A High-Performance Evolutionary Multiobjective Community Detection Algorithm

Guilherme O. Santos[1*], Lucas S. Vieira[1], Giulio Rossetti[3], Carlos H. G. Ferreira[1], Gladston J. P. Moreira[2]

[1]Department of Computing and Systems, Universidade Federal de Ouro Preto, João Monlevade, 35931-008, Minas Gerais, Brazil.
[2]Computing Department, Universidade Federal de Ouro Preto, Ouro Preto, 35402-136, Minas Gerais, Brazil.
[3]Istituto di Scienza e Tecnologie dellInformazione, National Research Council of Italy, Pisa, 56127, PI, Italy.

*Corresponding author(s). E-mail(s): guilherme.os1@aluno.ufop.edu.br;
Contributing authors: lucas.sv@aluno.ufop.edu.br;
giulio.rossetti@isti.cnr.it; chgferreira@ufop.edu.br; gladston@ufop.edu.br;

Abstract

Community detection in complex networks is fundamental across social, biological, and technological domains. While traditional single-objective methods like Louvain and Leiden are computationally efficient, they suffer from resolution bias and structural degeneracy. Multi-objective evolutionary algorithms (MOEAs) address these limitations by simultaneously optimizing conflicting structural criteria, however, their high computational costs have historically limited their application to small networks. We present HP-MOCD, a High-Performance Evolutionary Multiobjective Community Detection Algorithm built on Non-dominated Sorting Genetic Algorithm II (NSGA-II), which overcomes these barriers through topology-aware genetic operators, full parallelization, and bit-level optimizationsachieving theoretical $O(G \cdot N_p|V|)$ complexity. We conduct experiments on both synthetic and real-world networks. Results demonstrate strong scalability, with HP-MOCD processing networks of over 1,000,000 nodes while maintaining high quality across varying noise levels. It outperforms other MOEAs by more than 531 times in runtime on synthetic datasets, achieving runtimes as low as 57 seconds for graphs with 40,000 nodes on moderately powered hardware. Across 14 real-world networks, HP-MOCD was the only MOEA capable of processing the six largest datasets within reasonable time, with results competitive with single-objective approaches. Unlike single-solution methods, HP-MOCD produces a Pareto Front, enabling

1

individual-specific trade-offs and providing decision-makers with a spectrum of high-quality community structures. It introduces the first open-source `Python` MOEA library compatible with `networkx` and `igraph` for large-scale community detection.

Keywords: community detection, complex networks, multi-objective optimization, evolutionary algorithms, NSGA-II, parallel computing

1 Introduction

The study of complex systems, ubiquitous in social, technological, and biological domains, heavily relies on modeling their underlying connectivity patterns as complex networks (Boccaletti et al. 2006). Examples range from social media platforms and the World Wide Web to protein-protein interaction networks and critical infrastructures (Ravasz et al. 2002). A key characteristic of many such networks is their organization into communities, groups of nodes with dense internal connections and sparser connections to the rest of the network (Newman 2006). Figure 1 provides a canonical example of community structure. Identifying these communities is crucial
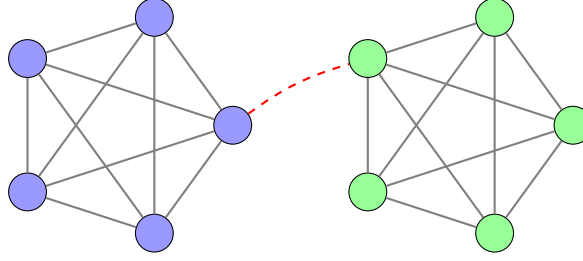
Fig. 1 A conceptual illustration of community structure. The network is partitioned into two distinct communities (blue and green). The solid gray lines represent dense **intra-community edges**, which create cohesive groups. The single dashed red line represents a sparse **inter-community edge** (or bridge), highlighting the weak connectivity between the groups. The goal of community detection is to identify such partitions.

for understanding network organization, function, and dynamics, making community detection a central research area in network science (Fortunato and Hric 2016; Flake et al. 2002; Blondel et al. 2008; Shi et al. 2010). The task is commonly formulated as a combinatorial optimization problem, where the goal is to partition the network into clusters that maximize internal cohesion while minimizing interconnections between groups (Newman and Girvan 2004a). Since this problem is NP-hard, most practical solutions rely on heuristic methods that optimize a predefined structural quality metric (Fortunato 2010).

Among these, single-objective approaches are particularly widespread due to their efficiency and simplicity. However, they often oversimplify the multifaceted nature of real-world networks by focusing on a single structural criterion. Two of the most

widely used single-objective methods are the Louvain algorithm (Blondel et al. 2008), which greedily maximizes modularity, and its improved variant, Leiden (Traag et al. 2019), which addresses issues such as disconnected communities through an enhanced refinement process.

While effective and computationally efficient, these traditional heuristic methods exhibit notable limitations. Their reliance on a single metric like modularity introduces structural biases, including the well-documented resolution limit, which causes small but meaningful communities to be absorbed into larger ones (Fortunato and Barthélemy 2007; Newman and Girvan 2004a). Modularity-based methods may also suffer from degeneracy, producing multiple partitions with similar scores but different topologies (Traag et al. 2019). Although Leiden mitigates some of these issues by ensuring better-connected communities and more stable partitions, it still operates under a fixed optimization criterion. This inherently restricts their capacity to navigate the complex trade-offs that arise in real-world networks, where no single metric fully captures the richness of the underlying structure (Azevedo et al. 2024).

Recognizing the limitations of traditional heuristic methods, the field has progressively shifted towards multi-objective evolutionary algorithms (MOEAs) as a more expressive and flexible framework for community detection (Shaik et al. 2021; Dabaghi-Zarandi et al. 2025). Unlike single-metric approaches, MOEAs aim to optimize multiple, often conflicting, objectives simultaneously. This multi-criteria formulation enables a more nuanced characterization of community structures by capturing diverse structural features of the network, such as internal density, inter-community sparsity, and modularity. By design, these algorithms produce a Pareto front, i.e., a set of non-dominated solutions, offering practitioners a range of alternatives rather than a single fixed partition (Moreira and Paquete 2019). This enables a more comprehensive exploration of the solution space, facilitating informed decisions based on application-specific trade-offs.

Early works such as MOGA-Net (Pizzuti 2009), which jointly optimized a community score and a fitness-based criterion, and Shi-MOCD (Shi et al. 2010), which explicitly targeted intra-community density and inter-community sparsity, demonstrated the potential of MOEAs to overcome structural limitations of modularity-based methods, including the resolution limit (Fortunato and Barthélemy 2007). Following these foundational works, the field has seen a range of proposals aimed at improving search efficiency and solution quality. In (Liu et al. 2024), the authors developed learning-based strategies to select nodes in each round of the multi-objective evolutionary process, called Local Search for Community Detection (LSCD). The work (Fu and Wang 2024) introduces a continuous-encoding-based approach with a clustering coefficient in a multi-objective framework, called CECC-Net, which represents the community structure. Focusing on the challenge of detecting overlapping communities, the work (Yusupov et al. 2021) presents an MOEA that utilizes an ensemble of different initializations. Despite their theoretical appeal, the practical adoption of those MOEAs for large-scale community detection remains limited, with applications to synthetic and real-world networks of no more than 10,000 nodes. A key challenge is their computational cost. Maintaining and evolving a population of candidate solutions requires repeated dominance comparisons, Pareto sorting, and fitness evaluations, all of which can be expensive in networks with thousands or millions of nodes and edges.

Many existing implementations exhibit time complexities on the order of $O(n^2)$ or worse, primarily due to operations involving pairwise comparisons and graph traversals (Shi et al. 2010; Pizzuti 2012). This makes them impractical for real-time or large-scale applications without specialized acceleration strategies.

More recent algorithms include KRM and CCM (Shaik et al. 2021), both based on the NSGA-III framework. KRM optimizes three objectives: Kernel-k-means, Ratio-cut, and Modularity using a reference-point guided selection mechanism, while CCM targets Community fitness, Community score, and Modularity. Another notable contribution is CDRME (Dabaghi-Zarandi et al. 2025), a four-phase evolutionary method that incorporates similarity-based initialization and random-walk-based seeding to balance cut density, ratio cut, and modularity across network scales.

Despite their theoretical appeal, the practical adoption of these MOEAs for large-scale community detection remains limited, with applications to synthetic and real-world networks of no more than 10,000 nodes. A key challenge is their computational cost. Maintaining and evolving a population of candidate solutions requires repeated dominance comparisons, Pareto sorting, and fitness evaluations, all of which can be expensive in networks with thousands or millions of nodes and edges. Many existing implementations exhibit time complexities on the order of $O(n^2)$ or worse, primarily due to operations involving pairwise comparisons and graph traversals (Shi et al. 2010; Pizzuti 2012). This makes them impractical for real-time or large-scale applications without specialized acceleration strategies.

Additionally, few implementations of MOEAs are publicly available for community detection, which limits accessibility for researchers and practitioners. Most available tools are either experimental, tied to specific datasets, or lack parallelization and optimization for modern hardware. Finally, designing effective genetic operators that preserve meaningful community structures while ensuring exploration of the vast combinatorial search space remains an open problem. Balancing exploitation and exploration in this context is especially difficult due to the discrete and topologically constrained nature of the community detection problem.

To address these challenges, this paper introduces the High-Performance Multi-Objective Community Detection (HP-MOCD) algorithm. HP-MOCD is a scalable evolutionary algorithm based on the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) framework (Deb et al. 2002), specifically engineered for accuracy and efficiency in large complex networks. It employs a parallel computing architecture and custom genetic operators that incorporate topological information to explore the solution space effectively. Instead of yielding a single partition, HP-MOCD produces a diverse set of Pareto-optimal solutions, each representing a different trade-off between conflicting structural objectives, thereby offering a more holistic understanding of network community structures. Moreover, this solution set can be flexibly analyzed under multiple evaluation metrics or selected according to the specific needs of a given application. Our primary contributions are as follows:

(i) We introduce a novel topology-aware crossover and mutation schemes that both refine community assignments and preserve population diversity, resulting in more accurate and structurally consistent community partitions.

(ii) We prove that, for the sparse graphs typically encountered in practice ($|E| = O(|V|)$) with fixed population size N_p, the per-generation cost reduces to $O(N_p |V|)$. Our experiments confirm runtimes remain well below worst-case bounds.

(iii) We design a fully multithreaded NSGA-II architecture that exploits modern multicore processors, drastically reducing runtime on largescale networks, .

(iv) We provide a Free/Libre and Open-Source Software (FLOSS) implementation of HP-MOCD, enabling researchers and practitioners to apply, extend, and benchmark our approach.

The remainder of this paper is structured as follows. Section 2 reviews foundational concepts in community detection, including definitions and key modeling assumptions. Section 3 surveys prior work, covering both traditional algorithms and recent multi-objective approaches. Section 4 formalizes the multi-objective problem and presents the HP-MOCD algorithm in detail, including its architecture, genetic operators, and parallel execution model. Section 5 describes the experimental setup and reports a comparative evaluation against several baselines on synthetic and real-world networks. Finally, Section 6 discusses the implications, limitations, and potential directions for future research.

2 Background

Community detection is a fundamental problem in network science, central to understanding the structure and function of complex systems. This section first introduces the foundational definitions and modeling assumptions adopted in this work. We then briefly review traditional heuristic methods, with emphasis on modularity-based algorithms such as Louvain and Leiden, analyzing their performance and limitations. Finally, we present an overview of multi-objective community detection, discussing its theoretical foundations, algorithmic strategies, and the key contributions from the literature that inform the design of our proposed method.

2.1 The Definition of Community

Community structure is a central concept in the analysis of complex networks, referring to the tendency of nodes to organize into groups with dense internal connections and relatively sparse connections to the rest of the network. While no universally accepted definition exists, one of the most common structural formulations is based on the relative density of intra- and inter-community links (Radicchi et al. 2004). In this perspective, communities are understood as subgraphs in which internal connectivity is significantly stronger than external connectivity.

Several formulations of community detection have emerged in the literature, depending on the nature of the data and the intended application (Fortunato 2010). Among the most common is node clustering, where the goal is to assign each node to a community. Other approaches include edge clustering, which focuses on grouping edges rather than nodes, and overlapping or fuzzy clustering, in which nodes may belong

to multiple communities, either to the same degree or with varying levels of association. More specialized formulations incorporate additional structural or semantic information (Lancichinetti and Fortunato 2009). Attributed clustering, for example, leverages both network topology and node-level features, while bipartite clustering targets networks composed of two distinct node types. Temporal clustering extends the task to dynamic networks, capturing how communities evolve over time. In directed acyclic graphs (DAGs), antichain clustering is used to identify groups based on partial order relationships (Rossetti et al. 2019).

In this work, we focus on node clustering under a crisp partitioning model, also referred to as hard clustering, where each node is assigned to exactly one community. This assumption of mutually exclusive group membership simplifies the structural analysis and remains the most widely adopted formulation in the evaluation of community detection algorithms. We further restrict our scope to undirected and unweighted networks, which are commonly used in benchmarks and allow clear structural interpretations based solely on topology.

Formally, let $G = (V, E)$ be an undirected and unweighted graph, where V is a finite, non-empty set of nodes and $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$ is the set of undirected edges.

$$\bigcup_{i=1}^{k} C_i = V \quad \text{and} \quad C_i \cap C_j = \emptyset \quad \forall \, i \neq j \tag{1}$$

2.2 Multi-Objective Optimization

A multi-objective optimization problem (MOP) involves simultaneously optimizing two or more, often conflicting, objective functions. Improving one objective may lead to the deterioration of another, making it impractical to obtain a single solution that is optimal for all objectives. Instead, MOP aims to find a set of solutions that represent optimal trade-offs among the objectives. In this study, optimization is approached as the minimization of objective values.

Formally, a MOP can be stated as (Ehrgott 2005):

$$\text{Minimize} \quad \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})) \text{ subject to } \mathbf{x} \in \Omega \tag{2}$$

Here, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is a vector of n decision variables residing in the n-dimensional decision space $\mathbb{R}^n$. The set $\Omega \subseteq \mathbb{R}^n$ represents the feasible region of the decision space, containing all permissible solutions. In the context of community detection, $\mathbf{x}$ typically represents a candidate partitioning of the network's nodes into communities. The function $\mathbf{f} : \Omega \to \mathbb{R}^m$ maps the decision space to the m-dimensional objective space, where each $f_i(\mathbf{x})$ is an objective function to be minimized. The image of the feasible set, $\mathbf{f}(\Omega) = \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in \Omega\}$, is the set of all possible objective vectors in the objective space $\mathbb{R}^m$. The problem can also be subject to additional constraints in Ω and/or $\mathbf{f}(\Omega)$.

Since the objective space $\mathbb{R}^m$ (for $m > 1$) is only partially ordered, the concept of dominance is used to compare solutions. Let $\mathbf{a} = (a_1, a_2, \ldots, a_m)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_m)$ be two objective vectors in $\mathbb{R}^m$. The vector $\mathbf{a}$ is said to dominate

b (denoted as $\mathbf{a} \prec \mathbf{b}$) if and only if $a_i \leq b_i$ for all $i \in \{1, \ldots, m\}$ and $a_j < b_j$ for at least one $j \in \{1, \ldots, m\}$. A solution $\mathbf{x}^* \in \Omega$ is Pareto optimal if there is no other solution $\mathbf{x} \in \Omega$ such that $\mathbf{f}(\mathbf{x})$ dominates $\mathbf{f}(\mathbf{x}^*)$. In other words, $\mathbf{x}^*$ is Pareto optimal if its corresponding objective vector $\mathbf{f}(\mathbf{x}^*)$ is non-dominated.

The set of all non-dominated objective vectors is known as the Pareto Front (PF), and the set of all Pareto optimal decision vectors is the Pareto Optimal set (PO):

$$PF = \{\mathbf{y} \in \mathbf{f}(\Omega) \mid \nexists \mathbf{y}' \in \mathbf{f}(\Omega) \text{ such that } \mathbf{y}' \prec \mathbf{y}\} \tag{3}$$

$$PO = \{\mathbf{x} \in \Omega \mid \mathbf{f}(\mathbf{x}) \in PF\} \tag{4}$$

The PF represents the set of optimal trade-off solutions where no objective can be improved without degrading at least one other objective (Moreira and Paquete 2019).

3 Related Work

Community detection is a vast field. Here, we compare and contrast HP-MOCD with the two main classes of algorithms relevant to our work: traditional algorithms and other MOEAs.

3.1 Traditional Algorithms

Traditional approaches to community detection in complex networks often rely on the optimization of a structural quality function, the most prominent being modularity. These methods aim to partition a network such that the density of edges within communities is significantly higher than between them. While modularity-based algorithms remain dominant due to their efficiency and simplicity, several alternative heuristics have also shaped the field. Among the earliest and most influential is the GirvanNewman algorithm, which detects communities by iteratively removing edges with the highest betweenness centrality, gradually decomposing the network into disconnected components (Girvan and Newman 2002). Despite its conceptual elegance, this method suffers from high computational cost and limited scalability.

Another prominent strategy is label propagation, a fast heuristic where each node adopts the most frequent label among its neighbors. This process converges to a partition where nodes in the same community share the same label (Raghavan et al. 2007a). Label propagation is highly scalable but can yield unstable results and lacks a clear objective function, making its outcomes harder to interpret. Among modularity-based methods, the Louvain algorithm proposed by Blondel et al. (2008) has become one of the most widely used due to its balance between accuracy and computational performance. It seeks to optimize the modularity metric (Newman 2006), which measures the quality of a partition by comparing the density of edges within communities to the expected density in a randomized null model that preserves the node degrees. In essence, modularity quantifies how much more densely connected the nodes are within communities than would be expected by chance. The modularity score theoretically ranges from $-\frac{1}{2}$ to 1, where higher values indicate a stronger community structure, and values near zero or negative suggest that the detected communities are no betteror even worsethan random.

Despite its popularity and efficiency, the Louvain algorithm exhibits several well-known limitations that affect both the quality and stability of the resulting partitions (Fortunato and Barthélemy 2007). Some works have attempted to address these limitations (Wang et al. 2019). One key issue is the production of disconnected communities, a consequence of how nodes are greedily reassigned based solely on local modularity gain without enforcing internal connectivity constraints (Fortunato and Hric 2016). This can lead to communities that are not internally cohesive, violating a fundamental structural assumption in community detection. Furthermore, Louvain is sensitive to the order in which nodes are processed during the local moving phase. Because the algorithm applies greedy updates, different node orderings may lead to different partitions with comparable modularity scores, introducing a level of non-determinism that compromises reproducibility.

A more fundamental limitation lies in the modularity function itself, particularly its susceptibility to the resolution limit problem. As demonstrated by Fortunato and Barthélemy (2007), modularity optimization tends to favor large communities and may fail to identify smaller but structurally meaningful groups, especially in networks with heterogeneous community sizes. This limitation arises because modularity evaluates gains relative to a global null model, which biases the optimization toward partitions that absorb smaller communities into larger ones. Additionally, the modularity landscape often contains many local maxima, leading to a high degree of degeneracy in the solution space. This means that multiple partitions with very different community structures can yield similar modularity values, complicating the interpretation and comparison of results across runs.

To address some of these issues, the Leiden algorithm was introduced by Traag et al. (2019). It preserves the modularity optimization framework but improves upon Louvain by enforcing connectedness within communities and refining the aggregation process. The algorithm proceeds in three stages. The first is local moving, in which nodes are reassigned to neighboring communities to improve a quality function. The second is refinement, which ensures that all communities remain internally connected, splitting any that are not. The final stage is aggregation, where communities are collapsed into single nodes, and the process restarts on the simplified graph.

While the Leiden algorithm was originally introduced with support for multiple objective functions, it often employs the Constant Potts Model (CPM) instead of modularity. CPM defines community quality in terms of internal edge density, controlled by a resolution parameter γ, and is given by:

$$H = \sum_c \left(e_c - \gamma \frac{n_c(n_c - 1)}{2} \right) \tag{5}$$

In this expression, e_c denotes the number of internal edges in community c, n_c is the number of nodes in that community, and γ sets the minimum internal density threshold. By tuning γ, the algorithm can control the granularity of the resulting partition, identifying larger or smaller communities accordingly. Unlike modularity, CPM does not suffer from the same theoretical resolution limit because it is a local

objective function, and its optimization does not depend on global graph properties such as the total number of edges.

Nevertheless, in practical applications, CPM introduces its own set of challenges. The performance of the algorithm can vary significantly with the choice of γ, which often requires careful, problem-specific calibration. Furthermore, when Leiden is configured to use modularity as its objective function, as is commonly done for compatibility with existing benchmarks, it inherits many of the same limitations as Louvain, including the resolution limit and modularity degeneracy. In either case, whether using modularity or CPM, Leiden remains fundamentally a single-objective optimization method.

Algorithms such as Louvain and Leiden represent the current state of practice in scalable, modularity-based community detection. Their combination of simplicity, computational efficiency, and availability in widely used libraries has made them widely adopted solutions in many real-world applications. Despite this practical success, their reliance on a single optimization criterion inherently limits their expressiveness. In particular, they are not designed to capture multiple, potentially conflicting structural properties simultaneously, nor do they offer the user flexibility in selecting solutions that trade off different qualities such as compactness, separation, or size balance. These methods produce only a single partition per execution, often overlooking alternative high-quality solutions that might better suit specific applications or structural goals. These limitations motivate the investigation of multi-objective optimization strategies, which aim to overcome the rigidity of single-objective heuristics by generating diverse sets of non-dominated solutions. In the following section, we review some of the recent advances in this area and highlight their potential to provide richer structural insights and greater flexibility for real-world community detection.

3.2 Multi-Objective Evolutionary Algorithms

The limitations of single-objective methods motivated the development of Multi-Objective Evolutionary Algorithms (MOEAs) for community detection. Unlike single-metric approaches, MOEAs optimize multiple, often conflicting, objectives simultaneously, such as internal density and external sparsity. This approach yields a Pareto front[1], a set of non-dominated solutionsoffering a range of robust alternatives rather than a single, potentially biased partition (Moreira and Paquete 2019; Coello et al. 2007). The use of MOEAs has become a prominent strategy due to their population-based nature, which is well-suited for exploring the complex and combinatorial solution space of community detection (Deb 2001; Handl and Knowles 2007).

Among the earliest applications of evolutionary multi-objective optimization in this domain is **MOGA-Net**, introduced by Pizzuti (2009). Her approach uses genetic algorithms to evolve network partitions that optimize both a community score and modularity. Although pioneering, the method remains dependent on modularity as a selection criterion, which reintroduces resolution bias. Additionally, the algorithm's computational cost is not thoroughly characterized, leaving its applicability to large networks uncertain.

---

[1]The proximity of the set of solutions obtained by an MOEA to the true PF is a measure of the algorithm's convergence.

The MultiObjective Community Detection (MOCD) algorithm by Shi et al. (2010) jointly minimizes two objectives related to intra-community density and inter-community sparsity. MOCD employs the PESA-II evolutionary framework and locus-based operators. However, its selection strategy and operators lead to high computational overhead, with a complexity of $O(gs^2(m+n))$, making it impractical for large networks.

More recent research has continued to explore the potential of MOEAs, introducing more sophisticated objective functions and hybrid strategies. The work by Shaik et al. (2021), introduced a customized NSGA-III framework to simultaneously optimize three objectives (e.g., Kernel k-means, Ratio-cut, and Modularity). Their contribution demonstrated the feasibility of using many-objective optimization to find community structures that represent a more nuanced trade-off between different structural properties. Dabaghi-Zarandi et al. (2025) proposed a hybrid method that first uses random walks to generate an initial set of high-quality partitions, which are then refined by a multi-objective evolutionary algorithm. This approach aims to accelerate convergence by seeding the evolutionary process in a more promising region of the search space.

4 The HP-MOCD Algorithm

We now describe the High-Performance Multi-Objective Community Detection (HP-MOCD) algorithm. Our method integrates the NSGA-II optimization framework with a parallel architecture and custom genetic operators designed for network topology. The implementation is written in Rust for performance and exposed to Python via PyO3. The full source code is publicly available on GitHub and the documention and usage are available on the Official Documentation. Algorithm 1 and Figure 2 summarizes the HP-MOCD's algorithm.

4.1 Overview and Design Rationale

Traditional single-objective approaches often suffer from resolution limit problems and can become trapped in local optima, as discussed in Section 2. To address these limitations, we need to use metrics that discover the entire Pareto front of solutions, ranging from coarse to fine partitions without biasing toward any particular resolution level.

Given a graph $G = (V, E)$ with node set V and edge set E, let $C = \{c_1, c_2, \ldots, c_k\}$ represent a partition of V into k communities. The set of all valid partitions of V is denoted as the feasible space $\Omega = \{C \mid C \text{ is a valid partition of } V\}$.

Following the formulation of Shi et al. (2010), we decompose the community detection problem into two complementary objectives that capture the essential trade-offs in community structure. Our method seeks to find partitions that simultaneously minimize both objective functions:

$$f_1(C) = 1 - \sum_{c \in C} \frac{|E(c)|}{m}, \quad f_2(C) = \sum_{c \in C} \Big(\frac{\sum_{v \in c} \deg(v)}{2m} \Big)^2 \tag{6}$$

where $m = |E|$ denotes the total number of edges, $\deg(v)$ represents the degree of node v, and $|E(c)|$ is the number of internal edges within community c.

The first objective function f_1 (or $intra(C)$) measures the fraction of edges that are *not* internal to communities, effectively penalizing partitions with poor internal cohesion. Minimizing f_1 encourages the formation of densely connected communities. The second objective function f_2 (or $inter(C)$) represents the sum of squared relative community sizes (measured by total degree), which penalizes the formation of overly large communities. Minimizing f_2 promotes more balanced partitions and prevents the emergence of dominant communities that absorb an excessive number of nodes.

These objectives are inherently conflicting; optimizing for strong internal cohesion (low f_1) often leads to fewer, larger communities, while optimizing for balanced community sizes (low f_2) typically results in many smaller communities with potentially weaker internal structure.

The problem is then stated as:

$$\min_{C \in \Omega} \mathbf{f}(C) \;=\; \min_{C \in \Omega} \big(f_1(C),\, f_2(C) \big). \tag{7}$$

Figure 2 illustrates the diagram of the proposed HP-MOCD algorithm, the initialization, evaluation, and evolutionary selection phases. Parallelism is applied during the evaluation of objectives and sorting operations, enhancing scalability. The evolutionary loop iteratively generates offspring, combines populations, and applies environmental selection until the maximum number of generations is reached. ....
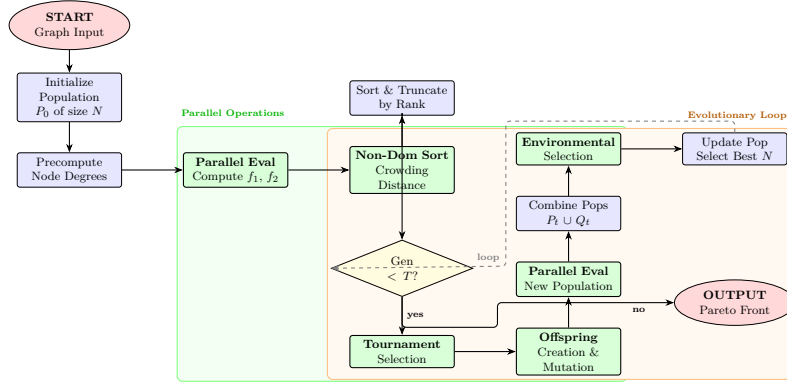
Fig. 2 Flowchart of the HP-MOCD algorithm. Green-highlighted boxes indicate parallel operations that enhance computational efficiency. The algorithm iteratively evolves populations through selection, crossover, mutation, and environmental selection until convergence.

Our HP-MOCD algorithm addresses this optimization problem, employing the NSGA-II framework and parallel architecture. The overall process is outlined in Algorithm 1. It begins by creating an initial population of random partitions, each representing a possible way to partition the graph into communities. Then, through successive generations, it employs genetic operators (selection, crossover, and mutation) to evolve the population, preserving the best non-dominated solutions.

11

Algorithm 1 High-Performance Multi-Objective Community Detection (HP-MOCD)


1: **procedure** HP-MOCD(Graph $G(V, E)$, Population Size N_p, Max Generations T, Crossover Probability C_P, Mutation Probability M_P, Ensemble Size E_S)
2:    **Input:** Undirected unweighted graph $G = (V, E)$ with $|V|$ nodes and $|E|$ edges
3:    **Output:** Pareto front $\mathcal{F}_1$ of non-dominated community partitions
4:                                     ▷ **Phase 1: Initialization and Preprocessing**
5:    $degrees \leftarrow$ PrecomputeDegrees(G)   ▷ $O(|V|)$ cache for $\deg(v), \forall v \in V$
6:    $m \leftarrow |E|$ ▷ Total number of edges
7:    $P^{(0)} \leftarrow$ InitializePopulation(G, N_p)   ▷ Random hash-map partitions
8:                                      ▷ **Phase 2: Initial Population Evaluation**
9:    **parallel for each** $p \in P^{(0)}$ **do**
10:      EvaluateIndividual($p, G, degrees$)   ▷ Compute (f_1, f_2)
11:    **end parallel**
12:                                 ▷ **Phase 3: Evolutionary Optimization Loop**
13:    **for** $gen \leftarrow 1$ **to** T **do**
14:                            ▷ `Multi-objective selection and ranking`
15:      FastNonDominatedSort($P^{(gen-1)}$)   ▷ $O(N_p \log N_p)$ for 2 objectives
16:      CrowdingDistanceAssignment($P^{(gen-1)}$)   ▷ Diversity preservation
17:      SortByRankAndDistance($P^{(gen-1)}, N_p$)   ▷ Pareto-based ranking
18:                               ▷ `Parallel offspring generation`
19:      **parallel map** $i \leftarrow 1$ **to** N_p **do**
20:        parents $\leftarrow$ TournamentSelection($P^{(gen-1)}, E_S, k = 2$)
21:        child $\leftarrow$ TopologyAwareCrossover(parents, C_P)   ▷ Algorithm 2
22:        TopologyAwareMutation(child, G, M_P)   ▷ Section 4.3.2
23:        **return** child
24:      **end parallel** $\rightarrow Q^{(gen)}$
25:                               ▷ `Parallel offspring evaluation`
26:      **parallel for each** $q \in Q^{(gen)}$ **do**
27:        EvaluateIndividual($q, G, degrees$)
28:      **end parallel**
29:                               ▷ `Environmental selection`
30:      $R^{(gen)} \leftarrow P^{(gen-1)} \cup Q^{(gen)}$ ▷ Combine parent and offspring populations
31:      $P^{(gen)} \leftarrow$ EnvironmentalSelection($R^{(gen)}, N_p$)   ▷ Select best N_p individuals
32:    **end for**
33:               ▷ **Phase 4: Pareto Front Extraction and Solution Selection**
34:    $\mathcal{F}_1 \leftarrow \{p \in P^{(T)} \mid \text{rank}(p) = 1\}$ ▷ Extract Pareto front
35:    $p^* \leftarrow$ SelectSolution($\mathcal{F}_1$)   ▷ Algorithm 3
36:    **return** $\mathcal{F}_1, p^*$ ▷ Return Pareto front and recommended solution
37: **end procedure**

4.2 Solution Representation

Consider the toy network $G(V, E)$ shown in Figure 3(a). The graph representation converts a `NetworkX` or `igraph` network into an optimized data structure that maintains multiple complementary representations of the same graph data: (i) An edge list for sequential access. (ii) An adjacency list for neighborhood queries. (iii) Pre-computed degree mappings. (iv) An edge lookup hash set that enables constant-time edge

existence queries. This design allows significant performance gains, providing $O(1)$ access times for operations.
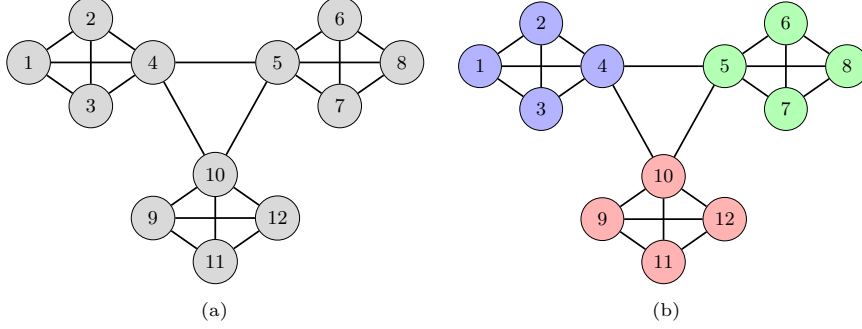
Fig. 3 (a) A simple network with some nodes that shows aspects of a community. (b) The same network, but with distinct communities. Nodes within each community (blue, green, and red) are more densely connected internally, with sparser connections between communities.

An individual in the population is a candidate solution, represented as a hash map that maps each node $v \in V$ to a community identifier, as a mapping $\mathcal{M}$.

$$\mathcal{M} : V \longrightarrow \{c_1, c_2 \dots c_k\}$$

For example of the Figure 3(a), consider the one possible mapping $\mathcal{M} = \{1\!:\!A,\ 2\!:\!A,\ 3\!:\!A,\ 4\!:\!A,\ 5\!:\!B,\ 6\!:\!B,\ 7\!:\!B,\ 8\!:\!B,\ 9\!:\!C,\ 10\!:\!C,\ 11\!:\!C,\ 12\!:\!C\}$ showed in Figure 3(b).

This representation is memory-efficient, allowing for fast lookups. Our custom crossover and mutation operators drive the evolution of these individuals.

4.3 Genetic Operators and Offspring Generation

4.3.1 Crossover

The crossover operator is responsible for combining multiple parent solutions to generate a new offspring. Let $P = (P_1, \ldots, P_{N_p})$ denote a collection of N_p parent individuals, where each individual P_i represents a partition of the graph $G = (V, E)$, that is, a mapping from each node $v \in V$ to a community label $c \in C$. The operator is controlled by a crossover threshold parameter $C_R \in [0, 1]$. Initially, a random number x is sampled uniformly from the interval $[0, 1]$. If $x > C_R$, the crossover is skipped, and one of the parent partitions is selected uniformly at random to serve as the offspring:

$$x > C_R \quad \Rightarrow \quad P_{\text{child}} = \text{Random}(P_1, \ldots, P_{N_p}). \tag{8}$$

If $x \leq C_R$, the offspring is constructed by aggregating community assignments from all parent individuals. The process starts by considering the common node set V, and initializing an empty offspring mapping P_{child}. For each node $v \in V$, the algorithm

counts the number of times each community label $c \in C$ has been assigned to v across all parent individuals, using the following expression:

$$\text{count}(c, v) = \sum_{i=1}^{N_p} \mathbb{1}\{P_i(v) = c\}, \tag{9}$$

where $\mathbb{1}\{\cdot\}$ denotes the indicator function. The community (or communities) with the highest count is then identified as

$$c^*(v) \in \arg \max_{c \in C} \text{count}(c, v). \tag{10}$$

In the event of a tie, one of the top-scoring communities is selected uniformly at random. The selected label $c^*(v)$ is then assigned to node v in the offspring partition, that is, $P_{\text{child}}(v) = c^*(v)$. This procedure is applied to all nodes in V, producing a complete offspring partition. The resulting solution tends to preserve dominant structural features from the parents while still allowing for variability and exploration through random tie-breaking. The crossover process is invoked for each of the N_p offspring to be created in the new population, as detailed in the loop of Algorithm 1 (lines 19-24), thus ensuring diversity and continuous exploration of the search space.

Algorithm 2 Crossover Operator

1: **procedure** CROSSOVER($P_1, \ldots, P_{N_p}, C_R$)
2:     **Input:** Collection of N_p parent individuals $P = (P_1, \ldots, P_{N_p})$, crossover threshold $C_R \in [0, 1]$
3:     **Output:** Offspring partition P_{child}
4:     Sample $x \sim \text{Uniform}[0, 1]$
5:     **if** $x > C_R$ **then**
6:         **return** Random($P_1, \ldots, P_{N_p}$)   ▷ Skip crossover, return random parent
7:     **end if**
8:     Initialize $P_{\text{child}} \leftarrow \emptyset$ ▷ Empty offspring mapping
9:            ▷ `Aggregate community assignments from all parents`
10:     **for** each node $v \in V$ **do**
11:            ▷ Count occurrences of each community label for node v
12:         **for** each community $c \in C$ **do**
13:             $\text{count}(c, v) \leftarrow \sum_{i=1}^{N_p} \mathbb{1}\{P_i(v) = c\}$
14:         **end for**
15:            ▷ Find community(ies) with highest count
16:         $c^*(v) \leftarrow \arg \max_{c \in C} \text{count}(c, v)$ ▷ Random tie-breaking if multiple maximizers
17:         $P_{\text{child}}(v) \leftarrow c^*(v)$ ▷ Assign selected community to node v
18:     **end for**
19:     **return** P_{child}
20: **end procedure**

4.3.2 Mutation

To introduce variation and avoid premature convergence, HP-MOCD applies a mutation operator after crossover. This operator modifies the community assignment of selected nodes based on the structure of their local neighborhood. The mutation operator uses the same individual representation as described in Section 4.3.1, along with a mutation threshold parameter $M_R \in [0, 1]$. Each node $v \in V$ is considered for mutation independently: a random number $r_v \in [0, 1]$ is sampled, and nodes satisfying $r_v < M_R$ are selected for mutation. The set of selected nodes is defined as:

$$V_m = \{v \in V \mid r_v < M_R\}. \tag{11}$$

For each selected node, the frequency of community assignments among its neighboring nodes is computed. To improve performance, the nodes are processed in batches. For a given node v, its set of neighbors $N(v)$ is retrieved, and for each neighbor $u \in N(v)$, the community to which u belongs is noted. The algorithm then determines the most frequent community using the same Equation 10.

4.4 Solution Selection

For the multi-objective optimization problem formulated in this study, the goal is to find solutions that "minimize" $\mathbf{f}(C)$ (see Equation 7) in the Pareto-optimality context. Algorithms such as NSGA-II do not return a single "optimal" solution. Instead, return a set of feasible, mutually non-dominated solutions that approximate as much as possible the Pareto set, which represents the optimal trade-off among the multiple objectives (Moreira and Paquete 2019).

The Decision Maker (DM) must choose the most suitable option from the approximation set to "implement".

Following the strategy proposed by Shi et al. (2010), we adopt a scalarization-based criterion to identify the solution that maximizes the following quality function:

$$Q(C) = 1 - f_1 - f_2 \tag{12}$$

This score favors solutions with low intra-community penalty and low inter-community connectivity, as defined in Equation 6, thus promoting partitions that are both cohesive and well-separated. Figure 4 illustrates this behavior: the red dot indicates the solution that maximizes $Q(C)$. We observe that this selection strategy consistently identifies partitions that also perform well under NMI and AMI validation metrics.

An important advantage of the Pareto-based approach is the flexibility it affords in selecting solutions based on application-specific preferences. While we adopt the scalarization function $Q(C)$ as a default selection criterion, practitioners are not constrained to this choice. Depending on the context, alternative metrics can be employed to guide selection from the Pareto front. For instance, one might prioritize solutions based on modularity, the number of detected communities, average community size, or even the variance in community sizes. Other structural properties, such as conductance, cut ratios, or expansion scores, may also be relevant in certain domains.

(a) Pareto Frontier

(b) $Q(C)$ vs. Number of Communities

(c) $Q(C)$ vs. NMI

(d) $Q(C)$ vs. AMI
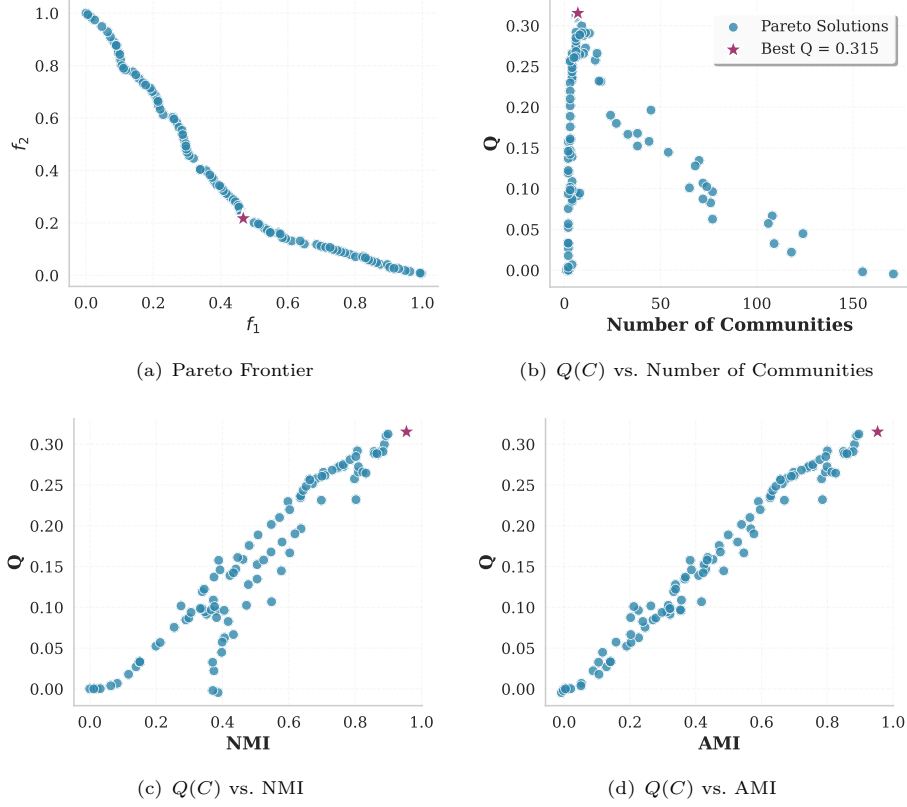
Fig. 4 Analysis of the scalarization-based selection method. The red dot highlights the solution with the highest value of $Q(C)$.

This flexibility allows users of HP-MOCD to serve as a general-purpose tool, as it provides not just one partition, but a rich set of candidate solutions from which the most appropriate can be chosen using auxiliary metrics or domain-specific constraints. This is a fundamental departure from single-objective methods, which collapse the search space into a single score and, therefore, offer limited interpretability and adaptability.

4.5 Complexity Analysis

Complexity time of HP-MOCD per generation is driven by its genetic operators (crossover and mutation) and the key NSGA-II routines (non-dominated sorting and crowding distance). Let $|V|$ be the number of nodes, $|E|$ the number of edges, N_p the population size, and M_R the mutation rate.

Algorithm 3 Solution Selection

1: **procedure** SELECTSOLUTION(Pareto front $\mathcal{F}_1$)
2:     **Input:** Set of non-dominated solutions $\mathcal{F}_1 = \{p_1, p_2, \ldots, p_k\}$
3:     **Output:** Recommended solution p^*
4:                                   ▷ `Scalarization-based selection criterion`
5:     selected_solution $\leftarrow -\infty$
6:     $p^* \leftarrow$ null
7:     **for** each solution $p \in \mathcal{F}_1$ **do**
8:         $Q(p) \leftarrow 1 - f_1(p) - f_2(p)$
9:         **if** $Q(p) >$ selected_solution **then**
10:            selected_solution $\leftarrow Q(p)$
11:            $p^* \leftarrow p$
12:         **end if**
13:     **end for**
14:                     ▷ `Domain-specific selection criteria can be applied here`
15:                ▷ `e.g., number of key nodes, communities, conductance, etc.`
16:     **return** p^*
17: **end procedure**

4.5.1 Operators Complexity Analysis

Crossover & Mutation

For each node in the graph, the algorithm crossover identifies the most frequent community label assigned to that node among a fixed number of parent solutions. Since hash map operations have an expected constant time complexity, $O(1)$, processing each node takes $O(1)$ time. Assigning the most frequent label also takes constant time per node. Consequently, for all $|V|$ nodes, the expected time complexity of a single crossover operation is linear in the number of nodes, i.e., $O(|V|)$.

The mutation iterates through each of the $|V|$ nodes, selecting a node for mutation with a given probability $M_R \in [0, 1]$. This selection phase involves a random draw for each node, costing $O(|V|)$ time, and, for each node selected for mutation, inspects its local neighborhood to determine a new community assignment (by finding the most frequent community label among its neighbors). This requires visiting all adjacent nodes, so the cost for mutating a single node v is proportional to its degree, $d(v)$. The expected number of nodes selected for mutation is $M_R \cdot |V|$. The total time for recalculating assignments for these mutated nodes is proportional to the sum of their degrees. In expectation, this sum is $M_R \sum_{v \in V} d(v) = M_R \cdot 2|E|$. Therefore, the expected time complexity for this part of the mutation is $O(M_R \cdot |E|)$. Combining the node selection phase and the assignment update phase, the total time complexity for one mutation operation is $O(|V| + M_R \cdot |E|)$. For sparse graphs, where $|E| = O(|V|)$, and assuming M_R is a constant, this simplifies to $O(|V|)$.

Non-Dominated Sorting & Crowding Distance Calculation

These operators act upon a combined population of $2N_p$ individuals (current population P_t and offspring population Q_t, each of size N_p). This process sorts the $2N_p$ individuals into different non-domination fronts. The non-dominated sorting of N individuals can

be performed in $O(N \log N)$ time with fixed 2 objectives. Thus, for the $2N_p$ individuals, this step takes $O((2N_p) \log(2N_p)) = O(N_p \log N_p)$ time.

For each front of size F_i, solutions are sorted based on each of the M objective functions. This takes $O(MF_i \log F_i)$ time per front. Summing over all fronts $\sum F_i = 2N_p$, the total time complexity for crowding distance calculation across all fronts is $O(M(2N_p) \log(2N_p))$. For $M = 2$, this becomes $O(N_p \log N_p)$.

Overall Complexity

The dominant cost per generation is the sum of: $T_{\text{gen}} = O(N_p|V|) + O(N_p \log N_p) + O(N_p \log N_p)$ which simplifies to $T_{\text{gen}} = O(N_p|V| + N_p \log N_p)$. Since $|V|$ in G is typically much larger than $\log N_p$, the $O(N_p|V|)$ term usually dominates. Thus, the complexity per generation effectively becomes: $O(N_p|V|)$ for sparse graphs. Over G generations, the total time complexity of HP-MOCD is: $O(G \cdot N_p|V|)$. A detailed analysis of the algorithm's space complexity, including the dominant memory terms across components, is provided in Appendix A.

5 Experimental Results

This section details the methodology and experimental evaluation of HP-MOCD. Our study explores four core dimensions: (i) detection performance on synthetic and real-world networks, (ii) scalability with increasing network size, (iii) parallel efficiency via multithreaded execution, and (iv) robustness under structural noise. Experiments are conducted using standard benchmarks, well-established metrics, and strong baseline algorithms, ensuring fair and reproducible comparisons. To ensure rigorous and interpretable comparisons, we perform pairwise statistical tests (paired t-test with initial $\alpha = 0.05$), applying Bonferroni correction to adjust for multiple comparisons. The goal is to determine, for each evaluation, whether HP-MOCD is statistically superior, equivalent, or inferior to the best-performing baseline. We classify each result accordingly and highlight them in **bold** (superior) in the tables.

### 5.1 Comparison Algorithms and Parameter Configuration

We compare HP-MOCD against a diverse set of classical heuristics and multi-objective evolutionary algorithms, described below:

- **MOCD** (Shi et al. 2010): A two-objective evolutionary method (based on PESA-II) optimizing intra-community density and inter-community sparsity.
- **MOGANet** (Pizzuti 2009): A modularity-oriented genetic algorithm evolving partitions directly.
- **Louvain** (Blondel et al. 2008): A multilevel greedy heuristic that locally optimizes modularity.
- **Leiden** (Traag et al. 2019): A refinement of Louvain ensuring connected communities and improved convergence.
- **ASYN-LPA** (Raghavan et al. 2007b): A fast, parameter-free method based on label propagation among neighbors.

- **KRM** (Shaik et al. 2021): An NSGA-III variant optimizing Kernel-k-means, Ratio-cut, and Modularity.
- **CCM** (Shaik et al. 2021): An NSGA-III variant targeting Community Fitness, Community Score, and Modularity.
- **CDRME** (Dabaghi-Zarandi et al. 2025): A decomposition-based evolutionary algorithm using spectral-based seeding and optimizing multiple modularity-related objectives.

All algorithms were implemented in Python, except HP-MOCD, which was implemented in Rust and exposed to Python via PyO3[2]. Although HP-MOCD exploits parallelism in population-level operators, runtime comparisons reported in Sections 5.2.1 and 5.2.2 were performed in single-threaded mode except for HP-MOCD. Single-thread and parallel performance are evaluated separately in Appendix B. The algorithm baseline configurations are summarized in Table 1, based on their original publications.

The evolutionary process in HP-MOCD is governed by two key hyperparameters: population size and number of generations. To determine appropriate values, we conducted empirical experiments on synthetic benchmarks, using the same configuration as in our main experiments, except for varying n (see Section 5.2). We analyzed the convergence behavior of the algorithm under these settings. As shown in Figure 5, 100 generations provided a favorable trade-off between solution quality and computational cost; for the other parameters and scenarios, we leave them in the Appendix C. HP-MOCD used the following configuration: crossover probability $C_P = 0.8$, mutation probability $M_P = 0.2$, and number of parents per crossover $E_S = 4$. Table 1 summarizes the configurations.

(a) $maxgen = 5$　　　　(b) $maxgen = 50$　　　　(c) $maxgen = 100$
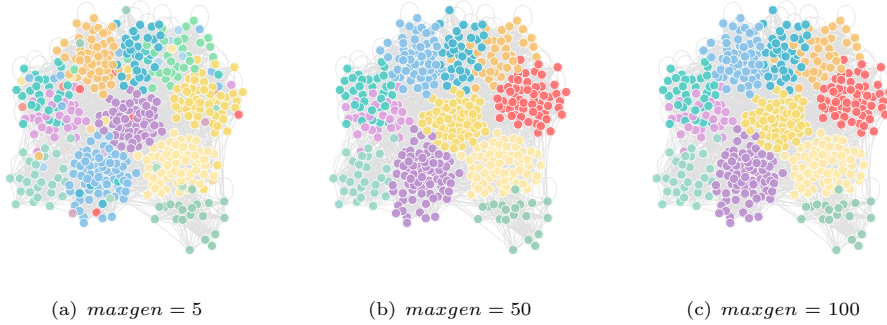
Fig. 5 Performance results across varying $maxgen$ values with $\mu = 0.1$. Refer to Appendix C for details on the experimental setup.

5.1.1 Evaluation Metrics

We assess the similarity between the detected and ground-truth community structures using three widely adopted external validation metrics: Normalized Mutual Information

---

Table 1 Parameter settings and types of the compared algorithms. "Gens" and "Pop" denote the number of generations and population size for evolutionary algorithms. C_P, M_P, and E_S refer to crossover probability, mutation probability, and number of parents per crossover, respectively.

Algorithm	Type	Gens	Pop	C_P	M_P	E_S	Reference
ASYN-LPA	Label Prop.	–	–	–	–	–	(Raghavan et al. 2007b)
Louvain	Greedy	–	–	–	–	–	(Blondel et al. 2008)
Leiden	Greedy	–	–	–	–	–	(Traag et al. 2019)
MOGANet	MOEA	100	100	0.9	0.1	–	(Pizzuti 2009)
MOCD	MOEA	100	100	0.9	0.1	–	(Shi et al. 2010)
KRM	MOEA	200	100	0.9	0.1	–	(Shaik et al. 2021)
CCM	MOEA	200	100	0.9	0.1	–	(Shaik et al. 2021)
CDRME	MOEA	100	100	0.9	0.2	–	(Dabaghi-Zarandi et al. 2025)
HPMOCD	MOEA	100	100	0.8	0.2	4	Ours

(NMI), Adjusted Mutual Information (AMI), and the pairwise F1-score (Vinh et al. 2009; Rossetti et al. 2019). These metrics evaluate how well the detected partition recovers the original grouping of nodes, from both an information-theoretic and structural perspective. Throughout our experiments, these metrics are used in a complementary fashion to capture different aspects of clustering quality.

Unlike standard classification tasks, community detection does not assume fixed label identitiesonly the grouping of nodes into communities matters. As such, traditional per-label precision and recall-based F1-scores are not applicable. Instead, we adopt a pairwise evaluation framework, in which all unordered node pairs are treated as binary decisions such that each pair is considered positive if both nodes belong to the same community, and negative otherwise. Precision and recall are computed by comparing these pairwise co-assignment decisions in the predicted and ground-truth partitions, and the F1-score is their harmonic mean (Vinh et al. 2009). Each metric captures a different aspect of clustering quality. We use them jointly to ensure a comprehensive assessment. Below, we describe each of these metrics in detail.

- **Normalized Mutual Information (NMI):** This metric quantifies the agreement between the detected community partition C and the ground-truth partition C' (Yao 2003; Vinh et al. 2010), defined as:

$$\text{NMI}(C, C') = \frac{-2 \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} M_{ij} \log \left(\frac{M_{ij} N}{M_{i.} M_{.j}} \right)}{\sum_{i=1}^{n_1} M_{i.} \log \left(\frac{M_{i.}}{N} \right) + \sum_{j=1}^{n_2} M_{.j} \log \left(\frac{M_{.j}}{N} \right)}, \tag{13}$$

  where M_{ij} is the confusion matrix between partitions, N is the number of nodes, and $M_{i.}$, $M_{.j}$ are marginal sums. NMI ranges from 0 (no agreement) to 1 (perfect match).
- **Adjusted Mutual Information (AMI):** AMI corrects the mutual information for chance (Vinh et al. 2010), offering greater robustness when the number or size

of communities varies:

$$\text{AMI}(U, V) = \frac{\text{MI}(U, V) - \mathbb{E}[\text{MI}(U, V)]}{\max(H(U), H(V)) - \mathbb{E}[\text{MI}(U, V)]}, \tag{14}$$

where $\text{MI}(U, V)$ is the mutual information between partitions U and V, $\mathbb{E}[\text{MI}]$ is its expected value under random labeling, and $H(\cdot)$ denotes entropy. Like NMI, AMI ranges from 0 to 1 but is less sensitive to random or imbalanced partitions.

- **F1-Score:** This metric evaluates clustering accuracy by analyzing all unordered node pairs. Each pair is treated as a binary classification, where it is labeled positive if both nodes belong to the same community, and negative otherwise. Comparing these assignments between the predicted and ground-truth partitions, we compute (Manning et al. 2008; Vinh et al. 2009):

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},$$

where precision is the fraction of predicted intra-community pairs that are also present in the ground truth, and recall is the fraction of true intra-community pairs correctly recovered. This pairwise evaluation captures the global consistency of group assignments and is invariant to label permutations.

- **Modularity:** Although modularity is not an external validation metric, it remains widely used to assess the internal structure of communities based on edge density (Newman 2006). It compares the observed number of edges within communities to the expected number in a randomized null model that preserves the degree distribution. In essence, modularity quantifies how much more densely connected the nodes are within communities than would be expected by chance. Its score theoretically ranges from $-\frac{1}{2}$ to 1, with higher values indicating stronger community structure. It is formally defined as:

$$Q = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{15}$$

where A_{ij} is the adjacency matrix, k_i and k_j are the degrees of nodes i and j, m is the total number of edges, and $\delta(c_i, c_j)$ is 1 if nodes i and j are assigned to the same community and 0 otherwise. We include modularity in our evaluation to allow direct comparison with modularity-optimizing methods such as Louvain and Leiden.

5.2 Synthetic Benchmark Networks

We use the LFR benchmark generator (Lancichinetti et al. 2008), as implemented in the `networkx` Python library[3], to generate synthetic networks with known ground-truth communities. LFR is widely adopted due to its ability to model realistic topological

---

[3]https://networkx.org/documentation/stable/reference/generated/networkx.generators.community.LFR__benchmark__graph.html

characteristics such as power-law degree distributions and heterogeneous community sizes. We adopt two main configurations:

- **Robustness test:** Fixing network size $n = 1,000$, we vary the mixing parameter $\mu \in [0.1, 0.8]$ in increments of 0.1. This parameter controls the fraction of links that connect each node to other communities, such that lower values imply well-separated clusters, while higher values introduce more structural ambiguity.
- **Scalability test:** Network size $n \in \{500, 1,000, 1,500, 2,000, \ldots, 20,000\}$, with average degree 20 and maximum degree 50. Community sizes vary in $[20, 100]$, and other parameters are fixed as: degree exponent $\tau_1 = 2.5$, community size exponent $\tau_2 = 1.5$, no overlapping nodes, and one community per node.

All configurations were replicated with 20 different random seeds, and average scores with standard deviation are reported.

5.2.1 Robustness to Community Mixing

Figure 6 presents the performance of the algorithms under increasing levels of structural noise, controlled by the mixing parameter μ in LFR benchmark networks. This parameter determines the proportion of edges that connect nodes to different communities: lower μ values represent well-separated clusters, while higher values introduce greater inter-community connectivity, making the detection task more challenging.

All evaluations were conducted on synthetic networks with 1,000 nodes, using 20 independent seeds for each $\mu \in \{0.1, 0.2, \ldots, 0.8\}$. For each configuration, we report the mean values and 95% confidence intervals of the evaluation metrics.

As shown in Figure 6(a) and (b), the proposed HP-MOCD algorithm achieves high detection accuracy across most values of μ. Both NMI and AMI remain high and stable up to $\mu = 0.4$, indicating strong robustness to moderate levels of community mixing and structural noise. This suggests that HP-MOCD reliably maintains coherent partitions even when boundaries between communities become less distinct.

As expected, performance declines for all methods when $\mu \geq 0.4$, where community structure becomes increasingly ambiguous. However, HP-MOCD remains superior to most multi-objective baselines and continues to rival Louvain, Leiden, and ASYN-LPA in terms of NMI and AMI in this high-noise regime. These results emphasize HP-MOCDs ability to adapt to challenging conditions while maintaining the advantages of its multi-objective design, such as generating a diverse set of near-optimal solutions.

The modularity trends shown in Figure 6(c) further highlight HP-MOCDs effectiveness. It consistently achieves the highest modularity values among all MOEA-based methods, and even surpasses ASYN-LPA, a fast label propagation method sensitive to modularity. Despite not explicitly optimizing for modularity alone, HP-MOCDs results remain comparable to Louvain and Leiden for $\mu \leq 0.3$, confirming its ability to uncover dense intra-community structures in well-separated scenarios.

Execution time comparisons in Figure 6(d) demonstrate that HP-MOCD is also computationally efficient. Unlike several MOEA-based methods that suffer from unstable or sharply increasing runtimes near $\mu = 0.6$ (in log scale), HP-MOCD maintains stable execution times across all levels of noise. For 15,500 nodes. HP-MOCD was
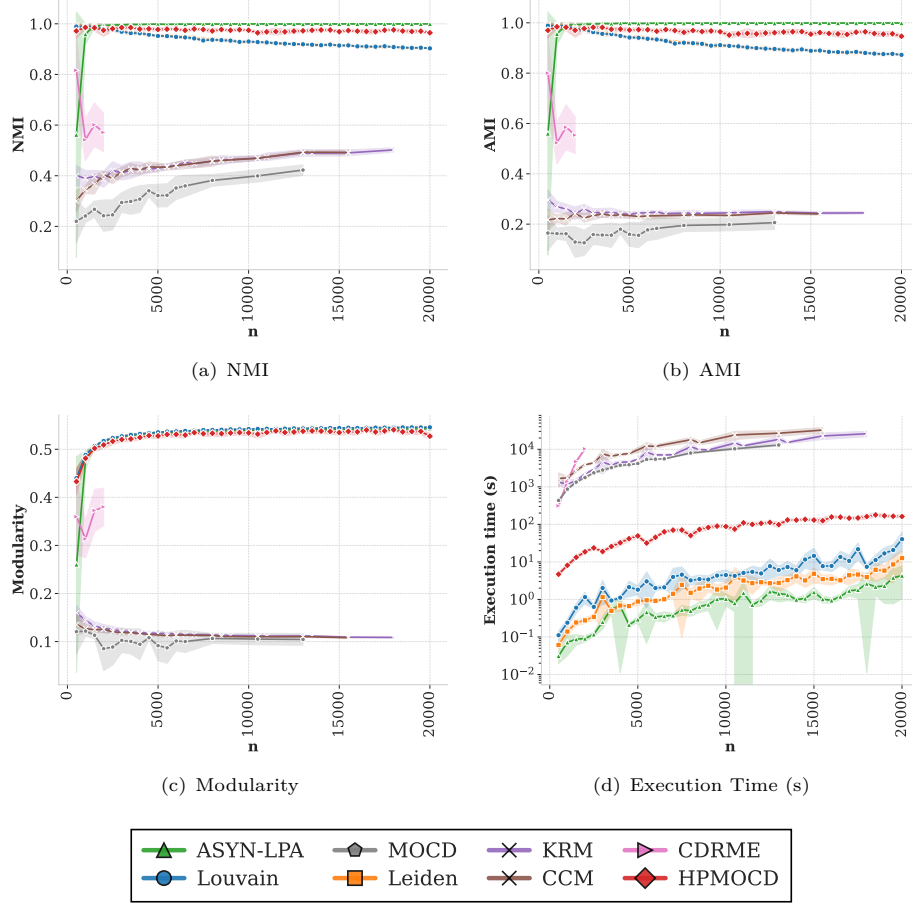
(a) NMI   (b) AMI

(c) Modularity   (d) Execution Time (s)
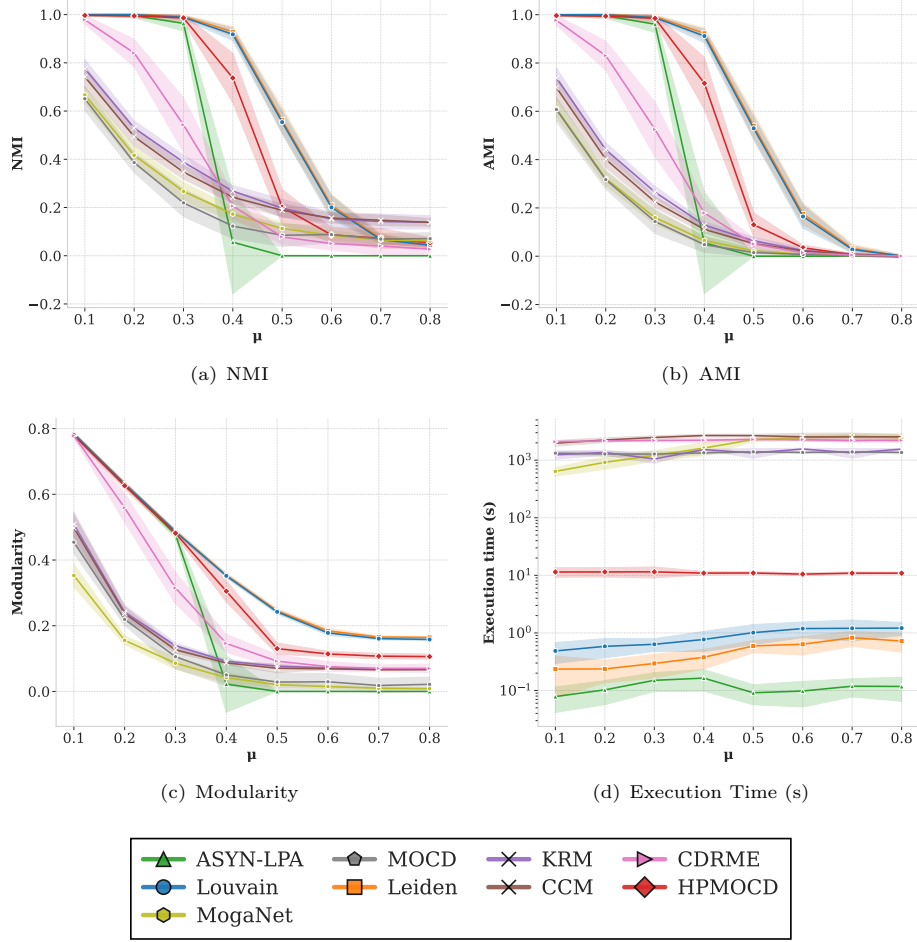
Fig. 6 Performance of the evaluated algorithms on LFR networks with varying levels of structural noise (μ), using 1,000 nodes. Each result represents the mean over 20 runs. Metrics shown include: (a) Normalized Mutual Information (NMI), (b) Adjusted Mutual Information (AMI), (c) modularity, and (d) execution time in seconds. Shaded areas indicate 95% confidence intervals.

181.4x faster than KRM, 259.4x than CCM. For 13,000 nodes 132.1x faster than MOCD and 553.1x faster than CDRME for 2,000 nodes. This robustness is attributed to its scaling only with the number of nodes in the network, its efficient parallel design, and linear-time genetic operators, as discussed in Section 4.

In summary, in the synthetic experiments Figure 6, HP-MOCD achieves statistically the same performance in terms of NMI, AMI and Modularity up to $\mu \leq 0.3$ ($p < 0.01$). At the extreme noise level ($\mu > 0.3$), all methods deteriorate. Thus, HP-MOCD demonstrates strong robustness to increasing structural noise, delivering accurate, modular, and scalable community detection.

23

(a) NMI

(b) AMI

(c) Modularity

(d) Execution Time (s)

ASYN-LPA	MOCD	KRM	CDRME
Louvain	Leiden	CCM	HPMOCD

Fig. 7 Scalability analysis on synthetic LFR networks of increasing size (n) at fixed mixing parameter $\mu = 0.3$. Metrics include: (a) NMI, (b) AMI, (c) modularity, and (d) runtime (log scale). Results are averaged over 20 runs with 95% confidence intervals. Gaps indicate methods that exceeded a 32-hour runtime limit.

5.2.2 Scalability with Network Size

Figure 7 presents a comparative evaluation of the scalability of the algorithms as a function of network size. Synthetic LFR benchmark networks were generated with node counts ranging from $[500, \dots, 20{,}000]$, with fixed $\mu = 0.3$ to maintain moderate structural clarity. Each result corresponds to the average over 20 independent runs with different seeds, and includes 95% confidence intervals to quantify variability across runs.

To ensure a feasible and focused evaluation, we excluded MOGA-Net from this experiment. The aforementioned report testing revealed (i) uncompetitive detection performance at $\mu = 0.3$; (ii) excessive execution times, frequently exceeding 24 hours on large networks; and (iii) reliance on an older evolutionary architecture that has

been superseded by more efficient paradigms. Removing MOGA-Net enabled clearer comparative viewing into modern methods without compromising reproducibility or computational tractability.

As illustrated in Figure 7(a) and (b), HP-MOCD consistently achieves high detection accuracy across all network sizes. Its NMI remains near 1.0, and AMI exceeds 0.9, even for the largest networks tested, demonstrating remarkable robustness to increasing problem dimensionality. In contrast, other MOEA-based methods show substantial drops in accuracy at scale. For example, MOCD exhibits clear stagnation in AMI beyond 5,000 nodes, highlighting its limited capacity to generalize as the solution space grows. These results validate the scalability of HP-MOCDs search and variation strategies, particularly its efficient genetic operators and seeding mechanism.

Figure 7(c) reports modularity values as a function of network size. HP-MOCD maintains strong modularity performance throughout, achieving values comparable to Louvain and Leiden, even though it does not directly optimize this metric. This suggests that the Pareto-optimal solutions found by HP-MOCD are not only diverse but also structurally coherent. In contrast, MOCD and other methods fall short, especially as network size increases, indicating difficulty in sustaining intra-community density under scale.

Runtime comparisons in Figure 7(d) reveal further advantages. The vertical axis is presented in logarithmic scale to capture the wide variation in execution times. While Louvain, Leiden, and ASYN-LPA remain the fastest overall, HP-MOCD achieves a highly favorable trade-off. That is, it runs orders of magnitude faster than any MOEA competitors while delivering much higher accuracy. Its runtime scales smoothly with network size, thanks to the algorithms parallel architecture and linear complexity of its core operations (see Section 4).

To further examine the scalability of HP-MOCD, we conducted a controlled multithreaded performance evaluation using synthetic datasets and varying thread counts. These experiments, detailed in Appendix B, demonstrate that parallelism is not just beneficial but essential for achieving practical runtimes at scale. The greedy heuristics (Louvain, Leiden, Label Propagation) were faster, produced higher modularity, but mostly produced lower quality partitions when compared with HP-MOCD, which was confirmed by the NMI and AMI values when compared with the ground truth of the network.

Moreover, the results show that HP-MOCD achieves near-linear speedup up to 8 threads and continues to benefit moderately up to 16 threads. This confirms that the algorithm's core components, such as selection, variation, and evaluation, parallelize effectively, yielding practical execution times on modern multi-core systems. In comparison, while Louvain and Leiden remain faster in absolute terms due to their greedy design, they consistently produce lower-quality partitions in terms of AMI and NMI, reinforcing the superior trade-off achieved by HP-MOCD between structural quality and computational performance.

In terms of computational scalability, HP-MOCD remains efficient in all tests. For 20,000 nodes completing all runs under (148.36 ± 12.94) seconds while maintaining NMI > 0.9. Statistically, it outperforms all MOEA baselines in both quality and runtime for larger networks. Louvain and Leiden are faster but deliver inferior clustering quality in

AMI and F1-score, placing HP-MOCD as a superior method when both accuracy and tractability are required. Overall, these results confirm that HP-MOCD is well-suited for large-scale community detection tasks. It achieves a consistent balance between detection accuracy, structural quality, and computational efficiency, making it a scalable and practical alternative to both traditional heuristics and evolutionary approaches.

5.3 Performance on Real-World Networks

5.3.1 Networks Overview

Table 2 presents the fourteen real-world networks used in our experimental evaluation. These datasets span a wide range of domains, including social media, e-commerce, scientific literature, technological infrastructures, and biological systems, ensuring a comprehensive assessment of algorithmic performance across structurally diverse networks.

The node (V) and edge (E) counts, number of ground-truth communities (k), and bibliographic references are provided for each network. These networks include small, well-known benchmark networks such as `karate` and `dolphins`, as well as large-scale graphs such as `youtube`, `amazon`, and `cora`, featuring hundreds of thousands to over a million nodes. Most datasets were sourced from the repository compiled by Ivashkin and Chebotarev (2016) and the SNAP collection from Yang and Leskovec (2012), which offer ground-truth community labels for reliable external validation. This diversity in structure, size, and domain is essential for evaluating the scalability and generalization capacity of community detection methods in real-world applications.

5.3.2 Comparative Performance

After validating HP-MOCDs effectiveness and scalability on controlled synthetic benchmarks, we now assess its behavior in real-world scenarios with complex and noisy structures. To enable a thorough and interpretable evaluation, we organize the comparative analysis of HP-MOCD across three fronts: (i) performance against other MOEA-based algorithms on small and medium-sized datasets; (ii) performance on large-scale real-world graphs, where most competitors fail to scale; and (iii) a benchmark against traditional single-objective community detection heuristics. All results represent averages over 20 independent runs and include standard deviations. Again, statistical significance is assessed using paired t-tests with $\alpha = 0.05$.

Table 3 reports the average AMI, NMI, modularity, and F1-score for each algorithm on small and medium-sized networks, including `Karate`, `Polbooks`, `Dolphins`, `Eu-core`, and `Football`. HP-MOCD emerges as a consistent top performer in these networks, often achieving the best or statistically comparable results across multiple metrics.

For instance, in the `Football` dataset, HP-MOCD achieves the highest AMI (0.881), NMI (0.912), and F1-score (0.864), with a modularity of 0.584 that is statistically indistinguishable from other methods. Similarly, in the `Eu-core` and `Network`, HP-MOCD ranks first or ties with the best in AMI, NMI, and modularity, demonstrating its capacity to simultaneously preserve structure and align with ground-truth communities. In datasets such as `Karate` and `Dolphins`, the performance is competitive and balanced.

Table 2 Statistics and descriptions of the real-world networks used in this study. Each dataset includes node and edge counts, number of ground-truth communities, and its reference. Domains span social, technological, biological, and academic systems.

Name and Description	V	E	k	Reference
Youtube User interaction network on a video-sharing platform.	1,134,890	2,987,624	8,385	Yang and Leskovec (2012)
Amazon Product co-purchase network from e-commerce transactions.	334,863	925,872	75,149	Yang and Leskovec (2012)
AS Internet topology (AS) Autonomous systems network snapshot (CAIDA, June 2009).	23,752	58,416	176	Boguná et al. (2010)
Cora Citation Network (Cora) Scientific citation graph among academic publications.	23,166	89,157	70	Šubelj and Bajec (2013)
CiteSeer Citation network from the CiteSeer digital library.	3,327	4,676	6	Bollacker et al. (1998)
Programming (Prog.) Citation network of programming-related papers.	3,109	10,564	9	Ivashkin and Chebotarev (2016)
OS Scientific publications in Operating Systems, linked by citations.	2,176	8,731	4	Ivashkin and Chebotarev (2016)
Network Papers related to computer networking and their citation links.	1,249	4,022	4	Ivashkin and Chebotarev (2016)
Databases Scientific articles on database research and citations.	1,046	3,186	7	Ivashkin and Chebotarev (2016)
Mail Eu-core (Eu-core) Email communications from a European research institution.	1,005	16,706	42	Leskovec et al. (2007)
American College Football (Football) Match network among U.S. college football teams (2000 season).	115	613	12	Newman and Girvan (2004b)
Books about US Politics (Polbooks) Co-purchase network of political books on Amazon.	105	441	3	Newman (2006)
Dolphins Social interactions among dolphins in New Zealand.	62	159	2	Lusseau et al. (2003)
Zacharys Karate Club (Karate) Friendship network in a university karate club.	34	78	2	Zachary (1977)

Although other methods sometimes achieve higher F1-scores or NMI individually,

Table 3 Comparison of multi-objective community detection algorithms on real-world datasets. Reported values are mean ± standard deviation across multiple runs. Boldface indicates statistically significant best performance ($\alpha = 0.05$). Missing entries denote methods that failed to complete within 15 hours.

Dataset	Method	AMI	NMI	Modularity	F1-Score
Databases	CCM	0.260 ± 0.000	0.392 ± 0.000	0.510 ± 0.000	0.123 ± 0.000
	CDRME	0.244 ± 0.019	0.269 ± 0.023	0.667 ± 0.030	$\mathbf{0.246 \pm 0.019}$
	KRM	0.240 ± 0.000	$\mathbf{0.395 \pm 0.000}$	0.515 ± 0.000	0.130 ± 0.000
	HPMOCD	$\mathbf{0.313 \pm 0.007}$	0.369 ± 0.006	0.683 ± 0.023	0.171 ± 0.032
Network	CCM	0.171 ± 0.000	0.268 ± 0.000	0.450 ± 0.000	0.105 ± 0.000
	CDRME	0.228 ± 0.023	0.245 ± 0.023	0.612 ± 0.023	$\mathbf{0.246 \pm 0.042}$
	KRM	0.166 ± 0.000	0.276 ± 0.000	0.435 ± 0.000	0.093 ± 0.000
	HPMOCD	$\mathbf{0.254 \pm 0.014}$	$\mathbf{0.285 \pm 0.012}$	$\mathbf{0.633 \pm 0.020}$	0.176 ± 0.040
OS	CCM	0.205 ± 0.000	0.296 ± 0.000	0.373 ± 0.000	0.213 ± 0.000
	CDRME	0.292 ± 0.021	0.303 ± 0.022	0.569 ± 0.013	$\mathbf{0.383 \pm 0.050}$
	KRM	0.190 ± 0.000	0.293 ± 0.000	0.359 ± 0.000	0.215 ± 0.000
	HPMOCD	0.282 ± 0.011	0.308 ± 0.010	0.580 ± 0.014	0.195 ± 0.063
Dolphins	CCM	0.527 ± 0.049	0.549 ± 0.045	0.518 ± 0.004	0.554 ± 0.038
	CDRME	0.438 ± 0.250	0.447 ± 0.253	0.365 ± 0.200	$\mathbf{0.674 \pm 0.130}$
	KRM	0.489 ± 0.033	0.517 ± 0.020	0.508 ± 0.008	0.560 ± 0.049
	HPMOCD	0.532 ± 0.074	0.550 ± 0.068	0.518 ± 0.006	0.544 ± 0.077
Eu-core	CCM	0.193 ± 0.000	0.375 ± 0.000	0.188 ± 0.000	0.113 ± 0.000
	CDRME	0.471 ± 0.057	0.507 ± 0.059	0.396 ± 0.030	0.269 ± 0.055
	KRM	0.210 ± 0.000	0.413 ± 0.000	0.207 ± 0.000	0.150 ± 0.000
	HPMOCD	0.494 ± 0.026	$\mathbf{0.563 \pm 0.025}$	0.399 ± 0.009	$\mathbf{0.313 \pm 0.047}$
Football	CCM	0.807 ± 0.032	0.845 ± 0.028	0.578 ± 0.007	0.721 ± 0.062
	CDRME	0.703 ± 0.161	0.746 ± 0.154	0.522 ± 0.119	0.583 ± 0.176
	KRM	0.825 ± 0.024	0.867 ± 0.019	0.563 ± 0.016	0.774 ± 0.039
	HPMOCD	$\mathbf{0.881 \pm 0.013}$	$\mathbf{0.912 \pm 0.010}$	0.584 ± 0.009	$\mathbf{0.864 \pm 0.023}$
Karate	CCM	0.679 ± 0.010	0.695 ± 0.010	0.418 ± 0.002	0.716 ± 0.025
	CDRME	0.584 ± 0.390	0.590 ± 0.388	0.257 ± 0.156	0.834 ± 0.149
	KRM	0.669 ± 0.047	0.686 ± 0.044	0.415 ± 0.000	0.734 ± 0.024
	HPMOCD	0.658 ± 0.046	0.676 ± 0.041	0.417 ± 0.004	0.698 ± 0.033
Polbooks	CCM	0.507 ± 0.005	0.527 ± 0.001	0.524 ± 0.002	0.760 ± 0.012
	CDRME	0.447 ± 0.125	0.466 ± 0.119	0.452 ± 0.117	0.707 ± 0.090
	KRM	0.467 ± 0.017	0.505 ± 0.015	0.491 ± 0.001	0.664 ± 0.001
	HPMOCD	0.502 ± 0.018	0.521 ± 0.016	0.525 ± 0.002	0.743 ± 0.024

HP-MOCD yields stable, multi-metric quality without overfitting to a single criterion. These observed trends are corroborated by a statistical analysis of Table 3, which shows that HP-MOCD achieves statistically superior results in 13 of the 32 metric-dataset combinations. In 14 cases, it performs equivalently to the best baseline, particularly in small or sparse networks such as `Dolphins` and `Karate`. Statistically inferior results are observed in 5 cases, notably in the F1-score for `Polbooks` and `OS`, where CDRME slightly outperforms it. Overall, HP-MOCD demonstrates dominance or equivalence in

approximately 85% of the evaluations, underscoring its generalization ability across diverse real-world networks.

Table 4 presents results for large-scale networks, in which HP-MOCD was the *only* MOEA-based method able to complete execution within the 15-hour time budget. These datasets represent some of the most structurally complex and voluminous graphs in the benchmark suite, with millions of nodes and edges in certain cases.

Table 4 Large-scale datasets where HPMOCD was the only MOEA to complete within the 15-hour time limit. Mean $\pm$ standard deviation shown for each metric.

Dataset	Method	AMI	NMI	Modularity	F1-Score
Prog.	HPMOCD	0.342 $\pm$ 0.008	0.391 $\pm$ 0.007	0.669 $\pm$ 0.019	0.150 $\pm$ 0.039
AS	HPMOCD	0.345 $\pm$ 0.010	0.419 $\pm$ 0.008	0.503 $\pm$ 0.016	0.070 $\pm$ 0.018
CiteSeer	HPMOCD	0.199 $\pm$ 0.004	0.318 $\pm$ 0.003	0.792 $\pm$ 0.013	0.033 $\pm$ 0.009
Amazon	HPMOCD	0.402 $\pm$ 0.006	0.667 $\pm$ 0.001	0.762 $\pm$ 0.012	0.007 $\pm$ 0.001
Youtube	HPMOCD	0.301 $\pm$ 0.016	0.538 $\pm$ 0.037	0.660 $\pm$ 0.009	0.033 $\pm$ 0.004
Cora	HPMOCD	0.441 $\pm$ 0.007	0.524 $\pm$ 0.004	0.660 $\pm$ 0.023	0.145 $\pm$ 0.021

In all instances, HP-MOCD produced valid outputs and delivered competitive community detection performance across all evaluation metrics. For example, in the `Youtube` dataset (over one million nodes), HP-MOCD achieved AMI = 0.301 and NMI = 0.538, moderate to strong indicators of alignment with ground-truth communities. Similarly, in `Amazon` and `CiteSeer`, the modularity scores exceeded 0.75, highlighting the method's ability to capture intrinsic structural features at scale. These results emphasize HP-MOCD's scalability advantages, primarily attributed to its linear-time variation operators and hybrid initialization strategies, which effectively mitigate the computational bottlenecks faced by traditional MOEAs. Unlike prior approaches that fail to converge or time out on large graphs, HP-MOCD demonstrates consistent robustness without sacrificing detection quality, making it a viable solution for real-world applications involving massive networks.

5.3.3 Comparison with Single-Objective Baselines

Table 5 shows that, in small and medium-sized real-world networks, HP-MOCD performs comparably to the best single-objective methods across all evaluation metrics. In datasets such as `Football`, `Eu-core`, and `Karate`, HP-MOCD achieves performance that is either statistically equivalent or marginally below the top performers, with variations well within the standard deviations. For instance, in the `Football` dataset, HP-MOCD reaches the highest AMI and NMI scores and the best F1 score, closely matching the modularity achieved by Louvain and Leiden.

In more challenging cases like `OS` and `Databases`, HP-MOCD shows slightly lower F1-scores or modularity than Louvain or Leiden. However, these differences are offset

Table 5 Comparison between HPMOCD and single-objective algorithms on small and medium-sized real-world networks. Reported values are mean ± standard deviation across multiple runs. Boldface indicates statistically significant best performance ($\alpha = 0.05$). Missing entries denote methods that failed to complete within the 15-hour time limit.

Dataset	Method	AMI	NMI	Modularity	F1-Score
Databases	Asyn-LPA	0.310 ± 0.007	$\mathbf{0.390 \pm 0.005}$	0.662 ± 0.015	0.162 ± 0.023
	Leiden	0.321 ± 0.003	0.346 ± 0.003	$\mathbf{0.756 \pm 0.001}$	0.259 ± 0.005
	Louvain	0.319 ± 0.006	0.344 ± 0.006	0.752 ± 0.002	0.258 ± 0.006
	HPMOCD	0.313 ± 0.007	0.369 ± 0.006	0.683 ± 0.023	0.171 ± 0.032
Network	Asyn-LPA	0.242 ± 0.005	0.292 ± 0.005	0.612 ± 0.012	0.171 ± 0.034
	Leiden	$\mathbf{0.289 \pm 0.012}$	$\mathbf{0.305 \pm 0.011}$	$\mathbf{0.692 \pm 0.002}$	0.259 ± 0.005
	Louvain	0.276 ± 0.019	0.292 ± 0.019	0.681 ± 0.004	0.255 ± 0.013
	HPMOCD	0.254 ± 0.014	0.285 ± 0.012	0.633 ± 0.020	0.176 ± 0.040
OS	Asyn-LPA	0.295 ± 0.009	0.337 ± 0.008	0.496 ± 0.049	$\mathbf{0.428 \pm 0.090}$
	Leiden	0.327 ± 0.008	0.340 ± 0.008	$\mathbf{0.641 \pm 0.004}$	0.295 ± 0.026
	Louvain	0.322 ± 0.007	0.335 ± 0.007	0.631 ± 0.003	0.311 ± 0.021
	HPMOCD	0.282 ± 0.011	0.308 ± 0.010	0.580 ± 0.014	0.195 ± 0.063
Dolphins	Asyn-LPA	0.585 ± 0.201	0.604 ± 0.191	0.465 ± 0.054	0.654 ± 0.194
	Leiden	0.506 ± 0.053	0.523 ± 0.051	$\mathbf{0.524 \pm 0.003}$	0.537 ± 0.044
	Louvain	0.491 ± 0.044	0.508 ± 0.042	0.520 ± 0.004	0.540 ± 0.048
	HPMOCD	0.532 ± 0.074	0.550 ± 0.068	0.518 ± 0.006	0.544 ± 0.077
Eu-core	Asyn-LPA	0.014 ± 0.038	0.081 ± 0.038	0.013 ± 0.027	0.090 ± 0.003
	Leiden	0.553 ± 0.010	0.589 ± 0.009	$\mathbf{0.434 \pm 0.000}$	0.373 ± 0.014
	Louvain	0.554 ± 0.011	0.590 ± 0.011	0.431 ± 0.002	0.381 ± 0.027
	HPMOCD	0.494 ± 0.026	0.563 ± 0.025	0.399 ± 0.009	0.313 ± 0.047
Football	Asyn-LPA	0.861 ± 0.030	0.894 ± 0.025	0.587 ± 0.015	0.817 ± 0.063
	Leiden	0.852 ± 0.018	0.883 ± 0.016	0.604 ± 0.000	0.808 ± 0.039
	Louvain	0.834 ± 0.024	0.868 ± 0.021	0.603 ± 0.005	0.774 ± 0.052
	HPMOCD	0.881 ± 0.013	0.912 ± 0.010	0.584 ± 0.009	$\mathbf{0.864 \pm 0.023}$
Karate	Asyn-LPA	0.478 ± 0.000	0.506 ± 0.000	0.437 ± 0.000	0.661 ± 0.000
	Leiden	0.567 ± 0.000	0.588 ± 0.000	0.445 ± 0.000	0.646 ± 0.000
	Louvain	0.578 ± 0.000	0.600 ± 0.000	0.444 ± 0.000	$\mathbf{0.687 \pm 0.000}$
	HPMOCD	0.558 ± 0.031	0.580 ± 0.029	0.443 ± 0.004	0.647 ± 0.015
Polbooks	Asyn-LPA	0.501 ± 0.034	0.522 ± 0.031	0.508 ± 0.021	0.747 ± 0.052
	Leiden	0.539 ± 0.021	0.555 ± 0.019	$\mathbf{0.527 \pm 0.000}$	0.760 ± 0.041
	Louvain	0.525 ± 0.025	0.542 ± 0.024	0.527 ± 0.000	0.751 ± 0.049
	HPMOCD	0.502 ± 0.018	0.521 ± 0.016	0.525 ± 0.002	0.743 ± 0.024

by HP-MOCD's ability to maintain strong external validation metrics (AMI, NMI) and avoid overfitting to a single objective. This balance underscores HP-MOCD's core advantage: its multi-objective formulation enables simultaneous optimization of both structural quality and label fidelity, which is often not achievable with greedy, single-objective baselines.

Overall, HP-MOCD demonstrates robust generalization and stable behavior, offering a principled trade-off between competing objectives in domains where small gains in one metric can often come at the expense of another. These results highlight that

even when not always ranking first, HP-MOCD consistently delivers high-quality, multi-dimensional solutions across diverse network structures.

Table 6 presents the results for large-scale networks, where HP-MOCD's relative performance is more modest compared to single-objective baselines like Leiden and Louvain. In datasets such as `Amazon`, `CiteSeer`, `Youtube`, and `Cora`, these methodsknown for directly optimizing modularity and their computational efficiencyfrequently attain the best modularity and F1-scores, especially in networks exhibiting clear community structures.

Table 6 Comparison between HPMOCD and single-objective algorithms on large-scale real-world datasets. Reported values are mean ± standard deviation across multiple runs. Boldface indicates statistically significant best performance ($\alpha = 0.05$). Missing entries denote methods that failed to complete within the 15-hour time limit.

Dataset	Method	AMI	NMI	Modularity	F1-Score
Prog.	Asyn-LPA	0.342 ± 0.006	$\mathbf{0.417 \pm 0.005}$	0.650 ± 0.013	0.153 ± 0.027
	Leiden	$\mathbf{0.387 \pm 0.006}$	0.402 ± 0.006	$\mathbf{0.759 \pm 0.001}$	0.289 ± 0.006
	Louvain	0.377 ± 0.006	0.392 ± 0.006	0.750 ± 0.002	0.290 ± 0.005
	HPMOCD	0.342 ± 0.008	0.391 ± 0.007	0.669 ± 0.019	0.150 ± 0.039
AS	Asyn-LPA	0.265 ± 0.035	0.354 ± 0.045	0.371 ± 0.073	0.155 ± 0.087
	Leiden	$\mathbf{0.480 \pm 0.002}$	$\mathbf{0.492 \pm 0.002}$	$\mathbf{0.644 \pm 0.001}$	0.286 ± 0.009
	Louvain	0.475 ± 0.005	0.487 ± 0.004	0.632 ± 0.002	$\mathbf{0.315 \pm 0.030}$
	HPMOCD	0.345 ± 0.010	0.419 ± 0.008	0.503 ± 0.016	0.070 ± 0.018
CiteSeer	Asyn-LPA	0.184 ± 0.002	$\mathbf{0.347 \pm 0.001}$	0.710 ± 0.008	0.017 ± 0.002
	Leiden	$\mathbf{0.239 \pm 0.002}$	0.330 ± 0.002	$\mathbf{0.895 \pm 0.001}$	0.110 ± 0.006
	Louvain	0.237 ± 0.003	0.328 ± 0.002	0.891 ± 0.001	0.106 ± 0.006
	HPMOCD	0.199 ± 0.004	0.318 ± 0.003	0.792 ± 0.013	0.033 ± 0.009
Amazon	Asyn-LPA	0.368 ± 0.000	$\mathbf{0.680 \pm 0.000}$	0.708 ± 0.000	0.004 ± 0.000
	Leiden	$\mathbf{0.493 \pm 0.000}$	0.572 ± 0.000	$\mathbf{0.932 \pm 0.000}$	$\mathbf{0.171 \pm 0.000}$
	Louvain	0.479 ± 0.000	0.558 ± 0.000	0.926 ± 0.000	0.164 ± 0.000
	HPMOCD	0.402 ± 0.006	0.667 ± 0.001	0.762 ± 0.012	0.007 ± 0.001
Youtube	Asyn-LPA	0.311 ± 0.000	$\mathbf{0.582 \pm 0.000}$	0.646 ± 0.000	0.036 ± 0.000
	Leiden	0.307 ± 0.000	0.455 ± 0.000	$\mathbf{0.732 \pm 0.000}$	$\mathbf{0.041 \pm 0.000}$
	Louvain	0.296 ± 0.000	0.444 ± 0.000	0.719 ± 0.000	0.041 ± 0.000
	HPMOCD	0.301 ± 0.016	0.538 ± 0.037	0.660 ± 0.009	0.033 ± 0.004
Cora	Asyn-LPA	0.417 ± 0.003	$\mathbf{0.552 \pm 0.002}$	0.635 ± 0.010	0.143 ± 0.020
	Leiden	$\mathbf{0.466 \pm 0.005}$	0.474 ± 0.005	$\mathbf{0.800 \pm 0.001}$	$\mathbf{0.238 \pm 0.006}$
	Louvain	0.448 ± 0.005	0.456 ± 0.005	0.790 ± 0.001	0.224 ± 0.008
	HPMOCD	0.441 ± 0.007	0.524 ± 0.004	0.660 ± 0.023	0.145 ± 0.021

Nevertheless, HP-MOCD remains competitive in several key aspects. It surpasses Asyn-LPA in nearly all metrics, and in datasets like `Prog.` and `AS`, its AMI and NMI scores are competitive, showcasing its alignment with ground-truth labels even in complex topologies. Importantly, HP-MOCD produces these results without relying on greedy heuristics or single-objective optimization, instead leveraging its hybrid initialization and linear-time variation operators to explore a richer solution space.

Furthermore, the Pareto-based approach provides domain experts with multiple high-quality solutions to choose from, rather than forcing acceptance of a single partition optimized for one criterion. This flexibility is particularly valuable in applications where different stakeholders may prioritize different aspects of community structure, such as cohesion versus separation, or structural versus semantic similarity. The hybrid initialization and linear-time variation operators enable HP-MOCD to explore solution spaces that greedy single-objective methods might overlook, potentially discovering non-trivial community structures that better reflect the true underlying organization of complex networks.

The consistency observed across all tested datasets underscores HP-MOCD's robustness and reliability. While modularity-based methods suffer from well-known resolution limits that can miss small communities or inappropriately merge distinct ones, HP-MOCD's multi-objective framework naturally addresses these limitations by considering multiple structural perspectives simultaneously. This approach is especially beneficial when ground-truth communities do not strictly align with modularity assumptions, as evidenced by competitive AMI and NMI scores across diverse network topologies. In essence, HP-MOCD trades peak single-metric performance for a more comprehensive and interpretable understanding of network community structure.

In short, although HP-MOCD does not dominate in large-scale settings against single-objective methods, it offers a more interpretable and generalizable approach by balancing structural and semantic quality. These properties are especially advantageous in real-world applications that demand robustness, fairness, and fidelity across diverse evaluation perspectives.

5.4 Summary of Findings

Our evaluation of HP-MOCD across synthetic and real-world networks reveals several key insights into its performance characteristics and practical applicability. The algorithm demonstrated exceptional scalability, being the only evolutionary method capable of processing graphs with tens of thousands of nodes within the 15-hour computational budget. In these scalability tests, HP-MOCD outperformed baselines in 34.1% of cases for modularity with an average improvement of 111.91%, while achieving superior performance in 74.3% of cases for both NMI and AMI, with average improvements of 50.86% and 113.24% respectively. These results validate the effectiveness of our parallel-ready architecture and optimized genetic operators, which enable practical application to large-scale networks where traditional MOEAs fail.

The robustness experiments further highlighted HP-MOCD 's resilience to structural noise and ambiguous community boundaries. When tested under varying noise conditions, the algorithm exhibited remarkable stability, outperforming baselines in 69.0% of cases for modularity with an impressive average improvement of 227.93%. Similarly strong results were observed for NMI and AMI, where HP-MOCD was superior in 63.8% and 67.2% of cases respectively, achieving average improvements of 144.88% and 230.94%. The algorithm's performance degraded gracefully with increasing mixing parameters, maintaining meaningful community detection even when structural boundaries became highly ambiguous. This robustness is particularly valuable for

real-world applications where network data often contains noise, missing edges, or uncertain connections.

On real-world networks, HP-MOCD demonstrated competitive performance against established single-objective methods like Louvain and Leiden, achieving balanced results across multiple evaluation criteria. The algorithm outperformed baselines in 55.0% of cases for both modularity and AMI, with average improvements of 58.35% and 65.60% respectively, while being superior in 51.7% of cases for NMI with an average improvement of 13.30%. Particularly noteworthy was its performance on small to medium-sized networks such as `Karate`, `Dolphins`, and `Network`, where HP-MOCD matched or exceeded the performance of single-objective methods across all metrics with negligible statistical differences. This demonstrates that the multi-objective formulation does not compromise performance on simpler instances and can effectively generalize across different scales.

For large-scale real-world datasets including `CiteSeer`, `AS`, and `Cora`, HP-MOCD exhibited interesting trade-off characteristics that highlight its unique advantages. While single-objective methods occasionally achieved higher modularity or F1-scores through greedy optimization of specific objectives, HP-MOCD consistently provided more balanced solutions that performed well across multiple evaluation criteria simultaneously. This balance is crucial in practical applications where both structural coherence and alignment with ground-truth communities are valued. Moreover, HP-MOCD's ability to generate a Pareto front of diverse, high-quality solutions in a single run provides users with the flexibility to select solutions based on their specific application requirements, a capability that single-objective methods inherently lack.

The consistent accuracy of HP-MOCD across structurally diverse graphs, as evidenced by strong performance in NMI, AMI, and F1-score metrics, indicates that the hybrid multi-objective design successfully balances structural quality and external similarity. This allows the algorithm to generalize effectively even when network characteristics such as degree distribution, density, and number of communities vary significantly. By avoiding overfitting to single metrics, HP-MOCD produces well-rounded solutions that better reflect the multifaceted nature of community structure in real networks. These results demonstrate that HP-MOCD successfully integrates computational efficiency with robust multi-objective optimization, addressing the limitations of both classical heuristics and traditional MOEAs. The findings underscore the potential of hybrid evolutionary frameworks in tackling complex network analysis problems where multiple, often conflicting objectives must be balanced, making the proposed method a practical and effective solution for modern community detection challenges.

6 Conclusion and Future Work

We introduced HP-MOCD, a high-performance evolutionary multi-objective algorithm for large-scale network community detection. Built upon the NSGA-II framework, HP-MOCD incorporates topology-aware crossover and mutation operators, coupled with a parallelized and efficient implementation that ensures practical scalability. In contrast to traditional single-objective heuristics and earlier MOEA-based methods, HP-MOCD

produces a Pareto front of diverse, high-quality partitions, enabling a principled exploration of trade-offs between structural cohesion and ground-truth alignment.

Extensive experiments on synthetic and real-world datasets show that HP-MOCD consistently outperforms prior evolutionary methods in both runtime and robustness, while delivering competitive accuracy, particularly in scenarios involving noisy, ambiguous, or large-scale networks. Even in cases where single-objective methods such as Louvain or Leiden yield higher scores in individual metrics, HP-MOCD maintains a more balanced performance profile across AMI, NMI, modularity, and F1-score, highlighting the benefits of its multi-objective formulation.

Future work can extend the algorithm to support overlapping communities and more general network types, such as weighted, directed, or temporal graphs. Another promising path involves improving robustness under high mixing regimes ($\mu \to 1$), potentially through hybrid strategies or advanced selection mechanisms. An additional direction includes investigating the resilience of communities under node or edge failures, as explored in prior work on failure-resilient communities (Wang and Liu 2018, 2019). We also envision extending our experimental evaluation by incorporating additional datasets and more baseline algorithms, such as those proposed by Ghoshal et al. (2019, 2021), to more thoroughly assess the strengths and limitations of HP-MOCD in diverse scenarios. Furthermore, accurate and scalable community detection can serve as a foundation for downstream applications such as diffusion source inference and diffusion containment in complex networks, which increasingly rely on community structure to enhance performance (Liu et al. 2023, 2024, 2025). Lastly, we plan to explore integration with NSGA-III (Deb and Jain 2014) and other recent MOEA frameworks to support higher-dimensional objectives, such as conductance and community balance, thereby expanding the scope of multi-objective community detection.

Acknowledgements

References

Azevedo, B.F., Rocha, A.M.A.C., Fernandes, F.P., Pacheco, M.F., Pereira, A.I.: Comparison between single and multi-objective clustering algorithms: Mathe case study. In: Optimization, Learning Algorithms and Applications, pp. 65–80. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-77426-3_5

Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment **2008**(10), 10008 (2008) https://doi.org/10.1088/1742-5468/2008/10/P10008

Bollacker, K.D., Lawrence, S., Giles, C.L.: Citeseer: An autonomous web agent for automatic retrieval and identification of interesting publications. In: Proceedings of the Second International Conference on Autonomous Agents, pp. 116–123 (1998)

Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.-U.: Complex networks: Structure and dynamics. Physics Reports **424**(4), 175–308 (2006) https://doi.org/10.1016/j.physrep.2005.10.009

Boguná, M., Papadopoulos, F., Krioukov, D.: Sustaining the internet with hyperbolic mapping. Nature communications **1**(1), 62 (2010)

Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation). Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-0-387-36797-2

Deb, K.: Multiobjective Optimization Using Evolutionary Algorithms. John Wiley & Sons, New York (2001). https://www.wiley.com/en-us/Multi+Objective+Optimization+using+Evolutionary+Algorithms-p-9780471873396

Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. IEEE Transactions on Evolutionary Computation **18**(4), 577–601 (2014) https://doi.org/10.1109/TEVC.2013.2281535

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002) https://doi.org/10.1109/4235.996017

Dabaghi-Zarandi, F., Afkhami, M.M., Ashoori, M.H.: Community detection method based on random walk and multi objective evolutionary algorithm in complex networks. Journal of Network and Computer Applications **234**, 104070 (2025)

Ehrgott, M.: Multicriteria Optimization, 2nd edn. Lecture Notes in Economics and Mathematical Systems, vol. 491. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/3-540-27659-9 . Second edition

Fortunato, S., Barthélemy, M.: Resolution limit in community detection. Proceedings of the National Academy of Sciences **104**(1), 36–41 (2007) https://doi.org/10.1073/pnas.0605965104

Fortunato, S., Hric, D.: Community detection in networks: A user guide. Physics Reports **659**, 1–44 (2016) https://doi.org/10.1016/j.physrep.2016.09.002

Flake, G.W., Lawrence, S., Giles, C.L., Coetzee, F.M.: Self-organization and identification of web communities. Computer (2002) https://doi.org/10.1109/2.989932

Fortunato, S.: Community detection in graphs. Physics Reports **486**(3), 75–174 (2010)

https://doi.org/10.1016/j.physrep.2009.11.002

Fu, J., Wang, Y.: An improved continuous-encoding-based multiobjective evolutionary algorithm for community detection in complex networks. IEEE Transactions on Artificial Intelligence **5**(11), 5815–5827 (2024)

Ghoshal, A.K., Das, N., Bhattacharjee, S., Chakraborty, G.: A fast parallel genetic algorithm based approach for community detection in large networks. In: COMSNETS, pp. 95–101 (2019)

Ghoshal, A.K., Das, N., Das, S.: Disjoint and overlapping community detection in small-world networks leveraging mean path length. IEEE Transactions on Computational Social Systems **9**(2), 406–418 (2021)

Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences **99**(12), 7821–7826 (2002) https://doi.org/10.1073/pnas.122653799

Handl, J., Knowles, J.: An evolutionary approach to multiobjective clustering. Evolutionary Computation, IEEE Transactions on **11**, 56–76 (2007) https://doi.org/10.1109/TEVC.2006.877146

Ivashkin, V., Chebotarev, P.: Do logarithmic proximity measures outperform plain ones in graph clustering? In: International Conference on Network Analysis, pp. 87–105 (2016). Springer

Lancichinetti, A., Fortunato, S.: Community detection algorithms: A comparative analysis. Phys. Rev. E **80** (2009) https://doi.org/10.1103/PhysRevE.80.056117

Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Phys. Rev. E **78**, 046110 (2008) https://doi.org/10.1103/PhysRevE.78.046110

Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: Densification and shrinking diameters. ACM transactions on Knowledge Discovery from Data **1**(1) (2007)

Lusseau, D., Schneider, K., Boisseau, O.J., Haase, P., Slooten, E., Dawson, S.M.: The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations: can geographic isolation explain this unique trait? Behavioral ecology and sociobiology **54**(4), 396–405 (2003)

Liu, B., Wang, D., Gao, J.: A multi-objective community detection algorithm with a learning-based strategy. International Journal of Computational Intelligence Systems **17**(1), 311 (2024)

Liu, Y., Wang, X., Wang, X., Wang, Z., Zhu, P.: Diffusion containment in complex networks through collective influence of connections. IEEE Transactions on Information

Forensics and Security (2023) https://doi.org/10.1109/TIFS.2023.3338423

Liu, Y., Wang, X., Wang, X., Wang, Z., Kurths, J.: Diffusion source inference for largescale complex networks based on network percolation. IEEE Transactions on Neural Networks and Learning Systems (2025) https://doi.org/10.1109/TNNLS.2023.3321767

Liu, Y., Wang, X., Wang, X., Wang, Z.: Observer-based source localization in complex networks: A minimum set approach. IEEE/ACM Transactions on Networking (2024) https://doi.org/10.1109/TNET.2024.3382546

Moreira, G., Paquete, L.: Guiding under uniformity measure in the decision space. In: 2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI), pp. 1–6 (2019). https://doi.org/10.1109/LA-CCI47412.2019.9037034

Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, ??? (2008). https://nlp.stanford.edu/IR-book/

Newman, M.E.J.: From the cover: Modularity and community structure in networks. Proceedings of the National Academy of Science **103**(23), 8577–8582 (2006) https://doi.org/10.1073/pnas.0601602103

Newman, M.E.: Modularity and community structure in networks. Proceedings of the national academy of sciences **103**(23), 8577–8582 (2006)

Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**, 026113 (2004) https://doi.org/10.1103/PhysRevE.69.026113

Newman, M.E., Girvan, M.: Finding and evaluating community structure in networks. Physical review E **69**(2), 026113 (2004)

Pizzuti, C.: A multi-objective genetic algorithm for community detection in networks. In: 2009 21st IEEE International Conference on Tools with Artificial Intelligence, pp. 379–386 (2009). https://doi.org/10.1109/ICTAI.2009.58

Pizzuti, C.: A multiobjective genetic algorithm to find communities in complex networks. Evolutionary Computation, IEEE Transactions on **16**, 418–430 (2012) https://doi.org/10.1109/TEVC.2011.2161090

Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E **76**, 036106 (2007) https://doi.org/10.1103/PhysRevE.76.036106

Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Physical Review EStatistical, Nonlinear, and Soft Matter Physics **76**(3), 036106 (2007)

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying

communities in networks. Proceedings of the National Academy of Sciences **101**(9), 2658–2663 (2004) https://doi.org/10.1073/pnas.0400054101

Rossetti, G., Milli, L., Cazabet, R.: CDLIB: a Python Library to Extract, Compare and Evaluate Communities from Complex Networks. Applied Network Science **4**(1), 52 (2019) https://doi.org/10.1007/s41109-019-0165-9

Ravasz, E., Somera, A.L., Mongru, D.A., Oltvai, Z.N., Barabási, A.-L.: Hierarchical organization of modularity in metabolic networks. Science **297**(5586), 1551–1555 (2002) https://doi.org/10.1126/science.1073374

Šubelj, L., Bajec, M.: Model of complex networks based on citation dynamics. In: International Conference on World Wide Web (2013)

Shaik, T., Ravi, V., Deb, K.: Evolutionary multi-objective optimization algorithm for community detection in complex social networks. SN Computer Science **2**(1), 1 (2021)

Shi, C., Zhong, C., Yan, Z., Cai, Y., Wu, B.: A multi-objective approach for community detection in complex network. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010). https://doi.org/10.1109/CEC.2010.5585987

Traag, V.A., Waltman, L., Eck, N.J.: From Louvain to Leiden: guaranteeing well-connected communities. Scientific Reports **9**(1), 5233 (2019) https://doi.org/10.1038/s41598-019-41695-z

Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 1073–1080 (2009)

Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. Journal of Machine Learning Research **11**, 2837–2854 (2010). http://jmlr.org/papers/v11/vinh10a.html

Wang, S., Liu, J.: Constructing robust community structure against edge-based attacks. IEEE Systems Journal **13**(1), 582–592 (2018)

Wang, S., Liu, J.: Community robustness and its enhancement in interdependent networks. Applied Soft Computing **77**, 665–677 (2019)

Wang, L., Zeng, Y., Li, Y., Liu, Z., Ma, J., Zhu, X.: Research on resolution limit of community detection in location-based social networks. In: 2019 International Conference on Networking and Network Applications (NaNA), pp. 90–95 (2019)

Yao, Y.Y.: Information-Theoretic Measures for Knowledge Discovery and Data Mining, pp. 115–136. Springer, Berlin, Heidelberg (2003). https://doi.org/10.1007/978-3-540-36212-8_6

Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. In: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics, pp. 1–8 (2012)

Yusupov, J., Palakonda, V., Mallipeddi, R., Veluvolu, K.C.: Multi-objective evolutionary algorithm based on ensemble of initializations for overlapping community detection. In: International Conference on Electronics, Information, and Communication (ICEIC), pp. 1–7 (2021)

Zachary, W.W.: An information flow model for conflict and fission in small groups. Journal of anthropological research **33**(4), 452–473 (1977)

Appendix A   Space Complexity Analysis

Let $|V|$ represent the number of nodes, $|E|$ the number of edges, N_p the population size, and $d_{\max} = \max_{v \in V} d(v)$ the maximum node degree. The graph G itself is stored once, requiring $O(|V| + |E|)$ space. Each individual in the population is a labeling of all $|V|$ nodes, thus one individual occupies $O(|V|)$ space. At any generation, HP-MOCD holds both the parent population P_t and the offspring population Q_t. This amounts to $2N_p$ individuals, consuming

$$O\big(2\,N_p \cdot |V|\big) \;=\; O\big(N_p\,|V|\big) \quad \text{space.}$$

Regarding algorithmic components:

- For *Crossover*, a hash-map is employed to count labels dynamically. This map is reused for each node, resulting in $O(1)$ auxiliary space per node, which does not accumulate across nodes.
- For *Mutation*, apart from storing the new individual (requiring $O(|V|)$), a frequency map of size at most the node's degree $(d(v))$ may be allocated when mutating a single node. Since this map is reused across nodes, the total auxiliary space for this operation is $O(d_{\max})$.

The non-dominated sorting and crowding distance calculations (as in NSGA-II) are performed on $2N_p$ individuals. The space required for these operations includes:

$$\underbrace{O\big(N_p\big)}_{\substack{\text{domination counts} \\ \text{and front indices}}} \;+\; \underbrace{O\big(N_p\big)}_{\substack{\text{crowding} \\ \text{distance array}}} \;=\; O\big(N_p\big).$$

A.1   Total Space Complexity

The total space complexity is the sum of these components:

$$
\begin{aligned}
&\; O\big(|V| + |E|\big) && \text{(graph storage)} \\
+\;& O\big(N_p\,|V|\big) && \text{(populations } P_t, Q_t) \\
+\;& O\big(d_{\max}\big) && \text{(mutation auxiliary space)} \\
+\;& O\big(N_p\big) && \text{(NSGA-II auxiliary space)} \\
=\;& O\big(N_p\,|V| + |V| + |E| + d_{\max} + N_p\big).
\end{aligned}
$$

In typical scenarios involving sparse graphs (where $|E| = O(|V|)$) and for realistic algorithm parameters where $N_p\,|V|$ is the overwhelmingly largest term (i.e., $N_p\,|V| \gg |V|, |E|, d_{\max}, N_p$), the dominant space complexity is

$$\boxed{O\big(N_p\,|V|\big).}$$

Appendix B   Role of Multi-Threaded Architecture in Algorithm Performance

The inherent parallel design of our algorithm necessitates an investigation into its scalability with an increasing number of logical threads and the identification of optimal threading levels. Understanding this relationship is paramount to harnessing the algorithm's full potential.

To quantify the impact of multi-threading, we benchmarked the algorithm on a high-performance system featuring an AMD Ryzen Threadripper 3960X processor (24 physical cores, 48 logical threads, @ 3.70GHz) and 128GB DDR4 RAM. The number of execution threads was systematically varied from 1 to 48. These tests utilized the same synthetic graph datasets with $n \in [10,000, 30,000, 50,000]$ nodes as detailed in our scalability analysis (Appendix C). Each configuration was executed five times, a sufficient number given the consistently low standard deviation observed in execution times.

Figure B1 illustrates the performance gains. Initially, increasing the number of threads yields substantial reductions in runtime. However, a performance plateau is reached beyond a certain threshold (approximately 8 threads for 10,000 nodes, 15 for 30,000 nodes, and 20 for 50,000 nodes). Beyond these points, adding more threads provides marginal benefits. This is likely attributable to the increased overhead associated with managing a large number of concurrent threads.

| (a) 10,000 nodes | (b) 30,000 nodes | (c) 50,000 nodes |
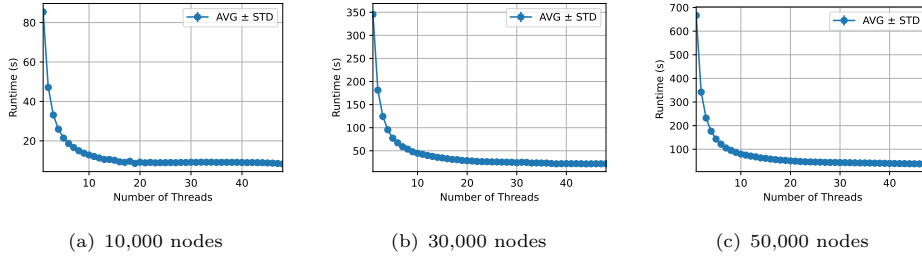
Fig. B1 Average runtime ($mean \pm \sigma$) as a function of the number of logical threads for graphs with 10,000 (a), 30,000 (b), and 50,000 (c) vertices. The results underscore the significant speedup achieved through multi-threading compared to single-threaded execution.

The most compelling evidence for the importance of the multi-threaded architecture lies in the stark contrast with single-threaded execution. As shown in Figure B1, a linear (single-thread) approach is dramatically slower across all scenarios. Specifically, when compared to utilizing the maximum available logical threads, single-threaded execution was **10.27** times slower for 10,000 nodes (Figure B1(a)), **16.45** for 30,000 nodes (Figure B1(b)), and **17.83** times for 50,000 nodes (Figure B1(c)). These figures unequivocally demonstrate that the algorithm's parallel processing capability is not merely an optimization but a fundamental requirement for achieving practical execution

times, rendering it viable and effective for real-world applications. Without its multi-threaded design, the algorithm would be impractically slow for datasets of meaningful scale.

Appendix C   Selection of Genetic Algorithm Parameters

The performance of the NSGA-II algorithm depends critically on two GA parameters: the population size P and the number of generations G. These parameters control the total search effort and must be tuned to trade off clustering qualitymeasured here via AMI and NMIagainst computational cost.

C.1   Aggregate Quality Metric

To summarize clustering quality in a single scalar, we compute the harmonic mean of AMI and NMI:
$$H = \frac{2 \left(\text{AMI} \times \text{NMI}\right)}{\text{AMI} + \text{NMI}},$$
which balances the two scores and more strongly penalizes configurations where either AMI or NMI is low. While AMI and NMI are widely used, the methodology remains compatible with other external or internal evaluation metrics, depending on application context.

C.2   GA Effort Metric

As a proxy for total computational investment, we define:

$$E = P \times G,$$

i.e. the total number of candidate evaluations across all generations. This single metric captures both population diversity (P) and evolutionary depth (G).

A systematic grid search was performed over various (P, G) pairs for graph sizes $n \in [10\,000,\ 20\,000,\ 30\,000; 40\,000]$. For each configuration, we recorded H and the execution time.

Figure C2(a) and Figure C2(b) display, respectively, the harmonic mean quality and runtime as functions of E, with separate curves for each n.

From these figures, we observe that clustering quality H increases rapidly with E up to approximately 10^4, after which gains become marginal. Meanwhile, execution time grows roughly linearly with E for all tested graph sizes. This behavior highlights a clear trade-off regime where further computational effort yields diminishing returns.

C.3   Chosen Configuration

To balance nearpeak quality with reasonable runtime, we select:
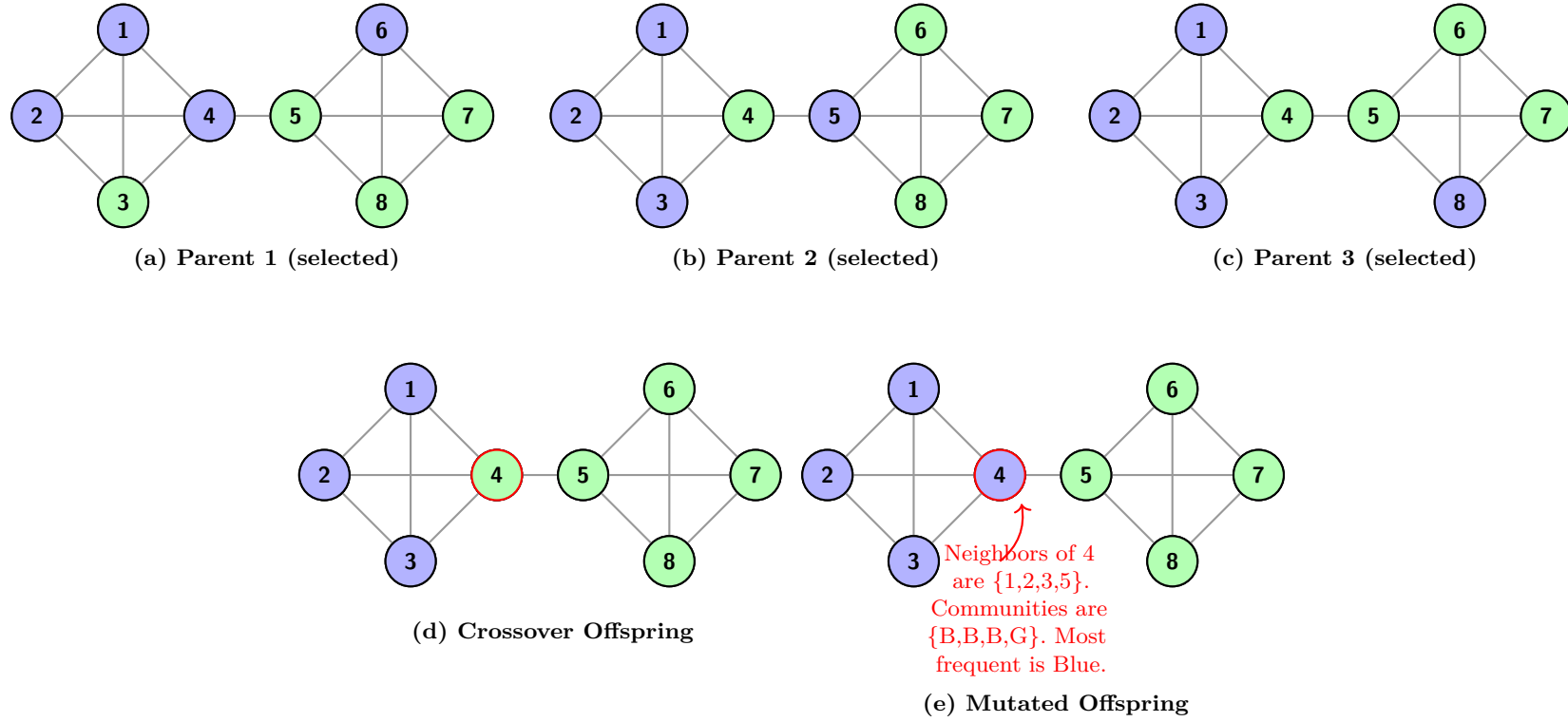
$$P = 100, \quad G = 100, \quad (E = 10^4)$$

(a) H versus E

(b) Execution Time versus E

n = 10,000   n = 20,000   n = 30,000   n = 40,000
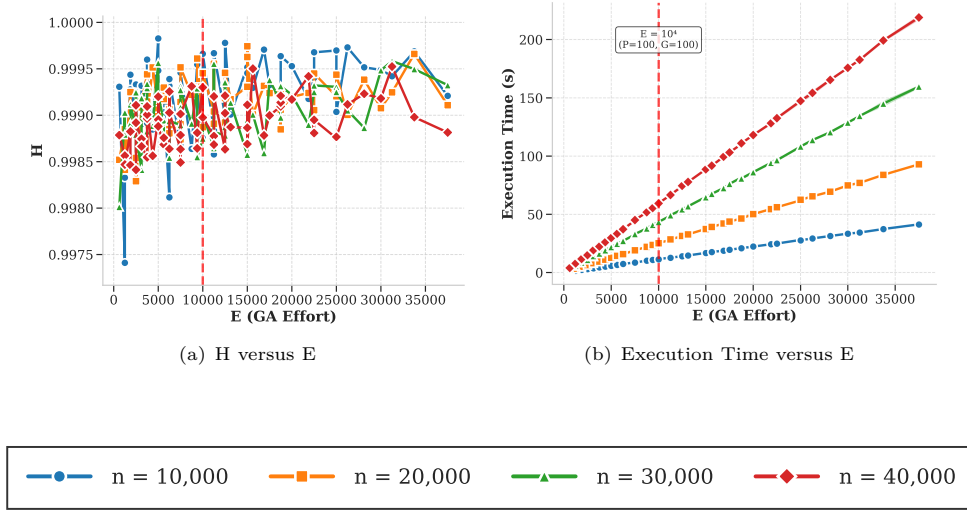
Fig. C2 Harmonic mean of AMI and NMI versus GA effort E, and execution time versus GA effort E, for different graph sizes.

as this setting reaches near-optimal H before the saturation point, while keeping execution time moderate and scalable.

Appendix D   Crossover and Mutation Example

Figure D3 provides a visual example of the crossover and mutation operators described in Section 4.3. These examples help illustrate how the genetic variation mechanisms alter individuals during the evolutionary process.

43

(a) Parent 1 (selected) **(b) Parent 2 (selected)** **(c) Parent 3 (selected)**

(d) Crossover Offspring

Neighbors of 4 are {1,2,3,5}. Communities are {B,B,B,G}. Most frequent is Blue.

(e) Mutated Offspring

Fig. D3 An illustrative example of the Crossover and Mutation operators. (a), (b) and (c) show three parent solutions selected. (d) The offspring is created by a majority vote for each node's community assignment, as determined by Equation 10. The result is a novel partition, different from all parents. (e) The mutation operator inspects the neighborhood of node 4, finds that blue is the most frequent community, and flips its color to perform a final correction, recovering the optimal partition.