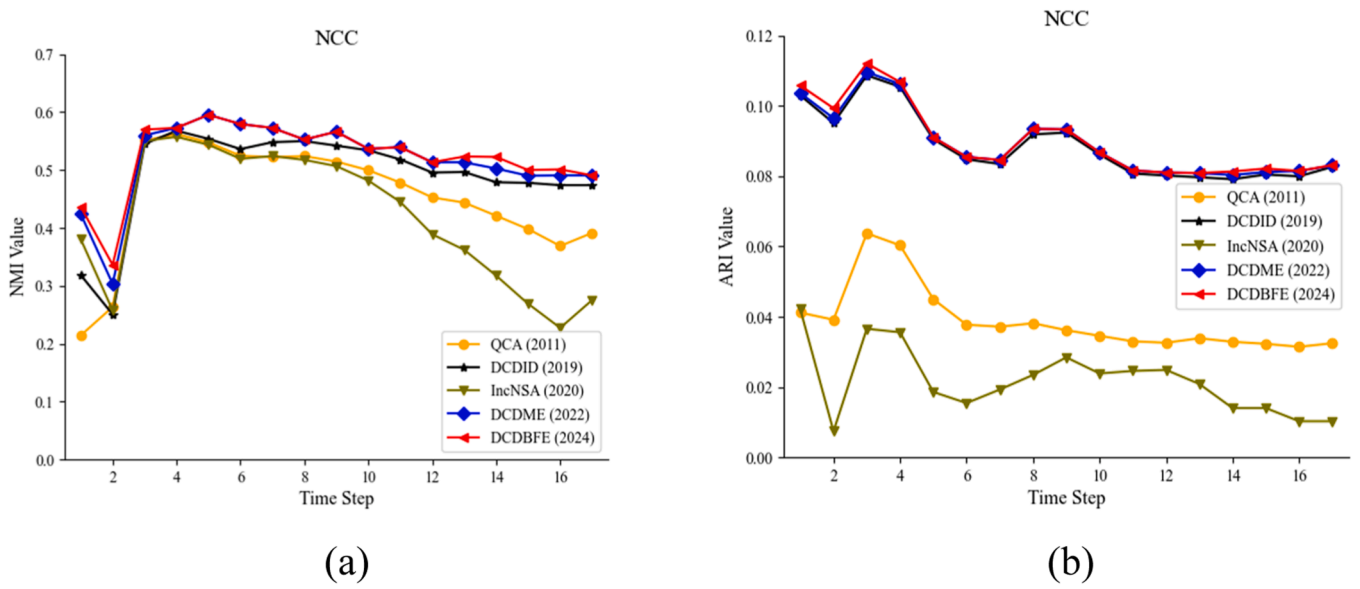


Fig. 28. Comparison methods on non-cummulative coauthor in term of NMI and ARI.

consistently achieved NMI values above 0.7, peaking at 0.75 at 6th time step, and ARI values around 0.75 at 5th time step, demonstrating exceptional clustering accuracy and stability over time. DCDME also maintained high performance, with NMI peaking at 0.7 at 7th time step and ARI around 0.65 at the same time step. These results highlight the stability and adaptability of these newer models in dynamic conditions, where older models like QCA showed significant declines in performance, with NMI values starting at 0.3 and peaking at 0.6, and ARI values starting at 0.2 and peaking at 0.5. The superior performance of DCDBFE and DCDME is attributed to their advanced design, which effectively handles frequent changes in network structure, thereby providing more stable and accurate clustering results.

Fig. 24 compares the performance of different clustering methods on primary school networks using NMI and ARI metrics. DCDME and DCDBFE maintain consistently high and stable values for both NMI and ARI across all time steps, demonstrating superior performance compared to other methods like DYNMOGA and IncSNA, which show significant fluctuations and lower overall clustering accuracy, suggesting they struggle to adapt effectively to the evolving structure of the network. NMI results of DCDBFE indicate their strong ability to handle dynamic changes in primary school networks, maintaining stable and accurate clustering performance.

Fig. 25 and Fig. 26 compare the performance of clustering methods in high school networks for the years 2011 and 2012, respectively, using NMI and ARI metrics. In both figures, DCDBFE and DCDME generally show superior performance, with higher and more stable NMI and ARI values compared to the other algorithms, although there are some fluctuations in certain time steps. Methods like QCA and DYNMOGA perform notably worse, showing consistently lower and less stable values, particularly in ARI, highlighting their reduced adaptability to dynamic changes in the high school networks over time.

Fig. 27 and Fig. 28 clearly demonstrate the superior performance of DCDBFE in both cumulative and non-cumulative coauthor networks. DCDBFE consistently maintains the highest NMI and ARI values, indicating its stability in capturing and preserving community structures as the network evolves. DCDME also performs exceptionally well, closely matching the performance of DCDBFE in both cumulative and non-cumulative coauthor networks. This algorithm demonstrates minimal fluctuations over time, outperforming other methods like QCA, IncNSA, and DCDID, which show significant declines in both metrics. Notably, DCDBFE achieves a balance between stability and accuracy across all time steps, particularly in comparison to older algorithms that fail to keep pace with the network's dynamic changes. This consistent performance across both networks positions DCDBFE as the most reliable and effective algorithm for dynamic community detection.

In conclusion, numerous studies on dynamic networks have shown that the DCDBFE algorithm remains capable and can effectively detect communities by leveraging network events. In synthetic dynamic networks, the difficulty of community detection is often influenced by the mixing parameter μ. When the values of the mixing parameter exceed 0.4, many community detection algorithms face significant challenges. In real-world network, the DCDBFE algorithm also performed well with varying characteristics.

5.3. *Limitation and challenges*

DCDBFE addresses the challenge of identifying community structures in dynamic networks. The challenge is from the dataset itself. For artificial networks, first parameter tuning remains a challenge due to changing network conditions over time. Artificial networks often require careful tuning of parameters to simulate realistic dynamic changes, which can be time-consuming and may not accurately reflect real-world scenarios. Second, the algorithm's time complexity increases as the number of time steps grows, impacting computational efficiency. While artificial networks provide controlled environments for testing, they may not capture the full complexity of real-world networks. This can lead to over-optimistic performance metrics that do not fully translate to practical applications. Third, there's an inherent trade-off between detection accuracy and computational speed. Striking the right balance is crucial. Fourth, real-world networks exhibit diverse characteristics, making it challenging to design an algorithm that performs consistently across different domains and scenarios.

The next problem we encountered occurred when running the events of birth 5 and death 5, which resulted in an error. A significant limitation in our analysis arose from an IndexError, specifically when attempting to access an element at position 990 in an array of size 990, where valid indices range from 0 to 989 due to zero-based indexing. This error highlights a common challenge in handling array indices, particularly in contexts requiring precise access to array elements within their bounds. Such errors typically result from off-by-one mistakes, where the code inadvertently exceeds the array's valid index range, leading to runtime failures. This limitation underscores the need for careful index management, especially when dealing with dynamic arrays or loops that iterate over data structures, to ensure robustness and prevent access

violations that could disrupt the analysis.

Challenges for the real-world networks, the data availability and quality are crucial. First, the real-world dynamic networks, such as those in primary schools or workplaces, can suffer from issues like incomplete data, noisy measurements, and irregular updates. These challenges can affect the accuracy and reliability of clustering results. Second, scalability. As real-world networks grow in size and complexity, ensuring the scalability of algorithms like DCDBFE becomes crucial. High computational and memory requirements can limit the practical applicability of these algorithms in large-scale networks. Third, evaluation Metrics. While NMI and ARI are robust metrics, they may not capture all aspects of clustering performance in dynamic networks. Developing new metrics that better reflect the intricacies of dynamic changes and community evolution remains a challenge.

In summary, the study highlights the exceptional performance of DCDBFE in dynamic network clustering, evidenced by its high NMI and ARI values. However, the limitations encountered during testing on both artificial and real-world networks underscore the need for further research to address data quality, scalability, and the development of more comprehensive evaluation metrics.

5.4. Future research

Based on the proposed DCDBFE algorithm, we may further analyze the algorithm with COVID-19 as a case study. The application of the DCDBFE algorithm to COVID-19 transmission networks provides significant insights into the dynamic evolution of community structures, revealing how persistent clusters, merging events, and the impact of interventions such as lockdowns can be analyzed to understand and manage virus spread. By tracking the stability and changes in communities over time, the algorithm helps identify sustained transmission within specific groups and the effectiveness of public health measures in breaking transmission chains. These findings have practical implications for public health officials, who can use this information to target interventions more effectively and design strategies for managing future pandemics. Additionally, the analysis opens avenues for future research, such as exploring the role of different network topologies in disease spread and refining dynamic community detection methods to enhance their predictive capabilities in real-world scenarios.

This contribution is directly aligned with several Sustainable Development Goals (SDGs). For example, SDG 9 (Industry, Innovation, and Infrastructure), where the development of robust

computational tools for managing evolving data in dynamic systems (e.g., telecommunications, transport, and energy grids) enhances infrastructure stability and fosters innovation, especially in real-time community detection for industries like smart cities and IoT applications. Another furter research is realted to SDG 11 (Sustainable Cities and Communities), which incremental dynamic community detection optimizes urban mobility, energy distribution, and resource management in smart cities, enabling planners to better predict and adapt to network changes. It also helps in identifying stable community structures in social, transportation, and communication networks, contributing to sustainable urban development.

6. Conclusions

This novel work presents dynamic community detection based on bird flock effect as an eco-system perspective, called DCDBFE. The ecosystem perspective in network analysis provides a comprehensive framework to understand the complex interactions and dynamics within a system, whether it's a flock of birds or an innovation ecosystem. DCDBFE is an inspired by bird flock effect using incremental community detection. The design of this algorithm imitates the three basic rules of bird flock: separation, alignment, and cohesion. DCDBFE successfully found high quality community structure naturally in dynamic networks, without settings any parameter. DCDBFE was tested against several well-known dynamic algorithms on the extended LFR and four real-world networks. A lot of tests showed that DCDBFE was good at finding groups in various types of dynamic networks by using NMI and ARI for checking the accuracy. It establishes a benchmark for evaluating the robustness and accuracy of dynamic clustering algorithms in maintaining high clustering accuracy despite frequent network changes, provides insights into the limitations of existing algorithms, and offers valuable guidance for practitioners and researchers by identifying the most reliable algorithms for dynamic environments. A future suggestion for the next researcher is to apply this algorithm to other possible real-world networks. Especially, in healthcare and medicine dataset or in the field of intelligent transportation, such as unnamed aerial vehicles (UAV) or drone movement and relate with the Sustainable Development Goals (SDGs).

CRediT authorship contribution statement

Norhazwani Mohd Yunos: Supervision. **Eko Arip Winanto:** Software. **Zejun Sun:** Supervision. **Mohd Fariduddin Mukhtar:** Writing – review & editing. **Iskandar Waini:** Funding acquisition. **Zuraida Abal Abas:** Supervision. **Siti Haryanti Hairol Anuar:** Writing – original draft, Methodology, Conceptualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was funded by Centre for Research and Innovation Management (CRIM), and Universiti Teknikal Malaysia Melaka (UTeM) (JURNAL/2022/FTKIP/Q00088).

References

[1] C. Zhang, B. Zheng, F. Tsung, Multi-view metro station clustering based on passenger flows: a functional data-edged network community detection approach, Data Min. Knowl. Discov. 37 (3) (May 2023) 1154–1208, https://doi.org/10.1007/s10618-023-00916-w.

[2] S. Chatterjee, B.S. Sanjeev, Community detection in Epstein-Barr virus associated carcinomas and role of tyrosine kinase in etiological mechanisms for oncogenesis, Microb. Pathog. 180 (May) (Jul. 2023) 106115, https://doi.org/10.1016/j.micpath.2023.106115.

[3] L. Jiang, L. Shi, L. Liu, J. Yao, B. Yuan, Y. Zheng, An efficient evolutionary user interest community discovery model in dynamic social networks for internet of people, IEEE Internet Things J. 6 (6) (Dec. 2019) 9226–9236, https://doi.org/10.1109/JIOT.2019.2893625.

[4] Z.A. Abas, et al., Analytics: a review of current trends, Future Appl. Chall. 9 (I) (2020).

[5] Q. Zhang, V.Y.F.F. Tan, C. Suh, Community detection and matrix completion with social and item similarity graphs, IEEE Trans. Signal Process. 69 (Dec. 2021) 917–931, https://doi.org/10.1109/TSP.2021.3052033.

[6] A. Daher, M. Coupechoux, P. Godlewski, P. Ngouat, P. Minot, A dynamic clustering algorithm for multi-point transmissions in mission-critical communications, IEEE Trans. Wirel. Commun. 19 (7) (2020) 4934–4946, https://doi.org/10.1109/TWC.2020.2988382.

[7] H. Damgacioglu, E. Celik, N. Celik, Intra-cluster distance minimization in DNA methylation analysis using an advanced tabu-based iterative kk-medoids clustering algorithm (T-CLUST),", IEEE/ACM Trans. Comput. Biol. Bioinforma. 17 (4) (2020) 1241–1252, https://doi.org/10.1109/TCBB.2018.2886006.

[8] D. Lin, Q. Wang, An energy-efficient clustering algorithm combined game theory and dual-cluster-head mechanism for WSNs, IEEE Access 7 (2019) 49894–49905, https://doi.org/10.1109/ACCESS.2019.2911190.

[9] F. Saggese, M. Moretti, A. Abrardo, A quasi-optimal clustering algorithm for MIMO-NOMA downlink systems, IEEE Wirel. Commun. Lett. 9 (2) (2020) 152–156, https://doi.org/10.1109/LWC.2019.2946548.

[10] M.S. Talib, et al., A center-based stable evolving clustering algorithm with grid partitioning and extended mobility features for VANETs, IEEE Access 8 (2020) 169908–169921, https://doi.org/10.1109/ACCESS.2020.3020510.

[11] N.H.M.M. Shrifan, G.N. Jawad, N.A.M. Isa, M.F. Akbar, Microwave nondestructive testing for defect detection in composites based on k-means clustering algorithm, IEEE Access 9 (2021) 4820–4828, https://doi.org/10.1109/ACCESS.2020.3048147.

[12] Z. Li, Y. Li, W. Lu, J. Huang, Crowdsourcing logistics pricing optimization model based on DBSCAN clustering algorithm, IEEE Access 8 (2020) 1-1, https://doi.org/10.1109/ACCESS.2020.2995063.
[13] C. Jiang, J. Wan, H. Abbas, An Edge computing node deployment method based on improved k -means clustering algorithm for smart manufacturing, IEEE Syst. J. 15 (2) (Jun. 2021) 2230–2240, https://doi.org/10.1109/JSYST.2020.2986649.
[14] B. Heinz, J. Henkel, Balancing wind energy and participating in electricity markets with a fuel cell population, Energy 48 (1) (Dec. 2012) 188–195, https://doi.org/10.1016/j.energy.2012.07.002.
[15] V. Alcácer, V. Cruz-Machado, Scanning the industry 4.0: a literature review on technologies for manufacturing systems, " *Eng. Sci. Technol. Int. J.* 22 (3) (2019) 899–919, https://doi.org/10.1016/j.jestch.2019.01.006.
[16] X. Xueshuo, et al., AWAP: Adaptive weighted attribute propagation enhanced community detection model for bitcoin de-anonymization, Appl. Soft Comput. 109 (Sep. 2021) 107507, https://doi.org/10.1016/j.asoc.2021.107507.
[17] S.H. Hairol Anuar, Z.A. Abas, M.F. Mukhtar, N.H. Miswan, Community detection in practice: a review of real-world applications across six themes, Int. J. Acad. Res. Bus. Soc. Sci. 14 (10) (Oct. 2024) 953–996, https://doi.org/10.6007/IJARBSS/v14-i10/23160.
[18] A. Karatas, S. Sahin, A Novel Efficient Method For Tracking Evolution Of Communities In Dynamic Networks, IEEE Access 10 (2022) 46276–46290, https://doi.org/10.1109/ACCESS.2022.3170476.
[19] J. Jia, L. Li, DynaMic Community Detection Based On Similarity Of Social Network Nodes. 2022 4th International Academic Exchange Conference on Science and Technology Innovation (IAECST), IEEE, Dec. 2022, pp. 1147–1151, https://doi.org/10.1109/IAECST57965.2022.10061958.
[20] N. Dakiche, F. Benbouzid-Si Tayeb, Y. Slimani, K. Benatchba, Tracking community evolution in social networks: A Survey, Inf. Process. Manag. 56 (3) (May 2019) 1084–1102, https://doi.org/10.1016/j.ipm.2018.03.005.
[21] S.H. Hairol Anuar, Z. Abal Abas, N. Md Yunos, M.F. Mukhtar, T. Setiadi, A. S. Shibghatullah, IdentifyIng Communities With Modularity Metric Using Louvain And Leiden Algorithms, Pertanika J. Sci. Technol. 32 (3) (Apr. 2024) 1285–1300, https://doi.org/10.47836/pjst.32.3.16.
[22] A. Karaaslanli, M. Ortiz-Bouza, T.T.K. Munia, S. Aviyente, Community detection in multi-frequency EEG networks, Sci. Rep. 13 (1) (May 2023) 8114, https://doi.org/10.1038/s41598-023-35232-2.
[23] K. Nallusamy, K.S. Easwarakumar, Classifying schizophrenic and controls from fMRI data using graph theoretic framework and community detection, Netw. Model. Anal. Heal. Inform. Bioinforma. 12 (1) (Apr. 2023) 19, https://doi.org/10.1007/s13721-023-00415-4.
[24] H. Gao, Q. Zhu, W. Wang, Optimal deployment of large-scale wireless sensor networks based on graph clustering and matrix factorization, EURASIP J. Adv. Signal Process. 2023 (1) (2023), https://doi.org/10.1186/s13634-023-00995-3.
[25] S. Wang, X. Yao, D. Gong, H. Tu, Overlapping community detection in software ecosystem based on pheromone guided personalized PageRank algorithm, Inf. Softw. Technol. 163 (June) (2023) 107283, https://doi.org/10.1016/j.infsof.2023.107283.
[26] N. Alotaibi, D. Rhouma, A review on community structures detection in time evolving social networks, *J. King Saud. Univ. - Comput. Inf. Sci.* 34 (8) (Sep. 2022) 5646–5662, https://doi.org/10.1016/j.jksuci.2021.08.016.
[27] F. Bouhatem, A.A. El Hadj, F. Souam, A. Dafeur, Incremental methods for community detection in both fully and growing dynamic networks, Acta Univ. Sapientia, Inform. 13 (2) (Dec. 2021) 220–250, https://doi.org/10.2478/ausi-2021-0010.
[28] M. Seifikar, S. Farzi, A comprehensive study of online event tracking algorithms in social networks, J. Inf. Sci. 45 (2) (Apr. 2019) 156–168, https://doi.org/10.1177/0165551518785548.
[29] S. Bansal, S. Bhowmick, P. Paymal, Fast community detection for dynamic complex networks, *Commun. Comput. Inf. Sci. (Conf. Pap.)* (2011) 196–207, https://doi.org/10.1007/978-3-642-25501-4_20.
[30] T. Hartmann, A. Kappes, D. Wagner, Clustering Evolving Networks, *Lect. Notes Comput. Sci. (Incl. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma.)* (2016) 280–329, https://doi.org/10.1007/978-3-319-49487-6_9.
[31] T. Aynaud, E. Fleury, J.-L. Guillaume, Q. Wang, Communities in evolving networks: definitions, detection, and analysis techniques, *Model. Simul. Sci., Eng. Technol.* (2013) 159–200, https://doi.org/10.1007/978-1-4614-6729-8_9.
[32] G. Rossetti, R. Cazabet, Community discovery in dynamic networks: a survey, ACM Comput. Surv. 51 (2) (Mar. 2019) 1–37, https://doi.org/10.1145/3172867.
[33] H. Jiang, et al., Exploring the inter-monthly dynamic patterns of Chinese urban spatial interaction networks based on baidu migration data, " *ISPRS Int. J. Geo-Inf.* 11 (9) (Sep. 2022) 486, https://doi.org/10.3390/ijgi11090486.
[34] S.S. Mishra, S.S. Singh, S.S. Mishra, B. Biswas, TCD2: Tree-based community detection in dynamic social networks, Expert Syst. Appl. 169 (September 2020) (May 2021) 114493, https://doi.org/10.1016/j.eswa.2020.114493.
[35] J. Liu, Y. Shao, S. Su, Multiple local community detection via high-quality seed identification over both static and dynamic networks, Data Sci. Eng. (May 2021), https://doi.org/10.1007/s41019-021-00160-6.
[36] K. Kadkhoda Mohammadmosaferi, H. Naderi, Evolution of communities in dynamic social networks: an efficient map-based approach, Expert Syst. Appl. 147 (Jun. 2020) 113221, https://doi.org/10.1016/j.eswa.2020.113221.
[37] D. Zhuang, et al., DynaMo: Dynamic community detection by incrementally maximizing modularity, IEEE Trans. Knowl. Data Eng. 33 (5) (May 2019) 1-1, https://doi.org/10.1109/TKDE.2019.2951419.
[38] H.S. Cheraghchi, A. Zakerolhosseini, S. Bagheri Shouraki, E. Homayounvala, A novel granular approach for detecting dynamic online communities in social network, Soft Comput. 23 (20) (Oct. 2019) 10339–10360, https://doi.org/10.1007/s00500-018-3585-z.
[39] D. Zhuang, M.J. Chang, M. Li, DynaMo: dynamic community detection by incrementally maximizing modularity, IEEE Trans. Knowl. Data Eng. 171 (March 2020) (Jun. 2019) 1-1, https://doi.org/10.1109/TKDE.2019.2951419.
[40] H.S. Cheraghchi, A. Zakerolhosseini, Toward a novel art inspired incremental community mining algorithm in dynamic social network, Appl. Intell. 46 (2) (Mar. 2017) 409–426, https://doi.org/10.1007/s10489-016-0838-3.
[41] R. Interdonato, A. Tagarelli, D. Ienco, A. Sallaberry, P. Poncelet, Local community detection in multilayer networks, Data Min. Knowl. Discov. 31 (5) (Sep. 2017) 1444–1479, https://doi.org/10.1007/s10618-017-0525-y.
[42] J. He, D. Chen, C. Sun, Y. Fu, W. Li, Efficient stepwise detection of communities in temporal networks, " *Phys. A Stat. Mech. its Appl.* 469 (Mar. 2017) 438–446, https://doi.org/10.1016/j.physa.2016.11.019.
[43] S. Ranjkesh, B. Masoumi, S.M. Hashemi, A novel robust memetic algorithm for dynamic community structures detection in complex networks, World Wide Web 27 (1) (Jan. 2024) 3, https://doi.org/10.1007/s11280-024-01238-7.
[44] H. Ma, K. Wu, H. Wang, J. Liu, Higher order knowledge transfer for dynamic community detection with great changes, IEEE Trans. Evol. Comput. 28 (1) (Feb. 2024) 90–104, https://doi.org/10.1109/TEVC.2023.3257563.
[45] L. Ni, Q. Li, Y. Zhang, W. Luo, V.S. Sheng, LSADEN: local spatial-aware community detection in evolving geo-social networks, IEEE Trans. Knowl. Data Eng. (2024) 1–16, https://doi.org/10.1109/TKDE.2023.3348975.
[46] S. Mishra, S.S. Singh, S. Mishra, B. Biswas, Multi-objective based unbiased community identification in dynamic social networks, Comput. Commun. 214 (ember 2023) (2024) 18–32, https://doi.org/10.1016/j.comcom.2023.11.021.
[47] R. Márquez, R. Weber, Dynamic community detection including node attributes, Expert Syst. Appl. 223 (March) (Aug. 2023) 119791, https://doi.org/10.1016/j.eswa.2023.119791.
[48] Y. Sun, X. Sun, Z. Liu, Y. Cao, J. Yang, Core node knowledge based multi-objective particle swarm optimization for dynamic community detection, Comput. Ind. Eng. 175 (December 2022) (2023) 108843, https://doi.org/10.1016/j.cie.2022.108843.
[49] W. Wang, Q. Li, W. Wei, An adaptive dynamic community detection algorithm based on multi-objective evolutionary clustering, Int. J. Intell. Comput. Cybern. (Oct. 2023), https://doi.org/10.1108/IJICC-07-2023-0188.
[50] X. Li, X. Zhen, X. Qi, H. Han, L. Zhang, Z. Han, Dynamic community detection based on graph convolutional networks and contrastive learning, Chaos, Solitons Fractals 176 (2022) (Nov. 2023) 114157, https://doi.org/10.1016/j.chaos.2023.114157.
[51] R. Márquez, R. Weber, Dynamic community detection including node attributes, Expert Syst. Appl. 223 (Aug. 2023) 119791, https://doi.org/10.1016/j.eswa.2023.119791.
[52] M.E. Samie, E. Behbood, A. Hamzeh, Local community detection based on influence maximization in dynamic networks, Appl. Intell. 53 (15) (Aug. 2023) 18294–18318, https://doi.org/10.1007/s10489-022-04403-5.
[53] N. Chinichian, et al., A fast and intuitive method for calculating dynamic network reconfiguration and node flexibility, Front. Neurosci. 17 (2023), https://doi.org/10.3389/fnins.2023.1025428.
[54] P. Jiao, T. Li, H. Wu, C.-D. Wang, D. He, W. Wang, HB-DSBM: modeling the dynamic complex networks from community level to node level, IEEE Trans. Neural Netw. Learn. Syst. 34 (11) (Nov. 2023) 8310–8323, https://doi.org/10.1109/TNNLS.2022.3149285.
[55] D. Li, X. Ma, M. Gong, Joint learning of feature extraction and clustering for large-scale temporal networks, IEEE Trans. Cybern. 53 (3) (Mar. 2023) 1653–1666, https://doi.org/10.1109/TCYB.2021.3107679.
[56] H. Long, X. Li, X. Liu, W. Wang, BBTA: Detecting communities incrementally from dynamic networks based on tracking of backbones and bridges, Appl. Intell. 53 (1) (Jan. 2023) 1084–1100, https://doi.org/10.1007/s10489-022-03418-2.
[57] L. Cai, J. Zhou, D. Wang, Improving temporal smoothness and snapshot quality in dynamic network community discovery using NOME algorithm, PeerJ Comput. Sci. 9 (2023) 1–21, https://doi.org/10.7717/peerj-cs.1477.
[58] A.R. Costa, C.G. Ralha, AC2CD: An actor–critic architecture for community detection in dynamic social networks, Knowl. -Based Syst. 261 (Feb. 2023) 110202, https://doi.org/10.1016/j.knosys.2022.110202.
[59] M. Mazza, G. Cola, M. Tesconi, Modularity-based approach for tracking communities in dynamic social networks, Knowl. -Based Syst. 281 (September) (Dec. 2023) 111067, https://doi.org/10.1016/j.knosys.2023.111067.
[60] X. Li, et al., Local node feature modeling for edge computing based on network embedding in dynamic networks, J. Parallel Distrib. Comput. 171 (Jan. 2023) 98–110, https://doi.org/10.1016/j.jpdc.2022.09.013.
[61] Y. Zhao, B.Y. Chen, F. Gao, X. Zhu, Dynamic community detection considering daily rhythms of human mobility, Travel Behav. Soc. 31 (December 2022) (Apr. 2023) 209–222, https://doi.org/10.1016/j.tbs.2022.12.009.
[62] T. Li, et al., Exploring Temporal Community Structure via Network Embedding, IEEE Trans. Cybern. 53 (11) (Nov. 2023) 7021–7033, https://doi.org/10.1109/TCYB.2022.3168343.
[63] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, B.L. Tseng, Analyzing communities and their evolutions in dynamic social networks, ACM Trans. Knowl. Discov. Data 3 (2) (Apr. 2009) 1–31, https://doi.org/10.1145/1514888.1514891.
[64] F. Folino, C. Pizzuti, An Evolutionary Multiobjective Approach for Community Discovery in Dynamic Networks, IEEE Trans. Knowl. Data Eng. 26 (8) (Aug. 2014) 1838–1852, https://doi.org/10.1109/TKDE.2013.131.
[65] S. Elhishi, M. Abu-Elkheir, A. Abou Elfetouh, Perspectives on the evolution of online communities, Behav. Inf. Technol. 38 (6) (Jun. 2019) 592–608, https://doi.org/10.1080/0144929X.2018.1546901.

[66] X. Su, J. Cheng, H. Yang, M. Leng, W. Zhang, X. Chen, IncNSA: Detecting communities incrementally from time-evolving networks based on node similarity, Int. J. Mod. Phys. C. 31 (07) (Jul. 2020) 2050094, https://doi.org/10.1142/S0129183120500941.
[67] Z. Sun, J. Sheng, B. Wang, A. Ullah, F. Khawaja, Identifying communities in dynamic networks using information dynamics, Entropy 22 (4) (Apr. 2020) 425, https://doi.org/10.3390/e22040425.
[68] Z. Sun, et al., Dynamic community detection based on the Matthew effect, Phys. A Stat. Mech. its Appl. 597 (4) (Jul. 2022) 127315, https://doi.org/10.1016/j.physa.2022.127315.
[69] Y. Shen, C. Wei, Multi-UAV flocking control with individual properties inspired by bird behavior, Aerosp. Sci. Technol. 130 (2022) 107882, https://doi.org/10.1016/j.ast.2022.107882.
[70] J. Shang, Y. Li, Y. Sun, F. Li, Y. Zhang, J. Liu, MOPIO: a multi-objective pigeon-inspired optimization algorithm for community detection, Symmetry (Basel) 13 (1) (Dec. 2020) 49, https://doi.org/10.3390/sym13010049.
[71] M.A. Peeples, R. J. Bischoff, Archaeological networks, community detection, and critical scales of interaction in the U.S. Southwest/Mexican Northwest, J. Anthropol. Archaeol. 70 (March) (Jun. 2023) 101511, https://doi.org/10.1016/j.jaa.2023.101511.
[72] B. Erfianto, I. Muchtadi-Alamsyah, Stability and vulnerability of bird flocking behaviour: a mathematical analysis, HAYATI J. Biosci. 26 (4) (2019) 179–184, https://doi.org/10.4308/hjb.26.4.179.
[73] A. Bellaachia, A. Bari, SFLOSCAN: A biologically-inspired data mining framework for community identification in dynamic social networks, *IEEE Ssci 2011 - Symp. Ser. Comput. Intell. - SIS 2011 2011 IEEE Symp. Swarm Intell.*, no. Ssci (2011) 156–163, https://doi.org/10.1109/SIS.2011.5952580.
[74] M.F. Mukhtar, et al., Integrating local and global information to identify influential nodes in complex networks, Sci. Rep. 13 (1) (Jul. 2023) 11411, https://doi.org/10.1038/s41598-023-37570-7.
[75] X. Huang, D. Chen, T. Ren, D. Wang, A survey of community detection methods in multilayer networks, Data Min. Knowl. Discov. 35 (1) (Jan. 2021) 1–45, https://doi.org/10.1007/s10618-020-00716-6.
[76] G. Beauchamp, Flocking in birds increases annual adult survival in a global analysis, Oecologia 197 (2) (Oct. 2021) 387–394, https://doi.org/10.1007/S00442-021-05023-5/METRICS.
[77] X. Liu and L. Qiu, "Bird Flocking Inspired Control Strategy for Multi-UAV Collective Motion," no. 1, pp. 1–7, Nov. 2019, [Online]. Available: ⟨http://arxiv.org/abs/1912.00168⟩.
[78] C.W. Reynolds, Flocks-Hers-and-Schools, Comput. Graph. (ACM). 21 (4) (1987) 25–34.
[79] D. Lijcklama à Nijeholt, "Control for Cooperative Autonomous Driving Inspired by Bird Flocking Behavior," 2020.
[80] T. Zhou, L. Lü, Y.-C. Zhang, Predicting missing links via local information, Eur. Phys. J. B 71 (4) (Oct. 2009) 623–630, https://doi.org/10.1140/epjb/e2009-00335-8.
[81] N.P. Nguyen, T.N. Dinh, Y. Xuan, M.T. Thai, Adaptive algorithms for detecting community structure in dynamic social networks, Proc. - IEEE INFOCOM (2011) 2282–2290, https://doi.org/10.1109/INFCOM.2011.5935045.
[82] W.H. Chong and L.N. Teow, "An incremental batch technique for community detection," Proc. 16th Int. Conf. Inf. Fusion, FUSION 2013, no. January 2013, pp. 750–757, 2013..
[83] L. Danon, A. Díaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification, J. Stat. Mech. Theory Exp. 2005 (09) (Sep. 2005) P09008, https://doi.org/10.1088/1742-5468/2005/09/P09008.
[84] W.M. Rand, Objective criteria for the evaluation of clustering methods, J. Am. Stat. Assoc. 66 (336) (Dec. 1971) 846, https://doi.org/10.2307/2284239.
[85] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, B.L. Tseng, "Facetnet,". in *Proceedings of the 17th international conference on World Wide Web*, ACM, New York, NY, USA, Apr. 2008, pp. 685–694, https://doi.org/10.1145/1367497.1367590.
[86] D. Greene, D. Doyle, P. Cunningham, Tracking the Evolution of Communities in Dynamic Social Networks. 2010 International Conference on Advances in Social Networks Analysis and Mining, IEEE, Aug. 2010, pp. 176–183, https://doi.org/10.1109/ASONAM.2010.17.
[87] N.X. Vinh, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance, J. Mach. Learn. Res. 11 (2010) 2837–2854.
[88] S. Souravlas, S.D. Anastasiadou, T. Economides, S. Katsavounis, Probabilistic community detection in social networks, IEEE Access 11 (January) (2023) 25629–25641, https://doi.org/10.1109/ACCESS.2023.3257021.
[89] L. Hubert, P. Arabie, Comparing partitions, *J. Classif.* 2 (1) (Dec. 1985) 193–218, https://doi.org/10.1007/BF01908075.
[90] A. Lancichinetti, S. Fortunato, Community detection algorithms: A comparative analysis, Phys. Rev. E 80 (5) (Nov. 2009) 056117, https://doi.org/10.1103/PhysRevE.80.056117.
[91] N.P. Nguyen, T.N. Dinh, Y. Shen, M.T. Thai, Dynamic social community detection and its applications, PLoS One 9 (4) (Apr. 2014) e91431, https://doi.org/10.1371/journal.pone.0091431.