Information Sciences 742 (2026) 123032

Contents lists available at ScienceDirect

Information Sciences

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

Null-adjusted persistence function for high-resolution community detection

Alessandro Avellone [a], Paolo Bartesaghi [b], Stefano Benati [c], Christos Charalambous [d], Rosanna Grassi [a,*]

[a] University of Milano - Bicocca, Via Bicocca degli Arcimboldi 8, Milano 20126, Italy
[b] University of Milano, Via Conservatorio 7, Milano 20122, Italy
[c] University of Trento, Via Verdi 26, Trento 38122, Italy
[d] University of Cyprus, Department of Economics, PO Box 20537, Nicosia 1678, Cyprus

HIGHLIGHTS

- We introduce a new objective function for community detection.
- It combines modularity and persistence probability using a null model comparison.
- It overcomes modularity's resolution and scaling limitations.
- It yields higher-resolution partitions on real and synthetic networks.

ARTICLE INFO

JEL classification:
C02
C38
C61

Keywords:
Community detection
Modularity
Persistence probability
Null-adjusted persistence

ABSTRACT

Modularity and persistence probability are two widely used quality functions for detecting communities in complex networks. In this paper, we introduce a new objective function called null-adjusted persistence, which incorporates features from both modularity and persistence probability, as it implies a comparison of persistence probability with the same configuration null model of modularity. We prove key analytical properties of this new function, demonstrating that it successfully addresses modularity's well-known scaling and resolution limitations, as well as the monotonic bias of persistence probability with respect to cluster size. To optimize this new function, we adapt the Louvain method and evaluate our approach on both synthetic benchmarks and real-world networks. Our results show that maximizing null-adjusted persistence consistently yields higher-resolution partitions than standard modularity maximization, particularly in large real networks.

1. Introduction

Community detection is an essential research area of network science: it seeks to uncover densely connected subgroups of nodes, interpreted as communities – also referred to as clusters or modules – within a given network. Loosely speaking, communities should be characterized by multiple links between their members and easy, fast communication between them. In contrast, they should have sparse connections to external nodes and less accessibility to information outside the groups.

Identifying these structures is critical to understanding the underlying organization and functionality of networks in various domains, including social [1,2] and economic and financial applications [3,4]. An essential contribution to community detection that shaped the modern study of the field is contained in [1]. To detect communities, the authors proposed an algorithm based on

* Corresponding author.
Email address: rosanna.grassi@unimib.it (R. Grassi).

https://doi.org/10.1016/j.ins.2025.123032
Received 16 May 2025; Received in revised form 23 December 2025; Accepted 23 December 2025
Available online 25 December 2025

hierarchical partitioning, in which arcs are progressively deleted depending on their edge betweenness. In the following two decades, the contributions to community detection by scientists from the fields of network theory, physics, computer science, operations research, and inferential statistics have been numerous and diverse, as evidenced by the surveys [5,6].

Among others, the following approach has been a recognizable research trend: first, a network statistic is elaborated to discern groups that can be interpreted as communities from the ones that cannot; then, the network community structure is determined through the maximization of the proposed statistic used as an objective function, that is, the community structure emerges as the partition that maximizes that statistic. The most widely used community statistic is modularity, for both weighted and unweighted networks [7,8]. Modularity evaluates the difference between the actual number of edges within communities and the expected number of such edges in a random network with the same degree distribution (the so-called *configuration model*, see [9]). Modularity has been maximized through exact and heuristic methods. One of the first and most popular heuristics is the Louvain algorithm [10], in which nodes and communities are progressively merged until a (local) maximal modularity has been found. The exact methods maximize modularity through a clique partitioning problem, formulated as an Integer Linear Programming (ILP) problem and solved with specific techniques [11,12]. However, maximizing modularity and clique partitioning are NP-hard problems. Only instances of moderate size can be solved, and therefore a large effort has been devoted to developing and improving heuristics, to the point that sometimes the purpose of maximizing the modularity has been lost (readers can refer to Aref et al. [13] for an accurate list of modularity maximization heuristic algorithms).

Nevertheless, modularity suffers from some drawbacks. Specifically, it is biased by the so-called resolution limit [14–17]. Although not apparent from its definition, the size of the network has an impact on the maximization of modularity. It has been proven that, if communities are small enough with respect to the network size, then they are not recognizable as they are merged into larger groups. To address this problem, some authors proposed corrections to modularity, such as the modularity density [18], or the z-modularity [19]. Another approach consists of improving the Louvain algorithm with the Leiden algorithm, which addresses major issues in the Louvain method, including the detection of arbitrarily weakly connected communities [20]. Other authors suggested applying a measure called the normalized cuts [21]. Alternatively, some research deals with the issue of finding communities by a minimization procedure. It minimizes the length of a random walk using the so-called *map equation*. A specific heuristic algorithm, Infomap, has been developed for this purpose [22].

In [23], a new index, called persistence probability, is proposed and described as the probability that a random walker starting in a given cluster will move to a node within the same cluster in the next iteration. As will be seen, persistence probability is defined as the ratio between the internal edges of a cluster and all the edges adjacent to that cluster; therefore, as is the case for modularity, a large number of internal arcs is an important feature of the community definition. However, the two measures are based on two different arguments. To define a community, modularity emphasizes static bonds *within* its members, while persistence emphasizes the role of dynamic communication *between* its members. A community with a high persistence probability reflects a scenario where information spreading across the network remains within a given community, and it is not shared with the rest of the network. Indeed, it has been used to analyze the world trade web structure [24], to characterize criminal gangs [25], to identify cohesive and persistent communities in dynamic social networks [26] and in datasets from platforms like Facebook and Twitter [27]. Algorithms and subroutines for the persistence probability are not as well-developed and tested as those for the modularity. The preliminary problem of finding *one* community that maximizes a slightly modified version of persistence probability has been discussed in [28]. It has been found that the problem can be formulated as a fractional integer programming problem, and therefore, exact and heuristic methods are proposed and applied to real network data.

In this paper, we deal with the problem of finding the graph partition with maximum persistence probability. In Section 2, we introduce the definition of persistence probability in the context of Markov chains and formulate the maximization problem. Since persistence probability tends to increase with respect to the size of the cluster, we propose a correction to persistence by defining a new function, which we call *null-adjusted persistence*. Drawing inspiration from modularity, it adjusts the persistence probability with a term representing the expected persistence under the configuration model, that is, a null hypothesis that assumes a random graph with no community structure. In Section 3, we present some analytical results. In particular, we highlight the differences between null-adjusted persistence and modularity, and we show that the former is scale-independent and not affected by the resolution limit. Then, we prove a necessary and sufficient condition for merging two distinct clusters and improving the null-adjusted persistence. In Section 4, we develop heuristic and exact algorithms to solve the proposed problem. We apply these algorithms to evaluate null-adjusted persistence on synthetic benchmark networks and real-world datasets. The primary aim is to conduct a comparative analysis between null-adjusted persistence and established community detection metrics. Specifically, we benchmark our approach against classical modularity optimization and other state-of-the-art community detection algorithms, such as Louvain, Leiden, and Infomap. Finally, in Section 5, we conclude and discuss possible future developments.

To summarize our findings, we highlight the following points:

- A thorough formal comparison of the structural properties of persistence and modularity is carried out. Unlike modularity, it is proven that null-adjusted persistence is not sensitive to network size. A necessary and sufficient condition is proven for improving the measure by merging two clusters.
- Optimal Integer Linear Programming and heuristic algorithms are provided to compute optimal or approximate values of the objective function and the corresponding communities.
- Computational tests on simulated and real data reveals two kinds of communities: large and sparsely connected communities, the former, small and densely connected communities, the latter. This empirical behavior, which supports the theoretical

results, highlights that there are applications in which persistence is better suited than modularity to reveal the hidden communities.

2. Persistence probability by Markov chains

In this section, we introduce the definition of persistence probability. Let $G = (V, E, W)$ be a weighted, undirected graph (or network), where V is the set of n vertices (or nodes), E is the set of edges and W is the set of edge weights. Let $n = |V|$ be the cardinality of V. The subgraph induced by $V' \subseteq V$ is the graph $G_{V'}$ whose vertex set is V' and whose edge set consists of all the edges in E that have both endpoints in V'. A *clustering* of G is a partition of the network nodes $\Pi = \{C_1, \dots, C_q\}$ where $C_\alpha \subseteq V$, $C_\alpha \neq \emptyset$ and the induced subgraph G_{C_α} is connected, for all $\alpha = 1, \dots, q$. We denote the set of all possible clusterings of a graph G with $\wp(G)$.

The information about the weights assigned to the edges is contained in the n-square matrix $\mathbf{W} = \left[w_{ij}\right]$, where $w_{ij} \geq 0$ is the weight of the edge between nodes i and j. The underlying graph, obtained by neglecting the arcs weights, is described by the adjacency matrix $\mathbf{A} = \left[a_{ij}\right]$, where $a_{ij} = 1$ if $w_{ij} > 0$ and $a_{ij} = 0$ otherwise. The strength and the degree of a node i are defined, respectively, by $s_i = \sum_{j=1}^{n} w_{ij}$ and $k_i = \sum_{j=1}^{n} a_{ij}$. We denote by $\mathbf{s}$ and $\mathbf{k}$ the vectors of the strengths and the degrees, respectively, and by $2S = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} = \sum_{i=1}^{n} s_i$ the total strength of the network.

Since persistence probability is based on random walks on a graph, we assume that the graph G is connected. We can place a homogeneous discrete-time Markov chain over the network, whose space of states coincides with the set of nodes V. Specifically, a discrete-time Markov chain on the network is any stochastic process represented by a sequence of n–state vectors $\pi(t) = \left(\pi_1(t), \dots, \pi_n(t)\right)$ such that the probability of being in any state at a given step t depends only on the state at the previous step $t-1$. The process is thus described by the equation $\pi(t+1) = \pi(t)\mathbf{P}$, where $\mathbf{P} = \left[p_{ij}\right]$ is an n-squared matrix and $0 \leq p_{ij} \leq 1$ is the conditional probability that a random walker jumps from node i to node j at each step, assuming it is in i. $\mathbf{P}$ is a row-stochastic matrix called transition probability matrix. The simplest choice of transition matrix is associated with the *natural* Markov chains. It assumes that the probability of transition from a node i is uniformly distributed over its neighbors, while it is zero for nodes not directly connected – i.e., not adjacent – to i, so that $p_{ij} = \frac{w_{ij}}{s_i}$. Notice that $\mathbf{s}$ is entrywise positive, as G is connected. The transition matrix can then be expressed as $\mathbf{P} = (\operatorname{diag}\mathbf{s})^{-1}\mathbf{W}$, with $\operatorname{diag}\mathbf{s}$ being the diagonal matrix whose diagonal entries are the elements of the vector $\mathbf{s}$. A discrete-time Markov chain is homogeneous if the one-step transition probabilities are invariant with respect to time. This implies that the k-step transition probabilities can be computed through the k-th power $\mathbf{P}^k$ of the transition matrix.

We also assume that the network G is non-bipartite, hence the Markov chain over G is ergodic, i.e it is always possible to go from any state to any other state and the chain does not exhibit periodic behavior. As a consequence, matrix $\mathbf{P}$ is irreducible, and there exists a unique stationary state solution of the eigenvalue equation $\pi(\infty) = \pi(\infty)\mathbf{P}$, of the form $\pi_i(\infty) = \frac{s_i}{\sum_{i=1}^{n} s_i} = \frac{s_i}{2S}$. Moreover, the sequence of matrices $\mathbf{P}^k$ converges to a rank-one matrix, that is $\mathbf{P}^\infty = \lim_{k\to\infty} \mathbf{P}^k$, and $\pi(\infty) = \pi(0)\mathbf{P}^\infty$, where $\mathbf{P}^\infty$ is the *infinite-time* transition matrix containing the transition probabilities of jumping from node i to node j in an infinite number of steps. The stationary probability flux ϕ_{ij} along the edge (i, j) is finally defined as the actual probability that the walker jumps from i to j in the stationary state, that is $\phi_{ij} = \pi_i(\infty)p_{ij}$.

Now, let $\mathscr{C}$ be a clustering of G and let C_α and C_β be two distinct elements in $\mathscr{C}$. Let us consider the global stationary probability flux from C_α to C_β: $\Phi_{C_\alpha C_\beta} = \sum_{i\in C_\alpha}\sum_{j\in C_\beta}\phi_{ij}$. By using this flux between clusters we can induce an aggregated process on a meta-network whose meta-nodes are clusters. This process is known in the literature as lumped Markov chain [23]. Since the random walker being in two different nodes at the steady state are incompatible events, the probability that it is in the meta-node C_α at the steady state is then given by $\pi_{C_\alpha}(\infty) = \sum_{i\in C_\alpha}\pi_i(\infty)$. Therefore, the transition probability from C_α to C_β is given by the conditional probability

$$p_{C_\alpha C_\beta} = \frac{\Phi_{C_\alpha C_\beta}}{\pi_{C_\alpha}(\infty)} = \frac{\sum_{i\in C_\alpha}\sum_{j\in C_\beta}\pi_i(\infty)p_{ij}}{\sum_{i\in C_\alpha}\pi_i(\infty)}. \tag{1}$$

These values represent the entries of the transition probability matrix of the lumped Markov chain. We denote the diagonal element $p_{C_\alpha C_\alpha}$ of this matrix as $\mathcal{P}_{C_\alpha}$ and we call it the persistence probability of the cluster C_α [23,29]. In the case of the natural Markov chain on a weighted undirected network, since $p_{ij} = \frac{w_{ij}}{s_i}$ and $\pi_i(\infty) = \frac{s_i}{2S}$, the persistence probability of the generic cluster C is given by

$$\mathcal{P}_C = \frac{\sum_{i,j\in C} w_{ij}}{\sum_{i\in C} s_i}. \tag{2}$$

Hence, the persistence probability for an undirected weighted network is the ratio between the total weight of the edges inside the community C and the total weight of the edges starting from one of the nodes in C and ending both inside and outside C. Finally, for the an unweighted graph, the persistence probability reduces to $\mathcal{P}_C = \frac{\sum_{i,j\in C} a_{ij}}{\sum_{i\in C} k_i}$.

2.1. Community detection through maximum persistence probability

The persistence probability can be used to measure the cohesiveness of a node subset and determine if it can be interpreted as a community. The first benchmark measure to compare it with is modularity, whose definition is $Q_C = \sum_{i,j\in C}\left(w_{ij} - \frac{s_i s_j}{2S}\right)$ [7]. Modularity compares the presence/absence of an edge between two nodes, that is, the term w_{ij}, with the probability of its existence under a null hypothesis, called the configuration model. The configuration model is a random graph with the same strength sequence as the original, but its connections are randomly rewired to destroy any form of endogenous community structure. Note that Q_C is

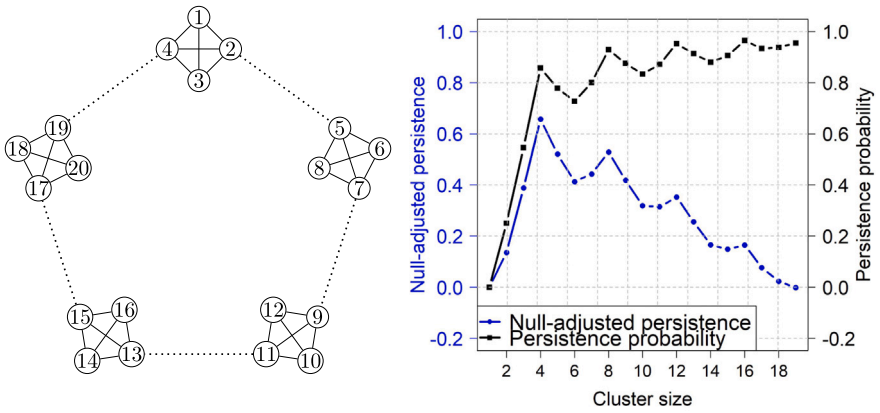

Fig. 1. Panel (a) : caveman graph. m_i internal links are represented by continuous lines; m_e external links are represented by dotted lines. Panel (b) : comparison between persistence probability (black line) and null-adjusted persistence (blue line).

defined for a single cluster; however, it can be extended as a global measure. For a given partition $\Pi = \{C_1, \ldots, C_q\}$, the modularity of Π is the (normalized) sum of the modularities of its clusters, and the community structure of a graph is the partition that solves the maximization problem:

$$\max_{\Pi} Q_{\Pi} = \max_{\Pi} \left[\frac{1}{2S} \sum_{C_\alpha \in \Pi} Q_{C_\alpha} \right]. \tag{3}$$

It is clear that persistence probability can play an analogous role to modularity in community detection. As in problem (3), network communities can be revealed by the partition that maximizes the sum of the persistences. Specifically, given a partition Π, the global persistence of Π is the sum of the persistence probabilities of its clusters:

$$\mathcal{P}_{\Pi} = \sum_{C_\alpha \in \Pi} \mathcal{P}_{C_\alpha}. \tag{4}$$

Then, the community structure of a graph can be revealed by the following maximization problem:

$$\max_{\Pi} \mathcal{P}_{\Pi} = \max_{\Pi} \sum_{C_\alpha \in \Pi} \mathcal{P}_{C_\alpha}. \tag{5}$$

Problem (5) can be formulated as a fractional integer programming problem, which can be reduced to mixed integer linear programming (the detailed formulation is reported in the Appendix A).

The preliminary problem of finding a *single* community that maximizes a slightly modified version of the persistence probability can be found in [28]. Exact and heuristic algorithms were given for that problem, and computational tests were carried out on artificial and real data. It was found that, for many test problems, the objective function studied in that paper tends to increase with respect to the size of the cluster, $|C|$. This global behavior has some troublesome consequences as high-sized clusters turn out to be the best candidate solutions. To amend this bias, the authors suggested plotting the maximum of the objective function as the cluster size $|C|$ varies, and selecting the local maxima rather than the global maximum, which corresponds to the trivial case $C = V$.

We now replicate a similar simulation using the persistence probability. Specifically, we refer to the caveman graph shown in Fig. 1, panel (a), which represents an instance of the small-world model proposed in [30]. The caveman graph in the figure is composed of five cliques of four nodes each, with each clique connected to two others by a single arc. The black line in the plot in panel (b) represents the persistence probability $\mathcal{P}_C$ as a function of the cluster size $|C|$. It can be seen that $\mathcal{P}_C$ tends to increase. As expected, there is a first local maximum for a cluster of size $|C| = 4$, followed by local maxima for clusters of sizes that are multiples of 4, which are the union of two or more cliques. Remarkably, the value of the local maxima increases with the size of the cluster. This could be problematic if we want to use the persistence probability to assess what the optimal community is. For example, for $|C| = 8$, $\mathcal{P}_C$ turns out to be larger than for $|C| = 4$, and this is clearly misleading, given the clique structure of the caveman graph. Actually, one may argue that, since problem (5) does not maximize the persistence of the single cluster but rather the sum of local persistence probabilities, local maxima of more than one clique cannot be optimal. The observation has a relevant consequence when one has to devise an efficient heuristic to solve problem (5). In fact, most heuristic algorithms for clustering find the optimal partition by joining the local optima: for example, the Louvain algorithm merges nodes into clusters in a greedy way, until the cluster modularity cannot be improved. However, this strategy is precluded for the persistence probability: the local maxima of the small clusters are almost always smaller than those of the large clusters. To overcome this problem, we then advise the necessity of adjusting the persistence probability in the formulation of the problem (5). We call this modified objective function *null-adjusted persistence* (NAP, for short). We will formally introduce this function in the next section, but for illustrative purposes, in Fig. 1 panel (b), we depict this new function (blue line). It can be seen that the function has a peak exactly where it ought to be, that is for the cluster of size 4.

2.2. Null-adjusted persistence function

Drawing inspiration from the definition of modularity, null-adjusted persistence compares the persistence probability to its expected value under the configuration model. In other words, it compares the probability to the value expected under the null hypothesis, which claims that there is no community structure.

Let $\mathbf{Z}$ be the matrix of elements $z_{ij} = \frac{s_i s_j}{2S}$.

Definition 1. The null-adjusted persistence $\mathcal{P}^\star_C$ of a cluster $C \subseteq V$ is:

$$\mathcal{P}^\star_C = \frac{\sum_{i\in C}\sum_{j\in C} w_{ij}}{\sum_{i\in C}\sum_{j=1}^{n} w_{ij}} - \frac{\sum_{i\in C}\sum_{j\in C} z_{ij}}{\sum_{i\in C}\sum_{j=1}^{n} z_{ij}} \tag{6}$$

where $\mathbf{W}$ is the weighted adjacency matrix of G and $\mathbf{Z}$ is the weighted adjacency matrix of the null model.

Note that $\mathcal{P}^\star_C$ is the difference between the actual persistence probability (from now on persistence, for short) of the nodes cluster C and the persistence of the same cluster under the null hypothesis of the configuration model, that is, assuming a random distribution of the edge weights among nodes. Let us note that the null model that enters formula (6) is the same one adopted in the definition of the classical modularity introduced by Newman and Girvan [7] and Newman [8]. This null model ensures that the observed community structure is not simply due to variations in node strength, and hence it provides a natural baseline for comparing network partitions.

In expression (6), $\sum_{i\in C}\sum_{j=1}^{n} w_{ij} = \sum_{i\in C} s_i$ represents the total strength of the cluster C, which we denote as S_C. The sum $\sum_{i\in C}\sum_{j\in C} w_{ij}$ is over the arc weights internal to the cluster, and it represents the internal strength of C, which we denote as $2S^{int}_C$. The difference between the total strength of cluster C and its internal strength represents the contribution of the arcs exiting C, so that we call $S^{ext}_C = S_C - 2S^{int}_C$ external strength of the cluster C. Moreover:

$$\frac{\sum_{i\in C}\sum_{j\in C} z_{ij}}{\sum_{i\in C}\sum_{j=1}^{n} z_{ij}} = \frac{\sum_{i\in C}\sum_{j\in C} s_i s_j}{\sum_{i\in C}\sum_{j=1}^{n} s_i s_j} = \frac{\left[\sum_{i\in C} s_i\right]^2}{\left[\sum_{i\in C} s_i\right]\cdot\left[\sum_{i=1}^{n} s_i\right]} = \frac{(S_C)^2}{(S_C)\cdot 2S} = \frac{S_C}{2S}.$$

Thus, the null-adjusted persistence $\mathcal{P}^\star_C$ of a cluster $C \subseteq V$ can be easily rewritten as follows:

$$\mathcal{P}^\star_C = \frac{2S^{int}_C}{S_C} - \frac{S_C}{2S}. \tag{7}$$

In the case of binary networks, $2S^{int}_C$ simply corresponds to twice the number of the internal edges, $2m_i$, and $S_C = 2S^{int}_C + S^{ext}_C$ corresponds to the sum of twice the number of internal arcs and the number of external of arcs of the cluster, $2m_i + m_e$. Therefore, expression (7) reduces to:

$$\mathcal{P}^\star_C = \frac{2m_i}{2m_i + m_e} - \frac{2m_i + m_e}{2m} \tag{8}$$

where m is the total number of arcs in the network.

From Eq. (6), which refers to a single cluster C, we can define the total null-adjusted persistence of the partition Π as $\mathcal{P}^\star_\Pi = \sum_{C\subseteq\Pi} \mathcal{P}^\star_C$. Problem (5) can be reformulated as follows:

$$\max_{\Pi} \mathcal{P}^\star_\Pi = \max_{\Pi} \sum_{C_\alpha\in\Pi} \mathcal{P}^\star_{C_\alpha}. \tag{9}$$

The null-adjusted persistence of the partition Π and the persistence of the same partition are related by the following:

Proposition 1. *Let $\mathcal{P}^\star_\Pi$ be the total null-adjusted persistence of the partition Π, and $\mathcal{P}_\Pi$ the total persistence of the same partition, then $\mathcal{P}^\star_\Pi = \mathcal{P}_\Pi - 1$.*

Proof. By Eq. (7), we have

$$\mathcal{P}^\star_\Pi = \sum_C \mathcal{P}^\star_C = \sum_C \left(\frac{2S^{int}_C}{S_C} - \frac{S_C}{2S}\right) = \sum_C \frac{2S^{int}_C}{S_C} - \frac{\sum_C S_C}{2S} = \mathcal{P}_\Pi - 1. \tag{10}$$

where the last equality holds observing that $\sum_C S_C = \sum_C \left(2S^{int}_C + S^{ext}_C\right) = 2S$. □

This result shows that functions $\mathcal{P}^\star_\Pi$ and $\mathcal{P}_\Pi$ computed on the same partition differ by a constant, and therefore a partition that maximizes one will also maximize the other. Thus, from the point of view of finding an optimal partition, they are equivalent. However, as will be shown in the next sections, they exhibit very different behaviors when computed on individual clusters. This has consequences on the way in which a heuristic to solve maximization problem should be designed.

3. Analytical results

3.1. Extreme values of the null-adjusted persistence

Here, we prove some analytical results about the null-adjusted persistence on weighted networks. First, we focus on the extreme values of the total null-adjusted persistence $\mathcal{P}^\star_\Pi$.

Proposition 2. *Let G be a network made up of $l > 1$ disconnected component C_α, $\alpha = 1, \dots, l$. The total null-adjusted persistence $\mathcal{P}^\star_\Pi$ with respect to the natural partition into the single components $\Pi = \{C_1, \dots, C_l\}$ is given by $\mathcal{P}^\star_\Pi = l - 1$.*

Proof. Since the network consists of $l > 1$ components that are connected internally but disconnected from each other, $2S^{int}_C = S_C$. The total null-adjusted persistence of the partition Π is then $\mathcal{P}^\star = \sum_C P_C{}^\star = \sum_C \left(\frac{2S^{int}_C}{S_C} - \frac{S_C}{2S} \right) = \sum_C \left(1 - \frac{S_C}{2S} \right) = l - (\frac{\sum_C S_C}{2S}) = l - 1$. □

Note that this result does not necessarily require that connected components have the same size.

Proposition 2 shows that $\mathcal{P}^\star_\Pi$ is unbounded from above, as $\mathcal{P}^\star_\Pi \to +\infty$ when $l \to +\infty$. The next result provides the minimum value for $\mathcal{P}^\star_\Pi$. At first, recall that a $k-$partite graph is a loopless graph whose vertices can be partitioned into k independent sets, that is sets of mutually non-adjacent vertices [31].

Proposition 3. *The total null-adjusted persistence $\mathcal{P}^\star_\Pi$ is bounded from below by $\mathcal{P}^\star_\Pi \geq -1$, and the minimum value -1 is attained by any multi-partite graph with respect to its canonical partition Π.*

Proof. The function $\mathcal{P}^\star_C = \frac{2S^{int}_C}{2S^{int}_C + S^{ext}_C} - \frac{2S^{int}_C + S^{ext}_C}{2S}$ is a strictly decreasing function of S^{ext}_C. The contribution of a cluster is minimized when $2S^{int}_C$ is zero and S^{ext}_C is as large as possible. Moreover, by Proposition 1, $\mathcal{P}^\star_\Pi = \sum_C \mathcal{P}^\star_C = \sum_C \mathcal{P}_C - 1$. Since $\mathcal{P}_C = \frac{2S^{int}_C}{2S^{int}_C + S^{ext}_C}$, if $S^{int}_C = 0$, for any C, that is the graph is any multi-partite graph, then we get $\sum_C \mathcal{P}^\star_C = -1$. Moreover, if $S^{int}_C > 0$, for some C, then $\sum_C \mathcal{P}^\star_C > -1$. This excludes the possibility that there may exist a partition with null-adjusted persistence less than -1. □

By previous results, we conclude that $-1 \leq \mathcal{P}^\star_\Pi < +\infty$.[1]

Note that greater persistence in one cluster compared to another does not guarantee higher null-adjusted persistence. For example, in the same notation as in Eq. (8), consider cluster C_1 with $m_i = 6$ and $m_e = 4$ in a binary graph with $m = 20$: this yields $\mathcal{P}_{C_1} = 0.75$ and $\mathcal{P}^\star_{C_1} = 0.35$. For a cluster C_2 in the same graph with $m_i = 8$ and $m_e = 4$, it is $\mathcal{P}_{C_1} = 0.80$ and $\mathcal{P}^\star_{C_1} = 0.30$. Therefore, increased persistence does not inherently correspond to higher null-adjusted persistence.

We now investigate the behavior of the null-adjusted persistence $\mathcal{P}^\star_C$ in comparison with the classical modularity function computed for a cluster C. For a weighted networks, the modularity Q_C of a given cluster $C \subseteq V$ can be expressed as (see [32])

$$Q_C = \frac{S^{int}_C}{S} - \left(\frac{S_C}{2S} \right)^2. \tag{11}$$

while, for a binary network, in the same notation as in Eq. (8), it becomes (see [15])

$$Q_C = \frac{m_i}{m} - \left(\frac{2m_i + m_e}{2m} \right)^2, \tag{12}$$

where the first term corresponds to the internal edge density and the second one to the expected edge density in the null model. For the sake of simplicity let us refer to the binary case in Eqs. (8) and (12). Studied as functions of the single variable m_i (fixing values m and m_e), they show quite similar behaviors. Indeed, both vanish at the same values of m_i, that is $\frac{m-m_e}{2} \pm \frac{1}{2}\sqrt{m(m-2m_e)}$. However, they have a maximum at different values of m_i. Specifically, the local maximum of $\mathcal{P}^\star_C$ is attained in $\hat{m}_i = \frac{1}{2}\left(\sqrt{2mm_e} - m_e \right)$ and it is equal to $\mathcal{P}^\star_C(\hat{m}_i) = 1 - \sqrt{\frac{2m_e}{m}}$. Conversely, the maximum for Q_C is attained in $\hat{m}_i = \frac{1}{2}\left(m - m_e \right)$ and it is equal to $Q_C(\hat{m}_i) = \frac{1}{4}\left(1 - \frac{2m_e}{m} \right)$. We conclude that while modularity Q_C is symmetric with respect to its maximum, persistence $\mathcal{P}^\star_C$ is not. The behavior of the two functions Q_C and $\mathcal{P}^\star_C$ with respect to the variable m_i is depicted in Fig. 2.

The different maxima for the two functions provide an interesting insight into our task. Indeed, if we use Q_C or $\mathcal{P}^\star_C$ to find network communities, all else being equal, the maximum of the two measures is attained with a different cluster size: the null-adjusted persistence will give more weight to clusters with fewer internal arcs than modularity. Therefore, if the network contains small-sized communities, the null-adjusted persistence is more appropriate than modularity to reveal its mesoscale structure and to bring out this structure at higher resolution. Finally, we stress that, by contrast, the persistence probability $\mathcal{P}_C$ is a monotonically increasing function with respect to m_i, and thus exhibits a behavior that, on the individual cluster, makes it not comparable to modularity.

In the next section, we highlight some further features of the null-adjusted persistence.

3.2. Scaling behavior of the null-adjusted persistence

In [15], the authors point out that the modularity exhibits the so-called sensitivity to satellites property: it identifies a clique as a natural cluster, but in presence of a clique with l leaves (precisely, satellites) the optimal Q_Π is attained for the clustering Π formed by l clusters. We can observe the same behavior with null-adjusted persistence, as the following example shows. Let us consider the

[1] We recall that the minimum value of the total modularity function $Q_\Pi = \sum_C Q_C$ is $-\frac{1}{2}$ and the minimum is attained only by a bipartite graph, when $Q_C = -\frac{1}{4}$ for both the clusters in the natural bipartition of the network G.

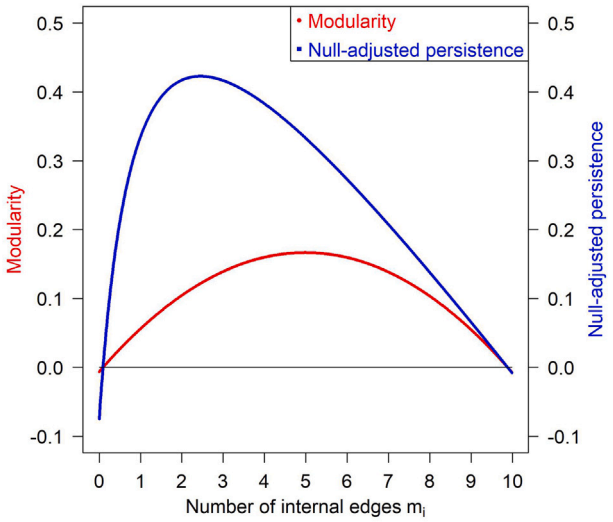

Fig. 2. Q_C (red line) and $\mathcal{P}_C^\star$ (blue line) as functions of m_i. For the sake of representation, m_e and m are set to values $m_e = 2$ and $m = 12$. Both functions vanish at the same values $m_i = 5 \pm 2\sqrt{6}$. Conversely, the modularity has a maximum for $m_i = 5$, whereas the null-adjusted persistence attains the maximum at $m_i = 2\sqrt{3} - 1 = 2.4641$.

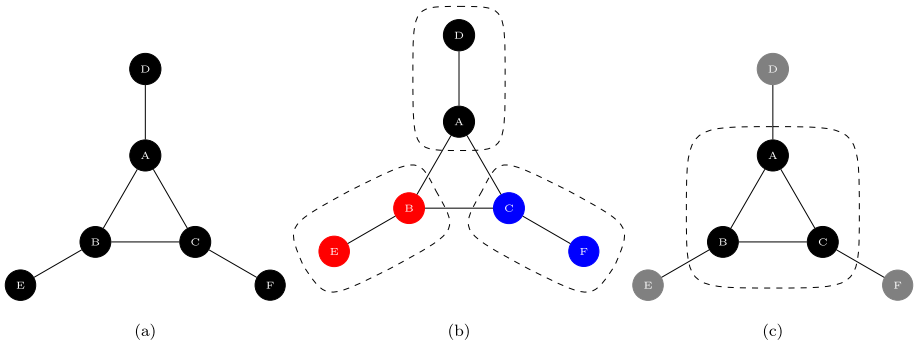

Fig. 3. Connected network G formed by the inner clique K_3 and three leaves (panels (a) and (c)). Optimal partition of the network G according to modularity and null-adjusted persistence (panel (b)).

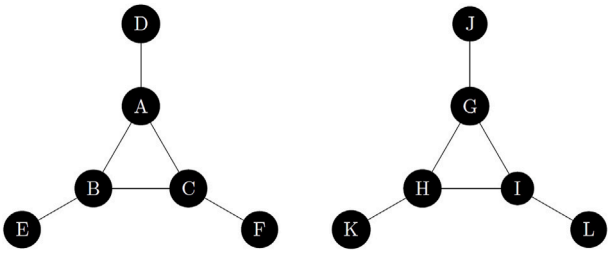

Fig. 4. Disconnected network G formed by two cliques K_3 with leaves.

complete clique K_3 with leaves, represented in Fig. 3, panel (a). Both modularity and null-adjusted persistence disaggregate the graph into three clusters, each containing a leaf as shown in panel (b), and do not preserve the inner clique unit in panel (c).

However, if we consider a non-connected network G originally composed by two identical and disjoint cliques K_3 with leaves, (see Fig. 4), the modularity identifies an optimal partition that keeps each clique in a single cluster containing the satellites as well. This result conflicts with what was previously found, where each clique was split into three components (see Fig. 5). In other words, as also [15] points out, modularity does not exhibit a scale-invariant behavior. Conversely, for the null-adjusted persistence, the optimal partition in which each clique is divided into three components is preserved even if we double the clique, as shown in Fig. 6, suggesting that the null-adjusted persistence *is* scale invariant.

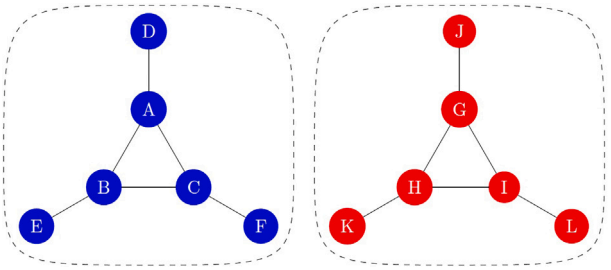

Fig. 5. Optimal partition Π of the disconnected network G formed by two cliques K_3 with leaves, according to modularity $Q(\Pi)$).

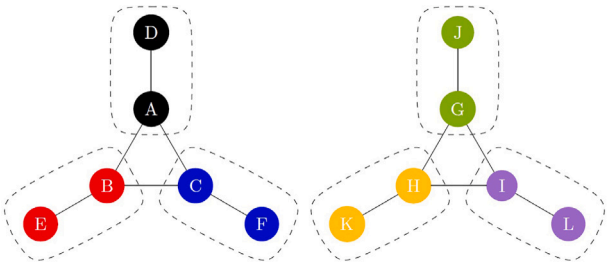

Fig. 6. Optimal partition of the two cliques K_3 with leaves according to the null-adjusted persistence $\mathcal{P}^{\star}_{\Pi}$.

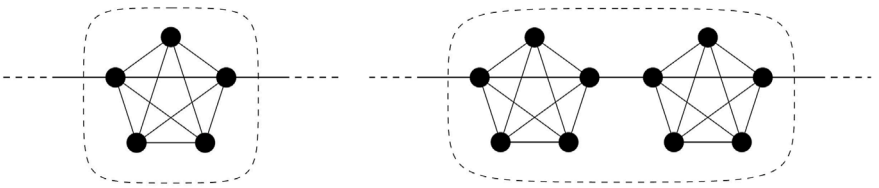

Fig. 7. Clusters in partition Π_1 (left) and in Π_2 (right).

3.3. The resolution limit

As pointed out above, modularity has an intrinsic dependence on scale, which limits the number and size of modules it can detect. We show in the next result that the null-adjusted persistence does not suffer from this limitation. To this end, it is sufficient to consider the binary case.

Proposition 4. *Let $G = (V, E)$ be a connected graph of $l > 1$, l even, identical cliques C connected in a circle, in such a way that each pair of adjacent cliques is connected by a single edge. Let Π_1 be the partition of G into the l cliques and Π_2 be the partition of G into the $\frac{l}{2}$ pairs of adjacent cliques. Then $\mathcal{P}^{\star}_{\Pi_1} > \mathcal{P}^{\star}_{\Pi_2}$.*

Proof. Let h be the number of arcs inside each clique C and consider the partition Π_1. In this case, $m_i = h$, $m_e = 2$ and $m = l(h+1)$. By formula (8) we obtain, for each cluster C,

$$\mathcal{P}^{\star}_{C} = \frac{2h}{2h+2} - \frac{2h+2}{2l(h+1)} = \frac{h}{h+1} - \frac{1}{l}.$$

The total null-adjusted persistence of the partition Π_1 is then $\mathcal{P}^{\star}_{\Pi_1} = \frac{hl}{h+1} - 1$.

Consider now the partition Π_2: $m_i = 2h+1$, $m_e = 2$ and $m = l(h+1)$. By Formula (8), we obtain, for each cluster C,

$$\mathcal{P}^{\star}_{C} = \frac{2(2h+1)}{2(2h+1)+2} - \frac{2(2h+1)+2}{2l(h+1)} = \frac{2h+1}{2h+2} - \frac{2}{l}.$$

The total null-adjusted persistence of the partition Π_2 is then $\mathcal{P}^{\star}_{\Pi_2} = \frac{l}{2}\left[\frac{2h+1}{2h+2} - \frac{2}{l}\right] = \frac{(2h+1)l}{4(h+1)} - 1$. Since the inequality

$$\frac{hl}{h+1} - 1 > \frac{(2h+1)l}{4(h+1)} - 1$$

equals $h > \frac{1}{2}$, which is satisfied for any l, we can conclude that $\mathcal{P}^{\star}_{\Pi_1} > \mathcal{P}^{\star}_{\Pi_2}$. □

Clusters belonging to partitions Π_1 and Π_2 of Proposition 4 are represented in Fig. 7.

A similar inequality for modularity function has been obtained in [15]. In particular, under the same hypothesis – i.e., considering the same partitions Π_1 and Π_2 – the authors show that $Q_{\Pi_1} > Q_{\Pi_2}$ if $l < \sqrt{2m}$. We provide here an alternative proof. In the clustering Π_1, for each cluster C, $Q_C = \frac{h}{l(h+1)} - \frac{1}{l^2}$ so that the total modularity is $Q_{\Pi_1} = l\left[\frac{h}{l(h+1)} - \frac{1}{l^2}\right] = \frac{h}{h+1} - \frac{1}{l}$. In the second clustering Π_2, for each cluster C, $Q_C = \frac{2h+1}{l(h+1)} - \frac{4}{l^2}$ so that the total modularity is $Q_{\Pi_2} = \frac{l}{2}\left[\frac{2h+1}{l(h+1)} - \frac{4}{l^2}\right] = \frac{2h+1}{2h+2} - \frac{2}{l}$. Therefore, the clustering Π_1 has higher modularity than the clustering Π_2, that is $Q_{\Pi_1} > Q_{\Pi_2}$, if

$$\frac{h}{h+1} - \frac{1}{l} > \frac{2h+1}{2h+2} - \frac{2}{l}$$

solved for $l < 2(h+1)$. Since $m = l(h+1)$, this condition equals $l < \frac{2m}{l}$, equivalent to $l < \sqrt{2m}$.

It is worth noting that this result is dependent on the number of cliques l and their size h. The modularity is able to discriminate cliques containing h arcs only if the number l of cliques does not exceed $2(h+1)$, and it fails when the number of cliques becomes large enough compared to the number of arcs contained in each clique. By contrast, the total null-adjusted persistence is able to discriminate cliques in the network regardless of their number, and, in the end, regardless of the size m of the network itself, overcoming the resolution limit typical of the modularity function. Although Proposition 4 refers to an extremely simplified and idealized graph, it nevertheless describes an intrinsic difference between the two objective functions, as will be highlighted in a very general context in the next subsection.

3.4. Merging clusters

A critical consideration in community detection is evaluating the benefit of merging clusters. Merging is beneficial if it improves the objective function. In this section, we quantify the gain or cost, in terms of null-adjusted persistence, in merging two distinct clusters. Let C_1 and C_2 be two clusters with internal strengths $2S_{C_1}^{int}$ and $2S_{C_2}^{int}$ and external strengths $S_{C_1}^{ext}$ and $S_{C_2}^{ext}$. The inter-cluster strength $S_{C_{1,2}}^{ext}$ is the sum of the weights of the arcs connecting the two clusters, that is the intensity of their mutual connection. The difference between the null-adjusted persistence of the merged cluster C and the sum of those of the two separate clusters C_1 and C_2, $\Delta\mathcal{P}_C^\star = \mathcal{P}_C^\star - \left(\mathcal{P}_{C_1} + \mathcal{P}_{C_2}\right)$, quantifies the merging gain or cost, and allows us to provide a threshold above which the merging operation is convenient, as we show in the following:

Proposition 5. *The null-adjusted persistence is increased by merging two clusters, i.e., $\Delta\mathcal{P}_C^\star > 0$, if and only if the inter-cluster strength satisfies*

$$S_{C_{1,2}}^{ext} > \frac{S_{C_2}}{S_{C_1}} S_{C_1}^{int} + \frac{S_{C_1}}{S_{C_2}} S_{C_2}^{int}. \tag{13}$$

Proof. Let $2S_C^{int}$ and S_C^{ext} be the internal and external strengths of the merged cluster C, respectively. We have: $S_C^{int} = S_{C_1}^{int} + S_{C_2}^{int} + S_{C_{1,2}}^{ext}$ and $S_C^{ext} = S_{C_1}^{ext} + S_{C_2}^{ext} - 2S_{C_{1,2}}^{ext}$. Therefore

$$\begin{aligned}\Delta\mathcal{P}_C^\star &= \mathcal{P}_C^\star - \left(\mathcal{P}_{C_1} + \mathcal{P}_{C_2}\right) \\ &= \frac{2\left(S_{C_1}^{int} + S_{C_2}^{int} + S_{C_{1,2}}^{ext}\right)}{\left(2S_{C_1}^{int} + S_{C_1}^{ext}\right) + \left(2S_{C_2}^{int} + S_{C_2}^{ext}\right)} - \frac{2S_{C_1}^{int}}{2S_{C_1}^{int} + S_{C_1}^{ext}} - \frac{2S_{C_2}^{int}}{2S_{C_2}^{int} + S_{C_2}^{ext}}\end{aligned}$$

By solving $\Delta\mathcal{P}_C^\star > 0$ with respect to $S_{C_{1,2}}^{ext}$ the inequality (13) immediately follows. □

Proposition 5 implicitly states that the threshold on the value of the inter-cluster strength $S_{C_{1,2}}^{ext}$ beyond which it becomes convenient to merge the two clusters does not depend on the total strength S of the overall network. This further supports the scale invariance of the null-adjusted persistence.

As a consequence of proposition 5, in a binary network, if C_1 and C_2 are two clusters with internal arcs $m_i^{(1)}$ and $m_i^{(2)}$, external arcs $m_e^{(1)}$ and $m_e^{(2)}$, and $m_e^{(12)}$ arcs in between connecting them, then the null-adjusted persistence is increased by merging the two clusters if and only if

$$m_e^{(12)} > \frac{2m_i^{(2)} + m_e^{(2)}}{2m_i^{(1)} + m_e^{(1)}} m_i^{(1)} + \frac{2m_i^{(1)} + m_e^{(1)}}{2m_i^{(2)} + m_e^{(2)}} m_i^{(2)}. \tag{14}$$

Note that the same does not hold for modularity. In fact, if we compute the merging cost for modularity, that is we solve $\Delta Q_C = Q_C - \left(Q_{C_1} + Q_{C_2}\right) > 0$, again restricting to the binary case, we get

$$m_e^{(12)} > \frac{\left(2m_i^{(1)} + m_e^{(1)}\right)\left(2m_i^{(2)} + m_e^{(2)}\right)}{2m} \tag{15}$$

which depends on the size m of the network, confirming the scale-dependent behavior.

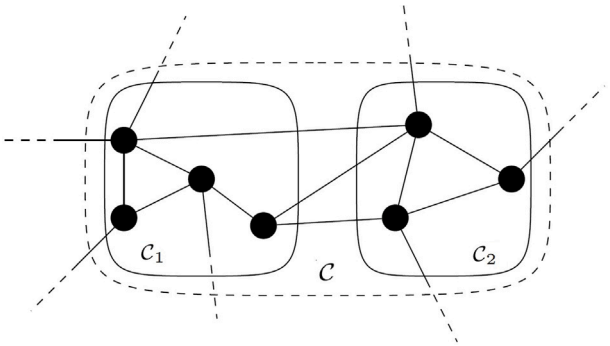

Fig. 8. Illustrative example of the merging condition of two clusters.

Fig. 8 illustrates the result of Proposition 5 and its corollary Eq. (15). In this case $m_i^{(1)} = 4$, $m_i^{(2)} = 3$, $m_e^{(1)} = 7$, $m_e^{(2)} = 6$ and $m_e^{(12)} = 3$, and we would need at least $m_e^{(12)} = \lceil \frac{139}{20} \rceil = \lceil 6.95 \rceil = 7$ arcs between C_1 and C_2 to conveniently merge them into a single cluster C. Note that we cannot measure the merging cost for modularity without knowing the number of arcs in the network, m.

Proposition 5 further justifies why null-adjusted persistence does not suffer from the same resolution limit as modularity. Indeed, under the hypotheses of Proposition 4, $m_i = h$, $m_e = 2$ for both cliques and $m = l(h+1)$, so that in the case of the null-adjusted persistence, $m_e^{(12)} > 2h$. This shows how difficult it is to merge two cliques when using the null-adjusted persistence and how this becomes increasingly difficult as the size of the two cliques increases. On the contrary, using modularity, $m_e^{(12)} > \frac{2(h+1)}{l}$. The threshold (15) for modularity becomes less than 1 when $l > 2(h+1)$ so that the presence of a single link between the two clusters is enough to make it convenient to merge them. In Appendix B, we prove a final analytical result related to the change in null-adjusted persistence caused by moving a single node from one cluster to another, as this is relevant to the algorithm to be discussed below.

4. Testing null-adjusted persistence optimization on benchmark and real networks

In this section, we test the null-adjusted persistence as a quality function to detect communities first on two classes of simulated graphs and then on two real-world networks. The scope is to assess its ability to capture the mesoscale structure, particularly in synthetic network classes that offer controlled environments, where the underlying community structure is known, and on ground-truth data, enabling a rigorous testing of how effectively algorithms can identify and distinguish these communities.

As can be seen in the Appendix A, solving for maximum persistence through integer fractional programming is only viable for small graphs. Therefore, a heuristic algorithm must be devised for our tests.[2] For comparison purposes, we adapted the classical Louvain algorithm for modularity [10] to the new objective function, and, following the tradition that most community algorithms have the name of a city, we will call it *Milano algorithm*. Actually, the principle by which two clusters merge is the same for both methods; the only difference is the use of null-adjusted persistence (NAP) instead of modularity as objective function. The detailed description of the proposed algorithm is reported in the Appendix C. The algorithm has been implemented in C++ and the R package **persistence** has been developed for the computation of the null-adjusted persistence.

The main results of our analysis can be summarized as follows:

- We tested NAP on synthetic Caveman and Lancichinetti-Fortunato-Radicchi (LFR) benchmark graphs. We compared its performance with modularity (via Louvain/Leiden) and Infomap. The results showed that NAP overcomes the resolution limit of modularity, correctly detecting the ground-truth communities in Caveman networks. NAP also outperformed Infomap on LFR networks across the ARI and NMI metrics.
- To assess its real-world application utility, the method was applied to both weighted and binary networks. The results highlighted that NAP identifies finer-grained community structures than modularity does, offering higher-resolution partitions. In particular:
  - Using Enron's internal email network (139 employees and 996 connections), NAP detected 42 communities. This is compared to six to seven communities with modularity and 61 communities with Infomap. This finer structure reveals micro-level clusters within functional groups, demonstrating NAP's sensitivity to hierarchical structures in weighted communication data.
  - Using Facebook ego-network dataset of 4039 nodes and 88,234 ties, NAP allows us to extract the separate clusters to which each ego node belongs from the 16 large communities identified by modularity via the Louvain algorithm. This method splits the giant clusters of modularity within them, preserves ego-centric structures and reveals nested communities missed by modularity.

[2] We emphasize that our goal here is to obtain an optimal partition that maximizes null-adjusted persistence. Thus, describing the best heuristic algorithm to achieve this optimal partition is beyond the scope of this paper and will be deferred to future work.

4.1. Community detection on benchmark networks

We begin by testing NAP on two different classes of benchmark networks. The first one is the class of the caveman graphs (see [30]), progressively modified through random rewiring. These graphs are built from cliques – fully connected subgraphs – that represent tightly-knit communities or *caves*, by shifting one edge from each clique and using it to connect to a neighboring one. These cliques are loosely connected through sparse inter-community arcs and, hence, exhibit clear and easily identifiable community boundaries that are progressively hidden by rewiring. Caveman graphs are particularly useful for testing algorithms under idealized conditions where communities are densely connected internally but sparsely linked externally. The second one is the class of simulated networks generated according to the methodology proposed by Lancichinetti et al. [33]. This procedure generates networks that are as close as possible to real networks, which are often characterized by heterogeneous distributions of node degree. The LFR networks provide a more complex and realistic scenario than caveman graphs. Designed to mimic the structure of real-world networks, LFR graphs feature power-law degree distributions and ground-truth community structures. A key factor of these graphs is the mixing parameter $\mu \in (0, 1]$, which determines the fraction of edges outgoing from a given node and pointing to nodes outside the community.

For both classes of networks, we compare the performance of the null-adjusted persistence computed by the Milano algorithm against the other mentioned methodologies for community detection. In particular, for caveman networks, NAP is compared with the classical modularity function via both Leiden and Louvain algorithms. For LFR benchmark networks, we compare NAP with the Map equation, the objective function of the Infomap algorithm. We recall that the Infomap idea is based on optimizing an objective function (the Map equation), which finds clusters by minimizing the length of a random walker's path on the network. The objective is to identify clusters in which the random walker remains for an extended period, making Infomap more similar to NAP than modularity. The efficiency of Infomap in describing the information flow also makes it sensitive to small clusters. As a result, it produces more clusters than modularity-based algorithms. Also the Milano algorithm tends to exhibit this property, intercepting small clusters and producing more fragmented partitions.

We consider a range of conditions and evaluate the performance of the different methods using two well-known measures of partition similarity: the Adjusted Rand Index (ARI) and the Normalized Mutual Information (NMI) ([34,35]; see Appendix D for the definitions of ARI and NMI). ARI index ranges from -1 to 1 and NMI index ranges from 0 to 1. For both indices, a value of 0 indicates random node assignment to a community, while a value of 1 indicates a perfect match between the partitions.

Fig. 9 depicts results for caveman graphs and compares the indices as a function of edge rewiring. Specifically, the initial configuration – the classical caveman structure – is progressively modified according to a rewiring mechanism that involves moving a randomly selected link while preserving the degree distribution. The structure of caveman graphs represents an ideal configuration for empirically verifying how our objective function overcomes the resolution limit problem that affects modularity. Panels (a) and (b) refer to a network of 60 nodes, distributed into 12 communities, of 5 nodes each. Results show that initially both algorithms identify the ground-truth partition, confirming that both the objective functions can intercept the natural community structure underlying the graph. Results differ only after the rewiring mechanism has produced substantial link shuffling, which makes it possible that the underlying structure is no longer well-defined. Panels (c) and (d) refer to a network of 120 nodes distributed into 24 communities, of 5 nodes each. In this case, the algorithms identify different partitions from the beginning, even before any rewiring has been done. Indeed, the modularity function fails in identifying the ground-truth community structure, as the number of communities leads to exceed the threshold (15), and it becomes convenient to merge adjacent clusters. Conversely, the null-adjusted persistence is still able to identify the community structure, as was theoretically proven in Proposition 4.

Concerning the second class of graphs, we generated LFR networks of 1000 nodes, where the degree and community-size distributions follow power laws with exponents $\tau_1 = 2$ and $\tau_2 = 2$, respectively, for mixing parameter μ that varies from 0.1 to 0.6. Note that $\mu = 0.5$ marks the threshold beyond which communities are no longer defined in the strong sense [33]. Fig. 10 illustrates the results. As shown, across both evaluation indices – ARI and NMI – the proposed null-adjusted persistence achieves higher values than Infomap, thereby confirming its improved ability to recover the ground-truth community structure.

4.2. Application to real networks

In this section, we apply our proposal to real networks. Our case studies focus on both weighted and binary networks. To this end, we present two applications to social networks, the Enron network and the Facebook network. The primary objective is to compare the outcomes of NAP optimization with other clustering methodologies, particularly those based on modularity and the Map equation.

4.2.1. Weighted case study: Enron network

The first network is based on the Enron company's email communications [3] The original dataset includes all email communications between Enron employees. The nodes are the email addresses, and there is an undirected edge (i, j) if an address i sent at least one email to address j (or vice-versa). The Federal Energy Regulatory Commission originally made this data public and posted it online during its investigation into the Enron scandal, which caused the bankruptcy of the Enron Corporation (see [36]). The original dataframe included emails involving people outside the Enron organization, such as friends or relatives of the employees, that are not relevant to the purpose of the Enron community detection. Therefore, we considered only the internal emails, focusing the analysis on just the communications between Enron employees. As a result, we considered the interactions between 139 employees with complete mailbox records. The resulting network is weighted, undirected, and connected with 139 nodes and 996 weighted edges (weights are

[3] Data is available at https://snap.stanford.edu/data.

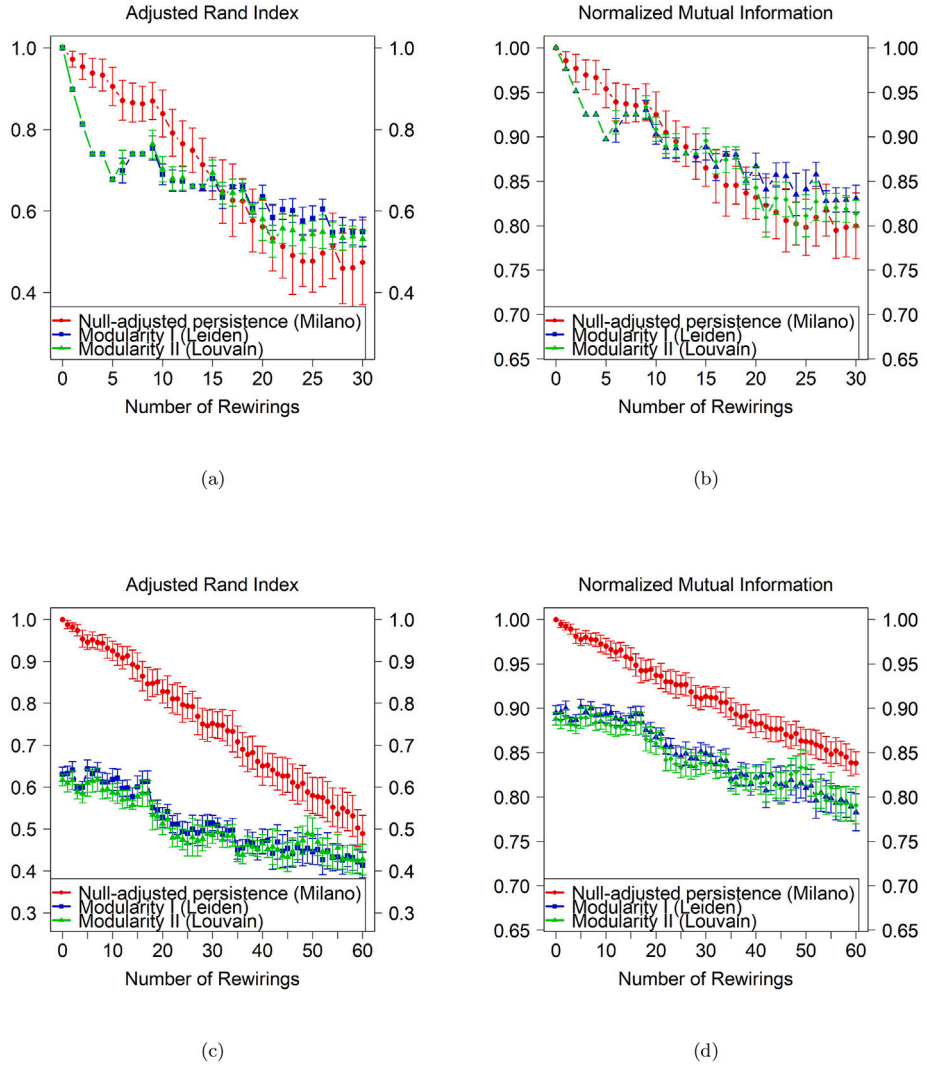

Fig. 9. Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) for caveman graphs with 12 cliques of 5 nodes (panels (a) and (b)) and 24 cliques of 5 nodes (panels (c) and (d).) Each curve represents the average on 40 different runs and vertical bars are the standard deviations. The red plot was obtained using the partition provided by null-adjusted persistence via the Milano algorithm, while the blue and green curves were obtained using the partition produced by modularity via the Leiden and Louvain algorithms, respectively.

the number of emails exchanged). Other structural properties of the Enron network are that the average node degree is approximately 7.2 and the overall density is about 0.1. Different centrality metrics identified the most influential nodes in the company, such as John Arnold, Paul Allen, John Lavorato, Geoff Storey, Barry Tycholiz, Kenneth Lay (CEO), and Susan Scott, among others. This highlights the central role of both middle managers and executive-level leaders in disseminating information within the company (see [37]).

We applied both the Louvain and Leiden algorithms for modularity, the Milano algorithm for NAP, and the Infomap algorithm for the Map equation. Results are synthesized in Table 1, where we report the number of communities and the maximum size of the clusters detected by each method. The Louvain and Leiden algorithms for modularity generate a partition into 7 and 6 communities, respectively, while the Milano algorithm generates a higher number of communities, that is, a partition of 42 clusters. This finding is consistent with the theoretical properties of null-adjusted persistence and modularity. As shown in Section 3, optimal modularity depends on network size, resulting in large communities. Conversely, null-adjusted persistence does not depend on network size, and, therefore, we expect smaller communities. It is interesting to note that Infomap, which is based on the Map equation, outputs a number of communities equal to 61, which is even larger than that produced by NAP. Actually, the Map equation is elaborated on probabilistic properties similar to those of the NAP, and the result suggests scalability properties for the Map equation similar to those of the null-adjusted persistence. The maximum sizes of the clusters detected by Louvain and Leiden are 26 and 43, respectively. They are consistent with the structure of the algorithms, as Leiden has been developed to improve Louvain by aggregating communities rather than separating them. The maximum sizes of Infomap and Milano are 4 and 8, respectively. Indeed, Infomap could improve

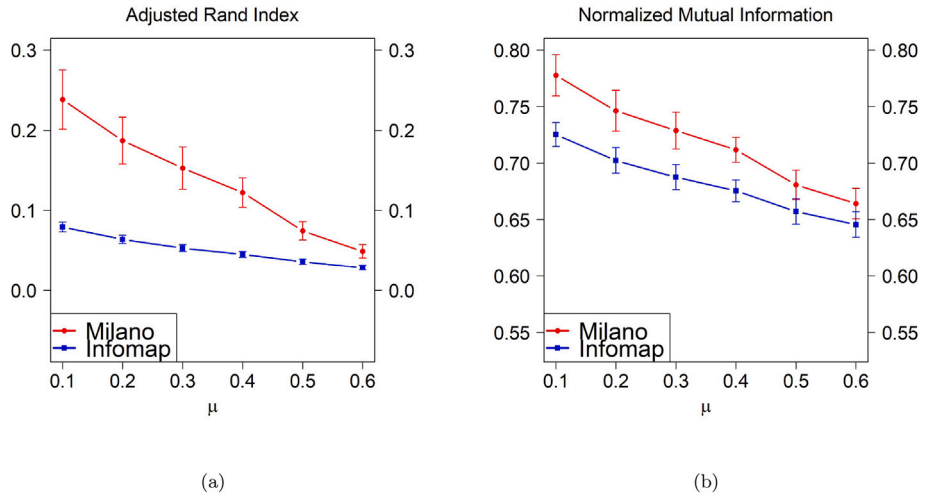

Fig. 10. Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) for LFR graphs with 1000 nodes and average degree 15. Each curve represents the average on 800 different runs, and vertical bars are the standard deviations. The red curve corresponds to the partition obtained by Milano algorithm, while the blue curve corresponds to the partition produced by Infomap.

Table 1
Comparison between community structures of the Enron company.

Heuristic methods	Number of communities	Maximum size
Louvain	7	26
Leiden	6	43
Infomap	61	4
Milano	42	8

dyads or triangles only marginally, as the maximum cluster size is 4. Conversely, the Milano algorithm yields a cluster up to 8 nodes, along with a cluster of size 7 and two of size 6. The community of 8 nodes identifies the cluster of the directors and employees of Enron Transportation Services (ETS, Enron's pipeline group and transportation services), while the community of 7 nodes reveals the cluster of the most important managers, such as D. Delainey, J. Lavorato, and S. Beck. Interestingly, CEO Ken Lay is in a dyad with Tom Donohoe, the financial trader. In conclusion, the clusters identified through the NAP are consistent with the roles of the employees in the organization.

4.2.2. Binary case study: facebook network

The second network is the Facebook social network described in [38]. The network results from the merging of ten ego-networks of friendship relationships. An ego-network is a subgraph consisting of a focal node (ego), its directly connected neighbors (alters), and all edges between these alters, or in other words, the subgraph induced by the ego and all its neighbors. Specifically, the ego-networks are merged by including all nodes and edges belonging to each ego's "social circle" in a single graph. This includes the ego users themselves, their friends (alters), the connections between egos and their alters, and the connections among the alters. The resulting network contains 4039 nodes and 88,234 connections. Each node represents an anonymized Facebook user from one of the ten friends' lists, and each edge corresponds to a friendship between two Facebook users. The network is unweighted and undirected because edges in Facebook encode only reciprocal ties.[4] Ego nodes typically have a high degree and aggregate their own friend lists, giving the network a structure of communities at a mesoscopic level, hence this network could serve as a preliminary ground truth for community detection. The ten egos are the nodes with the greatest number of connections, while their friends are in an ancillary position because some of their connections are missing. Consequently, we can expect the methods to identify at least ten distinct clusters, one for each ego.

The modularity-based Louvain and Leiden algorithms identified 16 and 17 communities, respectively. These communities will be referred to as *M-communities*. The NAP-based Milano algorithm identified 1016 clusters, which will be referred to as *P-communities*. The size of the clusters (M-Size) identified by both Louvain and Leiden methods ranges from a minimum of 19 nodes to a maximum of 548. In contrast, the size of the clusters identified by the Milano algorithm (P-Size) varies from a minimum of 2 to a maximum of 52 nodes. Specifically, the Milano algorithm identified 596 dyads and triads and 48 clusters with more than 10 nodes.

Data reported in Table 2 refer to the communities in which the egos are detected. A striking result of the methods is that the Milano algorithm could detect the ten separate clusters, corresponding to the ten egos, while Leiden found only eight communities,

[4] The dataset is available at this link: Stanford Facebook Dataset.

Table 2
Comparison of the community structure of Facebook network, as identified by the Louvain and Milano algorithms.

Ego	Deg	M-Comm	M-Dens	M-Size	P-Comm	P-Dens	P-Size
#1	347	#1	0.047	350	#1	0.133	15
#108	1045	#3	0.169	446	#35	0.167	12
#349	229	#2	0.066	430	#109	0.500	4
#415	159				#123	0.200	10
#687	170	#8	0.094	206	#218	0.667	3
#699	68				#197	0.333	6
#1685	792	#11	0.061	535	#443	0.333	6
#1913	755	#4	0.128	423	#500	0.250	8
#3438	547	#9	0.036	548	#843	0.143	14
#3981	59	#7	0.116	60	#997	0.250	8

as there are two communities containing two egos each. This result alone suggests that NAP could be a more accurate measure than modularity because ego nodes #349 and #415 are assigned to M-community #2, and nodes #687 and #699 are assigned to M-community #8. However, they correctly belong to four distinct P-communities. Additionally, important differences exist among the ego communities. We reported the main features of the M-communities and the P-communities: The first and second columns list the ego nodes (**Ego**), labeled with their number in the original network, and their degrees (**Deg**). The other columns report the density and the size of the communities to which the ego nodes belong, according to the Louvain algorithm for modularity (columns **M-dens** and **M-size**), and according to the Milano algorithm for NAP (columns **P-dens** and **P-size**). The internal density is computed by dividing the number of internal edges in a community by the maximum number of potential links in the cluster. Communities that do not contain ego nodes are omitted from the table.

For the eight communities in which both algorithms identified one ego node, the table shows that M-size is greater than P-size and P-dens is greater than M-dens. This suggests that NAP and Milano identify the ego communities as the egos' closest friends, who are also close to each other.

An informal but accurate analysis of all the other communities, that is those that do not contain an ego-network, revealed that most P-communities result from the additional partition of the same M-community. Conversely, very few P-communities are composed of nodes from distinct M-communities. Finally, the large difference in the number of clusters obtained by the two methods is an expected result due to the different nature of the objective functions. In a large scale-free network such as the one under analysis, modularity aggregates sparsely connected groups of nodes into giant communities according to its scaling behavior. Conversely, NAP filters out from the network only those clusters that are densely connected, breaking up the rest into a large number of small graphlets. However, the emerging clusters have a substantial meaning and highlight centers of aggregation of high density.

4.3. Comparative perspective with other methods

Several methods have been proposed to address the resolution limit of modularity. Among these methods, the persistence function corrected with the null model offers a different perspective, which can be understood by comparing it with existing approaches. For example, Infomap identifies communities by compressing random-walk trajectories, excelling at capturing flow patterns (see [22]). However, due to its lack of a null model, it cannot determine whether the observed retention exceeds what would be expected by chance. NAP overcomes this by benchmarking persistence against the configuration model, thereby providing a statistically interpretable measure of excess retention. Stochastic block models (SBMs; see [39,40]) take a generative route, assuming latent group assignments and probabilistic link patterns. While statistically elegant, they rely on model assumptions and face detectability thresholds; in contrast, NAP evaluates observed dynamics directly and yields community scores without hidden variables. Measures such as Surprise [41–43] shift attention to how improbable internal edge counts are under randomness, but their sensitivity can lead to overfragmentation in noisy or heterogeneous graphs. NAP avoids this pitfall by focusing not on edge counts alone but on the persistence of flows relative to null expectations. Similarly, Significance [43] quantifies deviations in internal density using information-theoretic measures, yet remains static in nature; on the contrary, NAP extends this idea by incorporating Markovian dynamics, identifying communities that retain information longer than chance would predict. Finally, OSLOM (Order Statistics Local Optimization Method; [44]) supports overlapping and statistically validated groups through local randomization tests, but depends on stochastic thresholds and repeated sampling, making results less reproducible. By contrast, NAP offers a deterministic, closed-form criterion that yields consistent partitions. Taken together, these distinctions position NAP as a theoretically grounded and practically robust alternative, uniquely capturing the dynamic excess of information retention relative to randomness, and thus well suited to uncovering fine-grained structure in dense or dynamically meaningful networks. Beyond null-model-based and walk-based criteria, other approaches to community detection have explored very different principles. Geometric methods built on discrete Ricci curvature use curvature flow to suppress boundary edges and thereby highlight modular structure [45], while subsampling frameworks such as SONNET assemble global partitions from overlapping local subnetworks to achieve scalability on massive graphs [46]. These methods provide valuable alternatives, yet they do not address the central issue considered here—measuring the strength of a community relative to

its expected behavior under a null model. In this respect, null-adjusted persistence offers a complementary advantage: by correcting observed persistence using its configuration-model expectation, it avoids size biases, achieves scale independence, and remains sensitive to smaller or highly cohesive communities that geometric or subsampling approaches may overlook.

5. Conclusions

This paper examines the relationship between persistence probability and the community structure of a weighted, undirected network. We introduce null-adjusted persistence, which compares persistence probability to the null configuration model. We adopt null-adjusted persistence as the objective function for the community detection optimization problem. This proposal incorporates features from persistence probability and modularity, offering significant advantages over both these functions. Indeed, it aligns with modularity-based approaches by separating observed persistence into contributions from the null model and deviations from it, thereby improving interpretability and integration with other network analysis methods. Its ability to take both positive and negative values allows us to distinguish cohesive clusters from those that are less cohesive than expected. This feature is particularly valuable in networks with strong degree heterogeneity, where this heterogeneity could otherwise produce misleading results. It also allows us to detect partitions with a higher resolution than modularity in large networks consisting of many medium or small communities. We then compared the partitions obtained using the Milano algorithm, developed specifically for the new objective function, with the results obtained using the classic Louvain, Leiden, and Infomap algorithms, highlighting the peculiarities of this objective function on real and simulated networks.

Our proposal outlines several directions for future research. For example, interpreting negative values of null-adjusted persistence values as indicators of anticommunity structure can enrich the analysis of mesoscale structures in complex networks. Extending our approach to temporal settings – more specifically, temporal baselines such as time-shuffled or activity-driven models – might be meaningful when the temporal dynamics of link formation carry important information. Another possible perspective is to test the stability of communities in the presence of outliers or missing data using node or edge samples to determine the most robust measure. Finally, developing more refined and optimized heuristics for searching for such communities with our objective function is the most pressing priority. With more sophisticated algorithmic tools, it will be possible to test the potential of null-adjusted persistence on large real networks and highlight its usefulness in improving the robustness and applicability of community detection methods.

CRediT authorship contribution statement

Alessandro Avellone: Writing – original draft, Validation, Software, Data curation. **Paolo Bartesaghi:** Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Stefano Benati:** Writing – original draft, Supervision, Software, Methodology, Funding acquisition, Conceptualization. **Christos Charalambous:** Writing – original draft, Validation, Methodology, Data curation. **Rosanna Grassi:** Writing – original draft, Supervision, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

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

Acknowledgments

A.A., P.B., S.B. and R.G. acknowledge financial support from the European Union – NextGenerationEU. Project PRIN 2022 “Networks: decomposition, clustering and community detection” code: 2022NAZ0365 – CUP H53D23002510006. PB and RG are members of the GNAMPA-INdAM group. CC has received funding from the European Union’s Horizon 2020 research and innovation program under the Marie Sklodowska-Curie Grant Agreement No. 101034403.

Appendix A. Mixed integer linear programming formulation of the optimal persistence partition

In the following, we report the formulation of problem (5) through a mixed integer linear programming. Note that, from Proposition (1), for any partition, the difference between the null-adjusted persistence and the persistence probability is a constant, therefore, any maximizer of the latter is also a maximizer of the former. Moreover, for the sake of clarity, we use the notation a_{ij} for the input parameters. These parameters can represent the presence/absence of an arc, that is, $a_{ij} = 1$ if $(i, j) \in E$, or the weight of the i, j-link, that is $a_{ij} = w_{ij}$. Suppose that V can be partitioned into $k = 1, \dots, n$ slots, then let the decision variable be $z_{ik} = 1$ if node i is assigned to slot k, 0 otherwise. Clearly, if for some k we have $z_{ik} = z_{jk} = 1$, then nodes i and j are in the same cluster k. If a slot k is empty, then it does not represent any cluster; therefore, in the optimal solution, many slots will be empty. The following objective function represents the persistence of a node-to-slot assignment that has to be maximized under the set of binary variables z:

$$\max_z \sum_{k=1}^{n} \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 2a_{ij}(z_{ik}z_{jk})}{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} [a_{ij}(z_{ik}z_{jk}) + a_{ij} \max\{z_{ik}, z_{jk}\}]} \tag{16}$$

In the objective function, the product $(z_{ik}z_{jk})$ is 1 if and only if both i and j are assigned to the same slot k, and therefore arc (i, j) is internal to the cluster represented by the slot k. The term $\max\{z_{ik}, z_{jk}\}$ is 1 if at least one between i and j is internal to the

slot k, and therefore either arc (i, j) is internal to the slot k, or is in the cut from the slot k to another slot. The numerator expresses the number of arcs internal to a cluster, counted twice; the denominator expresses the number of arcs internal to a cluster, plus the number of arcs exiting the cluster. The ratios of the objective function are written with the convention $0/0 = 0$.

A partition Π can be represented by many different node-to-slot assignments, simply by relabeling the slot representing a given cluster. Therefore, some constraints must be imposed to avoid multiple symmetric solutions (as they would increase the computational time exponentially). These constraints are:

$$\begin{aligned} &\sum_{k=i}^{n} z_{ik} = 1 \text{ for all } i \in V \\ &z_{ik} \leq z_{kk} \text{ for all } i \in V, k > i. \end{aligned} \tag{17}$$

The first kind of constraint requires that every node must be assigned to one slot, and that the index of the slot must be greater than or equal to the node index. The second kind of constraint imposes that node i can be assigned to a slot k only if k is not empty and node k has been assigned to slot k. Taken together, the two constraints impose that for a cluster $C_\alpha = \{i_1, \dots, i_r\}$, in which i_r is the greatest index, that is, $i_r > i_j$ for all $i_j \in C_\alpha, i_j$ the bin that contains/represents C_α can only be $k = i_r$. In fact, from the first constraint, i_r can be assigned only to slots with a label greater or equal to i_r, that is, it can be $z_{i_r,k} = 1$ only if $k \geq i_r$. Nevertheless, from the second constraint, we cannot have $z_{i_r,k} = 1$ for any $k > i_r$. Therefore, C_α can be represented only by $z_{i_r,i_r} = 1$. These constraints are important for numerical purposes to avoid symmetric solutions in branch&bound. They were previously used in [21,47]. Note that the optimal solution of (16) is made up of connected clusters. Indeed, by absurdity, if the optimal solution contained at least one bin representing an unconnected cluster, then it could be split into at least two connected components, improving the objective function, and thus contradicting its optimality.

The problem is not linear. However, it can be turned into a mixed integer linear programming problem with the appropriate constraints and linearization of the quadratic terms. The product terms of the objective function can be linearized as:

$$x_{ijk} = z_{ik} z_{jk} \iff \begin{cases} x_{ijk} \leq z_{ik} \\ x_{ijk} \leq z_{jk} \\ x_{ijk} \geq z_{ik} + z_{jk} - 1 \end{cases} \qquad \forall i, j \in V.$$

The max term of the objective function can be linearized as:

$$y_{ijk} = \max\{z_{ik}, z_{jk}\} \iff \begin{cases} y_{ijk} \geq z_{ik} \\ y_{ijk} \geq z_{jk} \\ y_{ijk} \leq z_{ik} + z_{jk} \end{cases} \qquad \forall i, j \in V.$$

The objective function now reads:

$$\max_{x,y} \sum_{k=1}^{n} \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 2a_{ij} x_{ijk}}{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} [a_{ij} x_{ijk} + a_{ij} y_{ijk}]}.$$

Note that the objective function is increasing with respect to variables x_{ijk} and decreasing with respect to y_{ijk}, therefore some of the above linearization terms are not necessary to characterize the optimal solution.

Now the objective function is the sum of ratios between two linear terms, then it can be linearized using the Charnes-Cooper linearization. Introduce a new variable u_k, defined as:

$$u_k = \frac{1}{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} [a_{ij} x_{ijk} + a_{ij} y_{ijk}]}$$

so we obtain the constraint:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} [a_{ij} x_{ijk} u_k + a_{ij} y_{ijk} u_k] = 1.$$

The constraint above is necessary only on the condition that bin k is non-empty, otherwise it must be 0 to respect the convention $0/0 = 0$. Considering that, from the anti-symmetric constraints, a slot k is non empty if and only if $z_{kk} = 1$, we have that the above condition is extended to all k with:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} [a_{ij} x_{ijk} u_k + a_{ij} y_{ijk} u_k] = z_{kk} \text{ for all k.}$$

Quadratic terms can be linearized:

$$x_{ijk} u_k = w_{ijk} \iff \begin{cases} w_{ijk} \leq u_k \\ w_{ijk} \leq x_{ijk} \\ w_{ijk} \geq u_k - (1 - x_{ijk}) \end{cases} \qquad \forall i, j \in V.$$

Table A.1
Computational times (seconds) of the Mixed Integer Linear Programming.

nodes	Caveman	Caveman rewired		
		min	average	max
15	4.9	6.5	9.3	13.6
20	57.5	63.6	212.4	1140.3
25	1069.1	1450.2	2022.8	2952.1

and similarly:

$$y_{ijk}u_k = q_{ijk} \Longleftrightarrow \begin{cases} q_{ijk} \le u_k \\ q_{ijk} \le y_{ijk} \\ q_{ijk} \ge u_k - (1 - y_{ijk}) \end{cases} \qquad \forall i, j \in V.$$

and now the objective function is:

$$\max \sum_{k=1}^{n} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 2a_{ij}w_{ijk}. \tag{18}$$

Combining the objective function (18) with the constraints representing the quadratic terms and the antisymmetric conditions, we obtain a MILP with $\mathcal{O}(n^3)$ constraints and $\mathcal{O}(n^3)$ variables. Indeed, there are $\mathcal{O}(n^2)$ symmetry breaking constraints, $\mathcal{O}(n^3)$ constraints derived by the linearization of the quadratic and max terms of the objective functions, and $\mathcal{O}(n^3)$ constraints derived by the Charnes-Cooper linearization. The number of variables is $\mathcal{O}(n^3)$ as well, as triple index variables are required by the linearization of the objective function. The actual dimension of an instance with 30 nodes is a problem with 108,930 variables and 324,931 constraints. Given the problem size, we can expect that any Integer Linear Programming solver can calculate the optimal partitions only for instances of moderate size, such as the networks arising from opinion surveys, see [2].

In the following test, we used Gurobi 11.02 in the R/RStudio environment. First, we simulate a Caveman graph, see [30], made up of cliques of five nodes. Then, cliques are multiplied by 3, 4 and 5, to obtain graphs of 15, 20 and 25 nodes. Caveman graphs have a clear community structure, making them the easiest to solve because there are not many competing solutions. Next, Caveman graphs are randomly rewired to hide the community structure and make instances harder to solve.

The computational times are reported in Table A.1, expressed in seconds. Assuming a time limit of 3600 seconds, we observe that instances of 20 nodes can be solved in a few minutes if the graph has a recognizable community structure. However, the time required increases significantly as the number of nodes increases. Instances of 25 nodes are solved within an hour, but after that size, that is, between 26 and 30 nodes, the time limit is often exceeded. It is worth noting that the optimization problem (16) is similar to the min-cut density clustering analyzed in [21]: both optimization functions are the sum of ratios, but the direction of the optimization is reversed, maximization vs minimization. The computational times of the two models are similar, but in [21] it has been shown that column generation with branch&cut can improve the computational times, and therefore it is an interesting direction for new research.

Appendix B. Further analytical properties

Proposition 6. *Let $\mathcal{P}^\star_\Pi$ be the total null-adjusted persistence of the partition Π, $\mathcal{P}^\star_{C_1}$ and $\mathcal{P}^\star_{C_2}$ the null-adjusted persistence of two clusters C_1 and C_2 in Π, respectively. Let us consider a node $v \in C_1$. The relocation of node v from cluster C_1 to cluster C_2 and the subsequent move to partition π' such that C_1 and C_2 are modified into $C_1 \setminus \{v\}$ and $C_2 \cup \{v\}$, with other clusters unchanged, produces the following variation of the total null-adjusted persistence:*

$$\Delta\mathcal{P}^\star = \left[\frac{2\left(m_i^{(1)} - m_i^{(1v)}\right)}{2m_i^{(1)} + m_e^{(1)} - k_v} - \frac{2m_i^{(1)}}{2m_i^{(1)} + m_e^{(1)}}\right] + \left[\frac{2\left(m_i^{(2)} + m_i^{(2v)}\right)}{2m_i^{(2)} + m_e^{(2)} + k_v} - \frac{2m_i^{(2)}}{2m_i^{(2)} + m_e^{(2)}}\right] \tag{19}$$

where $m_i^{(1v)}$ and $m_i^{(2v)}$ are the number of links from node v internal to C_1 and C_2, and k_v is the total degree of node v.

Proof. In the same notations as in the main text, when node $v \in C_1$ is re-assigned to cluster C_2, the following modifications in the number of the internal and external links for the two clusters occur:

$$\begin{aligned} m_i^{(1)} &\Longrightarrow m_i^{(1)} - m_i^{(1v)} \\ m_e^{(1)} &\Longrightarrow m_e^{(1)} - m_e^{(1v)} - m_e^{(2v)} + m_i^{(1v)} \\ m_i^{(2)} &\Longrightarrow m_i^{(2)} + m_i^{(2v)} \\ m_e^{(2)} &\Longrightarrow m_e^{(2)} + m_e^{(1v)} - m_e^{(2v)} + m_i^{(1v)} \end{aligned} \tag{20}$$

where $m_e^{(1v)}$ and $m_e^{(2v)}$ are the number of links exiting C_1 and C_2 from node v. Substituting the right hand sides in Eq. (20) into $\Delta\mathcal{P}^\star = \left(\mathcal{P}^\star_{C_1\setminus\{v\}} + \mathcal{P}^\star_{C_2\cup\{v\}}\right) - \left(\mathcal{P}^\star_{C_1} + \mathcal{P}^\star_{C_2}\right)$ and considering that $m_i^{(1v)} + m_i^{(2v)} + m_e^{(1v)} = k_v$, we get to Eq. (19) by simple algebra. □

We observe that the equation above represents the key formula for the move function of the proposed algorithm, which selects the single node that moved from the starting slot into a different slot, and guarantees the maximum increase in the objective function.

Appendix C. Cluster-Milano algorithm

We describe in detail the algorithm used for simulations on synthetic and real-world networks in Section 4. The algorithm falls into the category of standard "greedy" optimization algorithms, and it follows an approach similar to that of the Louvain method.

Algorithm 1: Cluster Milano community detection.

Input: Graph $G = (V, E)$ with edge weights w_{ij} and $\Pi_s = \{C_1, \dots, C_m\}$ a partition of V into clusters
Output: Partition of nodes into clusters

```
1  Build network G' = (V', E') where each clusters C_u ∈ Π_s is represented by a node u with weight w'_u and edge weights w'_uz
   equal to sum of the arc weights between clusters C_u and C_z
2  repeat
3      partition_changed ← false
4      repeat
5          improved ← false
6          foreach node u ∈ V' in random order do
7              current_cluster ← clusters(u)
8              best_cluster ← current_cluster
9              best_delta ← 0
10             foreach cluster z in {clusters(neighbors(u))} do
11                 Compute ΔL = change in NAP if u is merged with z
12                 if ΔL > best_delta then
13                     best_delta ← ΔL
14                     best_cluster ← z
15                 end
16             end
17             if best_cluster ≠ current_cluster then
18                 Merge u to best_cluster
19                 improved ← true
20                 partition_changed ← true
21             end
22         end
23     until not improved
24     if partition_changed then
25         Build new network G' = (V', E') where each cluster is a node
26         Edge weights equal sum of nodes' weight between cluster
27     end
28 until no further improvement of NAP
29 return Final partition of nodes into clusters
```

The algorithm steps are described in Algorithm 1. The procedure takes a partition Π_s as input, which is reduced to a graph G' with nodes V' representing communities and arcs E' representing connections between communities. If a node u represents a cluster C_u, then $w_u^{in} = \sum_{(i,j)\in C_u} w_{ij}$. The weight of an arc (u, z) is the sum of the arc weights between cluster C_u and C_z: $w'_{ij} = \sum_{i\in C_u, j\in C_z} w_{ij}$. Then, the weight of the external arcs of u is $w^{out} = \sum_{j\in} w'_{ij}$. If the initial partition is the original graph, then clearly $G' = G$, otherwise, the initial partition is a tentative solution that could be obtained by another algorithm, or by perturbing a local optimum that was calculated previously. The algorithm tries to improve a partition by merging two subsets: say $C_q = C_u \cup C_z$. This process is ruled by two nested iterations. The inner iteration is between lines 6 and 22. It selects a cluster u at random, then it searches for the cluster z that allows the maximum improvement of the null-adjusted persistence. If the improvement can be found, then the clusters are temporarily merged in line 11 . Then, from line 6, another improvement is attempted, and, since *best_cluster* has not been removed from the list, it could be separated from u and merged with another cluster. When the iteration on line 6 is concluded, a revision of the clusters is still attempted, in the iteration from 4 to 23. Here the iteration from 4 to 26 is repeated with the same node set V' but with different clustering Π. If no improvement can be found, then line 25 is reached and the reduced network G' is updated. Regarding the computational time, note that, to evaluate the null-adjusted persistence of a potential merge $C_q = C_u \cup C_z$, the sum of the internal arcs is calculated in constant time: $w'_q = w'_u + w'_z + w'_{uz}$. The sum of the external arcs is calculated in constant time too: $w_q^{out} = w_u^{out} + w_z^{out} - 2w'_{uz}$. These values can be used to calculate the improvement of the objective function. As there are $|E|$

edges, the iteration from line 6 to 22 takes $\mathcal{O}(|E|)$ operations to find at least one improvement, or to detect that there is none. The most external iteration, from lines 2 to 28, is repeated in the worst case $|V|$ times: note that, when line 25 is executed, the new network decreases the number of nodes by at least one. It remains to establish how many times the iteration between lines 4 and 23 is repeated. Unfortunately, we cannot bound this number. If in lines 6-22 the partition has been changed, still merged clusters could be reassigned, with the only property that the objective function improves, but without bounding a worst case for the number of iterations, even though the experimental results show that this number is very small. Let K be the number of iterations between lines 4 and 23 , the overall complexity is $\mathcal{O}(K \cdot |V| \cdot |E|)$.

Appendix D. Similarity measures for network partitions

Here we provide the formal definitions used in Section 4 to measure similarity between community structures, that is, to evaluate the similarity between the partition obtained by heuristic algorithms and the ground-truth partition.

Let $\Pi_1 = \{C_1, \dots, C_k\}$ and $\Pi_2 = \{C_1, \dots, C_h\}$ be two community partitions of the network, n the total number of nodes in the network, n_i the total number of vertices in cluster C_i, n_j the total number of vertices in cluster C_j, and n_{ij} the number of vertices that are simultaneously in both C_i and C_j.

Adjusted Rand Index (ARI) measures the similarity between Π_1 and Π_2 by correcting the Rand index for agreement by chance. The Rand index counts pairs of elements that are in the same or different clusters in the partitions. ARI adjusts the Rand index by subtracting the expected value of the Rand index for random clusterings and normalizing it. The formula is as follows:

$$\mathrm{ARI}(\Pi_1, \Pi_2) = \frac{\sum_{i=1}^{k}\sum_{j=1}^{h}\binom{n_{ij}}{2} - \left[\sum_{i=1}^{k}\binom{n_i}{2}\sum_{j=1}^{h}\binom{n_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left(\sum_{i=1}^{k}\binom{n_i}{2} + \sum_{j=1}^{h}\binom{n_j}{2}\right) - \left[\sum_{i=1}^{k}\binom{n_i}{2}\sum_{j=1}^{h}\binom{n_j}{2}\right]/\binom{n}{2}}$$

Normalized Mutual Information (NMI) is a normalized version of the Mutual Information index, an index measuring the shared information between two variables, by explaining how much the knowledge of a variable reduces uncertainty about the other one. In the network context, it can be used to quantify the similarity of partitions Π_1 and Π_2. Normalization is obtained by dividing MI index by the arithmetic mean of the entropies, which quantify the uncertainty of the partitions.

$$\mathrm{NMI}(\Pi_1, \Pi_2) = \frac{\sum_{i=1}^{k}\sum_{j=1}^{h}\frac{n_{ij}}{n}\log\left(\frac{\frac{n_{ij}}{n}}{\frac{n_i}{n}\frac{n_j}{n}}\right)}{\frac{1}{2}\left[\left(-\sum_{i=1}^{k}\frac{n_i}{n}\log\frac{n_i}{n}\right) + \left(-\sum_{j=1}^{h}\frac{n_j}{n}\log\frac{n_j}{n}\right)\right]}$$

Data availability

Data will be made available on request.

References

[1] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. 99 (12) (2002) 7821–7826.
[2] S. Benati, J. Puerto, A network model for multiple selection questions in opinion surveys, Qual. Quant. 58 (2) (2024) 1163–1179.
[3] F. Allen, A. Babus, Networks In Finance: Network-Based Strategies And Competencies, Chapter 21, Tech. rep, Working Paper, 2008.
[4] P. Bartesaghi, G.P. Clemente, R. Grassi, Community structure in the World Trade Network based on communicability distances, J. Econ. Interact. Coord. (2020) 1–37.
[5] S. Fortunato, D. Hric, Community detection in networks: a user guide, Phys. Rep. 659 (2016) 1–44.
[6] J. Li, S. Lai, Z. Shuai, Y. Tan, Y. Jia, M. Yu, Z. Song, X. Peng, Z. Xu, Y. Ni, et al., A comprehensive review of community detection in graphs, Neurocomputing 600 (2024) 128169.
[7] M.E. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2) (2004) 026113.
[8] M.E. Newman, Analysis of weighted networks, Phys. Rev. E 70 (5) (2004) 056131.
[9] M. Newman, D. Watts, S. Strogatz, Random graph models of social networks, Proc. Natl. Acad. Sci, 99 (supp. 1) (2002) 2566–2572.
[10] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. Theory Exp. 10 (2008) 10008.
[11] G. Agarwal, D. Kempe, Modularity-maximizing graph communities via mathematical programming, Eur. Phys. J. B 66 (3) (2008) 409–418.
[12] J. Zhu, B. Chen, Y. Zeng, Community detection based on modularity and k-plexes, Inf. Sci. 513 (2020) 127–142.
[13] S. Aref, M. Mostajabdaveh, H. Chheda, Heuristic modularity maximization algorithms for community detection rarely return an optimal partition or anything similar, Lect. Notes Comput. Sci. (Lect. Notes Artif. Intell. Lect. Notes Bioinform.) (2023) 612–626.
[14] S. Fortunato, M. Barthélemy, Resolution limit in community detection, Proc. Natl. Acad. Sci. 104 (1) (2007) 36–41.
[15] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, D. Wagner, On modularity clustering, IEEE Trans. Knowl. Data Eng. 20 (2) (2008) 172–188.
[16] A. Lancichinetti, S. Fortunato, Limits of modularity maximization in community detection, Phys. Rev. E 84 (6) (2011) 066122.
[17] X. Lu, B. Cross, B.K. Szymanski, Asymptotic resolution bounds of generalized modularity and multi-scale community detection, Inf. Sci. 525 (2020) 54–66.
[18] A. Costa, MILP formulations for the modularity density maximization problem, Eur. J. Oper. Res. 245 (1) (2015) 14–21.
[19] A. Miyauchi, Y. Kawase, Z-score-based modularity for community detection in networks, PLoS ONE 11 (1) (2016) e0147805.
[20] V.A. Traag, L. Waltman, N.J. Van Eck, From Louvain to Leiden: guaranteeing well-connected communities, Sci. Rep. 9 (1) (2019) 1–12.
[21] D. Ponce, J. Puerto, F. Temprano, Mixed-integer linear programming formulations and column generation algorithms for the minimum normalized cuts problem on networks, Eur. J. Oper. Res. 316 (2) (2024) 519–538.
[22] M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Natl. Acad. Sci. 105 (4) (2008) 1118–1123.
[23] C. Piccardi, Finding and testing network communities by lumped Markov chains, PLoS ONE 6 (11) (2011) 1–13.
[24] C. Piccardi, L. Tajoli, Existence and significance of communities in the World Trade Web, Phys. Rev. E 85 (6) (2012) 066119.
[25] F. Calderoni, D. Brunetto, C. Piccardi, Communities in criminal networks: a case study, Soc. Netw. 48 (2017) 116–125.
[26] N.P. Nguyen, M.A. Alim, T.N. Dinh, M. Thai, A method to detect communities with stability in social networks, Soc. Netw. Anal. Min. 4 (1) (2014) 224.
[27] M.M. Tulu, R. Hou, S.A. Gerezgiher, T. Younas, M.D. Amentie, Finding best matching community for common nodes in mobile social networks, Wirel. Pers. Commun. 114 (4) (2020) 2889–2908.
[28] A. Avellone, S. Benati, R. Grassi, G. Rizzini, On finding the community with maximum persistence probability, 4OR 22 (4) (2024) 435–463.

[29] A. Patelli, A. Gabrielli, G. Cimini, Generalized markov stability of network communities, Phys. Rev. E 101 (2020) 052301.
[30] D.J. Watts, Networks, dynamics, and the small-world phenomenon, Am. J. Sociol. 105 (2) (1999) 493–527.
[31] J.L. Gross, J. Yellen, P. Zhang, Handbook of Graph Theory, CRC Press, 2013.
[32] A. Arenas, A. Fernández, S. Gómez, Analysis of the structure of complex networks at different resolution levels, New J. Phys. 10 (5) (2008) 053039.
[33] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (2008) 046110.
[34] L. Hubert, P. Arabie, Comparing partitions, J. Classif. 2 (1985) 193–218.
[35] F. Taya, J. de Souza, N.V. Thakor, A. Bezerianos, Comparison method for community detection on brain networks from neuroimaging data, Appl. Netw. Sci. 1 (1) (2016) 8.
[36] J. Leber, The immortal life of the Enron e-mails, Technol. Rev. 116 (5) (2013) 15–16.
[37] G. Tsipenyuk, J. Crowcroft, An email attachment is worth a thousand words, or is IT? in: Proceedings of the 1st International Conference on Internet of Things and Machine Learning, 2017, pp. 1–10.
[38] J. Leskovec, J. Mcauley, Learning to discover social circles in ego networks, In F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, vol. 25, Curran Associates, Inc, 2012.
[39] P.W. Holland, K.B. Laskey, S. Leinhardt, Stochastic blockmodels: first steps, Soc. Netw. 5 (2) (1983) 109–137.
[40] L. Peel, D.B. Larremore, A. Clauset, The ground truth about metadata and community detection in networks, Sci. Adv. 3 (5) (2017) e1602548.
[41] R. Aldecoa, I. Marín, Deciphering network community structure by surprise, PLoS ONE 6 (9) (2011) 1–8.
[42] R. Aldecoa, I. Marín, Surprise maximization reveals the community structure of complex networks, Sci. Rep. 3 (1) (2013) 1060.
[43] V.A. Traag, G. Krings, P. Van Dooren, Significant scales in community structure, Sci. Rep. 3 (1) (2013) 2930.
[44] A. Lancichinetti, F. Radicchi, J.J. Ramasco, S. Fortunato, Finding statistically significant communities in networks, PLoS ONE 6 (4) (2011) 1–18.
[45] Y. Tian, Z. Lubberts, M. Weber, Curvature-based clustering on graphs, J. Mach. Learn. Res. 26 (52) (2025) 1–67.
[46] S. Chakrabarty, S. Sengupta, Y. Chen, Subsampling based community detection for large networks, Stat. Sin. 35 (3) (2025) 1–42.
[47] S. Benati, J. Puerto, A. Rodríguez-Chía, Clustering data that are graph connected, Eur. J. Oper. Res. 261 (1) (2017) 43–53.