

Article

Gain and Pain in Graph Partitioning: Finding Accurate Communities in Complex Networks [†]

Arman Ferdowsi ^{1,*}  and Maryam Dehghan Chenary ² 
¹ ECS Group, TU Wien, 1040 Vienna, Austria

² Department of Business Decisions and Analytics, University of Vienna, 1090 Vienna, Austria; maryam.dehghan.chenary@univie.ac.at

* Correspondence: aferdowsi@ecs.tuwien.ac.at

[†] This paper is an extended version of our paper published in the proceedings of the 18th Conference on Computer Science and Intelligence Systems (FedCSIS), Warsaw, Poland, 17–20 September 2023.

Abstract: This paper presents an approach to community detection in complex networks by simultaneously incorporating a connectivity-based metric and Max-Min Modularity. By leveraging the connectivity-based metric and employing a heuristic algorithm, we develop a novel complementary graph for the Max-Min Modularity that enhances its effectiveness. We formulate community detection as an integer programming problem of an equivalent yet more compact counterpart model of the revised Max-Min Modularity maximization problem. Using a row generation technique alongside the heuristic approach, we then provide a hybrid procedure for near-optimally solving the model and discovering high-quality communities. Through a series of experiments, we demonstrate the success of our algorithm, showcasing its efficiency in detecting communities, particularly in extensive networks.

Keywords: community detection; graph partitioning; complex network analysis; mathematical programming; approximation algorithm



Citation: Ferdowsi, A.; Dehghan Chenary, M. Gain and Pain in Graph Partitioning: Finding Accurate Communities in Complex Networks. *Algorithms* **2024**, *17*, 226. <https://doi.org/10.3390/a17060226>

Academic Editors: Marek Bolanowski, Andrzej Białas, Janusz Furtak and Andrzej Paszkiewicz

Received: 19 April 2024

Revised: 20 May 2024

Accepted: 21 May 2024

Published: 23 May 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction and Literature Review

In the intricate landscape of complex networks, the exploration of inherent structures, specifically the identification of cohesive subsets known as *communities*—has emerged as a foundational pursuit within network science and graph theory. Understanding communities and utilizing community detection in complex networks is pivotal for revealing real-world systems' intricate structures and interconnections. This process not only aids in deciphering the underlying dynamics of diverse networks, ranging from distributed communication networks to biological networks, but also empowers us to recognize outliers, predict the system's failures and future behaviors with greater precision, and optimize performance. These communities, comprising highly interactive vertices, underpin diverse systems and offer insights into complex relationships, thereby transcending domains such as social, biological, and cosmological network analyses; distributed computing; signal processing; and data mining [1–7]. The quest to unveil and understand these communities has given rise to a spectrum of methodologies, each addressing the challenges posed by networks of varying complexities.

At the heart of community detection lies the concept of partitioning a network, represented as a graph $G = (V, E)$, into distinct subsets or communities $C = \{C_1, C_2, \dots, C_k\}$, where vertices within each community are as densely connected as possible. Various forms of the problem arise based on factors such as whether the network is weighted or unweighted, directed or undirected, whether the desired number of communities is predetermined, and whether communities exhibit overlap or complete separation [8–13].

Crucially notable is that while there are non-optimization-based algorithms like the *Label Propagation* algorithm [14], the majority of community detection methods adhere to an optimization strategy. This involves first defining or selecting a specific metric, referred

to as a *quality function*, to assess the excellence of communities; then, the devising of an optimization algorithm to detect communities by optimizing the chosen quality function. Global optimization procedures, such as mathematical programming approaches [15,16], provide accurate results. However, they face computational challenges when dealing with large networks [17]. Heuristic techniques, on the other hand—like bottom-up (merging communities), top-down (splitting communities), and local search (migrating nodes)—present viable solutions, especially for extensive networks [18], but they come at the cost of losing accuracy. Although various community discovery approaches have surfaced in recent years [19–21], due to space limitations, our discussion concentrates on methods pertinent to our study.

Regarding optimization techniques, a wide variety of quality measures have been proposed, each offering a unique perspective to assess the goodness and accuracy of partitioning. Chakraborty et al. [22] categorize community quality functions into four broad types: *internal-connectivity-based* metrics (IC), *external-connectivity-based* metrics (EC), *mixed-internal-external* metrics (MIE), and metrics based on *network topology*. Table 1 outlines some well-known connectivity-based quality functions. For example, the *Average degree* function evaluates a community's quality based on both the number of edges with nodes inside the community and the number of nodes within the community.

Table 1. Several connectivity-based quality functions for measuring the quality of a community C . $|E_C^{out}|$, $|E_C^{in}|$, $|C|$, and $deg(i)$, respectively, represent the number of external connections of C , number of internal connections of C , the number of vertices belonging to C , and the degree of node i .

	Function	Definition
IC	Internal density [23]	$\frac{ E_C^{in} }{ C (C -1)/2}$
	Edge inside [23]	$ E_C^{in} $
	Average degree [23]	$\frac{2 E_C^{in} }{ C }$
EC	Expansion [23]	$ E_C^{out} $
	Cut Ratio [24]	$\frac{ E_C^{out} }{ C (V - C)}$
	Conductance [25]	$\frac{ E_C^{out} }{2 E_C^{in} + E_C^{out} }$
MIE	Normalized Cut [25]	$\frac{ E_C^{out} }{2 E_C^{in} + E_C^{out} } + \frac{ E_C^{out} }{2(E - E_C^{in}) + E_C^{out} }$
	Out-Degree Fraction [26]	$\max_{i \in C} \frac{ (i, j) \in E : j \notin C }{deg(i)}$
	Flake-ODF [26]	$\frac{ i \in C, (i, j) \in E : j \in C < deg(i)/2 }{ C }$

While connectivity-based metrics have achieved widespread recognition [22], their exclusive focus on link structures, such as the number of edges and degrees, limits their ability to consider other significant features of the network, such as potential relationships between nodes. Therefore, topological metrics, among which Newman's *Modularity* [27] stands out as a widely used and recognized one, have garnered remarkable attention. Modularity assesses the quality of a partition by measuring the disparity between the observed number of edges within communities and the expected number based on a null model. Nevertheless, the traditional Modularity metric has faced criticism for its limitation in exclusively considering existing edges, disregarding the possibility of connections between disconnected nodes within the same community. To overcome this drawback, an extension known as the *Max-Min Modularity* [9] introduces a complementary graph associated with the main network to capture relationships between disconnected nodes. The essential idea of Max-Min Modularity revolves around maximizing the Modularity value within the given network while concurrently minimizing this value across a complementary graph

consisting of edges between unrelated nodes. This adjustment penalizes the Modularity measure when unrelated nodes coexist within a community. This is while, despite its improved accuracy, current Max-Min Modularity approaches—like the one presented by Chen et al. [9]—depend on user-defined relation criteria that introduce potential faultiness into the community detection process.

The naive dependency on an oracle for determining the complementary graph, as well as the not-very-successful heuristic hierarchy algorithm used in [9] for maximizing the Max-Min Modularity measure, directed Ferdowsi et al. in [15,16] to provide a more structured way to address the deficiencies. Their focus was on methodically structuring an effective complementary graph that accurately reflects the absence of connections between nodes. They enhanced this graph by applying the optimal solution to the *linear relaxation* formulation of the Modularity maximization problem. Using this systematically obtained complementary graph, they devised an *integer programming* representation for the Max-Min Modularity maximization problem and solved it using various methods. In [16], they applied the rational solution of the model's linear relaxation version to a local search rounding technique to obtain integer solutions and identified communities. In contrast, Ref. [15] theoretically developed and solved an equivalent yet sparse integer model, determining the exact solution to the problem. Consequently, whereas [16] succeeded in deriving community structure in a prompt manner due to the heuristic fashion employed, Ref. [15] achieved more precise results by obtaining optimal solutions to the integer model.

Both approaches yielded promising results, particularly those presented in [15], as they optimally solve a corresponding integer model. Nevertheless, several potential downsides exist. A primary drawback, not exclusive to these two schemes, is that almost all community detection methodologies take only one quality function into consideration. As a matter of fact, regardless of the optimization technique, each quality function tends to emphasize some distinct aspects of the network's structure. For instance, as mentioned, whereas a connectivity-based measure is geared more toward determining communities with densely connected nodes, topological approaches are capable of also concentrating on grouping related nodes. Another source of inefficiency lies in time complexity, which is especially evident when comparing two optimization methods—e.g., a heuristic and an exact one—that both apply to the same quality function. Although the exact procedure outperforms the heuristic with regard to the quality of communities, it loses the ground to a heuristic approach in terms of time execution. Thus, enhancing time complexity remains a persistent challenge when employing an exact method.

1.1. Main Contributions

- (1) We synergistically integrate two highly effective community quality functions: (i) the connectivity-based metric introduced in [21] and (ii) the Max-Min Modularity [16]. Specifically, we employ the connectivity-based metric and the heuristic algorithm proposed in [21] to establish a novel complementary graph associated with the Max-Min Modularity and to obtain a high-quality initial solution to the Max-Min Modularity maximization problem.
- (2) We develop our community detection problem as an integer programming formulation of a revised Max-Min Modularity maximization problem (extending and improving the approach previously described in [15]).
- (3) To mitigate time complexity, by deriving analytical insights, we demonstrate the time efficiency of the employed heuristic method, thereby validating the efficiency of obtaining the refined complementary graph. Besides, we present a more compact yet equivalent model to the original integer formulation and demonstrate through theoretical arguments that the proposed sparse model yields the same optimal solution but with significantly fewer variables and constraints.
- (4) We design a bound for controlling the degree to which the algorithm is supposed to delve into optimally solving the model and when it should obtain assistance from the heuristic method. More specifically, taking the promising communities generated by

- the heuristic method as the initial solution to the sparse counterpart of the proposed Max-Min Modularity maximization problem's integer model, we devise a procedure for balancing between heuristic and exact optimization based on the size of the given network. This facilitates the detection of communities even in extensive networks.
- (5) Through various experiments, we showcase the success of our proposed algorithm.

1.2. Paper Organization

In Section 2, we review the heuristic approach introduced in [21] and propose some theoretical insights regarding its efficiency. We then recap the traditional Modularity and Max-Min Modularity methodologies. Afterward, in Section 2.3, we delve into presenting our approach. Finally, in Section 3, we bring forward the experimental results. Some conclusions are provided in Section 4.

2. Methodology

This section aims to amalgamate two optimization approaches: (1) a heuristic method for minimizing a connectivity-based quality metric; (2) an exact approach for maximizing a topology-based quality measure. The objective is to formulate a hybrid community detection algorithm capable of identifying accurate communities within a reasonable timeframe.

2.1. Heuristic but Fast

This section reviews the two-stage heuristic optimization-based community detection algorithm proposed in [21], which involves optimizing a connectivity-based quality metric. We first revisit the requirements and definition of the suggested quality metric, then outline the optimization algorithm. Afterward, we develop some theoretical arguments contributing to a more comprehensive evaluation of the algorithm's validity across varying network sizes.

What is worth noting about this fast yet accurate heuristic approach is our aim to (1) propose a systematic way to improve the complementary graph of the Max-Min Modularity and (2) obtain a high-quality initial solution for the exact optimization procedure that will be explained in Section 2.2.

Let the *degree* of a node and its set of neighbors be defined naturally. Additionally, the *inner* and *outer edges* of a community C , denoted as E_C^{in} and E_C^{out} , respectively, represent the edges with both endpoints and one endpoint within C . We proceed to define the *influential nodes*, which can be envisioned as vertices endowed with authority and the capacity to control and affect other nodes, particularly in our context, shaping communities. As elucidated in [21], nodes with high degrees that maintain an appropriate distance from each other emerge as suitable candidates for such roles. The intuition comes from the fact that the degree of nodes can substantially reflect the importance of vertices within the network. A high-degree vertex influences its surroundings more than a low-degree vertex. This is because it can pass messages to multiple nearby vertices and impose various features and information on them. For instance, a high-degree vertex in a communication network can significantly impact the spread of data and messages, affecting the behavior of its neighbors and the overall dynamics of the network. Think of the effect of a high-degree byzantine faulty process in a distributed computing network. Definition 1 gives a more formal characterization of the influential nodes.

Definition 1 (Influential node). Let $\alpha \in \{1, \dots, \text{diameter}(G)\}$ be an integer and define $P_\alpha \subseteq V$ as a sequence of nodes arranged in descending order according to their degrees, provided they are all at least α steps away from each other. P_α is referred to as the influential nodes of G with respect to α .

Note that the influential node of a community is not necessarily unique. In such a case, the influential node will be randomly picked among the candidates.

Definition 2 (Neighboring communities). Two communities C and C' are said to be neighbors if there exists an edge $(i, j) \in E$ connecting $i \in C$ to $j \in C'$. In this case, we refer to both i and j as boundary vertices of C and C' , respectively. Let $BV(C)$ denote the set of all boundary vertices of C .

Definition 3 (Distance to a community). The distance between a vertex $j \notin C$ and the community C is determined as the length of the shortest path from j to the influential node of C .

Definition 4 (Radius). The radius of a community C , denoted by $r(C)$, is the maximum distance from its influential node to other vertices within the community.

The influence a particular node exerts on others is evidently correlated with the distance between them. Valuable insights into this relationship can be gleaned from studies such as [16,28]. For instance, note the more significant impact that a high-degree node in a communication graph has on vertices connecting to it directly (distance = 1) than on those that are connected to it through several intermediaries (distance > 1). Consequently, it is natural to assume that a community with a small radius is likely to be more interactive and homogeneous. Subsequently, the authors of [21] proposed the following measure to compute the quality of a community C :

$$q(C) = \frac{|E_C^{out}| + r(C)}{|E_C^{in}|}. \quad (1)$$

Furthermore, given a set of communities $\mathbf{C} = \{C_1, \dots, C_k\}$, the quality of the partition $\mathbf{C}$ is defined as follows:

$$q_G^{\mathbf{C}} = \sum_{C \in \mathbf{C}} q(C). \quad (2)$$

Note that the quality of the partition $\mathbf{C}$ increases as $q_G^{\mathbf{C}}$ decreases. More specifically, the proposed community quality measure determines the quality of a community C by minimizing the sum of the following two terms: $\frac{|E_C^{out}|}{|E_C^{in}|}$ and $\frac{r(C)}{|E_C^{in}|}$. As minimizing the number of outer edges while simultaneously maximizing the number of inner edges naturally leads to enhancing the quality of community structures, reducing the radius yields a similar outcome. Recall the significant role of the vertex's degree in controlling and stimulating neighboring nodes due to high-degree vertices' capability to impose distinct attributes on surrounding nodes [29]. Specifically, many (complex) networks exhibit a small number of influential high-degree vertices, such as social media influencers, pivotal commodities in co-purchasing networks, or a high-degree processor in a distributed computing network. These high-degree vertices are typically surrounded by numerous lower-degree vertices [30,31]. In such networks, each vertex is either directly connected to or only a few links away from a high-degree node. With this in mind, having communities with small radii breaks the task into finding influential nodes that are at a proper distance from each other, as it turned out in [21] that an appropriate distance between high-degree nodes results in the formation of well-isolated communities.

For this, the proposed optimization algorithm works in such a way that, with the general goal of minimizing the quality measures (1) and (2), it first establishes a set of promising initial communities (Algorithm 1) that then get purified with a local search manner (Algorithm 2). Simply put, the first stage seeks the best α value leading to well-separated communities formed surrounding the set of influential vertices P_α . Equivalently, the goal of this phase restricts the objective to finding authoritative nodes that are at the right distance from each other, letting them freely absorb other nodes into their territory.

Algorithm 1: Creating Initial Communities (CIC)**Input:** Graph G .**Output:** Set of initial communities C and the corresponding set of influential nodes.

```

1 for  $\alpha = 2$  to  $\text{diameter}(G)$  do
2   Establish  $P_\alpha = \{p_1, p_2, \dots, p_m\}$ ; // based on Definition 1.
3    $C_\alpha = \{C_1 = \{p_1\}, \dots, C_{|P_\alpha|} = \{p_m\}\}$ ;
4   while  $\exists v \notin \bigcup_{i=1}^{|P_\alpha|} C_i$  do
5     Assign  $v \notin C_\alpha$  to its closest community  $C_i \in C_\alpha$ ;
6   Record  $C_\alpha$  and the corresponding  $P_\alpha$  and  $q_G^{C_\alpha}$ ;
7 Return a set of community  $C$  and the corresponding set of influential nodes  $P$  associated
    with the minimum obtained quality value  $q_G^C$ ;

```

The second phase, afterward, updates initial communities using a local search technique that iteratively selects a community, favoring worse quality values. It then greedily improves communities with one of three functions: *Merge*, *Split*, and *Swap*. While the *Merge* function assesses merging a community C into neighbor communities for quality improvement, the *Split* function checks if applying the CIC stage to the induced sub-graph G' formed by C can divide C into separate communities that result in a better quality value. The *Swap* function, on the other hand, optimizes node swaps between C and its neighbors. The procedure updates communities and influential vertices after each iteration, and it continues until either no further local search improvements are possible or a certain number of iterations have been performed.

This heuristic two-stage community detection algorithm stands out as a powerful approach to identifying well-separated communities within complex networks concerning minimizing the introduced quality measure. Evidently, depending on the quality of initial communities obtained in the first step, one can effectively reduce the need for reforms in the revising stage. This is particularly significant when time constraints are critical, resulting in fewer executions of the revising stage. This is while, according to [21], the first stage indeed returns high-quality communities, enabling us to confidently decrease the number of iterations in the second phase or even consider skipping it. In the theoretical explorations below, we state some mathematical foundations of this algorithm, aiming to unveil essential insights into its efficiency and solution quality. Particularly, since the algorithm's first phase aims to establish initial communities by determining the best set of influential nodes P_α , to maintain efficiency, $|P_\alpha|$ should be manageable.

Lemma 1. Consider a graph $G = (V, E)$ and let α be an integer such that $2 \leq \alpha \leq \text{diameter}(G)$. Let P_α be the set of corresponding influential nodes, meaning the set of nodes arranged in descending order according to their degrees, provided they are all at least α steps away from each other. Then, the time complexity of building P_α is of $O(n(n + m))$.

Proof. Let n be the number of nodes in G . To analyze the time complexity of building P_α , we need to consider the steps involved in constructing this set:

1. Sorting nodes by degree: sorting n nodes based on their degrees takes $O(n \log(n))$ time.
2. Constructing P_α : we iterate through the n nodes. For each node v , we need to check its distance to all nodes in P_α . In other words, for each node v , we should verify that it is at least α steps away from all nodes in P_α . To do this, we can use *Breadth-First Search* (BFS) starting from v . BFS runs in $O(n + m)$ time, where m is the number of edges in Graph G .

Thus, the total time complexity for constructing P_α is dominated by the distance verification step, giving us $O(n(n + m))$. $\square$

Algorithm 2: Updating Communities (UC)

Input: Graph G , set of initial communities $\mathbf{C}$, set of influential vertices P , integer k .
Output: Set of revised communities $\mathbf{C}$.

```

1  $q \leftarrow q_G^{\mathbf{C}}$ ;
2  $q_{temp} \leftarrow 0$ ;
3  $Iteration \leftarrow 1$ ;
4 while  $q \leq (1 - \epsilon)q_{temp} \wedge Iteration \leq k$  do
5    $q_{temp} \leftarrow q$ ;
6   select a community  $C \in \mathbf{C}$  with probability  $\frac{q(C)}{q_G^{\mathbf{C}}}$ ;
7    $(\mathbf{C}, P) \leftarrow \text{BestMovement}(C, \mathbf{C}, P)$ ;
8    $q \leftarrow q_G^{\mathbf{C}}$ ;
9    $Iteration++$ ;
10 Return  $(\mathbf{C})$ ;

```

Functions declaration:

```

13 BestMovement  $(C, \mathbf{C}, P)$ 
14    $(\mathbf{C}_{temp\_M}, P_{temp\_M}) \leftarrow \text{Merge}(C, \mathbf{C}, P)$ ;
15    $(\mathbf{C}_{temp\_S}, P_{temp\_S}) \leftarrow \text{Split}(C, \mathbf{C}, P)$ ;
16    $(\mathbf{C}_{temp\_Sw}, P_{temp\_Sw}) \leftarrow \text{Swap}(C, \mathbf{C}, P)$ ;
17   Among the returned partitions, retrieve the best set of communities  $\mathbf{C}$  corresponding to
   the minimum obtained quality value alongside its set of influential vertices  $P$ ;
18   Return  $(\mathbf{C}, P)$ ;
19 Merge  $(C, \mathbf{C}, P)$ 
20   for every  $C' \in \mathbf{C} - C$  such that  $C'$  is a neighbor community for  $C$  do
21     merge  $C$  and  $C'$ ;
22     update and remember  $\mathbf{C}$  and also the set of influential vertices;
23   retrieve the set of communities  $\mathbf{C}$  and its set of influential vertices  $P$ , corresponding to
   the minimum obtained quality value;
24   Return  $(\mathbf{C}, P)$ ;
25 Split  $(C, \mathbf{C}, P)$ 
26    $IG \leftarrow$  induced sub-graph of  $G$  formed by the subset  $C$  of vertices;
27    $(C', P') \leftarrow \text{CIC}(IG)$ ;
28   if  $|C'| > 1 \wedge q_{G'}^{\mathbf{C}'} \leq q(C)$  then
29      $\mathbf{C} \leftarrow (\mathbf{C} - C) \cup C'$ ;
30     update  $P$ ;
31   Return  $(\mathbf{C}, P)$ ;
32 Swap  $(C, \mathbf{C}, P)$ 
33   for every community  $C' \in \mathbf{C}$  that is neighbor of  $C$  do
34     for every pair of nodes  $(i, j) \in BV(C, C')$  do
35       swap the communities of  $i$  and  $j$  if it leads to a decrease in the quality value;
36   Return  $(\mathbf{C}, P)$ ;

```

* //small constant ϵ guarantees that running time for this stage remains polynomial [32,33].

Theorem 1 (Time efficiency of the first phase of the algorithm). *For a graph $G = (V, E)$ with n nodes and m edges, the first phase of the algorithm, which involves the identification of α -far nodes and the creation of initial communities, operates in $O(n^2(n + m))$ time, where k is a constant.*

Proof. The first phase involves constructing P_α and creating initial communities from α -far nodes. As shown in Lemma 1, P_α can be determined in $O(n(n + n))$ time. Hence, as the assignment of the remaining nodes to communities can take place in at most $O(n)$ time, the construction and initialization steps require $O(n^2(n + m))$ time. $\square$

Having the polynomial time complexity of the first phase of the algorithm proved, one can deduce the similar complexity for the whole algorithm.

Theorem 2 (Time efficiency of the whole algorithm). *For a graph $G = (V, E)$ with n nodes, the whole heuristic algorithm consisting of establishing initial communities and refining them operates in $O(n^4(n + m))$ time.*

Proof. Assume that the second phase iterates for p times. Then, the worst case complexity of this stage would be of $O(3pn^2)$, since all the three revising operations—Merge, Split, and Swap—in this stage occur in $O(n^2)$. Therefore, taking the time complexity of the first phase into account, we can conclude that the whole approach functions in $O(n^4(n + m))$. $\square$

Theorem 3 (An approximation community detection algorithm). *For a graph $G = (V, E)$, the heuristic algorithm provides an approximation algorithm for community detection.*

Proof. Note that a sufficiently small value of ϵ makes the second phase pass through all combinations of possible partitioning, pushing it to ultimately visit the optimal set of communities C_{OPT} with the quality value of $q^* = q_G^{COPT}$. Suppose that it happens after a finite number of iterations l . Moreover, due to the improvement factor $1 - \epsilon$, it is evident that after r iterations of this phase, the quality value will be less than or equal to $(1 - \epsilon)^{r-1}q_{init}$, where q_{init} is the quality value of the set of initial communities achieved by the first phase. Now, point out that in order for the two-stage algorithm to be a β -approximation algorithm, one needs to find a constant β such that βq^* becomes less than or equal to the quality value of the partitioning of the algorithm. Assuming that the algorithm runs for a given k_1 times results that $\beta q^* = \beta(1 - \epsilon)^{l-1}q_{init} \leq (1 - \epsilon)^{k_1-1}q_{init}$, which leads to $\beta < (1 - \epsilon)^{k_1-l}$. $\square$

We conclude this section by highlighting the leverage of the first phase not only to yield promising communities that stand alone as reliable results but also to achieve such outcomes in a sufficiently rapid manner (i.e., in $O(n^2(n + m))$). In addition, joining it to the practical second phase for improving the communities provides an approximation algorithm guaranteeing a final set of accurate communities. What is crucial to mention at the end is our choice in determining the extent to which the algorithm delves into the second phase. This pivotal decision allows for a reasonable equilibrium, finely attuned to the structure and magnitude of the network. Consequently, scaling up the network size offers a seamless avenue to curtail the number of iterations required for the second phase. As will be illustrated in Section 3, this strategic approach significantly optimizes computational efficiency.

2.2. Toward Exactness

Inevitably, the primary objective of any community detection algorithm is to identify communities as accurately as possible within a reasonable timeframe. For the former, it seems necessary to shift the focus from heuristic methods to exact optimization techniques, even though it would fight against the time complexity. On the other hand, utilizing a mix of quality metrics could also lead to accuracy improvement, as each quality metric can be tailored to suit specific graph structures within a defined range.

Accordingly, in this section, to enhance the quality of communities, we first concentrate on *Max-Min Modularity*, which is recognized as one of the most applicable topology-based metrics. Using this metric, we then express the Max-Min Modularity maximization as a mathematical programming problem. Solving this formulation allows us to optimize the measure and, consequently, detect highly accurate communities. For this, let us briefly explain the foundational principles of Modularity. This framework serves as the basis for the introduction of various Max-Min Modularity variants, which we will also discuss subsequently.

Modularity, as introduced by Newman [27], is widely acknowledged as a measure in network analysis. For a network G with n vertices and m edges, the Modularity value (Q) of a given partitioning $\mathbf{C}$ is mathematically expressed as follows:

$$Q(\mathbf{C}) = \frac{1}{2m} \sum_{i,j \in V} [a_{ij} - \frac{d_i d_j}{2m}] \sigma(i, j). \quad (3)$$

Here, $A = (a_{ij})$ is the adjacency matrix of G , where a_{ij} is set to one if an edge exists between node i and node j , and zero otherwise. d_i represents the degree of node i , defined as the sum of all entries in the i -th row of the adjacency matrix. Additionally, $\sigma(i, j)$ is one if nodes i and j are in the same community and zero otherwise. In simpler terms, Modularity quantifies the number of edges within a community minus the expected number of such edges. Communities with higher Modularity values are considered to be of better quality. Therefore, maximizing Modularity aids in identifying high-quality communities within a network.

However, despite the widespread use of Modularity, it has known limitations (refer to [22,34] for details). Notably, Modularity only accounts for existing edges in the network, evaluating the goodness of a community solely based on its fit with observed edges. It overlooks disconnected nodes (absent edges) within the same community, which is a drawback since node disconnection does not necessarily imply an absence of underlying relations between them. To address this limitation, an extension of Modularity called *Max-Min Modularity* [9] has been developed, aiming to enhance accuracy by penalizing Modularity when disconnected nodes are present in the same community. The core concept behind Max-Min Modularity involves maximizing the Modularity value within the given network G while simultaneously minimizing this value across a complementary graph G' . The latter has an identical set of nodes as G , and the presence of an edge between nodes i and j in G' indicates that, firstly, i and j are not connected in G , and secondly, these nodes are relatively unrelated. The rationale behind Max-Min Modularity is quite intuitive: by refining communities to reduce the occurrence of unrelated nodes, nodes not only disconnected but also unrelated within the same communities, the quality of the partitioning is significantly enhanced.

With this in mind, the Max-Min Modularity (QMM) of a given partition $\mathbf{C}$ of V can be defined as follows:

$$Q_{MM}(\mathbf{C}) = \sum_{i,j \in V} [\frac{1}{2m} (a_{ij} - \frac{d_i d_j}{2m}) - \frac{1}{2m'} (a'_{ij} - \frac{d'_i d'_j}{2m'})] \sigma(i, j), \quad (4)$$

where $A' = (a'_{ij})$ is the adjacency matrix of G' and d'_i is the degree of node i in G' . Besides, m' represents the number of edges in G' .

Particularly relevant in this regard is that despite different approaches for optimizing the Max-Min Modularity metric, a pivotal concern lies in defining an appropriate complementary graph G' capable of accurately representing the absence of relations between nodes. For instance, [9] naively relied on an expert to determine such a graph. In a more systematic fashion, Refs. [15,16] managed to obtain the complementary graph by employing the optimal solution to the linear relaxation formulation of the Modularity maximization problem. Utilizing the acquired complementary graph, the authors of both [15,16] formulated an integer representation of the Max-Min Modularity maximization problem and approached its solution through distinct methods. In the case of [16], the rational solution derived from the linear relaxation of the problem was input into a local search rounding technique to determine integer solutions and extract communities. On the other hand, [15] theoretically formulated an equivalent yet sparse integer model to the primary model, facilitating the direct solution of the integer model. While the adoption of the linear programming (LP) model and a heuristic rounding approach by [16] expedited the community identification

process, [15] achieved notably more accurate results by optimizing the solution of the integer model.

Remarkably, despite the promising outcomes from both methodologies, we identified potential for enhancement, particularly in terms of simultaneously reducing execution time and refining community quality. Our primary avenue for such improvement stems from a modified formulation of the Max-Min Modularity maximization problem, which we introduce in the subsequent subsection.

2.3. The Proposed Hybrid Approach

In this section, we bring forward a successful approach for obtaining accurate communities by optimally optimizing a revised version of the well-known topological community quality metric, *Max-Min Modularity*. What we intend to do is use the already introduced heuristic method to propose a new complementary graph (Definition 5) and then apply our theoretical findings in [15] to derive a sparse integer model for the revised Max-Min Modularity, which we then try to solve optimally using the branch-and-bound technique.

Definition 5 (Complementary graph G'). Let $G = (V, E)$ be a given network with a corresponding set of communities $\mathbf{C} = \{C_1, \dots, C_l\}$ identified by the heuristic algorithm. We associate with $G = (V, E)$ a complementary graph $G' = (V, E')$ such that $(i, j) \in E'$ if and only if $(i, j) \notin E \wedge \{i, j\} \notin C_r, r = 1, 2, \dots, l$.

In essence, two disconnected nodes i and j in G establish an edge in the complementary graph G' , provided that i and j do not belong to a same community specified by the heuristic algorithm introduced in Section 2.1. It is crucial to emphasize the advantages of incorporating this complementary graph into the Max-Min Modularity metric. Firstly, the high-quality communities already obtained by the heuristic algorithm validate the absence of connections between nodes belonging to different communities. Secondly, this approach allows the simultaneous consideration of two quality metrics. While maximizing the topology-based metric, Modularity, over the main network has been proven in the literature to yield highly accurate communities, minimizing it over a graph consisting of unrelated nodes identified by a connectivity-based metric can significantly enhance the results. Additionally, in terms of time complexity, our theoretical findings regarding the heuristic method, coupled with the ability to confidently adjust the number of iterations in the revision phase of the algorithm or even omit it due to the favorable quality of the initial phase, enable us to achieve results in a shorter timeframe.

Having G' defined, we now provide the integer formulation of our *Revised Max-Min Modularity maximization problem* (IP-RMM):

$$\max \sum_{(i,j) \in I_{all}} c_{ij}(1 - x_{ij}) \quad (\text{IP-RMM})$$

$$x_{ij} + x_{jk} - x_{ik} \geq 0 \quad \forall (i, j), (j, k), (i, k) \in I_{all} \quad (5)$$

$$x_{ij} - x_{jk} + x_{ik} \geq 0 \quad \forall (i, j), (j, k), (i, k) \in I_{all} \quad (6)$$

$$-x_{ij} + x_{jk} + x_{ik} \geq 0 \quad \forall (i, j), (j, k), (i, k) \in I_{all} \quad (7)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in I_{all} \quad (8)$$

where the binary variable x_{ij} indicates if nodes i and j belong to the same community ($= 0$) or not ($= 1$). Moreover, $I_{all} = \{(i, j) \in V^2 \mid i < j\}$, $c_{ij} = \frac{q_{ij}}{m} - \frac{q'_{ij}}{m'}$ with $q_{ij} = a_{ij} - \frac{d_i d_j}{2m}$, $q'_{ij} = a'_{ij} - \frac{d'_i d'_j}{2m'}$, $d'_i = \sum_{l=1}^n a'_{il}$, and $m' = \sum_{(i,j) \in I_{all}} a'_{ij}$ for each $(i, j) \in I_{all}$. Constraints (5)–(7) satisfy the triangle inequalities and guarantee that if i and j are in the same community and j and k are in the same community, then so are i and k .

The primary consideration is that solving (IP-RMM) falls into the category of *NP-hard* problems, presenting a challenge for optimal resolution. Consequently, we embarked on an investigation to explore the possibility of simplifying (IP-RMM) while preserving the same set of optimal solutions. To attain this objective, we derived the following conceptual insights from the notion of row generation, all borrowed from [15]: the subsequent lemma illustrates that the optimal solution remains unaffected when concentrating solely on variables x_{ij} with $c_{ij} > 0$, and the subsequent theorem establishes that the optimal solution to (IP-RMM), without constraints involving x_{ij} where $c_{ij} \leq 0$, is equivalent to the optimal solution to (IP-RMM).

Theorem 4. *Excluding variables x_{ij} and the corresponding constraints with $c_{ij} \leq 0$ from the integer programming problem (IP-RMM) does not change the optimality—that is, the optimal solution of the modified problem remains the same as the one for (IP-RMM).*

Proof. Suppose we have a binary variable x_{ij} that satisfies constraints (5)–(7). We show that if $c_{ij} \leq 0$, then the variable x_{ij} will not contribute to maximizing the objective function (IP-RMM). Consider the term $c_{ij}(1 - x_{ij})$ in the objective function (IP-RMM). If $c_{ij} \leq 0$, irrespective of the value of x_{ij} (0 or 1), the term $c_{ij}(1 - x_{ij})$ will be non-positive. Consequently, incorporating x_{ij} in the objective function with $c_{ij} \leq 0$ does not play any role in maximizing the objective. Conversely, if $c_{ij} > 0$, including x_{ij} in the objective function can potentially enhance the objective value by setting x_{ij} to 0 (i.e., nodes i and j belong to the same community) since $c_{ij}(1 - 0) = c_{ij}$. Therefore, it suffices to consider only the variables x_{ij} for which $c_{ij} > 0$ in the objective function (IP-RMM).

In addition, constraints (5)–(7) ensure that the variables x_{ij} adhere to the triangle inequalities, maintaining consistency in community assignments. Removing constraints involving x_{ij} with $c_{ij} \leq 0$ does not change the feasible region of the problem because these constraints do not affect the feasibility of the solution.

Thus, excluding variables x_{ij} and corresponding constraints where $c_{ij} \leq 0$ does not change the optimal solution. $\square$

Taking into account the above theorem, we can simplify the original model (IP-RMM) by focusing only on the variables x_{ij} , where $c_{ij} > 0$. Letting $I_{pos} = \{(i, j) \in I_{all} \mid c_{ij} > 0\}$, the revised integer programming model can be formulated as follows:

$$\max \sum_{(i,j) \in I_{pos}} c_{ij}(1 - x_{ij}) \quad (\text{IPs-MM})$$

$$x_{ij} + x_{jk} - x_{ik} \geq 0 \quad \forall (i, j), (j, k), (i, k) \in I_{pos} \quad (9)$$

$$x_{ij} - x_{jk} + x_{ik} \geq 0 \quad \forall (i, j), (j, k), (i, k) \in I_{pos} \quad (10)$$

$$-x_{ij} + x_{jk} + x_{ik} \geq 0 \quad \forall (i, j), (j, k), (i, k) \in I_{pos} \quad (11)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in I_{pos} \quad (12)$$

An encouraging observation is that (IPs-MM) exhibits significantly fewer constraints and variables in comparison to the original (IP-RMM) model while maintaining an equivalent set of optimal solutions. Nevertheless, obtaining the optimal solution for average to large-scale networks might still pose challenges. Undoubtedly, a viable approach to expedite the branch-and-bound technique, especially when utilizing the CPLEX (CPLEX is a powerful optimization software package developed by IBM that employs advanced algorithms to solve linear, mixed-integer, and quadratic programming problems efficiently.) solver, is to provide it with a sufficiently good initial solution. Commencing with a judicious selection from the feasible solution space has proven to significantly enhance performance. In this context, we once again leverage the proposed heuristic algorithm. Specifically, we employ the communities generated by the heuristic approach as an initial solution for solving (IPs-MM).

With all elements considered, Algorithm 3 frames our method for (near) optimally identifying communities. It is essential to note the integer inputs k_1 and k_2 of the algorithm, which control the number of iterations in the second phase of the heuristic algorithm and the number of repetitions considered for solving (IPs-MM), respectively. The latter parameter is intertwined with the number of the model's constraints involved in the optimization. Note that due to the NP-hard nature of (IPs-MM), a row generation technique is employed. This technique initially solves the problem without considering any constraints and then iteratively adds those that are not satisfied by the obtained solution. Whereas assuming $k_2 = \infty$ lets the algorithm terminate whenever it finds the optimal solution, this could prolong execution times for huge networks. To address this, we came to introduce a bounding mechanism, limiting the number of times the algorithm is allowed to add violated constraints. While this bounding introduces some inaccuracy, due to the smart choice of the initial solution, we do not end far from the optimal solution. Furthermore, the promising initial solution reduces the likelihood of encountering cases where the algorithm would reach the optimal solution with all constraints satisfied after k_2 iterations.

Algorithm 3: The proposed algorithm for community discovery

Input: Graph G and integers k_1 and k_2 .
Output: Set of communities C .
1 $C \leftarrow UC(G, CIS(G), k_1)$; // Heuristic approach (Algorithms 1 and 2)
2 Iteration $\leftarrow 1$;
3 Formulate (IPs-MM) without any constraints;
4 **while** true **do**
5 Solve (IPs-MM) using the branch-and-bound method to obtain the optimal solution x^* ;
6 **if** all constraints of (IPs-MM) are satisfied by $x^* \vee (Iteration > k_2)$ **then**
7 Break;
8 **else**
9 Add the violated constraints to the model;
10 Iteration++;
11 **Return** the set of community C associated with the solution x^* ;

As a complement to the above discussion, one might point out that whereas the golden standard for determining optimal communities is to let both k_1 and k_2 be sufficiently large, it does not sound logical, especially when dealing with large networks. What is important to note here is the advantage one can take from the trade-off between these two parameters. More specifically, despite the promising quality of the initial communities, the greater the number of k_1 , the better the communities become, which leads to less concern regarding the optimality of the (IPs-MM). In contrast, the fewer reforms performed due to the small value of k_1 seem to require more care with finalizing the communities, which highlights the importance of achieving an optimal solution to (IPs-MM) with as many satisfied constraints as possible, which itself depends on picking a large value of k_2 . To summarize, one can conclude that picking a large value of k_1 can relax a large choice of k_2 and vice versa. On the other hand, as already observed in [21], k_1 does not need to be large due to the already high-quality initial communities obtained by the first phase of the heuristic algorithm, which has turned out to be efficient also in terms of time execution. This provides us with a final useful hint: choosing generally a relatively small value of k_1 , especially in the case of dealing with large networks, and presuming k_2 to be large enough to make a safe margin about having the majority of constraints of (IPs-MM) satisfied, could lead us to obtain (near) optimal community structures. The experiments in the next section validate this assertion.

3. Evaluation of Accuracy

In this section, we delve into verifying the efficiency of our algorithm throughout various experiments. Due to our method having been built upon the heuristic and the exact method introduced in [15,21], we came to, first of all, compare outcomes with the one determined by those two approaches.

It is worth noting that [21] succeeded in surpassing various state-of-the-art heuristic community detection algorithms, like Distributed Louvain (DLouvain) [35], Variable Neighborhood Search (VNS) [36], Simulated Annealing (SA) [37], and FastGreedy [13] algorithms, both in terms of execution time and communities quality. In addition, Ref. [15] managed to detect communities superior to the ones obtained by [9,16], where in the former, the optimal solution to the linear relaxation model of ordinary Max-Min Modularity was fed into a local search manner rounding procedure for obtaining the communities. In the latter, communities were discovered by using the user-defined complementary graph introduced in [9] and also applying its proposed hierarchical heuristic algorithm for maximizing the Max-Min Modularity.

With this in mind, if our algorithm can outperform all the above approaches, it would substantiate its efficiency. For this reason, we take all the network benchmarks used in [15,21] and compare the results of all algorithms with each other. Furthermore, we also generate and use six *Lancichinetti Fortunato Radicchi* (LFR) artificial benchmarks [38], which closely resemble real-world complex networks by incorporating a pre-defined community structure. Generating these artificial graphs, which will be explained later, is based on the user-defined parameters fully elucidated in [38]. What makes the comparison fair is that all the under-study graphs, including the real-world networks presented in Table 2 and the artificial ones, are among the recognized and commonly utilized networks in this context. More importantly, each possesses the corresponding ground truth, representing the optimal community structures. This allows us to evaluate the effectiveness of a community detection algorithm by measuring the similarities between the algorithmically derived communities and the optimal structures. For this evaluation, we employ the widely acknowledged performance metric, *Normalized Mutual Information* (NMI), which we explain in Section 3.1.

Table 2. Networks under-study.

ID	Network	n	m
1	Zachary's karate club [39]	34	78
2	Mexican Politicians [40]	35	117
3	Dolphin network [41]	62	159
4	Les Misérables [41]	77	254
5	p53 protein [42]	104	226
6	Books about U.S. politics [43]	105	441
7	American college football [8]	115	613
8	Citation graph drawing [44]	311	640
9	USAir97 [45]	332	2126
10	C. Elegans [46]	453	2025
11	Erdos collaboration [47]	472	1314
12	Electronic circuit [48]	512	819
13	Email-Eu-core [49]	986	16,064
14	Amazon network [50,51]	334,863	925,872
15	LiveJournal social network [51,52]	3,997,962	34,681,189

Subsequently, we focus exclusively on the proposed algorithm, delving into the role of its input parameters, namely, k_1 and k_2 , delineated in Algorithm 3, in improving community establishments.

It is worth mentioning that all the experiments are conducted on a computer system with an Intel(R) Core i9-12900KF processor @ 3.2GHz, 16.6 Core(s) and 128 GB of RAM. Algorithms are implemented with C++, and CPLEX optimizer 12.9 is used for solving linear programming.

3.1. Normalized Mutual Information (NMI)

Whereas Normalized Mutual Information (NMI) (We have additionally utilized the *Adjusted Rand Index* (ARI) [53], a recognized metric for community evaluation. However, as the results were consistent with those obtained using NMI, we have opted not to include them in order to maintain conciseness in the paper.), as defined in [54], stands as a widely acknowledged metric for assessing the similarity between clusters, it is particularly effective in measuring the agreement between optimal communities and those discovered by an algorithm. Consider a network G with n nodes, where $\mathcal{C}(\mathcal{A}) = C_1, \dots, C_k$ represents the communities obtained by algorithm $\mathcal{A}$, and $\mathcal{C}' = C'_1, \dots, C'_{k'}$ denotes the ground truth communities. The NMI value corresponding to algorithm $\mathcal{A}$ can be computed as follows:

$$NMI(\mathcal{A}) = \frac{-2 \sum_{x=1}^{|\mathcal{C}|} \sum_{y=1}^{|\mathcal{C}'|} \frac{|C_x \cap C'_y|}{n} \log\left(\frac{n|C_x \cap C'_y|}{|C_x||C'_y|}\right)}{\sum_{x=1}^{|\mathcal{C}|} \frac{|C_x|}{n} \log\left(\frac{|C_x|}{n}\right) + \sum_{y=1}^{|\mathcal{C}'|} \frac{|C'_y|}{n} \log\left(\frac{|C'_y|}{n}\right)}. \quad (13)$$

When the identified communities precisely match the ground truth, NMI achieves its maximum value of one. Conversely, if there is no resemblance between the two sets, the NMI score drops to zero. Generally, a larger NMI value signifies a more precise and effective unveiling of community structures.

3.2. Experiments

In our initial series of experiments, our objective is to assess the proposed algorithm in comparison to those presented in [15,21] using the 15 real-world networks. Figure 1 portrays the computed NMI values for each methodology. To offer a more comprehensive perspective on the advancements introduced by each algorithm, the results from the competitive algorithms utilized in [15,21] are also depicted in the figure.

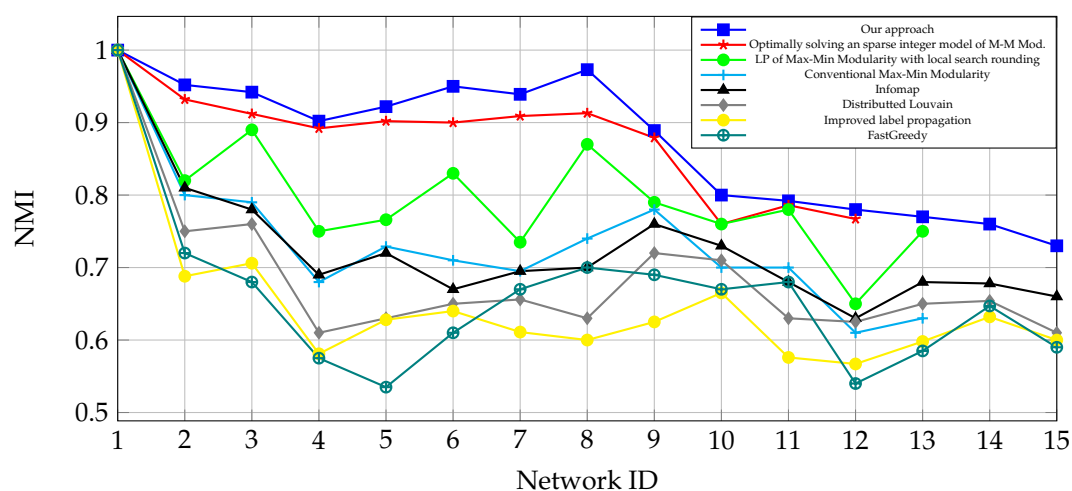


Figure 1. Comparison between NMI values achieved by our approach, the blue curve, and optimally solving a sparse integer model of M-M Mod. [15], and other understudy algorithms (from the top, second curve to bottom: [15], [16], [55], [9], [35], [56], and [13], using real-world networks presented in Table 2.

While the superiority of our proposed algorithm against all other methodologies across various real-world benchmarks is evident, it is crucial to highlight the relatively small values of the parameters k_1 and k_2 required to achieve these results. Table 3 outlines the used parameter values. Whereas the selection of k_1 can be made relatively freely, often contingent on the network size, the utilized values of k_2 here represent the minimum number of iterations required to solve the corresponding (IPs-MM) with all constraints satisfied.

Table 3. Used values for the algorithm’s input parameters k_1 and k_2 associated with each of the real-world networks depicted in Table 2.

Parameter	Network ID														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
k_1	0	0	0	2	2	2	2	3	4	4	4	4	5	5	5
k_2	2	2	3	3	3	3	5	6	8	8	11	15	18	25	29

The small values of these parameters affirm, firstly, the existence of high-quality initial communities that necessitate minimal restructuring. Secondly, they substantiate the claim that all constraints of (IPs-MM) are swiftly satisfied in the optimal solution to the model, particularly within the initial iterations of the applied row generation technique. This revelation not only bodes well for achieving refined communities but also ensures a reasonable timeframe for the algorithm. Bring to mind that our baseline methods [15,16] exhibit limitations in producing results within a reasonable timeframe, particularly when applied to relatively large networks. Specifically, the former experiences complete failure when confronted with networks 13, 14, and 15, while the latter collapses when applied to networks 14 and 15 within a reasonable timeframe.

By the same token, this breakthrough can be extrapolated to artificial networks. As mentioned, the generation process of the artificial graphs we take into account relies on what was introduced in [38]. These networks adhere to power-law distributions for both the degree and community size, with user-defined exponents γ and β , respectively. Additionally, the mixing parameter μ dictates that each vertex in the graph should share a fraction of $1 - \mu$ of its links with other nodes in its community and a fraction of μ with other nodes in the network. The parameter μ effectively controls the noise, with higher values leading to less-distinct community structures. Thus, a higher μ poses challenges for community detection algorithms in identifying optimal communities [31]. In this study, we utilize four different LFR graphs with distinct mixing parameters μ ($\mu = 0.1, 0.3, 0.6$, and 0.9), each comprising 1000 nodes and 5 communities. All other parameters are set to default values, as detailed in [38].

We employ these benchmarks for a dual purpose: to assess the precision of our algorithms in comparison to other competing methods and to investigate the impact of the algorithm’s input parameters, namely, k_1 and k_2 . Figure 2 illustrates the resultant NMI values corresponding to each scenario.

As can be discerned, in scenarios $\mu = 0.1$ and $\mu = 0.3$, which indicate less-intricate community interconnections in the synthesized networks, all algorithms more or less effortlessly detected structures closely aligned with the ground truths. Particularly noteworthy is the superior performance of the proposed approach, even with relatively small values of k_1 and k_2 (depicted by the blue curve). The selection of k_1 underscores the algorithm’s ability to bypass unnecessary iterations in the community revision step to sufficiently improve the complementary graph. Meanwhile, the small choice of k_2 demonstrates the algorithm’s rapid convergence toward the optimal solution by satisfying all the constraints of (IPs-MM).

In contrast, as the level of noise within the network’s structure increases, notably with an elevated value of μ , especially for $\mu = 0.9$, all competing algorithms encounter difficulties in capturing the optimal community structure. Intriguingly, the proposed method also faces challenges in this scenario when k_1 and k_2 are not sufficiently large—specifically, for the black curve, wherein $k_1 = k_2 = 3$, the proposed algorithm exhibits a complete degradation against other techniques.

From a time complexity perspective, as illustrated in Figure 3, our algorithm operates significantly faster compared to other methods. In previous research [15], it was demonstrated that our baseline model (depicted by the black curve in Figure 3) holds an advantage over all competing algorithms except for the one introduced in [16]. The latter solves the LP relaxation version of the model and employs a rounding algorithm to determine the integer solution and, consequently, the community structures (red curve in Figure 3). Notably, our

augmented method (the blue curve) not only surpasses the baseline model but also closely approaches a fast heuristic but not very accurate approach presented in [16].

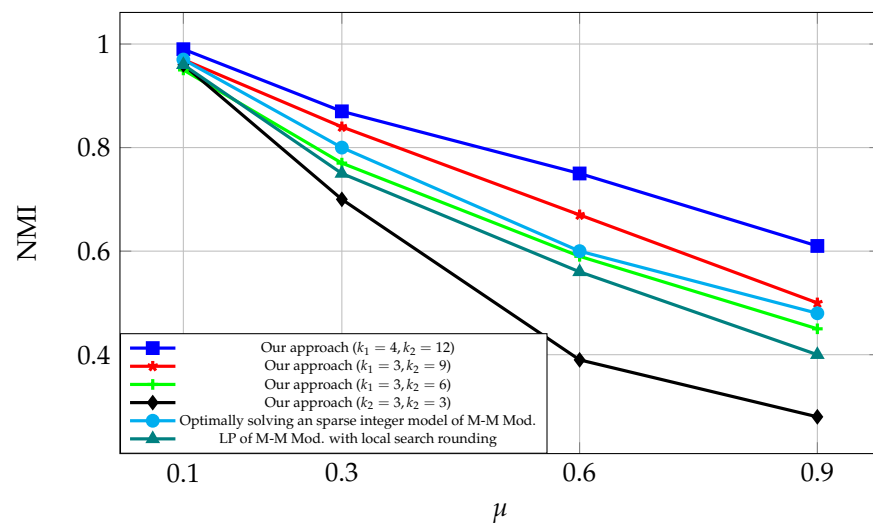


Figure 2. Comparison between NMI ranks of different algorithms consisting of our method for four different parameter settings and also optimally solving a sparse integer model of M-M Modularity [15] and LP of Max-Min Modularity with local search rounding [16], all applying to the LFR synthetic graphs, generated by considering four different mixing parameters μ .

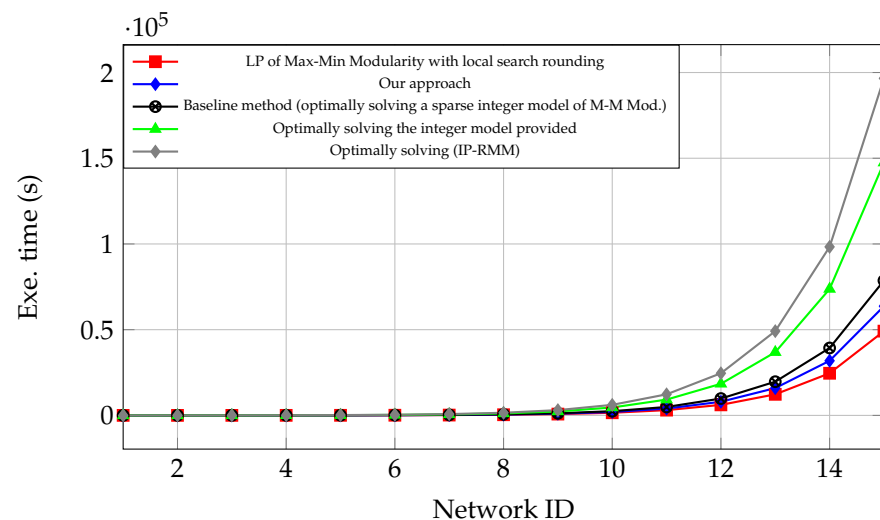


Figure 3. Execution time associated with each algorithm (from top, [16], our approach, [15], [16], and optimally solving (IP-RMM)) applied to the real-world networks presented in Table 2.

We state that, as expected, not only does our proposed augmented method outperform our baseline approach [15] in terms of time complexity (as depicted in Figure 3), it also yields higher quality communities that closely align with the ground truths (as depicted in Figures 1 and 2). This is achieved through a heuristic design for enhancing the Max-Min Modularity complementary graph and expediting solving the sparse yet equivalent integer counterpart we introduced for the problem. As a final note regarding the selection of parameters, as networks increase in size, a smaller value for k_1 becomes more reasonable. This is attributed primarily to the high quality of the initial communities obtained in the first phase of the heuristic method and, secondly, to the time efficiency of this phase, as elucidated in Section 2.1. As for k_2 , although the sparse model (IPs-MM) expedites the solving process, opting for a larger value may not be necessary, especially when dealing with massive graphs. This is particularly evident and plausible given the typically swift

convergence of the model, wherein all constraints are satisfied by an optimal solution obtained in the early iterations.

4. Conclusions

Communities and community detection in complex networks are crucial for uncovering hidden structures and relationships, which enhance our understanding of various real-world systems, from communication networks to biological systems. By identifying these communities, we can improve strategies for targeted interventions, optimize network functionalities, and predict system behaviors more accurately. This paper introduced a novel approach to community detection in complex networks, combining the strengths of a connectivity-based metric and the Max-Min Modularity. The synergistic incorporation of these two effective community quality functions resulted in the development of a robust algorithm. By leveraging the connectivity-based metric incorporated into a heuristic algorithm, a novel complementary graph associated with Max-Min Modularity was established, demonstrating enhanced effectiveness in community detection. The analytical insights derived substantiated the time efficiency of the employed heuristic method, thus addressing the time complexity concerns. The formulation of community detection as an integer programming problem and the presentation of a more compact yet equivalent model contributed to the optimization of computational resources. The proposed sparse model significantly reduced the number of variables and constraints. Furthermore, depending on the network size, the paper introduced a systematic procedure to balance between optimally solving the sparse model and relying on the heuristic method to facilitate community detection in very large networks. Through comprehensive experiments, the success of the algorithm was showcased, highlighting its efficiency in detecting communities.

Author Contributions: The authors' contributions are delineated as follows: Both authors jointly managed the conceptualization, methodology, software development, and manuscript preparation. A.F. specifically undertook the tasks of validation, formal analysis, data curation, and the review and editing process. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Jalali, M.; Tsotsalas, M.; Wöll, C. MOFSocialNet: Exploiting Metal-Organic Framework Relationships via Social Network Analysis. *Nanomaterials* **2022**, *12*, 704. [\[CrossRef\]](#) [\[PubMed\]](#)
2. Jiang, L.; Shi, L.; Liu, L.; Yao, J.; Yousuf, M.A. User interest community detection on social media using collaborative filtering. *Wirel. Netw.* **2019**, *28*, 1169–1175. [\[CrossRef\]](#)
3. Hu, L.; Zhang, J.; Pan, X.; Yan, H.; You, Z.H. HiSCF: Leveraging higher-order structures for clustering analysis in biological networks. *Bioinformatics* **2021**, *37*, 542–550. [\[CrossRef\]](#) [\[PubMed\]](#)
4. Zhao, D.; Li, J.; Jiang, Z. Effects of link perturbation on network modularity for community detections in complex network systems. *Mod. Phys. Lett. B* **2021**, *35*, 2150214. [\[CrossRef\]](#)
5. Gawande, N.; Ghosh, S.; Halappanavar, M.; Tumeo, A.; Kalyanaraman, A. Towards scaling community detection on distributed-memory heterogeneous systems. *Parallel Comput.* **2022**, *111*, 102898. [\[CrossRef\]](#)
6. Ramakrishna, R.; Scaglione, A. Grid-graph signal processing (grid-GSP): A graph signal processing framework for the power grid. *IEEE Trans. Signal Process.* **2021**, *69*, 2725–2739. [\[CrossRef\]](#)
7. Magnani, M.; Hanteer, O.; Interdonato, R.; Rossi, L.; Tagarelli, A. Community detection in multiplex networks. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–35. [\[CrossRef\]](#)
8. Girvan, M.; Newman, M.E. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 7821–7826. [\[CrossRef\]](#) [\[PubMed\]](#)
9. Chen, J.; Zaïane, O.R.; Goebel, R. Detecting communities in social networks using max-min modularity. In Proceedings of the 2009 SIAM International Conference on Data Mining, Sparks, NV, USA, 30 April–2 May 2009; pp. 978–989.
10. Boudebza, S.; Cazabet, R.; Azouaou, F.; Nouali, O. OLCPM: An online framework for detecting overlapping communities in dynamic social networks. *Comput. Commun.* **2018**, *123*, 36–51. [\[CrossRef\]](#)

11. Shen, H.W.; Cheng, X.Q.; Guo, J.F. Quantifying and identifying the overlapping community structure in networks. *J. Stat. Mech. Theory Exp.* **2009**, 2009, P07042. [\[CrossRef\]](#)
12. Devi, J.C.; Poovammal, E. An analysis of overlapping community detection algorithms in social networks. *Procedia Comput. Sci.* **2016**, 89, 349–358. [\[CrossRef\]](#)
13. Newman, M.E. Fast algorithm for detecting community structure in networks. *Phys. Rev. E* **2004**, 69, 066133. [\[CrossRef\]](#)
14. Raghavan, U.N.; Albert, R.; Kumara, S. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **2007**, 76, 036106. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Ferdowsi, A.; Chenary, M.D. Toward an Optimal Solution to the Network Partitioning Problem. In Proceedings of the 2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS), Warsaw, Poland, 17–20 September 2023; pp. 111–117.
16. Ferdowsi, A.; Khanteymoori, A. Discovering communities in networks: A linear programming approach using max-min modularity. In Proceedings of the 2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS), Sofia, Bulgaria, 2–5 September 2021; pp. 329–335.
17. Schaeffer, S.E. Graph clustering. *Comput. Sci. Rev.* **2007**, 1, 27–64. [\[CrossRef\]](#)
18. Cheikh, S.; Sara, B.; Sara, Z. A Hybrid Heuristic Community Detection Approach. In Proceedings of the 2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), Novi Sad, Serbia, 24–26 August 2020; pp. 1–7.
19. Gao, Y.; Yu, X.; Zhang, H. Overlapping community detection by constrained personalized PageRank. *Expert Syst. Appl.* **2021**, 173, 114682. [\[CrossRef\]](#)
20. Sahu, S.; Rani, T.S. A neighbour-similarity based community discovery algorithm. *Expert Syst. Appl.* **2022**, 206, 117822. [\[CrossRef\]](#)
21. Ferdowsi, A.; Dehghan Chenary, M.; Khanteymoori, A. Tscda: A dynamic two-stage community discovery approach. *Soc. Netw. Anal. Min.* **2022**, 12, 46. [\[CrossRef\]](#)
22. Chakraborty, T.; Dalmia, A.; Mukherjee, A.; Ganguly, N. Metrics for community analysis: A survey. *ACM Comput. Surv. (CSUR)* **2017**, 50, 54. [\[CrossRef\]](#)
23. Radicchi, F.; Castellano, C.; Cecconi, F.; Loreto, V.; Parisi, D. Defining and identifying communities in networks. *Proc. Natl. Acad. Sci. USA* **2004**, 101, 2658–2663. [\[CrossRef\]](#)
24. Wei, Y.C.; Cheng, C.K. Towards efficient hierarchical designs by ratio cut partitioning. In Proceedings of the 1989 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers, Santa Clara, CA, USA, 5–9 November 1989; pp. 298–301.
25. Shi, J.; Malik, J. Normalized cuts and image segmentation. *Dep. Pap. (CIS)* **2000**, 22, 888–905.
26. Flake, G.W.; Lawrence, S.; Giles, C.L. Efficient identification of web communities. In Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, USA, 20–23 August 2000; pp. 150–160.
27. Newman, M.E. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA* **2006**, 103, 8577–8582. [\[CrossRef\]](#)
28. Alozie, G.U.; Arulselvan, A.; Akartunali, K.; Pasiliao, E.L., Jr. Efficient methods for the distance-based critical node detection problem in complex networks. *Comput. Oper. Res.* **2021**, 131, 105254. [\[CrossRef\]](#)
29. Hamilton, W.L. Graph representation learning. *Synth. Lect. Artificial Intell. Mach. Learn.* **2020**, 14, 1–159.
30. Orman, G.K.; Labatut, V. A comparison of community detection algorithms on artificial networks. In Proceedings of the International Conference on Discovery Science, Porto, Portugal, 3–5 October 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 242–256.
31. Ferdowsi, A.; Abhari, A. Generating high-quality synthetic graphs for community detection in social networks. In Proceedings of the 2020 Spring Simulation Conference, Fairfax, VA, USA, 18 May 2020; pp. 1–10.
32. Arya, V.; Garg, N.; Khandekar, R.; Meyerson, A.; Munagala, K.; Pandit, V. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.* **2004**, 33, 544–562. [\[CrossRef\]](#)
33. Gupta, A.; Tangwongsan, K. Simpler analyses of local search algorithms for facility location. *arXiv* **2008**, arXiv:0809.2554.
34. Ferdowsi, A. An Integer Programming Approach Reinforced by a Message-passing Procedure for Detecting Dense Attributed Subgraphs. In Proceedings of the 2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS), Sofia, Bulgaria, 4–7 September 2022; pp. 569–576.
35. Ghosh, S.; Halappanavar, M.; Tumeo, A.; Kalyanaraman, A.; Lu, H.; Chavarria-Miranda, D.; Khan, A.; Gebremedhin, A. Distributed louvain algorithm for graph community detection. In Proceedings of the 2018 IEEE international parallel and distributed processing symposium (IPDPS), Vancouver, BC, Canada, 21–25 May 2018; pp. 885–895.
36. Aloise, D.; Caporossi, G.; Hansen, P.; Liberti, L.; Perron, S.; Ruiz, M. Modularity maximization in networks by variable neighborhood search. *Graph Partitioning Graph Clust.* **2012**, 588, 113.
37. Xie, J.R.; Wang, B.H. Modularity-like objective function in annotated networks. *Front. Phys.* **2017**, 12, 128903. [\[CrossRef\]](#)
38. Lancichinetti, A.; Fortunato, S.; Radicchi, F. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **2008**, 78, 046110. [\[CrossRef\]](#) [\[PubMed\]](#)
39. Zachary, W.W. An information flow model for conflict and fission in small groups. *J. Anthropol. Res.* **1977**, 33, 452–473. [\[CrossRef\]](#)
40. Gil-Mendieta, J.; Schmidt, S. The political network in Mexico. *Soc. Netw.* **1996**, 18, 355–381. [\[CrossRef\]](#)
41. 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. *Behav. Ecol. Sociobiol.* **2003**, 54, 396–405. [\[CrossRef\]](#)
42. Guimera, R.; Danon, L.; Diaz-Guilera, A.; Giralt, F.; Arenas, A. Self-similar community structure in a network of human interactions. *Phys. Rev. E* **2003**, 68, 065103. [\[CrossRef\]](#) [\[PubMed\]](#)

43. Mahajan, A.; Kaur, M. Various approaches of community detection in complex networks: A glance. *Int. J. Inf. Technol. Comput. Sci. (IJITCS)* **2016**, *8*, 35–41. [\[CrossRef\]](#)
44. Meghanathan, N. A greedy algorithm for neighborhood overlap-based community detection. *Algorithms* **2016**, *9*, 8. [\[CrossRef\]](#)
45. Batagelj, V.; Mrvar, A. Pajek Datasets. 2009. Available online: <http://vlado.fmf.uni-lj.si/pub/networks/data/> (accessed on 20 May 2024).
46. Cangelosi, A.; Parisi, D. A neural network model of caenorhabditis elegans: The circuit of touch sensitivity. *Neural Process. Lett.* **1997**, *6*, 91–98. [\[CrossRef\]](#)
47. Mrvar, A.; Batagelj, V. *Pajek: Programs for Analysis and Visualization of Very Large Networks: Reference Manual: List of Commands with Short Explanation Version 5.10*; Mrvar, A., Ed.; University of Ljubljana: Ljubljana, Slovenia, 2020.
48. Chand, S.; Mehta, S. Community detection using nature inspired algorithm. In *Hybrid Intelligence for Social Networks*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 47–76. [\[CrossRef\]](#)
49. Leskovec, J.; Kleinberg, J.; Faloutsos, C. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data (TKDD)* **2007**, *1*, 2. [\[CrossRef\]](#)
50. Mkhitarian, K.; Mothe, J.; Haroutunian, M. Detecting communities from networks: Comparison of algorithms on real and synthetic networks. *Int. J. Inf. Theor. Appl.* **2019**, *26*, 231–267.
51. Yang, J.; Leskovec, J. Defining and evaluating network communities based on ground-truth. *Knowl. Inf. Syst.* **2015**, *42*, 181–213. [\[CrossRef\]](#)
52. Shi, X.; Lu, H.; He, Y.; He, S. Community detection in social network with pairwise constrained symmetric non-negative matrix factorization. In Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Paris, France, 25–28 August 2015; pp. 541–546.
53. Yip, K.Y.; Cheung, D.W.; Ng, M.K. Harp: A practical projected clustering algorithm. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 1387–1397. [\[CrossRef\]](#)
54. Danon, L.; Diaz-Guilera, A.; Duch, J.; Arenas, A. Comparing community structure identification. *J. Stat. Mech. Theory Exp.* **2005**, *2005*, P09008. [\[CrossRef\]](#)
55. Rosvall, M.; Bergstrom, C.T. Maps of random walks on complex networks reveal community structure. *Proc. Natl. Acad. Sci. USA* **2008**, *105*, 1118–1123. [\[CrossRef\]](#) [\[PubMed\]](#)
56. Xu, G.; Guo, J.; Yang, P. TNS-LPA: An Improved Label Propagation Algorithm for Community Detection Based on Two-Level Neighbourhood Similarity. *IEEE Access* **2020**, *9*, 23526–23536. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.