ARTICLE TEMPLATE

On Finding the Community with Maximum Persistence Probability

A. Avellone[a], Stefano Benati[b], Rosanna Grassi[a] and Giorgio Rizzini[a]

[a]Dipartimento di Statistica e Metodi Quantiativi, Università degli Studi di Milano-Bicocca, Milan, Italy; [b]Dipartimento di Sociologia e Ricerca Sociale, Università degli Studi di Trento, Trento, Italy

ARTICLE HISTORY
Compiled December 22, 2022

ABSTRACT
The persistence probability is a statistical index that has been proposed to detect one or more communities embedded in a network. Even though its definition is straightforward, e.g, the probability that a random walker remains in a group of nodes, it has been seldom applied possibly for the difficulty of developing an efficient algorithm to calculate it. Here, we propose a new mathematical programming model to find the community with the largest persistence probability. The model is integer fractional programming, but it can be reduced to mixed-integer linear programming with an appropriate variable substitution. Nevertheless, the problem can be solved in a reasonable time for networks of small size only, therefore we developed some heuristic procedures to approximate the optimal solution. First, we elaborated a randomized greedy-ascent method, taking advantage of a peculiar data structure to generate feasible solutions fast. After analyzing the greedy output and determining where the optimal solution is eventually located, we implemented improving procedures based on a local exchange, but applying different long term diversification principles, that are based on variable neighborhood search and random restart. Next, we applied the algorithms on simulated graphs that reproduce accurately the clustering characteristics found in real networks to determine the reliability and the effectiveness of our methodology. Finally, we applied our method to two real networks, comparing our findings to what found by two well-known alternative community detection procedures.

KEYWORDS
Persistence probability; Integer fractional programming; Community detection

1. Introduction

The analysis of real networks, as they emerged as a structural model in disciplines as different as biology, economics, social sciences, engineering and so on, brought about a growing and thriving interest in developing new tools and methods to uncover the networks hidden characteristics, such as their communities (Fortunato and Hric (2016)), their core-periphery structure (Tang, Zhao, Liu, and Yan (2019)), their node centrality (Das, Samanta, and Pal (2018)). From the greatly cited Girvan's contribution (Girvan and Newman (2002)) community models and problems are often concerned on

Email: alessandro.avellone@unimib.it
Email: stefano.benati@unitn.it
Email: rosanna.grassi@unimib.it
Email: giorgio.rizzini@unimib.it

networks decomposition, that is, finding *all* the communities that form the network. However, it can be the case that just one or few more communities are actually hidden in the largest realm of unstructured connections between nodes having no peculiar structure. Therefore, we focus here on the problem of finding *one* community embedded in the largest networks, without explicitly requiring that the rest of the nodes should be interpreted as communities. The first concept that has been used to identify a community was the clique, that is, a subset of nodes forming a complete graph, see Luce and Perry (1949). However, the clique is a concept that is too rigid to determine a community, as some vertices could be part of the same community even though they do not share a link. Therefore some variants on clique were proposed. For example, the n-clique (connected vertices with minimum distance n) was suggested in Mokken (1979) and, in the same paper, the n-clique is further restricted to define n-clans and n-clubs. The k-plex, a group of nodes with peculiar connectivity constraint, is defined in Seidman (1978); other generalizations appeared in the literature, among others the quasi-clique Pattillo (2013), the s-defective clique, Yu (2006), the l-triangle k-club, Almeida and Brs (2019). Moreover, combining edge density with the subgraph diameter is useful to characterize subgraphs having the small-world property, as proposed in Kim, Veremyev, Boginski, and Prokopyev (2020). Finally, a recent taxonomy of various clique generalization for complex network analysis is provided in Pattillo, Youssef, and Butenko (2013). Given these characterizations, a community is then identified as the nodes subset for which one of the above generalized clique indexes is maximum, that is, solving an optimization problem with integral variables. Since the clique problem is NP-complete, exact optimization for quasi-clique problems is of practical use for small-sized instances, while arge instances can be solved by heuristic procedure. Exact methods relies on formulating the problem as mixed integer linear programming: see the quasi-clique in Mahdavi Pajouh, Miao, and Balasundaram (2014), see the k-club solution in Moradi and Balasundaram (2018); Veremyev and Boginski (2012), see the k-plex in Balasundaram, Butenko, and Hicks (2011). Heuristic solution for quasi-clique are calculated with variants of meta-heuristic, for example GRASP is used in Abello, Resende, and Sudarsky (2002), Tabu-search is used in Djeddi, Haddadene, and Belacel (2019); Zhou and Hao (2017), genetic/memetic algorithms are used in Pinto, Ribeiro, Rosseti, and Plastino (2018); Zhou, Benlic, and Wu (2020), bee colony is used in Peng, Wu, Wang, and Wu (2021). Relevant to our contributions are the application of Variable Neighborhood search to modularity maximization and graph-connected clustering Aloise, Caporossi, Hansen, Liberti, Perron, and Ruiz (2013); Benati, Puerto, and Rodríguez-Chía (2017); Dami, Aloise, and Mladenovi (2019).

The above characterizations looked at the structure of the arcs *within* the community, but one could also consider that communities are not only composed of well connected members, but are also *separated* by the rest of the graph. Therefore, connections to external vertices should be considered as well. For example, combining the out-degree and the in-degree of a community leads to the definitions of strong and weak community, see Hu, Chen, Zhang, Li, Di, and Fan (2008); Radicchi, Castellano, Cecconi, Loreto, and Paris (2004). The former definitions of community rely on counting internal and external edges of a community, possibly requiring additional structural properties in term of distances, cohesion and so on. However, it could be argued that the real issue could be summarized in term of probabilities. Namely, the vertices of a community should form a link between them with probability higher than forming a link with an external node, see Fortunato and Hric (2016), page 7. Following this approach, in this work we focus on a specific measure, the *persistence probability*, that has been proposed to find one or more communities embedded in a network, see

Piccardi (2011). Loosely speaking, given a subset of nodes, its persistence probability is the probability that a random walker, located by chance in one of these nodes and moving randomly across the links, will remain in another node of the subset. The ratio is that this statistic should be able to detect community nodes as well connected with each other to form a community: the highest the persistence, the highest fraction of links are directed towards internal nodes to the detriment of the external ones. The measure aroused some interest among scholars: for example, it has been used in Della Rossa, Dercole, and Piccardi (2013) to detect the core-periphery structures in many real networks such as the Karate Club, the co-authorship, the proteins and the World Trade networks. Next, the persistence has been used to analyze the World Trade network, Piccardi and Tajoli (2012), and to identifies the *locali* (the local mobs) of the *n'drangheta* criminal networks in Calderoni, Brunetto, and Piccardi (2017).

The persistence probability has a clear and appealing definition and it is flexible enough to be applied for both community detection and core-periphery analysis. However, its application is still undervalued, maybe due to the fact that computational methods have not been developed yet, at least to the best of our knowledge. In this contribution, we try to fill the gap and we propose a new mathematical programming model to find the community with the largest persistence. It results in a fractional programming model that can be converted to mixed-integer using standard variable substitutions. Next, one of the difficulty of the model is imposing connectivity on the community nodes, but we apply here the linear constraints that were effective in a similar problem: the graph-connected clique, Benati et al. (2017). Unfortunately, but predictability as the problem is NP-hard, the problem can be solved exactly only when the network size is small, therefore we developed some heuristic procedures to approximate the optimal solution. We had to consider that the arithmetic behind the maximum persistence problem imposes that the correct community size k must be known in advance, a case that rarely occurs in practice. Therefore, the implementation of a heuristic must consider that it should be able to provide the persistence values for a whole range of parameters k in a single run. Then, looking at the peaks of the persistence function, the correct value k could be guessed. For this purpose, we elaborated on a randomized greedy-ascent method, proposed previously for a similar problem in Benati, Ponce, Puerto, and Rodriguez-Chia (2022). After determining the right value of k, the greedy outcome can be improved by local exchange and long-term diversification strategies. Here, we adopt diversification based on variable neighborhood search and random restart, but with some variation due to the problem structure. Variable neighborhood search has been implemented of the reduction of the cluster to its spanning tree, random restart is controlled by preliminary diversification.

We test the whole methodology, its accuracy and computational time, on graphs simulated through the procedure proposed in Lancichinetti, Fortunato, and Radicchi (2008), as was done also in Piccardi (2011). This procedure simulates synthetic networks with the same characteristics found in real networks, therefore they are a severe and realistic benchmark. As it can be seen, the right size k is often correctly determined after the greedy, and then the diversification heuristic improves the incumbent solution (when possible) in a fast way and hidden communities embedded in the network are detected. In the end, we apply our method to two real networks to test its ability in identifying communities, comparing its results with what found by two alternative methods, e.g., the Walktrap and the Louvain, see Blondel, Guillaume, Lambiotte, and Lefebvre (2008); Pons and Latapy (2005), and we will see that the use of the persistence complements well the findings of the other methods.

The paper is organized as follows. In Section 2 the definition of persistence probability is formalized. In Section 3, finding the node subset with maximum persistent is formulated as an optimization problem, that after some modification is turn into mixed-integer linear programming. In Section 4, some heuristic algorithms are introduced: the first is a greedy procedure with some randomized steps, able to calculate the optimal persistence for subsets of varying size k, next the greedy results are improved with the interchange heuristic and some version of long-term diversification. In Section 5, computational tests are carried on to explain how to use the persistence probability and what are the best algorithms to find it. Finally, in Section 6, some empirical experiments on two real networks are explained. Conclusions follow (Section 7).

2. The Persistence probability

Let $G = (V, E)$ be a simple, undirected and connected graph (or network) where V is the set of nodes and E is the set of the edges (or links). Let $n = |V|$ be the cardinality of V. Consider a node subset $V_C \subseteq V$ and assume that the subgraph G_C, induced [1] by V_C is connected. Let E_C be the edge set of the subgraph and $e_{ij} = 1$ if $(i, j) \in E$, $e_{ij} = 0$ otherwise. In Piccardi (2011) and Della Rossa et al. (2013), the persistence probability $\alpha(V_C)$ is proposed as a measure of cohesiveness of subset V_C. Formally, $\alpha(V_C)$ is defined as:

$$\alpha(V_C) = \frac{\sum_{(i,j)\in E_C} e_{ij}}{\sum_{i\in V_C} \sum_{j\in V} e_{ij}}, \tag{1}$$

expressing the ratio between the number of links connecting nodes inside V_C, e.g. the internal links, and all the links emanating from V_C, e.g. the internal plus the external links.

Communities are defined as the node subsets with maximum persistence probability, however, one should be careful. By definition, $\alpha(V_C)$ takes value in $[0, 1]$. The extreme cases refer to the situation in which V_C is a singleton, e.g. $\alpha(V_C) = 0$, and $V_C = V$, e.g. $\alpha(V_C) = 1$. As a result, the plain optimization of $\alpha(V_C)$ is misleading, as no subset can be better than the whole set V. Rather, best values of $\alpha(V_C)$ can be calculated by constraining the cardinality blue of the subset V_C to a bound k and then determining the community from the trade-off between k and $\alpha(V_C)$.

An example about the use of the persistence is reported in Figure 1, where two subsets V_1 and V_2 with different size are considered.

3. Integer programming problem formulation

Given the graph $G = (V, E)$ previously defined, the aim of this section is to formulate the maximization of the persistence probability $\alpha(V_C)$ as mathematical programming problem. In doing so, we unveil the community with the highest persistence probability selecting k nodes from V to be included in V_C under the constraint that the induced subgraph G_C is connected.

Problem variables are the boolean x_i, $i = 1, \ldots, n$, taking value 1 if $i \in V_C$, 0 otherwise.

[1] The induced subgraph G_C is the graph whose vertex set is V_C and whose edge set consists of all the edges in E that have both endpoints in G_C.

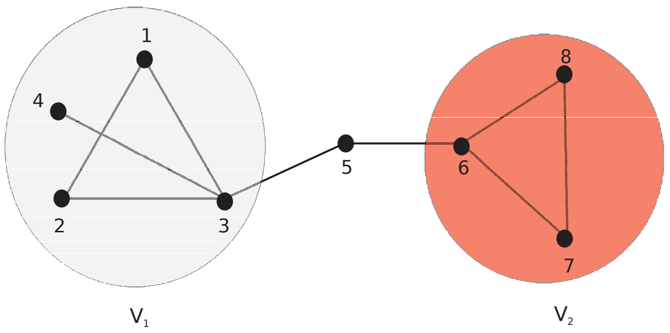


Figure 1.: Different persistence probabilities of the subsets V_1 and V_2 for a graph G. The persistence probabilities are $\alpha(V_1) = \frac{4}{5}$ and $\alpha(V_2) = \frac{3}{4}$, respectively.

Given the trade-off between size k and $\alpha(V_C)$, the following constraint should be imposed:

$$\sum_{i \in V} x_i = k. \tag{2}$$

Next, for any edge $(i,j) \in E$, let:

$$z_{ij} = x_i x_j, \quad \forall (i,j) \in E, \tag{3}$$

and

$$w_{ij} = \max\{x_i, x_j\}, \quad \forall (i,j) \in E, \tag{4}$$

so that $z_{ij} = 1$ if both nodes i and j are in the subset V_C, whereas $w_{ij} = 1$ if at least one of the nodes i and j belongs to V_C. By using variables w and z, we rewrite the persistence probability in Formula 1 as:

$$\alpha(V_C) = \frac{\sum\limits_{(i,j) \in E} z_{ij}}{\sum\limits_{(i,j) \in E} w_{ij}} \tag{5}$$

As the persistence must be calculated for connected G_C only, we must impose the connectivity. In Benati et al. (2022), a similar problem is addressed, that is finding the optimal graph-connected clique. Here, we adopt one of the methods proposed there to impose node connectivity: the flow-based approach and, for clarity sake, we briefly describe such method.

The subgraph $G_C = (V_C, E_C)$ is connected if a node in V_C, denoted as source node, can send a unit of flow to any other node of V_C through an auxiliary digraph $G_D = (V, A)$, being A the set of arcs defined in a way that if $(i,j) \in E$, then $(i,j), (j,i) \in A$. Variables f_{ij} for all pairs $i,j = 1,\dots,n$ must be introduced, they will correspond to the flow from node i to node j. For a given V_C, the source is identified as the node j with maximum index, that is $j = \max\{i | x_i = 1\}$. The flow leaves j to satisfy a demand of one unit flow from all the other nodes of the subset V_C. To be G_C connected, V_C must allow a feasible solution to these set of constraints:

$$\sum_{j \in V:(i,j)\in A} f_{ij} \leq (k-1)x_i, \; \forall i \in V \tag{6}$$

$$\sum_{i \in V:(i,p)\in A} f_{ip} - \sum_{i \in V:(p,i)\in A} f_{pi} \geq x_p + (n-2)(x_j - 1), \; \forall p, j \in V : j > p, \tag{7}$$

$$f_{ij} \geq 0, \quad \forall (i,j) \in A. \tag{8}$$

Constraint (6) states an upper bound on the amount of flow leaving any node $i \in V$. The right hand side can be positive or null. If $i \notin V_C$, then $x_i = 0$ and no flow can leave the node i. Conversely, if $i \in V_C$ and $x_i = 1$, then the flow can be positive and as large as to satisfy the flow demands from the other nodes of V_C, so its maximum value is bounded by the cardinality of the subset V_C, k. Constraint (7) expresses a flow conservation law. It is defined for all pairs of nodes $j, p \in V$, with $j > p$, as j can be identified as the source. If both j and p are in V_C then a unit flow must remain in p. If $j \notin V_C$ or $p \notin V_C$, then the constraint is not active. Note that if p is the source, there is no $j \in V_C$, such that $j > p$ and the only constraint affecting its flow is Equation (6). Finally, condition (8) states a non-negative constraint for the unit of flow.

Hence, the maximum persistence problem is defined as follows. It is an instance of integer fractional programming, due to equation (5), and it is denoted as Problem **P**.

$$\mathbf{P} : \max_{x_i} \frac{\sum_{(i,j)\in E} z_{ij}}{\sum_{(i,j)\in E} w_{ij}} \tag{9}$$

s.t.

$$\sum_{i \in V} x_i = k, \tag{10}$$

$$z_{ij} = x_i x_j \quad \forall i, j \in V, \tag{11}$$

$$w_{ij} = \max\{x_i, x_j\} \quad \forall i, j \in V, \tag{12}$$

$$\sum_{j \in V:(i,j)\in A} f_{ij} \leq (k-1)x_i \quad \forall i \in V, \tag{13}$$

$$\sum_{i \in V:(i,p)\in A} f_{ip} - \sum_{i \in V:(p,i)\in A} f_{pi} \geq x_p + (n-2)(x_j - 1), \; \forall p, j \in V : j > p, \tag{14}$$

$$f_{ij} \geq 0, \quad \forall (i,j) \in A, \tag{15}$$

$$x_i \in \{0,1\}, \quad \forall i \in V. \tag{16}$$

The objective function in (9) expresses the persistence probability in terms of variables z_{ij} and w_{ij} defined in (3) and (4), which, in turn, are formally represented by constraints (11) and (12), respectively. (10) is the constraint on the cardinality of

V_C. Constraints (13), (14), and (15) ensure that the subgraph G_C is connected. The variable x_i is defined by constraint (16).

The problem **P** can be converted into a mixed-integer linear problem through the Clique Partitioning problem inequalities and the Charnes-Cooper linearization, (Charnes and Cooper, 1962).

An auxiliary variable u is introduced such that:

$$u = \frac{1}{\sum_{(i,j)\in E} w_{ij}} \tag{17}$$

and set $h_{ij} := u\, z_{ij}$ and $l_{ij} := u\, w_{ij}$ for all $(i,j) \in E$. The Charnes-Cooper linearization allows us to write the objective function in (9) as

$$\sum_{(i,j)\in E} h_{ij}. \tag{18}$$

Variables z_{ij} and w_{ij} can be described by the linear constraints:

$$z_{ij} = x_i x_j \iff \begin{cases} z_{ij} \leq x_i \\ z_{ij} \leq x_j \\ z_{ij} \geq x_i + x_j - 1 \end{cases} \qquad \forall i, j \in V$$

$$w_{ij} = \max\{x_i, x_j\} \iff \begin{cases} w_{ij} \geq x_i \\ w_{ij} \geq x_j \\ w_{ij} \leq x_i + x_j \end{cases} \qquad \forall i, j \in V.$$

As $h_{ij} = u$ and $l_{ij} = u$ only in cases in which $z_{ij} = 1$ and $w_{ij} = 1$ respectively, and noting that $u \leq 1$, then the quadratic terms are linearized as follows

$$h_{ij} = u z_{ij} \iff \begin{cases} h_{ij} \leq u \\ h_{ij} \leq z_{ij} \\ h_{ij} \geq u - (1 - z_{ij}) \end{cases} \qquad \forall (i, j) \in E$$

and

$$l_{ij} = u w_{ij} \iff \begin{cases} l_{ij} \leq u \\ l_{ij} \leq w_{ij} \\ l_{ij} \geq u - (1 - w_{ij}) \end{cases} \qquad \forall (i, j) \in E.$$

Note that if $z_{ij} = 1$ the inequalities are satisfied only for $h_{ij} = u$, whereas if $z_{ij} = 0$ then the inequalities are satisfied only for $h_{ij} = 0$, that is exactly the meaning of $h_{ij} = u z_{ij}$ (similar considerations are valid for l_{ij}).

Hence, the linearized problem can be written as following:

$$\mathbf{P1}: \max_{u} \sum_{(i,j)\in E} h_{ij} \tag{19}$$

s.t.

$$\sum_{i\in V} x_i = k. \tag{20}$$

$$z_{ij} \leq x_i \quad \forall (i,j) \in E \tag{21}$$

$$z_{ij} \leq x_j \quad \forall (i,j) \in E \tag{22}$$

$$z_{ij} \geq x_i + x_j - 1 \quad \forall (i,j) \in E \tag{23}$$

$$w_{ij} \geq x_i \quad \forall (i,j) \in E \tag{24}$$

$$w_{ij} \geq x_j \quad \forall (i,j) \in E \tag{25}$$

$$w_{ij} \leq x_i + x_j \quad \forall (i,j) \in E \tag{26}$$

$$h_{ij} \leq u, \quad \forall (i,j) \in E \tag{27}$$

$$h_{ij} \leq z_{ij}, \quad \forall (i,j) \in E \tag{28}$$

$$h_{ij} \geq u - (1 - z_{ij}), \quad \forall (i,j) \in E \tag{29}$$

$$h_{ij} \geq 0 \tag{30}$$

$$l_{ij} \leq u, \quad \forall (i,j) \in E \tag{31}$$

$$l_{ij} \leq w_{ij}, \quad \forall (i,j) \in E \tag{32}$$

$$l_{ij} \geq u - (1 - w_{ij}), \quad \forall (i,j) \in E \tag{33}$$

$$l_{ij} \geq 0, \tag{34}$$

$$\sum_{(i,j)\in E} l_{ij} = 1, \tag{35}$$

$$\sum_{j\in V:(i,j)\in A} f_{ij} \leq (k-1)x_i \quad \forall i \in V, \tag{36}$$

$$\sum_{i\in V:(i,p)\in A} f_{ip} - \sum_{i\in V:(p,i)\in A} f_{pi} \geq x_p + (n-2)(x_j - 1), \forall p, j \in V : j > p, \tag{37}$$

$$f_{ij} \geq 0, \quad \forall (i,j) \in A \tag{38}$$

$$x_i \in \{0,1\}, \quad \forall i \in V \tag{39}$$

The objective function in (19) represents the persistence probability in terms of the variable h_{ij} defined through the auxiliary variable u in (17). Constraints (21),(22), and (23) reply the constraint (10) with the Clique Partitioning inequalities of z_{ij}. The quadratic term l_{ij} is linearized by the constraints (31),(32), (33), and (34). Constraint (35) writes in terms of equality the auxiliary variable u introduced in (17). Finally, constraints (27),(28),(29), and (30) express the linearization of h_{ij}.

4. Heuristic algorithms for the optimal persistent community

In this section we present some heuristic algorithms for finding the community with maximum persistence probability. The algorithms that we introduce are (in order of computational times):

- A randomized-greedy procedure called Random Shrink (RS).
- A merge procedure called Random Shrink Interchange (RSI).
- Two long-term heuristic search called Random Shrink Variable Neighborhood Search (RSVNS) and Constrained Random Restart (CRR).

One practical difficulty of applying the persistent index to community detection is that a user does not know in advance what is the size k of the optimal community. Moreover, as we discussed previously, the persistent index tends to increase as the size of a community increases as well, ranging from 0, when the community is a singleton, to 1, when the community is composed of all nodes in V. As we will show in Section 5, at least in some cases, there is a way to determine k, but only after that approximate values of the persistence have been calculated for all k belonging to the set $\{k_l, \ldots, k_u\}$. Therefore, the first need of a user is a fast and reliable algorithm to calculate the persistence blue probability for communities of size k in a range, and this is the purpose of algorithm Random Shrink. Next, after determining k, the incumbent solution calculated by Random Shrink can be improved replacing a node of a community with an external one, using the interchange function. When the interchange function cannot improve a solution, we say that the solution is a local optimum, and we call "basin of attraction" the set of all communities that calculate the same local optimum when interchange is applied. Of course, the interchange subroutine can be applied to many starting solutions to find different local optima, but many strategies are possible and the most effective depends on the problem. We test two of these strategies, one based on variable neighbors and the other on random restart. In variable neighbors search the best solution found is slightly perturbed to escape the local optimum and to continue the interchange in another basin of attraction, that is close to the previous one. If the local optima are close to each other, and the optimal community is similar to (e.g. it overlaps) other communities that are only locally optimal, this strategy can be effective. Conversely, if local optima are distant, so that the optimal solution does not overlap with other local optima, then it is more convenient to forget about them and continue the search though random restart. Hopefully, the new starting solution is a community of a complete different basin of attraction.

4.1. Random Shrink

In most application, the researcher does not know in advance the correct size k of the community with the optimal persistence. Rather, as often happens with clustering algorithms, optimization should consider various levels of k before deciding the best one. It may be quite time expensive to engage the algorithms in a thorough calculation of the optimal communities before knowing the exact value of k. Instead, it would be convenient to be satisfied with approximate values of α calculated quickly, hence to concentrate the computational resources after that k has been determined.

The Algorithm 1 that is presented below takes advantage from the fact that, for two non-overlapping communities: 1) it takes linear computational time to check whether their union is a connected subset; 2) it takes constant computational time to calculate the resulting value α. Hence, starting from a collection of non overlapping communities, they can be progressively merged to find (almost) optimal α's for a whole range of k. Moreover, to enhance diversification, the process can be repeated many times with different starting solutions with a cheap computational cost. The method has been used before in Benati et al. (2022), where it was found to be an effective and reliable tool to obtain a collection of almost optimal communities quickly.

In a general step of the algorithm, nodes of V are partitioned into groups $\mathcal{C} = \{C_1, \ldots, C_m\}$, with $m \leq n$. Let be $C_q, C_l \in \mathcal{C}$, the following quantities are given:

- $\mathcal{E}_q^{in} = \#\{$edges with both end nodes in $C_q\}$;
- $\mathcal{E}_q^{out} = \#\{$edges with one end node in C_q and the other end node not in $C_q\}$;
- $\mathcal{A}_{ql} = \#\{$edges with one end node in C_q and the other end node in $C_l\}$.

Given those quantities, two clusters C_q and C_l can be merged if $\mathcal{A}_{ql} \geq 1$, and next it is easy to calculate the coefficient $\alpha(C_{ql})$ of the group $C_{ql} = C_q \cup C_l$:

$$\alpha(C_{ql}) = \frac{\mathcal{E}_q^{in} + \mathcal{E}_l^{in} + \mathcal{A}_{ql}}{\mathcal{E}_q^{in} + \mathcal{E}_l^{in} + \mathcal{E}_q^{out} + \mathcal{E}_l^{out} - \mathcal{A}_{ql}} \tag{40}$$

So, values $\alpha(C_{ql})$ can be calculated for all q, l pairs and then the best one is selected. Next, the partition $\mathcal{C}$ is updated by deleting C_q, C_l from it, but inserting C_{ql} and the process repeated until $|\mathcal{C}| = 1$, that is, all subsets are merged. If a data structure containing $\mathcal{E}_q^{in}, \mathcal{E}_q^{out}, \mathcal{A}_{ql}$ is available from the beginning, then data can be updated in linear time whenever two subsets are merged.

The pseudo code of the Algorithm Random Shrink is presented in Algorithm 1. Merging begins with clusters containing one node, see Line 4. While instructions lead the choice of C_q and C_l: in first iterations, e.g, $t \leq$ *max_random_step* in Line 7, subsets are chosen at random (Line 8), in order to diversify the search between different starts to explore different basins of attraction. After that, the algorithm continues in a greedy way, merging clusters that obtain the best local solution α_k (updated in Line 11). When some α_k is an optimum, values $\overline{\alpha}_k$ are updated (Line 15). Of course, when some $\overline{\alpha}_k(C_k)$ is updated, the corresponding optimal set C_k is updated as well (not reported in the pseudocode). Finally, the process is repeated *max_start* times, see Line 2, and the fact that the first merging are random could guarantee a sufficient diversification of the search.

When the Algorithm Random Shrink concludes, the researcher has at disposal persistence values $\overline{\alpha}_k$ for a range of k's, from which to select the best size k. How to select k is explained in the computational tests section (Section 5).

4.2. Random Shrink Interchange

After selecting the community size k, computational resources can be invested to improve the objective function α_k. The Interchange function (Function 2) attempts to maximize the α-value of a k-connected subset V_C of V by replacing one node at a time while keeping the connectivity. The function begins with the initial subset V_C of V containing the candidate nodes and continues to exchange an inner node h ($h \in V_C$) with an outer node h' ($h' \in (V - V_C)$) to obtain a better α-solution (updated in Line 3). When no more α-improvement can be found the function ends, see Line 7, returning the current k-connected subset V_C. The computational time of the function

Algorithm 1: Random Shrink

```
Input: $G = (V, E)$.
Result: $V_C = \{C_2^*, \dots, C_{n-1}^*\}$ where $C_k^*$ $(2 \leq k \leq n-1)$ is a $k$-connected
        subset of $V$.
Parameters: $max\_start$ repetition numbers, $max\_random\_step$, number of
            random choices.
$\overline{\alpha}^k = 0$ for $k = 2, \dots, n-1$
for $s = 1, \dots, max\_start$ do
    $\alpha^k = 0$ for $k = 2, \dots, n-1$
    $\mathcal{C}^0 = \{\{1\}, \dots, \{n\}\}$
    t = 0
    while $|\mathcal{C}^t| > 1$ do
        if  $t \leq max\_random\_step$ then
            Select randomly $C_q, C_l \in \mathcal{C}^t$, such that $G[C_q \cup C_l]$ is connected
            $\alpha(C_{ql}) = \alpha(C_q \cup C_l)$
        else
            $\alpha(C_{ql}) = \max\{\alpha(C_q \cup C_l) | C_q, C_l \in \mathcal{C}^t$, $G[C_q \cup C_l]$ connected $\}$
        $\mathcal{C}^{t+1} = (\mathcal{C}^t - C_q - C_l) \cup C_{ql}\}$
        $k = |C_{ql}|$: $\alpha_k = \max\{\alpha(C_{ql}), \alpha_q\}$
        t = t + 1
    $\overline{\alpha}^k = \max\{\overline{\alpha}^k, \alpha^k\}$, for $k = 2, \dots, n-1$.
```

can be high, due to the fact that, while it is fast to check whether a candidate entering node is connected to other nodes in V_C, a candidate exiting node can leave V_C only if it will not break the connectivity, that must be checked by an appropriate subroutine.

Function 2: Interchange

```
Interchange($G$, $V_C$)
    Input: $G = (V, E)$, $V_C$ is a $k$-connected subset of $V$.
    Result: a $k$-connected subset of $V$.
    while  True  do
        $\alpha(C_{h'}^h) = \max\{\alpha(C_i^j) | \forall i \in V_C, j \in (V - V_C)$ and $C_i^j$ connected$\}$
        if $\alpha(V_C) < \alpha(C_{h'}^h)$ then
            remove $h$ from $V_C$ and add $h'$ to $V_C$
        else
            return $V_C$
```

We combine the algorithm Random Shrink with the Interchange function obtaining a new algorithm that we will call Random Shrink Interchange described in Algorithm 3.

4.3. Random Shrink Variable Neighborhood Search

After the application of the Interchange function, a first optimal solution $V_C \subset V$, $|V_C| = k$ has been determined. The Tree Variable Neighborhood Search function (Func-

```
Algorithm 3: Random Shrink Interchange
  Input: $G = (V, E)$, $k$ community size.
  Result: $C_k^*$ $(2 \leq k \leq n-1)$ is a $k$-connected subset of $V$.
  Parameters: $max\_start$ repetition numbers, $max\_random\_step$, number of
              random choices.
  Let $C_k$ be the $k$-connected subset of $V$ resulting from the Algorithm 1 applied
    to $G$
  $C_k^* =$ Interchange$(G, C_k)$
```

tion 4) attempts to improve the α-value of V_C by replacing a subset $\mathcal{R}$ of V_C with a random subset of nodes $\mathcal{I}$ taken from the neighbors of V_C and obtaining a new solution $V_C' = (V_C - \mathcal{R}) \cup \mathcal{I}$. Next, the Interchange function is applied to V_C', hopefully to find a new local optimum. If V_C is almost optimal, this way of diversifying the search can be effective.

Calculating V_C' from V_C can be computationally demanding as $V_C' = (V_C - \mathcal{R}) \cup \mathcal{I}$ must be connected, that is the reason why we implement a subroutine that exploits the connectivity of a random spanning tree of G_C. Specifically, $\mathcal{R}$ is obtained in the following way. First, a random node $v \in V_C$ (Line 3) is selected and a random spanning tree $T = (V_C, E_T[G_C])$ is constructed from the root v. Next, we determine the set $\mathcal{L}$ of nodes that are leaves of T and select a random number h in the range $h \in \{2, \ldots, |\mathcal{L}|\}$. Finally, the exiting nodes are a random set $\mathcal{R} \subseteq \mathcal{L}$, such that $|\mathcal{R}| = h$. The way in $\mathcal{R}$ is selected guarantees that the subgraph having vertices set $V_C - \mathcal{R}$ and edges set $E[V_C - \mathcal{R}]$ is connected. The subset $\mathcal{I} \subset V$ such that $|\mathcal{I}| = h$ is selected at random from the neighboring nodes (different from those belonging to the set $\mathcal{R}$) of $V_C - \mathcal{R}$. The subset of V so obtained is optimized with the Interchange function (Line 6). Finally, the process is repeated max_start times to obtain different starting solutions V_C', see Line 2.

```
Function 4: Tree Variable Neighborhood Search
1 VNS($G$, $V_C$)
      Input: $G = (V, E)$, $V_C$ is a $k$-connected subset of $V$.
      Result: a $k$-connected subset of $V$.
      Parameters: $max\_start$ repetition numbers.
2     for $s = 1, \ldots, max\_start$ do
3         select randomly a node $v \in V_C$
4         select randomly a subset $\mathcal{R} \subset V$ build from the leaf of a random
            spanning tree of $v$
5         select randomly a subset $\mathcal{I}$ from the neighbors of $V_C - \mathcal{R}$ of the same
            size of $\mathcal{R}$
6         $\overline{V_C} =$ Interchange$(G, (V_C - \mathcal{R}) \cup \mathcal{I})$
7         if $\alpha(V_C) < \alpha(\overline{V_C})$ then
8             replace $V_C$ with $\overline{V_C}$
9     return $V_C$
```

We combine the algorithm Random Shrink Interchange with the Tree Variable Neighborhood Search function obtaining a new algorithm that we will call Random

Shrink Variable Neighborhood Search described in Algorithm 5.

```
Algorithm 5: Random Shrink Variable Neighborhood Search
  Input: $G = (V, E)$, $k$ community size.
  Result: $C_k^*$ $(2 \leq k \leq n-1)$ is a $k$-connected subset of $V$.
  Parameters: max_start_greedy repetition numbers greedy alg ,
              max_random_step, number of random choices, max_start_vns
              repetition numbers vns function .
  Let $C_k$ be the $k$-connected subset of $V$ resulting from the Algorithm 3 applied
    to $G$
  $C_k^* =$ VNS$(G, C_k)$
```

4.4. Constrained Random Restart

Even though Random Shrink Variable Neighborhood Search algorithm is a flexible tool to explore local optima that are close to the incumbent solution, still it could be the case that the global optimum is much farther away and it can be detected only selecting a completely different starting solution. As the Algorithm 5, Constrained Random Restart algorithm tries to improve a set of starting solutions through the Interchange function, but in this new algorithm the starting solutions are selected at random. Still, as it is important to explore different basins of attraction, we have included a distance constraint about how new starting solutions are selected. First, a new starting node c must have a distance from the previously selected nodes (set $\mathcal{P}$) of at least *min_distance* (Line 9) and then a new community is built at random around c. When, due to distance constraints, c cannot be determined, a new V_C is determined from scratch and the process is repeated until a number of *max_start* solutions is attempted (Algorithm 6).

5. Computational experiments

All algorithms have been implemented in `C++` language. The exact solution is calculated using the solver GuRoBi described in Gurobi Optimization, LLC (2022). The simulations have been performed on an iMac Pro 3.2 GHz 8-Core Intel Xeon W with 32 GB of ram.

5.1. Integer Programming versus the random Shrink Algorithm

Here we compare the effectiveness of the integer programming problem formulation (ILP) with the shrink heuristic. The former is an exact method whose solution time will turn out prohibitively large as the problem size overcomes a certain threshold. The latter is an approximate method in which the suboptimal solutions are calculated fast time. The trade-off between accuracy and speed can be established as long as the ILP can reach the solution in a reasonable time. For the following experiment, we simulate a random graph with a hidden community. The random graph is obtained as an Erdős-Rényi process with parameter $p = 0.1$. A subset of nodes of size $k = \lceil |V|/2 \rceil$ forms a community, characterized by being another Erdős-Rényi graph with parameter

Algorithm 6: Constrained Random Restart

```
Input: $G = (V, E)$, $k$ community size.
Result: $V_C^b$ a $k$-connected subset of $V$.
Parameters: $\mathcal{D}$ distance function, $min\_distance$ minimum distance between
            the starting node, $max\_start$ repetition numbers.
$it = 0$
while $it < max\_start$ do
    randomly select a node $c \in V$
    $\mathcal{P} = \{c\}$
    randomly built a $k$-connected subset $V_C$ of $V$
    $V_C^b = \texttt{Interchange}(G, V_C)$
    do
        randomly select a node $c \in V$ such that $\mathcal{D}(\mathcal{P}, c) \geq min\_distance$
        if $c$ is found then
            $it = it + 1$
            add $c$ to $\mathcal{P}$
            randomly built a $k$-connected subset $V_C$ of $V$
            $\overline{V_C} = \texttt{Interchange}(G, V_C)$
            if $\alpha(\overline{V_C}) > \alpha(V_C^b)$ then
                $V_C^b = \overline{V_C}$
            end
        else
            break
        end
    while $it < max\_start$
end
```

$p_c = 0.3$. The purpose of the experiment is to compare the trade-off between solution quality and computational time for the ILP model and the Random Shrink heuristic. Experiments have been run for $n \in \{25, 30, 35, 40\}$. The ILP model has been run with the true value $k = \lceil |V|/2 \rceil$. The RS heuristic has been run with parameters *max_start* = 1000 and *max_random_step* = $|V|/3$.

Computational results are reported in Tables 1 and 2. In particular, for each set of graphs, tables report the objective functions of the ILP model and RS heuristic, with the corresponding computational times. Additionally, for graphs of major size, the objective function of the upper bound has been computed. In Table 1 we can see how computational times increase as the number of nodes of V increase. We can see that exact and heuristic objective functions are the same in almost all the problems. The only exception is the fifth problem in the set of graphs with $n = 35$, in which the approximate persistence is 0.7% less than the optimal, and the second problem in the set of graphs with $n = 40$, in which the approximation is 0.6% less. It can be seen that the RS computation times increase smoothly, from around 0.2 seconds to 0.5 seconds, making the method fully practical. Conversely, as expected, the ILP model takes more time. Time is around 2 seconds for $n = 25$, 10 seconds for $n = 30$, and next we see that it is greatly variable. For $n = 35$ we see that 8 problems are solved in less than one minute, but one required more than 2 minutes, while for $n = 40$ we observe 5 problems solved in less than one minute, but one required almost one hour. This suggests that $n = 40$ is the threshold beyond which we observe the exponential growth of computational times, therefore, to get the results of Table 2 we imposed a time limit of 600 seconds. In this table it can be seen that exact (or truncated exact) and heuristic persistence values are very close. In 12 instances they reach the same solution, while in 6 cases ILP values are better than the RS, the converse occurs 2 times. On average, the two methods are very close in term of the objective function. Of course, they strongly differs for what concerns the computing times: in 5 instances for $n = 45$ and 8 instances for $n = 50$ the time limit of 10 minutes has been overcome, while the heuristic always concluded in less than 1 second. Moreover, not reported in Tables, it is worth to note that the RS heuristic terminates with approximate persistence values α_k for all $k = 2, \ldots, n-1$, values that could be calculated by the ILP model only though separate runs for k. This test suggests that the RS heuristic is a fast and reliable method to calculate persistence values *for all* $k = 2, \ldots, n-1$. The strategic value of this method is discussed in the following section.

5.2. Determining the correct community size k

In this section, we test the proposed algorithms of Section 4 on a family of simulated networks that have been generated according to the methodology proposed in Lancichinetti et al. (2008). That procedure generates synthetic networks that are as close as possible to real networks, which are often characterized by a high variability in the nodes degree. We simulate networks of n nodes so that each node degree is a random value taken from a power law distribution with parameter γ, minimum and maximum degree k_{min} and k_{max}, respectively, and average degree $\langle k \rangle$. Then, nodes of the graph are partitioned into communities, using the mixing parameter $\mu \in (0, 1]$. This parameter represents the fraction of edges that starting from a given node points to nodes outside the community. Conversely, the complement $1 - \mu$ is the fraction of edges outgoing from a node and pointing to nodes inside the community. The size of each community is a random value taken from a power law distribution with parameter β, and it ranges between the minimum s_{min} and the maximum s_{max}. The procedure partitions nodes of the network with each node being assigned to only one community. We simulate $N = 1000$ graphs of size $n \in \{20, 25, 30, 40, 50, 100, 150, 200\}$. For each graph, we set the mixing parameter $\mu = 10\%$, the average node degree $\langle k \rangle = 30\%$, and s_{min} and s_{max} equal to 20% and 50%, respectively, parameters γ and β are set to 2 and 1, respectively, that are the lowest values of the intervals indicated in Lancichinetti et al. (2008).

As we discussed earlier, the computation of the optimal persistence is flawed by the mere fact that the index tends to increase just as the cardinality of the communities increases. In particular, the value of α is close to zero, when communities are small and connected with many other nodes outside the community, and it approaches to one, when communities are composed of almost all the nodes of the network connected mostly with nodes inside the community. An example is reported in Figure 2, that depicts, for a simulated graph, the curve of the persistence α_k as the community size k increases. As it can be seen, the trend of the curve is almost increasing, providing an evidence that the global maximum of the function cannot be the unique criterion to select communities. Nevertheless, a closer inspection of the figure reveals that there are some local maxima determined for intermediate value of k: as better motivated later, we guess that they are the correct values of k that determine communities. Therefore, it is important relying on a fast and accurate method for drawing histograms as in Figure 2, that we will call *persistence curve*.

Nodes	Arcs	f_{LP}	$time_{LP}$	f_H	$time_H$
25	59	0.630	2.156	0.630	0.106
	54	0.667	2.732	0.667	0.133
	51	0.703	2.228	0.703	0.132
	57	0.700	2.703	0.700	0.113
	52	0.711	2.127	0.711	0.131
	54	0.739	1.686	0.739	0.120
	44	0.735	1.507	0.735	0.137
	43	0.700	1.579	0.700	0.131
	47	0.735	1.220	0.735	0.108
	49	0.700	1.786	0.700	0.125
30	78	0.677	8.353	0.677	0.232
	80	0.689	14.532	0.689	0.201
	76	0.704	5.411	0.704	0.227
	75	0.673	22.090	0.673	0.202
	70	0.717	3.788	0.717	0.201
	69	0.776	1.777	0.776	0.184
	62	0.653	14.295	0.653	0.176
	65	0.667	24.833	0.667	0.204
	59	0.698	10.453	0.698	0.190
	66	0.679	14.351	0.679	0.207
35	102	0.671	42.136	0.671	0.659
	111	0.644	134.007	0.644	0.325
	102	0.737	16.069	0.737	0.326
	102	0.648	107.378	0.648	0.302
	94	0.676	59.733	0.671	0.315
	99	0.730	6.270	0.730	0.300
	87	0.698	16.997	0.698	0.284
	85	0.703	12.907	0.703	0.276
	85	0.672	33.147	0.672	0.301
	102	0.705	25.897	0.705	0.282
40	131	0.680	133.242	0.680	0.412
	142	0.624	5000.075	0.620	0.782
	126	0.703	44.065	0.703	0.350
	140	0.673	333.786	0.673	0.345
	126	0.709	37.604	0.709	0.357
	134	0.651	380.774	0.651	0.388
	109	0.667	115.943	0.667	0.420
	114	0.678	51.680	0.678	0.375
	109	0.693	46.568	0.693	0.346
	132	0.704	36.186	0.704	0.335

Table 1.: Objective functions and computation times of the ILP model and the RS heuristic.

Nodes	Arcs	f_{LP}	$time_{LP}$	f_H	$time_H$	f_{UB}
45	169	0.676	572.419	0.676	0.879	0.676
	178	0.630	TL	0.630	0.547	0.764
	167	0.733	109.914	0.733	0.493	0.733
	174	0.703	303.744	0.697	0.528	0.703
	171	0.710	229.366	0.698	0.503	0.710
	169	0.620	TL	0.625	0.540	0.695
	152	0.658	TL	0.656	0.543	0.700
	144	0.681	TL	0.681	0.508	0.734
	152	0.680	TL	0.667	0.569	0.750
	164	0.676	249.637	0.676	0.735	0.676
mean		0.678		0.674	0.584	0.714
50	207	0.648	TL	0.648	0.728	0.730
	205	0.624	TL	0.628	0.647	0.732
	208	0.716	273.679	0.716	0.649	0.716
	204	0.680	TL	0.680	0.602	0.737
	202	0.683	TL	0.683	0.654	0.726
	207	0.637	TL	0.630	0.592	0.724
	197	0.621	TL	0.621	0.633	0.739
	185	0.677	537.094	0.677	0.591	0.677
	190	0.650	TL	0.650	0.628	0.898
	210	0.695	TL	0.695	0.607	0.713
mean		0.663		0.663	0.633	0.739

Table 2.: Objective function, upper bunds and computation times of the ILP model and the RS heuristic, time limit TL = 600.

Algorithm 1 has been developed for the purpose. It has been tested for graphs of size ranging from $n = 20$ to $n = 200$. As already pointed out, for each size, 1000 graphs have been generated and the algorithm is run, determining an approximate value of α_k for all $k \leq n$. We set the number *maxit* equal to 100, 1000 and 10000 starting solutions and we summarize computational results in Table 3. The left side of the table reports the objective function (i.e., the persistence probability). It is measured as follows: for each instance (and depending on the size n), we calculate the index[2] $f_n = \sum_{k=2}^{n-1} \alpha_k$, then, we compute the average of f_n on all the instances for each value of n.

The two central columns report the probability of improvement when the number of starting solutions increases from 100 to 1000 and from 100 to 10000. It is measured, for

[2] In the sum we exclude the extreme cases $k = 1$ and $k = n$ that correspond to trivial cases.

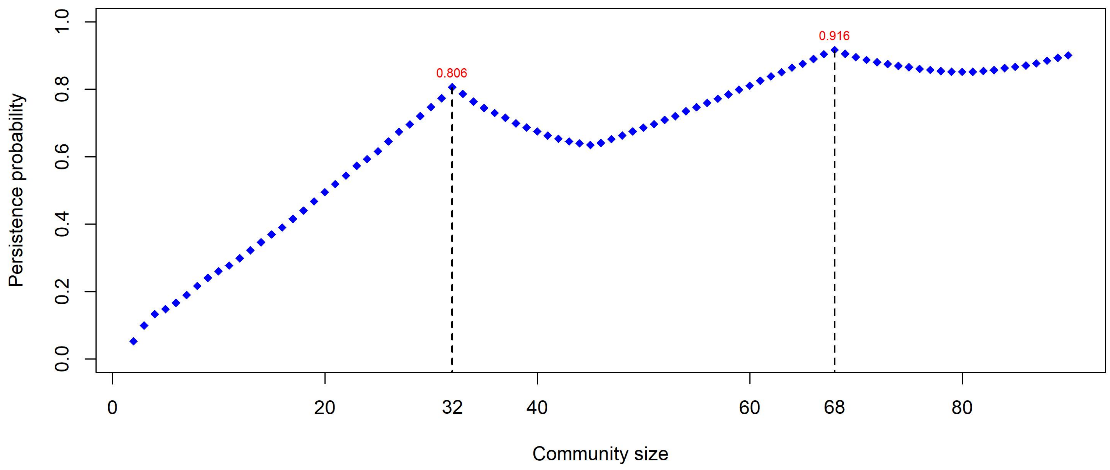


Figure 2.: Example of persistence curve.

each instance of size n, as the relative frequency in which α_k computed for $maxit = 1000$ (or for $maxit = 10000$, respectively) is greater than α_k computed for $maxit = 100$. Then, we calculate the average on all the instances for each value of n. It can be noticed that, when $maxit$ increases, the corresponding increase of the persistence probability is quite marginal, always less than 1%. However, these results have to be read in the light of the probability of improvement. This probability significantly increases as n grows, for instance passing from 4% for $n = 20$ to 58% for $n = 200$ in the case in which $maxit$ passes from 100 to 1000.

Computational times are reported as average values in the right side of the table. Even though they linearly increase, they could be severe when more starting solution are allowed. For example, considering $n = 200$, solutions with $maxit = 100$ has been obtained in 2 minutes, with $maxit = 1000$ in 20 minutes, with $maxit = 10000$ in more than 3 hours (note that times are multiplied by 10). However, the purpose of Algorithm 1 is to obtain a quick histogram of the α_k values, and the table reveals that reliable data can be obtained with $maxit = 100$.

From the analysis of a persistence curve, as the one reported in Figure 2, a researcher can guess the correct community size k^* as the one corresponding to a local maximum. Actually, the persistence curve can contain more than one peak, because the networks can contain more than one community or by mere numerical reasons. In practical applications, we guess that some further substantive analyses may be carried on the communities corresponding to each peak to establish whether they are realistic clusters. Nevertheless, some indications can be inferred by using some automatic procedure. In our tests, we apply two selection rules. Suppose that $\{k_1^*, \ldots, k_l^*\}$ is the set of the sizes corresponding to the local maxima of the persistence curve. The first rule is taking the smallest value k_1^*, the second one is taking the median value k_m^* (with $m = \lfloor (l+1)/2 \rfloor$). The experimental graphs simulated by the algorithm of Lancichinetti et al. (2008) usually contain more than one community, so let $\{k_1, \ldots, k_r\}$ be the set of sizes of the r simulated communities. Therefore, to check the effectiveness of our procedure we control whether $k_i^* \in \{k_1, \ldots, k_r\}$ with $i = 1$ or m. Computational results are described in Table 4. There, in the first two columns and for the 1000

	Persistence			Probab.		Times		
$maxit$ n	100	1000	10000	1000	10000	100	1000	10000
20	11.79	11.82	11.82	0.04	0.04	0.05	0.64	6.61
25	14.95	15.01	15.02	0.07	0.07	0.13	1.32	14.81
30	18.42	18.51	18.52	0.10	0.11	0.26	2.68	29.52
40	23.97	24.11	24.13	0.15	0.17	0.83	8.46	87.18
50	29.14	29.32	29.35	0.20	0.24	2.01	20.30	207.10
100	54.94	55.27	55.50	0.38	0.47	31.25	315.24	3175.45
150	81.52	82.10	82.40	0.49	0.60	152.96	1540.27	15277.62
200	105.93	106.80	107.24	0.58	0.69	480.37	4880.58	47114.58

Table 3.: Persistence mean and computational times Algorithm 1 depending on graph size.

n	p.k first	p.k median	p.k atleast	p.k all
20	0.812	0.735	0.948	0.465
25	0.771	0.665	0.956	0.426
30	0.716	0.606	0.949	0.470
40	0.824	0.711	0.988	0.602
50	0.843	0.814	0.997	0.673
100	0.866	0.860	1	0.754
150	0.823	0.846	1	0.757
200	0.777	0.779	1	0.682

Table 4.: Probabilities of finding a correct value of community size k.

graphs generated by each network size n, we have reported the relative frequency with which it has been observed $k_i^* \in \{k_1, \ldots, k_r\}$. As it can be seen, for networks of small dimension, k_1^* is better than k_m^*, but the relation reverses for the largest networks. Nevertheless, and in both cases, they are correct guesses for the large majority of the instances. In more than 10% of the cases, k_i^*, $i = 1$ or m are a wrong prediction, but further substantive analyses show that (third column) almost always at least one $k_i, i = 1, \ldots, r$ is between the guessed ones $k_i^*, i = 1, \ldots, l$. Moreover, in the fourth column we report how many times all values $k_i, i = 1, \ldots, r$ are contained in the guessed set $\{k_1^*, \ldots, k_l^*\}$ and this probability significantly increases when the size n grows. To summarize, most of the selected values k_i^* corresponds to true community sizes k_i and our algorithm can reveal them.

5.3. Interchange heuristics

Our final tests regard the computational analysis of the interchange heuristics. Next algorithm take as input the subset size k^* and the subset V_C determined by algorithm Random Shrink, then V_C is attempted to be improved first by the Interchange function, next by perturbing the optimal solution to restart the interchange from different initial clusters. Tested procedures require a minimal amount of parameters: in the Tree Variable Neighborhood Search function, the number of nodes that are replaced from the incumbent solution is a random number between 2 and k. The numbers of initial solution that are tested by Algorithm 5 and Algorithm 6 is 100. In Table 5 it can be seen the comparison between the three heuristics. Results are average values on 1000 test problems, in times the use of Random Shrink is included. In the first group of columns data about the plain Random Shrink Interchange are reported. In the first column (P.BtRS: Probability Better than Random Shrink), we report the relative frequency in which the Random Shrink Interchange could improve the Random Shrink solution; it can be seen that it happens, but less frequently than what expected. In the smallest sized problems, only 4% of the Random Shrink solutions were improved by the Random Shrink Interchange. Frequency of improved solutions increases with problem size; nevertheless, only 16% of the times the Random Shrink has been increased when the network size is $n = 200$. As whole, these data points that the Random Shrink is an effective heuristic. The second column (M.diff: mean difference), reports the relative increase of the objective function calculated only for the cases in which an improvement actually occurs. It can be seen that the improvement is more substantial on the smallest problem size than for the largest, indicating that there are cases in which the Random Shrink fails to find the optimal solution and that the improvement can be substantial. The last column (Time) reports the computational times, in which it can be seen that they are rather negligible, as the largest instances are solved in a few more than one second.

We compare the restricted diversification, e.g., Random Shrink Variable Neighborhood Search, in which starting solutions are generated close to the local optimum, with the free diversification, e.g. Constrained Random Restart, in which starting solutions are generated through consideration about their distance from previous analyzed regions. Looking at the frequency in which the Random Shrink solution has been improved, we can see that data are in favor of Constrained Random Restart, as they are improved from 9% of the times when the network size is 50 to 22% when the network size is 200 with respect to 6% to 18%. Solutions quality is better as well, improving of some 13% in all problem size with respect to less 10%. The only comparison in favor of Random Shrink Variable Neighborhood Search is about times that are some half the ones of Constrained Random Restart.

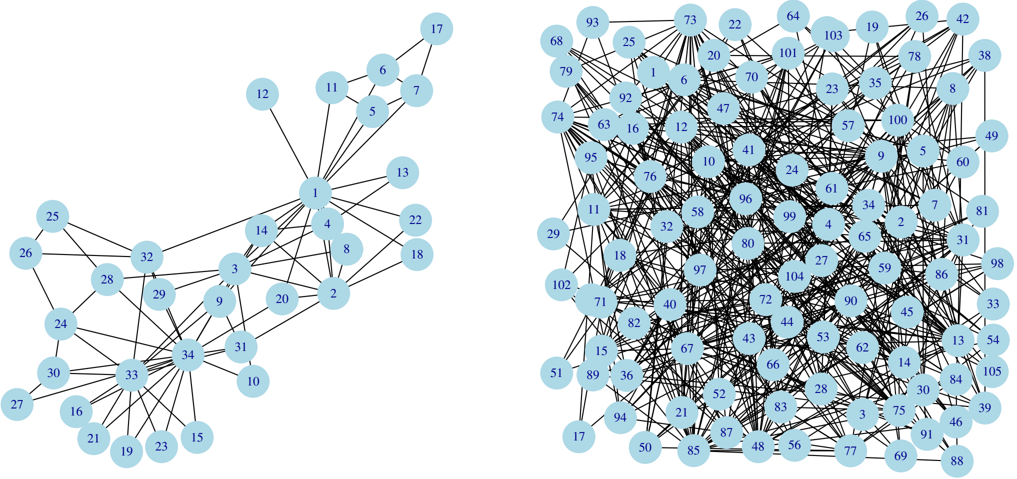
This is due to the fact that starting solutions generated closer to a local optimum are close to an other local optimum as well, but this implies that the diversification strategy is less effective. In summary, data are suggesting that the Constrained Random Restart is the most effective method to improve the results of the Random Shrink.

n	RSI 3			RSVNS 5			CRR 6		
	P.BtRS	M.diff	Time	P.BtRS	M.diff	Time	P.BtRS	M.diff	Time
50	0.04	0.07	0.02	0.06	0.10	0.05	0.09	0.12	0.31
100	0.10	0.03	0.15	0.13	0.04	0.72	0.16	0.14	2.85
150	0.14	0.02	0.46	0.16	0.04	2.69	0.20	0.13	10.02
200	0.16	0.01	1.14	0.18	0.03	11.30	0.22	0.13	28.60

Table 5.: Comparison between the number of times that a combined algorithm found a better solution w.r.t. the Random Shrink algorithm.

6. Application to real data

In this section, we make some computational tests on two simple real networks to verify the reliability of the persistence α to real applications, that is, if it can identify homogeneous groups of nodes interpreted as communities. Considered networks are the Zachary Karate Club (Zachary (1977)) and the political books (Krebs (2004)). The Zakary karate club is the network of friendships between the members of a club in a US university, while the political books is the network of co-purchased books about US politics published around year 2004 and sold online by Amazon.com. Figure 3 represents the topology of these networks.



(a) Karate Club network.

(b) U.S. Political books network.

Figure 3.: Network topologies.

These networks are classical examples used in many papers to compare new community detection proposals with methods already existing in the literature. Indeed, the two networks are quite different in terms of the number of nodes, average degree and density. The latter allows us to test the efficiency and robustness of the method and algorithms proposed in this work. Table 6 reports the basic characteristics of the networks.

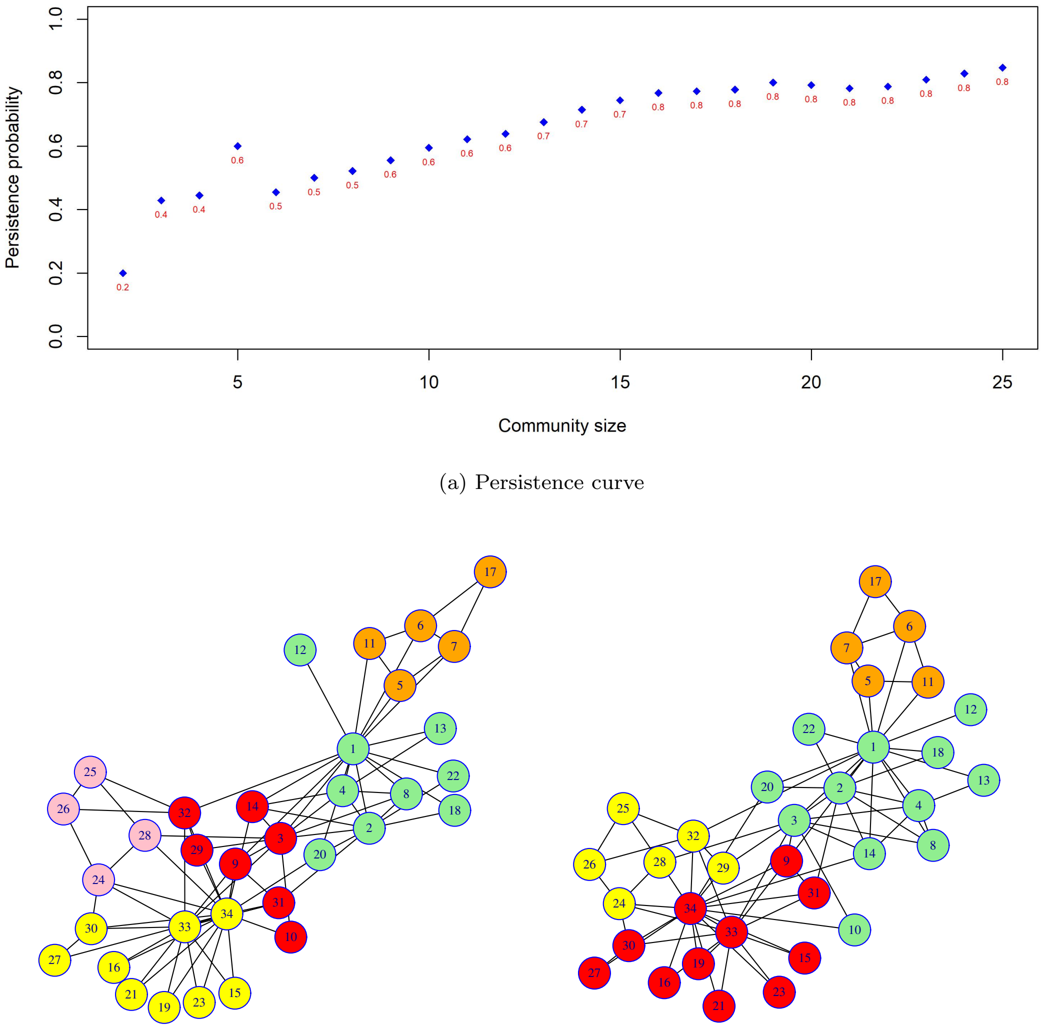
Network	N. of nodes	N. of edges	Average degree	Density
Karate Club	34	78	4.58	0.14
Political books	105	441	8.4	0.081

Table 6.: Network characteristics.

At first, we apply our methodology, e.g. Algorithm 1, to compute the peaks of the persistence probability α and then we improve those results with Constrained Random Restart. Finally, we compare our results with two well-known community detection methods, i.e. Walktrap and Louvain method ((Pons and Latapy, 2005) and (Blondel et al., 2008)). Similar to the idea of which the persistence probability the Walktrap method assumes that a random walk on a graph tends to remain inside a community. Conversely to the persistence probability, the community is detected using the specific structural distance between vertices and then a hierarchical clustering algorithm is applied. By analogy with the persistence probability, the Louvain method is based on the maximisation of an index, the modularity score, but the community is detected using spectral decomposition methods and not by optimization. These methods are designed to make community detection, that is, finding a partition of nodes, and therefore their output is composed of many communities, while our method is designed to find just one community. Nevertheless, we can discuss the consistency of the results obtained by the three methods.

Figure 4a reports the persistence probability (y-axis) varying k (x-axis) for the Zachary karate club network (figure 3a). The curve shows that there is an evident peak of the persistence probability for $k = 5$ corresponding to subset $V_C = \{5, 6, 7, 11, 17\}$, in which $\alpha(V_C) = 0.60$. This value is well above the persistence of communities of size 4 and 6 showing that those nodes form a well-separated cluster. The Walktrap and the Louvain method (see Figures 4b and 4c), detect this community as well: it is the group with orange color, a specific peripheral group of friends densely connected, but with few links with the rest of graph. Next, the persistence curve reveals a second local maximum, corresponding to the subset $V'_C = \{3, 9, 10, 15, 16, 19, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34\}$ of size $k' = 19$. With respect to the Walktrap communities, V'_C is the union of all the yellow nodes, all the pink nodes, all the red nodes except node 14; with respect to the Louvain communities, V'_C corresponds to all the red nodes, all the yellow nodes, and moreover the green nodes 3 and 10. It is worth noting that we also found a group of 10 nodes ($\tilde{k} = 10 = n - (k + k')$ in the persistence curve) that, even though it does not correspond to a local peak, it is composed of the remaining nodes of the green community, that is, excluding nodes 3 and 10. In conclusion, the persistence index revealed communities similar to the other methods, possibly allowing some aggregation and with some peripheral nodes resolved differently. Moreover, in this case, even though the method purpose is not finding a partition, still the outcome can be interpreted as such.

A similar representation is reported in Figure 5a for the network of the political books (see Figure 3b). We can observe three main peaks of persistence probability,



(a) Persistence curve

(b) Walktrap algorithm

(c) Louvain methodology

Figure 4.: Comparison of community detection methods and persistence probability for Zachary karate club network.

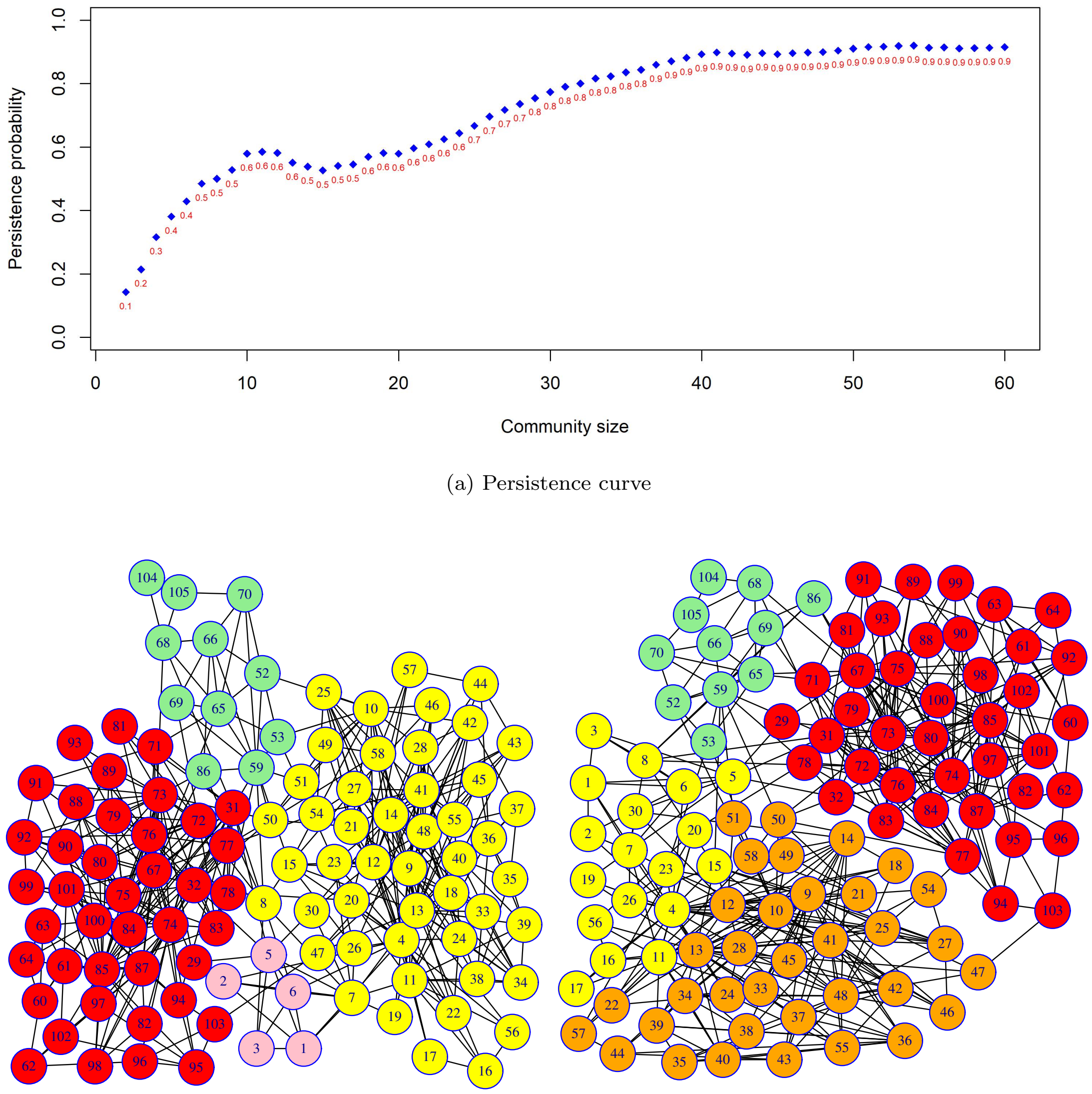
of values $\alpha(V_C) = 0.43$, $\alpha(V_C) = 0.58$ and $\alpha(V_C) = 0.89$, respectively, corresponding to $k = 5$, $k = 11$ and $k = 41$. Two of these communities corresponds to two communities determined by the Walktrap and Louvain methods as well. They are the two communities with $k = 11$, coloured in green in Figures 5b and 5c, and $k = 41$, coloured in red. For $k = 5$, the third community that we detected by the persistence index is different from what has been found by the two methods. The Walktrap method detects one community with the same size, $k = 5$, but different from the one we found, while the Louvain detects nothing. Our methods detects the community $V_C = \{60, 61, 63, 64, 100\}$ while the Walktrap detects $V_C = \{1, 2, 3, 5, 6\}$ (coloured in pink), but it is interesting to observe that for $k = 6$ our method detects the similar cluster $V_C = \{1, 2, 3, 5, 6, 7\}$.

It is worth observing that there is a structural difference between the results of our model and those of both the Walktrap and Louvain method. We do not require that all nodes of the networks belongs to a community, while the latter models do: our method is more flexible without loosing the property of analyzing the network as a whole, while the strict partition could be a too restrictive condition, as it is fully realistic that data reveal that some nodes of the graph are very cohesive between them. The rest of the nodes could be too loosely connected to claim that they are forming exclusive cliques, e.g. they do not show any relevant membership. This is exemplified by the small group detected in the Karate Club, that, at a close inspection, reveals few links outside the group, a property that is not shared by the rest of the nodes. Of course, we are not claiming that what was found by the other methods is wrong, but it is exactly what persistence probability does: it shows what communities are the most separated from the rest of the graph. This property is further supported through the analysis of the Political books. There, three different groups of books have been found: two of them correspond to clusters also found by the two other methods. This suggests that they are the two most customers' segmentation found by the other methods. On the other side, our method selected a third segment, or cluster, that for numerical reasons has not been detected by the other two. Again, and as for the Karate club, this small group could feature structural properties more consistent than what can be obtained through strict node partitioning.

7. Conclusions

In this work we presented the integer programming formulation of the problem of finding the node subset with maximum persistence probability and developed heuristic algorithms as well. Next, we showed how this methodology helps in discovering communities embedded in a real network by comparison of our findings with well-known methods of community detection.

There are two main difficulties in applying the persistence index. The first is that the optimal solution is hard to find. Actually, this is a problem shared by many other network statistics, but further research about heuristic procedures is worthwhile. The second difficulty is that the persistence index tends to increase with the subset size k and determining the value of k that corresponds to a community can be problematic. We overcome this issue by locating local peaks on the persistence curve, but further research could be devoted to determine other empirical rule to determine the exact value of k. Finally, our model is devoted to finding one community, but it can be used as a subroutine for a graph partitioning model.



(a) Persistence curve

(b) Walktrap algorithm

(c) Louvain methodology

Figure 5.: Comparison of community detection methods and persistence probability for U.S. political books network.

Acknowledgements

Research by Stefano Benati has been supported by the financial aid of NetMeetData: Ayudas Fundacion BBVA a equipos de investigacion cientifica 2019.
Research by Rosanna Grassi and Giorgio Rizzini has been financially supported by Fondo di Ateneo Quota Competitiva 2021, University of Milano - Bicocca.

References

Abello, J., M. Resende, and S. Sudarsky (2002). Massive quasi-clique detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2286*, 598–612.

Almeida, M. T. and R. Brs (2019). The maximum l-triangle k-club problem: Complexity, properties, and algorithms. *Computers and Operations Research 111*, 258–270.

Aloise, D., G. Caporossi, P. Hansen, L. Liberti, S. Perron, and M. Ruiz (2013). Modularity maximization in networks by variable neighborhood search. *Graph Partitioning and Graph Clustering 588*, 113–127.

Balasundaram, B., S. Butenko, and I. Hicks (2011). Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research 59*(1), 133–142.

Benati, S., D. Ponce, J. Puerto, and A. M. Rodriguez-Chia (2022). A branch-and-price procedure for clustering data that are graph connected. *European Journal of Operational Research 297*(3), 817–830.

Benati, S., J. Puerto, and A. Rodríguez-Chía (2017). Clustering data that are graph connected. *European Journal of Operational Research 261*(1), 43–53.

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

Calderoni, F., D. Brunetto, and C. Piccardi (2017). Communities in criminal networks: A case study. *Social Networks 48*, 116–125.

Charnes, A. and W. W. Cooper (1962). Programming with linear fractional functionals. *Naval Research Logistics Quarterly 9*(3-4), 181–186.

Dami, D., D. Aloise, and N. Mladenovi (2019). Ascentdescent variable neighborhood decomposition search for community detection by modularity maximization. *Annals of Operations Research 272*(1-2), 273–287.

Das, K., S. Samanta, and M. Pal (2018). Study on centrality measures in social networks: a survey. *Social Network Analysis and Mining 8*, 1–11.

Della Rossa, F., F. Dercole, and C. Piccardi (2013). Profiling core-periphery network structure by random walkers. *Scientific Reports 3*(1), 1–8.

Djeddi, Y., H. Haddadene, and N. Belacel (2019). An extension of adaptive multi-start tabu search for the maximum quasi-clique problem. *Computers and Industrial Engineering 132*, 280–292.

Fortunato, S. and D. Hric (2016). Community detection in networks: A user guide. *Physics Reports 659*, 1–44.

Girvan, M. and M. E. Newman (2002). Community structure in social and biological networks. *Proceedings of the national academy of sciences 99*(12), 7821–7826.

Gurobi Optimization, LLC (2022). Gurobi Optimizer Reference Manual. http://www.gurobi.com.

Hu, Y., H. Chen, P. Zhang, M. Li, Z. Di, and Y. Fan (2008). Comparative definition of community and corresponding identifying algorithm. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics 78*(2).

Kim, J., A. Veremyev, V. Boginski, and O. A. Prokopyev (2020). On the maximum small-world subgraph problem. *European Journal of Operational Research 280*(3), 818–831.

Krebs, V. (2004). Books about U.S. politics. Unpublished http://www.orgnet.com/.

Lancichinetti, A., S. Fortunato, and F. Radicchi (2008). Benchmark graphs for testing community detection algorithms. *Physical Review E 78*(4), 046110.

Luce, R. and A. Perry (1949). A method of matrix analysis of group structure. *Psychometrika 14*(2), 95–116.

Mahdavi Pajouh, F., Z. Miao, and B. Balasundaram (2014). A branch-and-bound approach for maximum quasi-cliques. *Annals of Operations Research 216*(1), 145–161.

Mokken, R. (1979). Cliques, clubs and clans. *Quality & Quantity 13*(2), 161–173.

Moradi, E. and B. Balasundaram (2018). Finding a maximum k-club using the k-clique formulation and canonical hypercube cuts. *Optimization Letters 12*(8), 1947–1957.

Pattillo, J., N. Youssef, and S. Butenko (2013). On clique relaxation models in network analysis. *European Journal of Operational Research 226*(1), 9–18.

Pattillo, J., V.-A. B. S. B. V. (2013). On the maximum quasiclique problem. *Discrete Applied Mathematics 161*, 244–257.

Peng, B., L. Wu, Y. Wang, and Q. Wu (2021). Solving maximum quasi-clique problem by a hybrid artificial bee colony approach. *Information Sciences 578*, 214–235.

Piccardi, C. (2011). Finding and testing network communities by lumped markov chains. *Plos One 6*(11), e27028.

Piccardi, C. and L. Tajoli (2012). Existence and significance of communities in the world trade web. *Physical Review E 85*, 066119.

Pinto, B., C. Ribeiro, I. Rosseti, and A. Plastino (2018). A biased random-key genetic algorithm for the maximum quasi-clique problem. *European Journal of Operational Research 271*(3), 849–865.

Pons, P. and M. Latapy (2005). Computing communities in large networks using random walks. In p. Yolum, T. Güngör, F. Gürgen, and C. Özturan (Eds.), *Computer and Information Sciences - ISCIS 2005*, Berlin, Heidelberg, pp. 284–293. Springer Berlin Heidelberg.

Radicchi, F., C. Castellano, F. Cecconi, V. Loreto, and D. Paris (2004). Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America 101*(9), 2658–2663.

Seidman, S. B., B. L. F. (1978). A graph theoretic generalization of the clique concept. *Journal of Mathematical Sociology 6*(1), 139–154.

Tang, W., L. Zhao, W. Liu, and B. Yan (2019). Recent advance on detecting core-periphery structure: a survey. *CCF Transactions on Pervasive Computing and Interaction 1*, 175–189.

Veremyev, A. and V. Boginski (2012). Identifying large robust network clusters via new compact formulations of maximum k-club problems. *European Journal of Operational Research 218*(2), 316–326.

Yu, H., P. A. T. V. G. M. (2006). Predicting interactions in protein networks by completing defective cliques. *Bioinformatics 22*, 823–829.

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

Zhou, Q., U. Benlic, and Q. Wu (2020). An opposition-based memetic algorithm for the maximum quasi-clique problem. *European Journal of Operational Research 286*(1), 63–83.

Zhou, Y. and J.-K. Hao (2017). Frequency-driven tabu search for the maximum s-plex problem. *Computers and Operations Research 86*, 65–78.