Community Detection by Modularity Maximization using GRASP with Path Relinking ⋆

Mariá C. V. Nascimento[1] and Leonidas S. Pitsoulis[2]

[1] Instituto de Ciência e Tecnologia, Universidade Federal de São Paulo (UNIFESP)
Rua Talim, 330 - Vila Nair, São José dos Campos, SP, Brasil
mcv.nascimento@unifesp.br

[2] Department of Mathematical and Physical Sciences, Aristotle University of Thessaloniki, Thessaloniki, Greece
pitsouli@gen.auth.gr

Abstract. Detection of community structure in graphs remains up to this date a computationally challenging problem despite the efforts of many researchers from various scientific fields in the past few years. The modularity value of a set of vertex clusters in a graph is a widely used quality measure for community structure, and the relating problem of finding a partition of the vertices into clusters such that the corresponding modularity is maximized is an NP-Hard problem. A Greedy Randomized Adaptive Search Procedure (GRASP) with path relinking is presented in this paper, for modularity maximization in undirected graphs. A new class of $\{0,1\}$ matrices is introduced which characterizes the family of clusterings in a graph, and a distance function is given which enables us to define an l-neighborhood local search which generalizes most of the related local search methods that have appeared in the literature. Computational experiments comparing the proposed algorithm with other heuristics from the literature in a set of some well known benchmark instances, indicate that our implementation of GRASP with path relinking consistently produces better quality solutions.

Keywords: Complex systems, community structure, graph clustering, modularity, GRASP, path relinking.

1 Introduction

Community detection in graphs is an interdisciplinary subject with a vast spectrum of applications, that has attracted the interest of many researchers in various fields in the past few years (for an extensive survey on the topic see [13]). Although there is no rigorous mathematical definition of the concept of community structure, the modularity value of a given vertex partition suggested in [27],

⋆ This work is a modified and improved version of [23] that is currently in submission is currently the most popular function used to quantify community structure in a graph.

There are few exact approaches in the literature for modularity maximization, and they can solve problems with a limited number of vertices [39, 7, 3]. On the contrary there is a plethora of heuristic algorithms which have appeared mainly within the physics community. A large class of heuristic algorithms is that of *greedy methods*, where initially every vertex of the graph is considered to be a cluster, and at every step of the algorithm a merge between two clusters is performed in a greedy fashion. Many variations of this theme have been suggested in order to decrease the computational effort, or improve the solution quality [8, 35, 40, 9, 36, 5]. Greedy methods can be characterized as being capable of handling large graphs in the order of 10^6 vertices, but with worse solution quality with respect to other methods. Other heuristic algorithms for community detection by modularity maximization are simulated annealing methods [16, 31, 22], extremal optimization [10], spectral clustering methods [26, 38], hybrid methods [29], integer rounding heuristics [1], genetic algorithms [20] and tabu search [4].

In this paper a heuristic procedure for finding a clustering of maximum modularity in a graph is presented, which can be characterized as a randomized multistart local search algorithm. A new class of $\{0, 1\}$ matrices is introduced which is in one-to-one correspondence with the clusterings in a graph, and it enables us to define a distance function that forms a metric space together with the family of clusterings. The paper is organized as follows. Section 2 contains all the preliminary information needed. Section 3 is where the GRASP with path relinking algorithm is developed which is composed of three main phases, a construction phase, a local search phase and a path relinking phase. In Section 3.1 we describe how to construct a clustering starting from the empty set, by following a randomized greedy strategy. Local search is described in Section 3.2, where initially we establish a notion of distance between two clusterings by providing an algebraic characterization of the clusterings, while in Section 3.3 we present the memory mechanism incorporated to our algorithm so as to use previous information to improve the current solution. Computational experiments that compare our algorithm with other algorithms from the literature in benchmark graphs are given in Section 4. Conclusions and further research are given in Section 5.

2 Modularity

Let $G = (V, E)$ be a graph where $V(G) := \{1, 2, \ldots, n\}$ is its set of nodes and $E(G)$ its set of edges, while $|E(G)| = m$. By a *clustering* $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ we mean a partition of $V(G)$, and we will refer to the $C_i \in \mathcal{C}$ as the clusters. *Modularity* is a quality function introduced by Newman and Girvan in [27] which quantifies the community structure by providing a value for every clustering of a given graph. The main idea is to employ a random graph on the same vertex set that does not have any community structure, and compare the edge density of the clusters in the original graph with the edge density of the clusters in the random graph. The greater the difference between the two edge densities, the more community structure the given clustering describes. The modularity of a given clustering $\mathcal{C}$ for some graph G, is defined as

$$Q(\mathcal{C}) := \frac{1}{2m} \sum_{C \in \mathcal{C}} \sum_{i,j \in C} \left(a_{ij} - \frac{d_G(i) d_G(j)}{2m} \right) \tag{1}$$

where a_{ij} is the number of edges between nodes i and j, and $d_G(i)$ is the degree if node i. In the case of a weighted graph the definition of modularity in (1) can be easily generalized as it is indicated by [28]. Given a weight function $w : E(G) \to \mathbb{R}$ on the edges of a graph, we can define the *strength* of a vertex $i \in V(G)$ as

$$s_G(i) := \sum_{j \in V(G)} w(i,j). \tag{2}$$

We can then write for the modularity of a clustering $\mathcal{C}$ for some weighted graph G

$$Q_w(\mathcal{C}) := \frac{1}{2 \sum_{e \in E(G)} w(e)} \sum_{C \in \mathcal{C}} \sum_{i,j \in C} \left(w(i,j) - \frac{s_G(i) s_G(j)}{2 \sum_{e \in E(G)} w(e)} \right). \tag{3}$$

To ease the notation we will employ the so called *modularity matrix*, which is an $n \times n$ matrix $M = (m_{ij})$ with entries defined as

$$m_{ij} := \begin{cases} a_{ij} - \frac{d(i)d(j)}{2m} & \text{if } G \text{ is unweighted,} \\ w(i,j) - \frac{s_G(i)s_G(j)}{2\sum_{e \in E(G)} w(e)} & \text{if } G \text{ is weighted,} \end{cases} \tag{4}$$

for $i, j = 1, \ldots, n$.

3 GRASP with Path Relinking for Community Detection

Greedy Randomized Adaptive Search Procedure (GRASP) is a randomized multistart local search algorithm which has been applied to a plethora of combinatorial optimization algorithms with favorable computational results [30]. In each iteration a solution is constructed starting from an empty solution in a greedy randomized fashion and a local search is applied to some specified neighborhood of the constructed solution. This solution is then used to examine the space formed with previously found solutions for further improvement.

The proposed algorithm GraspPR for community detection problem in networks by modularity maximization, is shown in pseudocode in Figure 1. The inputs to the algorithm is the graph under consideration, the parameter `iter` which is the maximum number of iterations, a random number seed `seed`, the size of the elite set of solutions `elitesize`, the greedy parameter α which indicates the degree of greediness in the construction phase of the algorithm, and the neighborhood size l for the local search. In lines 1 and 2 we initialize the best clustering $\mathcal{C}^*$ and the set of solutions $\mathcal{S}$ upon which path relinking will take

Algorithm: GraspPR

```
Input : graph $G(V,E)$, maximum iterations iter, random number seed
     seed, size of elite set elitesize, greedy parameter $\alpha \in (0,1)$, neigh-
     borhood size $l$
Output: clustering $\mathcal{C}^* = \{C_1, C_2, \ldots, C_k\}$ with maximum $Q(\mathcal{C}^*)$

 1. $\mathcal{C}^* := \emptyset$
 2. $\mathcal{S} := \emptyset$
 3. for $i = 1, \ldots,$ iter do
 4.    $\mathcal{C}^c :=$ ConstructClustering($G(V,E)$, seed,$\alpha$)
 5.    $\mathcal{C}^l :=$ LocalSearch($G(V,E), \mathcal{C}^c$, $l$)
 6.    $\mathcal{C}^p :=$ PathRelinking($G(V,E), \mathcal{S}, \mathcal{C}^l$, elitesize)
 7.    UpdateElite($\mathcal{S}, \mathcal{C}^p$)
 8.    if $Q(\mathcal{C}^p) > Q(\mathcal{C}^*)$ then
 9.       $\mathcal{C}^* := \mathcal{C}^p$
10.    end if
11. end for
12. return $\mathcal{C}^*$
```

Fig. 1. GRASP with path relinking for maximum modularity clustering

place. In the main iterations of the algorithm (lines 3 through 11) a solution $\mathcal{C}^c$ is constructed from the empty set in line 4, and it is passed to local search in line 5 where a l-neighborhood local maximum solution $\mathcal{C}^l$ will be computed. The resulting solution $\mathcal{C}^l$ is then passed to the path relinking procedure in line 6, where a search for a clustering with higher modularity will be performed in the space of solutions spanned by the difference of $\mathcal{C}^l$ and every solution from the elite set $\mathcal{S}$ with sufficient distance from $\mathcal{C}^l$. In line 7 the elite set used in the path relinking procedure is updated, while the best solution $\mathcal{C}^*$ found throughout the iterations is stored in lines 8 through 10. All three procedures in lines 4, 5 and 6 as well as the updating of the elite set of solutions, will be discussed in detail in the sections that follow.

3.1 Construction of a Clustering

The first task in every iteration of the GraspPR algorithm is to construct a clustering starting from the empty set, in a greedy randomized fashion. In the discussion that will follow let $\mathcal{C} = \{C_1, \ldots, C_{|\mathcal{C}|}\} \subseteq 2^{|V(G)|}$ denote a *partial clustering*, that is a clustering of the vertices of G that does not include the whole of $V(G)$. Consider the set of vertices not in $\mathcal{C}$ defined as

$$L(\mathcal{C}) := \{i \in V(G) : i \notin C, \forall C \in \mathcal{C}\}.$$

Let us now define a *gain* function $g_{\mathcal{C}} : V(G) \to \mathbb{R}$ on the elements of $L(\mathcal{C})$ for some partial clustering $\mathcal{C}$, which will indicate the increase in the modularity value $Q(\mathcal{C})$ upon the insertion of some vertex from $L(\mathcal{C})$ to the partial clustering $\mathcal{C}$. Note that some $i \in L(\mathcal{C})$ can be augmented to $\mathcal{C}$ either as a separate cluster $\{i\}$ or as a member of an existing cluster $C \in \mathcal{C}$. Thereby the gain function for some $i \in L(\mathcal{C})$ is defined as

$$g_{\mathcal{C}}(i) := \max \begin{cases} m_{ii}, & \text{if } \{i\} \text{ is a cluster in } \mathcal{C}, \\ \max_k \left(\sum_{j \in C_k} 2m_{ij} \right) + m_{ii}, & \text{if } i \text{ to be a included in some} \\ & \text{existing cluster in } \mathcal{C}, \end{cases} \tag{5}$$

where the m_{ij} are the entries of the modularity matrix defined in (4). Using the gain function $g_{\mathcal{C}}$ we can order the set $L(\mathcal{C})$ such that its elements appear in descending order, that is

$$\boldsymbol{L}(\mathcal{C}) := (i_{(1)}, i_{(2)}, \ldots, i_{(|\mathcal{C}|)}), \tag{6}$$

where

$$g_{\mathcal{C}}(i_{(1)}) \geq g_{\mathcal{C}}(i_{(2)}) \geq \cdots \geq g_{\mathcal{C}}(i_{(|\mathcal{C}|)}).$$

```
Algorithm: ConstructClustering

Input : graph $G(V, E)$, random seed seed, $\alpha \in (0, 1)$
Output: clustering $\mathcal{C}^c = \{C_1, C_2, \ldots, C_{|\mathcal{C}^c|}\}$

1. $\mathcal{C}^c := \emptyset$
2. $L(\mathcal{C}^c) := V(G)$
3. while $L(\mathcal{C}^c) \neq \emptyset$ do
4.    Compute $\boldsymbol{L}(\mathcal{C}^c)$
5.    Choose $i$ randomly among the first $\alpha$ elements of $\boldsymbol{L}(\mathcal{C}^c)$
6.    $\mathcal{C}^c := \mathcal{C} \cup \{i\}$
7.    $L(\mathcal{C}^c) := L(\mathcal{C}^c) - \{i\}$
8. end while
9. return $\mathcal{C}^c$
```

Fig. 2. Constructing a clustering

We can now state the algorithm ConstructClustering which is shown in Figure 2. In line 1 we initialize the partial clustering $\mathcal{C}^c$ to the empty set. At every iteration in lines 2-8, the ordered set of candidate vertices is computed in line 4 using the gain function as defined in (5), and from this set we choose some vertex i to be added to our clustering, among the best $\alpha \times |\mathcal{C}^c|$ vertices. Note here that the range of $\alpha \in (0, 1)$ defines the degree of greediness or randomness our constructed solution will have. That is for values of α close to 0 the algorithm behaves greedily while for values close to 1 it behaves randomly. In line 7 the selected vertex is added to the partial clustering, in a way so defined by its gain function.

3.2 Local Search

By local search for a optimization problem, we refer to the process upon which an exhaustive search is performed in the neighborhood of a given solution in order to derive a local optima. In order to specify the structure of the neighborhood of a solution, the notion of distance between any two solutions has to be defined. The following definition of a class of $\{0, 1\}$ matrices will provide us with the means for an algebraic treatment of clusterings in a graph.

Definition 1. *A matrix* $S = (s_{ij}) \in \{0, 1\}^{k \times n}$ *is called a* **basic clustering matrix** *if*

i) *it has no zero rows*
ii) $\sum_{i=1}^{k} s_{ij} = 1$ *for all* $j = 1, \ldots, n$
iii) *if* s_{ij} *is the first nonzero element of row* i *then* $s_{lt} = 0$ *for* $l = i + 1, \ldots, n$ *and* $t = 1, \ldots, j$.

If only conditions i) and ii) are satisfied then the matrix is called **clustering matrix**.

It is easy to show that given some graph with n vertices, there is a one-to-one correspondence between the set of clusterings of size k and the $\{0, 1\}^{k \times n}$ basic clustering matrices. For any clustering $\mathcal{C}$ we will denote with $S_{\mathcal{C}}$ its corresponding basic clustering matrix, and for any basic clustering matrix S we will denote with $\mathcal{C}_S$ its corresponding clustering. For a basic clustering matrix S we will denote with $\mathcal{M}(S)$ the set of $k!$ clustering matrices which can be generated by permuting its rows. Given any two clustering matrices $S \in \{0, 1\}^{k_1 \times n}$ and $T \in \{0, 1\}^{k_2 \times n}$ we define their *difference set* as the set

$$\Delta(S, T) := \{j : S_{ij} \neq T_{ij}, i = 1, \ldots, \min\{k_1, k_2\}, j = 1, \ldots, n\}, \tag{7}$$

which is the set of columns upon which these matrices differ. The *distance* between any two basic clustering matrices $S^1 \in \{0, 1\}^{k_1 \times n}$ and $S^2 \in \{0, 1\}^{k_2 \times n}$ is thus defined as

$$d(S^1, S^2) := \min\{|\Delta(S, T)| : S \in \mathcal{M}(S^1), T \in \mathcal{M}(S^2)\}. \tag{8}$$

So $d(S^1, S^2)$ is the minimum number of *moves* of elements within the clusters in the clusterings associated with the basic clustering matrices S^1 and S^2, needed to transform one clustering to another. The difference set of elements which defines the distance between S^1 and S^2 will be called the *minimum difference set*, and will be denoted by $\Delta^*(S^1, S^2)$. It is easy to show that the set of all basic clustering matrices for a given graph along with the distance function as defined in (8) forms a **metric space**.

Observe that direct computation of the distance as defined in (8) requires $(\min\{k_1, k_2\})!$ steps, since it suffices to permute the rows of the smaller basic clustering matrix in order to find the difference set of minimum cardinality. The constructive proof of the following theorem provides a polynomial time algorithm to compute the distance function defined (8).

Theorem 1. *Given two basic clustering matrices $S^1 \in \{0,1\}^{k_1 \times n}$ and $S^2 \in \{0,1\}^{k_2 \times n}$ their distance $d(S^1, S^2)$ can be computed in $\mathcal{O}(k^3)$ time, where $k := \min\{k_1, k_2\}$.*

Proof. The computation of the distance will be reduced to a linear assignment problem. Given the two basic clustering matrices $S^1 = (s^1_{ij})$ and $S^2 = (s^2_{ij})$, construct a $k \times k$ cost matrix $C = (c_{ij})$ as follows

$$c_{ij} := \sum_{l=1}^{n} |s^1_{il} - s^2_{jl}|, \tag{9}$$

for $i, j = 1, \ldots, k$. Consider now the following combinatorial optimization problem

$$\min_{p \in \mathcal{P}_k} \sum_{i=1}^{k} c_{p(i)i} \tag{10}$$

where $\mathcal{P}_k$ is the set of all permutations of the set $\{1, \ldots, k\}$. The optimum permutation to (10) corresponds to the permutation of the rows of the smaller basic clustering matrix, needed to produce the clustering matrix that defines the distance between S^1 and S^2. The problem in (10) is the so called linear assignment problem which can be solved efficiently in $\mathcal{O}(k^3)$ computational time (see [19]). □

The l-neighborhood of a clustering $\mathcal{C}$ can now be defined as

$$N_l(\mathcal{C}) := \{\mathcal{C}' \subseteq 2^{|V(G)|} : d(S_{\mathcal{C}}, S_{\mathcal{C}'}) \leq l\} \tag{11}$$

where $S_{\mathcal{C}}, S_{\mathcal{C}'}$ are the basic clustering matrices of $\mathcal{C}$ and $\mathcal{C}'$ respectively. So $N_1(\mathcal{C})$ contains all those clusterings which can be generated by $\mathcal{C}$ by moving one element from one cluster into another, including the case where this element can be removed from its cluster and be a cluster of its own. The set $N_2(\mathcal{C})$ contains $N_1(\mathcal{C})$, as well as all those clusterings generated by $\mathcal{C}$ by moving two elements from their clusters to other clusters etc.

We are now ready to describe the local search procedure in which starting from a clustering $\mathcal{C}^c$ as given by the algorithm ConstructClustering in Figure 2, we derive a clustering $\mathcal{C}^l$ which satisfies

$$\mathcal{C}^l = argmax\{Q(\mathcal{C}) : \mathcal{C} \in N_l(C^l)\}.$$

The local search algorithm LocalSearch is shown in Figure 3. In the main iterations of the algorithm in lines 2-11, we repeatedly derive the local optimum solution $\mathcal{C}^l$ in lines 5-10 in the l-neighborhood of our current solution $\mathcal{C}^*$. When the current solution is the local optimum solution we terminate the procedure. In this work we employed the 1-neighborhood for all local searchers, since the size of the neighborhood grows exponentially with l and for values of $l \geq 2$ it is computationally intensive to perform an exhaustive search.

Algorithm: LocalSearch

```
Input  : graph $G(V,E)$, clustering $\mathcal{C}^c$, neighborhood size $l$
Output: clustering $\mathcal{C}^l = \{C_1, C_2, \ldots, C_{|\mathcal{C}^l|}\}$

 1. $\mathcal{C}^l := \mathcal{C}^c$
 2. repeat
 3.    $\mathcal{C}^* := \mathcal{C}^l$
 4.    $N := N_l(\mathcal{C}^*)$
 5.    for all $\mathcal{C} \in N$ do
 6.       if $Q(\mathcal{C}) > Q(\mathcal{C}^l)$ then
 7.          $\mathcal{C}^l := \mathcal{C}$
 8.       end if
 9.       $N := N \setminus \{\mathcal{C}\}$
10.    end for
11. until $\mathcal{C}^l \neq \mathcal{C}^*$
12. return $\mathcal{C}^l$
```

Fig. 3. Local search

3.3 Path Relinking

Path relinking has been incorporated in GRASP for the improvement of the solution quality as well as for faster convergence to various combinatorial problems (see [11, 2, 33, 12, 24]). A thorough survey of how path relinking can be applied to GRASP is given in [32]. In path relinking we explore the space of solutions spanned by two good quality solutions, in hope of finding a better solution. Although it can be viewed as another type of local search where the region to be searched is defined by two solutions instead of one, its main function is to provide a memory mechanism to the algorithm since one of the solutions will be some good quality solution obtained from a previous iteration.

Given two clustering matrices $S_{\mathcal{C}_1}$ and $S_{\mathcal{C}_2}$ from the respective clustering $\mathcal{C}^1$ and $\mathcal{C}^2$ we define a second *difference set* as the set

$$\Delta'(S_{\mathcal{C}_1}, S_{\mathcal{C}_2}) := \{C_j^2 : \min_{1 \le i \le |\mathcal{C}_1|} ||S_{\mathcal{C}_1}^i - S_{\mathcal{C}_2}^j|| > 0, i = 1, \ldots, |\mathcal{C}_2|\}. \tag{12}$$

In describing the path relinking procedure, consider that we have two solutions $\mathcal{C}_1$ and $\mathcal{C}_2$. The space of clusterings *spanned* by $\mathcal{C}_1$ and $\mathcal{C}_2$ is

$$span(\mathcal{C}_1, \mathcal{C}_2) := \{\mathcal{C} \in 2^{|V(G)|} : \Delta^*(S_{\mathcal{C}}, S_{\mathcal{C}_1}) \subseteq \Delta^*(S_{\mathcal{C}_1}, S_{\mathcal{C}_2})\},$$

as defined by the set of all $2^{|d(S_{\mathcal{C}_1}, S_{\mathcal{C}_2})|}$ subsets of $\Delta^*(S_{\mathcal{C}_1}, S_{\mathcal{C}_2})$ which can be ordered by inclusion. Since the size of this space is exponentially large with respect to $d(S_{\mathcal{C}_1}, S_{\mathcal{C}_2})$, instead of generating all the solutions in this space in search of the one with the maximum modularity, we will trace in a greedy fashion

Algorithm: PathRelinking

```
Input  : graph $G(V,E)$, clustering $\mathcal{C}^l$, elite solution set $\mathcal{S}$, elitesize
Output: clustering $\mathcal{C}^p = \{C_1, C_2, \ldots, C_{|\mathcal{C}^p|}\}$

1.  $\bar{\mathcal{S}} := \{\mathcal{C} \in \mathcal{S} : d(S_{\mathcal{C}}, S_{\mathcal{C}^l}) \geq 4\}$
2.  if $|\mathcal{S}| <$ elitesize or $\bar{\mathcal{S}} = \emptyset$ then
3.      return $\mathcal{C}^l$
4.  end if
5.  for all $\mathcal{C} \in \bar{\mathcal{S}}$ do
6.      $S^0 := S_{\mathcal{C}}$
7.      $S^* := S_{\mathcal{C}}$
8.      for $k = 0, \ldots, d(S_{\mathcal{C}}, S_{\mathcal{C}^l}) - 2$ do
9.          max:=0
10.         for all $C_i^l \in \Delta^{'}(S^k, S_{\mathcal{C}^l})$ do
11.             $S$ :=change$(S^k, S_{\mathcal{C}^l}, C_i^l)$
12.             if $Q(\mathcal{C}_S) >$ max then
13.                 $i^* := i$
14.                 max:= $Q(\mathcal{C}_S)$
15.             end if
16.         end for
17.         $S^{k+1}$ :=change$(S^k, S_{\mathcal{C}^l}, C_{i^*}^l)$
18.         $\mathcal{C}$ := LocalSearch$(G(V,E), \mathcal{C}_{S^{k+1}})$
19.         if $Q(\mathcal{C}) > Q(\mathcal{C}_{S^*})$ then
20.             $S^* := S_{\mathcal{C}}$
21.         end if
22.     end for
23. end for
24. return $\mathcal{C}_{S^*}$
```

Fig. 4. Path relinking

a path of solutions from $\mathcal{C}_1$ to $\mathcal{C}_2$, perform a local search in the l-neighborhood of any solution in the path and keep the best solution found overall.

The path relinking procedure is shown in pseudocode in Figure 4. It will receive as inputs the graph under examination, a clustering $\mathcal{C}^l$ generated by the local search procedure as described in Section 3.2, the set of elite solutions $\mathcal{S}$, updated throughout the GRASPPR iterations. In line 1 we initialize the set $\bar{\mathcal{S}}$ which contains all solutions from $\mathcal{S}$ that have sufficient distance from $\mathcal{C}^l$. Note that $\mathcal{C}^l$ and every $\mathcal{C} \in \mathcal{S}$ are 1-neighborhood local maximum solutions, therefore if $d(\mathcal{C}^l, \mathcal{C}) \leq 3$ none of the solutions in $span(\mathcal{C}^l, \mathcal{C}) - \{\mathcal{C}^l, \mathcal{C}\}$ will have a higher modularity value than $Q(\mathcal{C}^l)$ or $Q(\mathcal{C})$. In lines 2-4 we will return the solution $\mathcal{C}^l$ unaltered as the result of the PATHRELINKING procedure, if either the size of the elite set is less than the parameter `elitesize` or there are no solutions of sufficient distance from $\mathcal{C}^l$. This will take place in the initial iterations of the GRASPPR algorithm where the elite set of solutions is constructed. In the loop defined in lines 5 through 23, a path of solutions from $\mathcal{C}$ to $\mathcal{C}^l$ will be generated and searched, for any clustering $\mathcal{C} \in \bar{\mathcal{S}}$. In lines 6 and 7 we initialize the starting and best solution, respectively. In each iteration k of the loop in lines 8 through 23 a solution S^k is generated such that

$$S^k := argmax\{Q(\mathcal{C}_S) : d(S^{k-1}, S) = |\Delta^{'}(S^{k-1}, S)|\},$$

until we reach $S_{\mathcal{C}^l}$. The local search is performed since we want to ensure that the solution produced by the path relinking procedure will be a 1-neighborhood local maximum.

Note that for the two basic clustering matrices S_1, S_2 from the respective clusterings C^1 and C^2 and some clustering $C_j^2 \in \mathcal{C}^2$ with $j = 1, \ldots, |\mathcal{C}^2|$, the function `change`$(S_1, S_2, C_j^2)$ in line 11 and 17 returns a basic clustering matrix S such that $\Delta^*(S, S_2) = \Delta^*(S_1, S_2) - C_j^2$. In the procedure `UpdateElite` in line 7 of the GRASPPR algorithm shown in Figure 1, any solution $\mathcal{C}^p$ produced by the path relinking procedure will be considered for possible insertion in the elite set of solutions $\mathcal{S}$. This will be done even if no path generation takes place in the case that the elite set is not of maximum size, or there are no solutions of sufficient distance from the current solution. If $|\mathcal{S}| <$ `elitesize` then any solution $\mathcal{C}^p$ will be added to the elite set. If $|\mathcal{S}| =$ `elitesize`, then the only criterion used in deciding whether a solution will be added to the elite set, thereby replacing an existing solution of the set, is the modularity value. So if

$$\mathcal{C}_{min} := argmin\{Q(\mathcal{C}) : \mathcal{C} \in \mathcal{S}\},$$

then a solution $\mathcal{C}^p$ will replace $\mathcal{C}_{min}$ if $Q(\mathcal{C}^p) > Q(\mathcal{C}_{min})$.

4 Computational experiments

In this section we will present the computational experiments that we performed in order to examine the performance of the GRASPPR algorithm for modularity maximization. In what follows, we will present the graphs used in the computational experiments, the heuristics used for comparison reasons, as well as the parameter settings of GRASPPR.

4.1 Experimental Results

The set of benchmark graphs used in our experimental study is presented in Table 1, where the number of vertices and edges is given. The parameter values for

Table 1. Benchmark graphs used in the computational experiments.

Data set	**Ref.**	n	m
karate	[41]	34	78
dolphins	[21]	62	159
lesmis	[18]	77	254
polbooks	-	105	441
adjnoun	[26]	112	425
afootball	[14]	115	613
jazz	[15]	198	2742
collab	[25]	235	415
celegans_neural	[37]	297	2148
celegans_metabolic	[10]	453	2025
email	[6]	1133	5451

the GraspPR are `iter`=500, $\alpha \in [0.1, 0.7]$ and `elitesize`=1. Since GraspPR has a random component, we run the heuristic 10 times and considered the median result. The heuristic algorithms used for comparison reasons can be found in the works of [10, 1, 34, 4].

Columns 2 through 5 in Table 2 report for each benchmark graph, the best known modularity value for the algorithm in the given reference, the number of clusters in the corresponding solution as well as the required time when available. Columns 6 through 8 give the same information for the proposed GraspPR algorithm, while columns 9 and 10 present the currently known upper bounds as reported in [1] and [3]. The former upper bound corresponds to the linear relaxation of a integer program whereas the latter corresponds to the optimum solution obtained by a column generation strategy.

It is worth mentioning that, in most of the cases, the best result achieved by [1] corresponds to the solution of two different heuristics (LP and VP). For this reason, in such cases, we reported the runtime of VP heuristic, that presented a lower average runtime than LP. Moreover, Rotta [34] proposed two different multi-level strategies (ML), one based on density ML-KL-density, and another based on random walks ML-KL-rw. We differentiate them by using a * when referring to a solution obtained by the ML-KL-density strategy. We do not report any computational times of the algorithms in [34] in Table 2. However, by the analysis given by the author in his PhD thesis, it is possible to conclude that, as a multi-level strategies they require less computational time than most metaheuristics. Nevertheless both strategies make use of the intensification algorithm

Table 2. Results of the comparison between algorithms for modularity maximization.

	Best known solution				GraspPR			Upper Bound	
Data set	source	$Q(\mathcal{C})$	$\|\mathcal{C}\|$	time	$Q(\mathcal{C})$	$\|\mathcal{C}\|$	time	source	$Q(\mathcal{C})$
karate	[1, 34]	0.420	4	6	0.420	4	0	[3]	0.420
dolphins	[1]	0.529	-	4	0.529	5	0	[3]	0.529
lesmis	[1]	0.560	-	4	0.560	6	0	[3]	0.560
polbooks	[1, 34]	0.527	5	12	0.527	5	1	[3]	0.527
adjnoun	[34]	0.310	-	-	0.311	6	1	-	-
afootball	[1, 34]	0.605	10	23	0.605	10	1	[3]	0.605
jazz	[10]	0.445	5	24	0.445	4	0	[1]	0.446
collab	[1]	0.803	-	105	0.803	51	18	[1]	0.805
celegans_neural	[34]	0.503	5	-	0.503	4	2	-	-
celegans_metabolic	[34]	0.451*	9	-	0.452	9	98	-	-
email	[34]	0.581	9	-	0.582	13	1565	-	-

from Kernighan and Lin [17], thereby it is expected that they require higher computational time than conventional ML algorithms.

According to the results of Table 2, we observe that the proposed hybrid metaheuristic, GraspPR is competitive with the best algorithms found in literature for the tested benchmark graphs. Moreover, the best results of the past heuristics for the benchmark graphs karate, dolphins, lesmis, polbooks and afootball, according to the corresponding upper bounds, cannot be improved since they are the optimal solutions. For these graphs, GraspPR also found an optimum solution. Concerning the other six benchmark graphs, in three of them GraspPR achieved the same solution value as past heuristics, while it outperformed them in the remaining three. Specifically, for the graph adjnoun, Rotta [34] found a partition with modularity 0.31078, whereas the solution found by GraspPR had a modularity of 0.311048. An improvement in the best solution found by past heuristics was also observed for the graphs celegans_metabolic. A more precise value of the solution found by GraspPR was 0.451687, whereas the solution found by ML-KL-density that was the best solution among past heuristics, was 0.45090. GraspPR achieved a very good result with the benchmark graph email. A more precise value of the modularity obtained was 0.581695 that is better than the best reported heuristic result, by [34], 0.58137.

Considering the other three graphs for which the proposed heuristic found competitive results, more precise values enable a better analysis of the solutions found. For the benchmark graph jazz, GraspPR achieved a slightly worse result than to the best reported. In this case, GraspPR found a partition with modularity 0.44514, whereas the best reported past heuristic result, by [1], was 0.4452. On the other hand, the solution value found by GraspPR for the graph collab was exactly the same as the best past heuristic. For the graph celegans_neural, the solution found by GraspPR was 0.503485, better than the best past heuristic, that achieved the solution value 0.50295.

According to the presented results, the use of GraspPR is recommend for small and medium graphs. It has to be noted that this version of the GraspPR is an improvement over the previous version that appears in [23], that did not perform as well for large graphs.

5 Conclusions

Computational results in a set of benchmark graphs, indicate that the proposed algorithm is superior and robust with respect to solution quality, in the sense that for the majority of the cases it produces better solutions than other heuristics from the literature while it had the least deviation in the cases where it did not find the best solution. Furthermore, for a set of frequently used benchmark graphs from the literature, it produced the best solution among all heuristics tested. The proposed algorithm however required greater computational time. Further research will be directed towards employing sophisticated data structures to reduce the computational time of the algorithm, so as to be able to solve within a few seconds problems involving large scale graphs (i.e. for $n \geq 10^6$).

Acknowledgements

The first author gratefully acknowledge FAPESP for its financial support.

References

1. G. Agarwal and D. Kempe. Modularity-maximizing graph communities via mathematical programming. *European Physical Journal B*, 66:409–418, 2008.
2. R.M. Aiex, M. Resende, P.M. Pardalos, and G. Toraldo. GRASP with path-relinking for the three-index assignment problem. *Parallel Computing*, 29:393–430, 2003.
3. D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, L. Liberti, and S. Perron. Column generation algorithmsfor exact modularity maximization in networks. *Physical Review E*, 82(4):046112, 2010.
4. A. Arenas, A. Fernández, and S. Goméz. Analysis of the structure of complex network at different resolution levels. *New Journal of Physics*, 10:053039, 2008.
5. V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E. Lefebrvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics*, P10008, 2008.
6. M. Boguña, R. Pastor-Satorras, and A. Arenas. Models of social networks based on social distance attachment. *Phys. Rev. E*, 70:056122, 2004.
7. U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikolosk, and D. Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20:172–188, 2008.
8. A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111–[6 pages], Dec 2004.
9. L. Danon, A. Díaz-Guilera, and A. Arenas. Effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 11010, 2006.

10. J. Duch and A. Arenas. Community identification using extremal optimization. *Physical Review E*, 72:027104, 2005.
11. P. Festa, P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. GRASP with path-relinking for the weighted maxsat problem. *ACM Journal of Experimental Algorithmics*, 2006.
12. P. Festa and M.G.C. Resende. GRASP: An annotated bibliography. In C. C. Ribeiro and P. Hansen, editors, *Essays and Surveys on Metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.
13. S. Fortunato. Community detection in graphs. *Physics Reports*, 486, 2010.
14. M. Girvan and M. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99:7821–7826, 2002.
15. P. Gleiser and L. Danon. Community structure in jazz. *Advances in Complex Systems*, 6:565–573, 2003.
16. R. Guimerá, M. Sales-Pardo, and L.A.N. Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2):025101, 2004.
17. B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49:291–307, 1970.
18. D.E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing.* Addison-Wesley, Reading, MA, 1993.
19. Eugene Lawler. *Combinatorial Optmization: Networks and matroids.* Dover Publications, Inc., 1976.
20. X. Liu, D. Li, S. Wang, and Z. Tao. Effective algorithm for detecting community structure in complex networks based on ga and clustering. In *Proceedings of the 7th International Conference on Computational Science*, pages 657–664. Springer-Verlag, 2007.
21. D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54:396–405, 2003.
22. A. Medus, G. Acu na, and C.O. Dorso. Detection of community structures in networks via global optimization. *Physica A*, 358:593–604, 2005.
23. M. C. V. Nascimento and L. O. Pitsoulis. Community detection by modularity maximization using GRASP with path-relinking. Submitted to Computers & Operations Research, 2011.
24. M.C.V. Nascimento, F.M.B. Toledo, and A.P.L.F. Carvalho. Investigation of a new GRASP-based clustering algorithm applied to biological data. *Computers & Operations Research*, 37:1381–1388, 2010.
25. M. Newman. Scientific collaboration networks. II. shortest paths, weighted networks, and centrality. *Physical Review E*, 64:016132, 2001.
26. M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74:036–104, 2006.
27. M .E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113–[15 pages], 2004.
28. M.E.J. Newman. Analysis of weighted networks. *Physical Review E*, 70:056131, 2004.
29. A. Noack and R. Rotta. Multi-level algorithms for modularity clustering. In *Proceedings of the 8th International Symposium on Experimental Algorithms (SEA 2009)*, volume 5526 of *Lecture Notes in Computer Science*, pages 257–268. Springer, 2009.
30. L. Pitsoulis and M.G.C. Resende. Greedy randomized adaptive search procedures. In *Handbook of Appied Optimization*, pages 168–181. Oxford University Press, 2002.

31. J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74:016110, 2006.
32. M. Resende and C. Ribeiro. GRASP with path-relinking: Recent advances and applications, 2005.
33. M.G.C. Resende and C.C. Ribeiro. A GRASP with path-relinking for private virtual circuit routing. *Networks*, 41(1):104–114, 2003.
34. R. Rotta. *A Multi-Level Algorithm for Modularity Graph Clustering*. PhD thesis, Brandenburgische Technische Universität Cottbus, 2008.
35. P. Schuetz and A. Caflisch. Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Physical Review E*, 77(046112), 2008.
36. K. Wakita and T. Tsurumi. Finding community structure in a mega-scale social networking service. In *Proceedings of IADIS International Conference on WWW/Internet*, pages 153–162, 2007.
37. D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.
38. S. White and P. Smyth. A spectral clustering approach to finding communities in graphs. In *Proceedings of SIAM International Conference on Data Mining*, pages 76–84, 2005.
39. G. Xu, S. Tsoka, and L.G. Papageorgiou. Finding community structures in complex networks using mixed integer optimisation. *Eur. Phys. J. B*, 60:231–239, 2007.
40. Z. Ye, S. Hu, and J. Yu. Adaptive clustering algorithm for community detection in complex networks. *Physical Review E*, 78:046115, 2008.
41. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.