Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 96 (2016) 485 – 494

Procedia Computer Science

20th International Conference on Knowledge Based and Intelligent Information and Engineering Systems

Growing Neural Gas as a memory mechanism of a heuristic to solve a community detection problem in networks

Camila Pereira Santos[a], Mariá C. V. Nascimento[a,*]

[a]*Instituto de Ciência e Tecnologia, Universidade Federal de São Paulo (UNIFESP)*
Av. Cesare M. G. Lattes, 1201, Eugênio de Mello, São José dos Campos-SP, CEP: 12247-014, Brasil

Abstract

Iterative heuristics are commonly used to address combinatorial optimization problems. However, to meet both robustness and efficiency with these methods when their iterations are independent, it is necessary to consider a high number of iterations or to include local search-based strategies in them. Both approaches are very time-consuming and, consequently, not efficient for medium and large-scale instances of combinatorial optimization problems. In particular, the community detection problem in networks is well-known due to the instances with hundreds to thousands of vertices. In the literature, the heuristics to detect communities in networks that use a local search are those that achieve the partitions with the best solution values. Nevertheless, they are not suitable to tackle medium to large scale networks. This paper presents an adaptive heuristic, named GNGClus, that uses the neural network *Growing Neural Gas* to play the role of memory mechanism. The computational experiment with LFR networks indicates that the proposed strategy significantly outperformed the same solution method with no memory mechanism. In addition, GNGClus was very competitive with a version of the heuristic that employs an elite set of solutions to guide the solution search.

Peer-review under responsibility of KES International

Keywords: Growing Neural Gas; Community Detection in Networks; Heuristic Methods

1. Introduction

Many real networks, commonly represented by graphs, are characterized by groups of highly related vertices. To detect communities in these networks, one may employ strategies based on the optimization of the well-known modularity measure [1]. The modularity maximization problem is NP-Complete [2], mostly approached by heuristics. The primary reason is that a lot of instance problems are intractable by optimization solvers that aim at proving their optimality.

Strategies optimizing modularity are widely employed achieving effective community structures in networks. The existing modularity maximization-based heuristics, besides aiming to find good heuristic solutions, usually tackle the issue of overcoming some unexpected partitions achieved by solving this problem. One reason is that, in spite of

* Corresponding author. Tel.: +55-12-3924-9500
E-mail address: mcv.nascimento@unifesp.br.

Peer-review under responsibility of KES International
doi:10.1016/j.procs.2016.08.110

the good quality of solutions found through this approach, maximizing modularity may be a bad choice to detect communities in networks. The drawbacks in modularity appear in networks whose scale of the community structures with regard to the size of the networks reaches a limit[3]. There are a few attempts to address this issue by adjusting the modularity in the literature.

This paper proposes an adaptive strategy that employs the neural network *Growing Neural Gas* (GNG) to optimize the adjusted modularity. The mapping of the search space by GNG to guide the solution method is a form of avoiding a local search strategy that is too costly. The computational experiment carried in this paper indicates that the introduced strategy outperformed the same strategy without memory mechanism. Moreover, it was very competitive with a version guided by the elite set of solutions, suggested in[4]. The performance analysis was in accordance with the performance profiles introduced in[5].

2. The modularity maximization problem

The community detection problem in networks can be defined as a combinatorial optimization problem, whose goal is to find a partition with the largest possible value for a quality measure. Among the existing measures to perform this task found the literature, it is worth mentioning the modularity[1] and the map equation[6].

Let $G = (V, E, \phi)$ be a weighted simple graph, where $V(G)$ is its set of vertices and $E(G)$ its set of edges. An element from $E(G)$ is represented by a tuple (u, v), where $u, v \in V(G)$. The function $\phi : E(G) \to \Re$ is a function that assigns a weigh to each edge of $E(G)$. The neighborhood of a vertex v, denoted by $\mathcal{N}(v)$, is composed of all vertices z such that $(v, z) \in E(G)$. Therefore, if $y \in \mathcal{N}(v)$, then y is a neighbor of v. A modularity formulation is presented in Equation (1).

$$q(\mathscr{C}) = \frac{1}{2m} \sum_{C \in \mathscr{C}} \sum_{i,j \in C} \left(\phi(i,j) - \lambda \frac{d_G(i) d_G(j)}{2m} \right) \tag{1}$$

where m is the total weight on the edges of the graph, $\mathscr{C}$ is a vertex partition and $d_G(i)$ is the degree of vertex i, defined as $d_G(i) = \sum_{j=1}^{|V(G)|} \phi(i,j)$ and λ is a parameter to be adjusted according to the graph topology (considering the original modularity formulation, it is set as 1). In unweighted graphs, the weight of each edge is unitary and null, if there is no edge between a pair of vertices. The expected weight between a pair of vertices i and j according to the null model is $\frac{d_G(i) d_G(j)}{2m}$. Then, the measure evaluates if a pair of vertices in the same community has a high clustering tendency if in a random graph with the same degree sequence as G this pair has a low expected number of edges between them.

The parameter λ aims at controlling the resolution limit pointed out in[3]. Reichardt et al.[7] proposed this parameter adjustment and, roughly, the lower its value, the better the measure for larger communities. Among the few attempts to automatically adjusting this measure is that proposed in[8]. The authors proposed the use of a multi-layer perceptron to adjust λ depending on the graph topology.

3. Proposed Solution Method

The *Growing Neural Gas* (GNG) network, proposed in[9], is a Self Organizing Map (SOM) network that combines characteristics of the *Neural Gas* (NG)[10] and the *Competitive Hebbian Learning* (CHL)[11]. According to[12], the network grows and self-adapts as the data are presented to it.

In this paper, the neural network will be modeled as a graph $H = (A, F)$, where A represents the nodes of the neural network, and F represents the edges between the nodes of the network. Each node $a \in A$ from the neural network stores a partition of the network G under consideration. The age of an edge c of the neural network is denoted by age_c.

To address the community detection problem, the proposed strategy considers the pairwise relationship between vertices in a partition the weights on the GNG network nodes. Each node of the GNG network is represented by a $n \times n$ matrix of weights w_a. An element in row i and column k in this matrix is denoted by $w_a^{i,k}$ and is valued equal of greater than 0.5 if vertices i and k are in the same community, and less than 0.5, otherwise. It is noteworthy that w_a is symmetric. Therefore, we can just consider the superior triangular matrix resulting in $\frac{|V|^2}{2} - |V|$ memory positions. The structure *nodes* is a vector of pointer to matrices of weights of the partitions corresponding to the nodes of the GNG network. The matrix of weights w_a corresponding to a partition of node a is obtained by $w_a = nodes(a)$.

GNG will be employed in the proposed community detection algorithm to insert an adaptive condition in the strategy. The general pseudocode of the proposed strategy, named GNGClus, is presented in Algorithm 1.

Algorithm 1: GNGClus

Data: $G = (V,E)$, maxIter
Output: Partition $\mathscr{C}*$

```
Define 𝒞 as singletons
nodes:= structure that contains the nodes of the GNG network
H = (A,F), A = {a1;a2}, nodes(a1), nodes(a2) := random partitions
for iter = 1 to maxIter do
    𝒞_C = Construction Phase(G, 𝒞, Q, α)
    if iter < 15 then
        GNG(H, nodes, iter_GNG , 𝒞_C)
    end
    else
        if Q(𝒞_C) ≥ 0.6Q̄ then
            GNG(H, nodes, iter_GNG, 𝒞_C)
        end
        if (i+1) mod 3 then
            Contractions GNG(G, H, nodes)
        end
    end
end
w*:= node with highest error in the GNG network
𝒞*:= partition obtained from w*.
```

Algorithm 1 has as input the graph G to detect communities and the maximum number of iterations of GNGClus. It returns the final communities in the set $\mathscr{C}^*$. Each node $a \in A$ of the neural network stores a partition of G. The GNG network initiates with two nodes, a and b, with random partitions. In order to define a random partition, each position of the weight matrix is randomly selected with real numbers between 0 and 1.

In the first iterations of GNGClus, every solution obtained is presented to the GNG network as an input. After a number of iterations of the proposed strategy, however, only if the current solution, $\mathscr{C}_C$, has an adjusted modularity higher than or equal to 60% of the average values obtained on the iterations, indicated as $\bar{Q}$, the partition $\mathscr{C}_C$ is presented as an input to the GNG network.

Additionally, at each three iterations, the vertices that appear in the same community in the three nodes with the highest errors of the GNG network are definitively contracted by the procedure Contractions GNG. This idea was suggested in[4], by considering an elite set containing the best solutions found by their algorithm. In the end, the node of the GNG network with the highest error, w^*, indicates the final communities.

The task of mapping the vertices to the communities according to this matrix is not direct. The reason is that the transitivity rule does not necessarily hold for the weight matrix. Therefore, this paper proposes a procedure to produce $\mathscr{C}^*$: for each vertex $i \in V(G)$, if i was not assigned to a community in $\mathscr{C}$, it must be assigned to a new community. Then, for each vertex $j \in V(G), i \neq j$, j is assigned to the same community as vertex i, if $w_{h^*}^{i,t} > 0.5$ for at least half of the vertices t already assigned to this community. The computational complexity of the proposed strategy is $O(|V|^2|E| + |V||E|^2)$ and each of its phases and procedures is explained in the next sections.

3.1. Construction Phase

In the proposed Construction Phase, the contraction of edges results in an operation of creating supernodes in the network. At each iteration of this phase, an edge is contracted until the stop criterion is reached. When it does, the resulting supernodes are the communities of the network, here denoted as $\mathscr{C}$. A community of $\mathscr{C}$ is referred as C.

The quality of an edge contraction, that means the partition resulting from the contraction of a pair of vertices v and c with regard to the former communities in $\mathscr{C}$, must be calculated. Equation (2) assesses this variation in the adjusted modularity, if v and c will be considered in the same community.

$$\Delta Q(\mathscr{C}, v, c) = \frac{1}{m}\left((\phi(v,c) - \lambda \frac{d_G(v)d_G(c)}{2m}\right) \tag{2}$$

Notice that both v and c might represent supernodes from previous edge contractions. For this reason, the vertex degrees and edge weights must be carefully updated after edge contractions in order to properly represent the supernodes in the graph.

In the first iteration of the Construction Phase, the vertices are singletons. Algorithm 2 shows a pseudocode of this phase.

Algorithm 2: Construction Phase

Data: $G = (V, E)$, initial partition $\mathscr{C}$, adjusted modularity value Q, α
Output: partition $\mathscr{C}_C$

```
Let M be the list of all possible edge contractions (v,c) sorted in decreasing order of ΔQ(C,v,c)
repeat
    Create RCL with the ⌊α|M|⌋ first elements of M.
    Pick at random an element of RCL, denoting such element as (v,c)
    Update C by assigning c to the same community as v
    d(v) = d(v) + d(c)
    Update (v,i) ∈ M, ∀i ∈ N(v).
    for k ∈ N(c) do
        if (v,k) ∉ E(G) then
            E(G) = E(G) + (v,k)
            φ(v,k) = φ(k,c)
            Insert (v,k) in M in the corresponding position according to ΔQ(C,v,k)
        end
        else
            φ(v,k) = φ(v,k) + φ(k,c)
            Update (v,k) ∈ M according to ΔQ(C,v,k)
        end
    end
    Remove (c,j) from M, ∀j ∈ N(c)
    V(G) = V(G) − c
    Q(C) = Q(C) + ΔQ(v,c)
until ΔQ(v,c) > 0 and |M| > 0;
```

The inputs of Algorithm 2 are the graph G under consideration, an initial partition $\mathscr{C}$, in this case, the singletons, its adjusted modularity value Q, and the $\alpha \in [0,1]$ value, which defines the degree of randomness on the movement choice. The closer to 1 the parameter α is, the more random the strategy. The parameter α was set as 0.5, as suggested in[4].

Initially, the algorithm constructs a list M of all possible edge contractions, sorted in decreasing order of adjusted modularity gain. The Restricted Candidate List (RCL) contains the best $\lfloor \alpha|M| \rfloor$ edge contractions from M. The stop criterion of the strategy is either there exists no element in M or the corresponding edge contractions of the list have negative adjusted modularity gain. For each edge contraction (v,c), the node c is moved to the community of v and node c is contracted in the supernode v. For each vertex k adjacent to c, $\phi(c,k)$ is added to $\phi(v,k)$, if (v,c) already exists. Otherwise, a new edge (v,k) is created with $\phi(v,k) = \phi(c,k)$. In both cases, the new modularity gain of

contracting (v,k) is calculated and the edge contraction is inserted or updated to keep M ordered. The edges incident to v are also updated in M, since the degree of v has changed. Finally, vertex c and all its edge contractions are effectively removed from the graph and the list M. The adjusted modularity value of the partition is incremented and the *RLC* is updated.

Máximo et al.[13] proposed an algorithm named *Intelligent-Guided Adaptive Search* to solve the maximal location covering problem. IGAS consists in a hybridization of the GNG network with the well-known metaheuristic *Greedy Randomized Search Procedure* (GRASP)[14]. GRASP is a two-phase iterative method. The first phase builds a solution from the scratch, using a semi-greedy-based strategy, the construction phase. The solution is then refined in the local search phase.

Máximo et al.[13] observed that the GRASP heuristic is iteration independent since it has not any memory mechanism to guide next iterations. Bearing this in mind that the authors embedded in the construction phase of GRASP a learning stage considering the GNG network. The best solutions obtained by GRASP are presented to the GNG network as an input signal.

In this paper, instead of guiding the choice by elements to insert in the partial solutions, the GNG network guides which pair of vertices should belong to the same cluster in every posterior iterations. Accordingly, the GNG network indicates the pair of nodes to be contracted in the graph under consideration. Moreover, as the local search in community detection problems has a very poor performance, being inefficient, this stage will not be considered in the proposed strategy.

3.2. Growing Neural Gas

The pseudocode in Algorithm 3 details the adaptation phase of the GNG network.

Algorithm 3: Adaptation Phase

```
Data: the GNG network H = (A, F) and structure nodes, γ
γ := inputSignal()
s1, s2 := the closest nodes to γ
error(s1) := error(s1) + (||w_s1 − γ||)^2
Δ(w_s1) := μ1(γ − w_s1)
Δ(w_v) := μ2(γ − w_v) , ∀v ∈ N(s1)
if z = (s1, s2) ∈ C(H) then
    age_z = 0
end
for c ∈ C(H) do
    age_c := age_c + 1
    if age_c > g_max then
        H := H − c
        if d_g(v) = 0, for some v, end of c then
            H := H − v
        end
    end
end
```

The proximity of a pair of nodes v and u is the variation of the average squared error between the matrices of weights of the nodes, w_v and w_u. The values of the errors of the nodes are used to identify the regions of the network with the largest errors. This study employs the mechanism proposed in[12] that, during the iterations the edge ages are incremented to identify the oldest ones. Accordingly, the algorithm has a mechanism to remove old (unvisited) edges of the network, i.e., those older than a threshold g_{max}. The parameter g_{max} was empirically set to 20. If the removal of an edge results in isolated vertices, those are removed from the GNG network.

After θ adaptations, the algorithm performs the growing neural phase, detailed in Algorithm 4.

Algorithm 4: Growing Neural Phase

Data: network $H = (A, C)$

$q :=$ node with the largest acummulated error.
$f := f \in \mathcal{N}(q)$ with the largest accumulated error.
$A(H) \cup r$, $w_r := 0.5(w_q + w_f)$.
$C(H) \cup \{(r,q),(r,f)\}$
$C(H) - (q,f)$
$error(q) := error(q)\chi$
$error(f) := error(f)\chi$
$error(r) := error(q)$
$\forall a \in A(H), error(a) := error(a)\beta$

Algorithm 5: GNG

Data: Network $H = (A,F)$, the set of weight matrices *nodes*, $iter_{GNG}$, partition $\mathscr{C}$

$\gamma =$ weight matrix of $\mathscr{C}$
$H =$ Adaptation Phase($H, nodes, \gamma$)
if $iter_{GNG}$ mod θ **then**
  $H =$ Growing Neural Phase($H, nodes$)
end
$iter_{GNG} := iter_{GNG} + 1$

Therefore, the GNG strategy is summarized in Algorithm 5.

As observed in Algorithm 5, $\mathscr{C}$ is used to obtain a matrix of weights by defining, for each $i, j \in V(G)$, $w_\gamma^{i,j} = 1$, if i and j are in the same community and 0, otherwise. Next, γ is used in the Adaptation Phase of GNG. The asymptotic complexity of adaptation phase is $O(|A||V|^2)$.

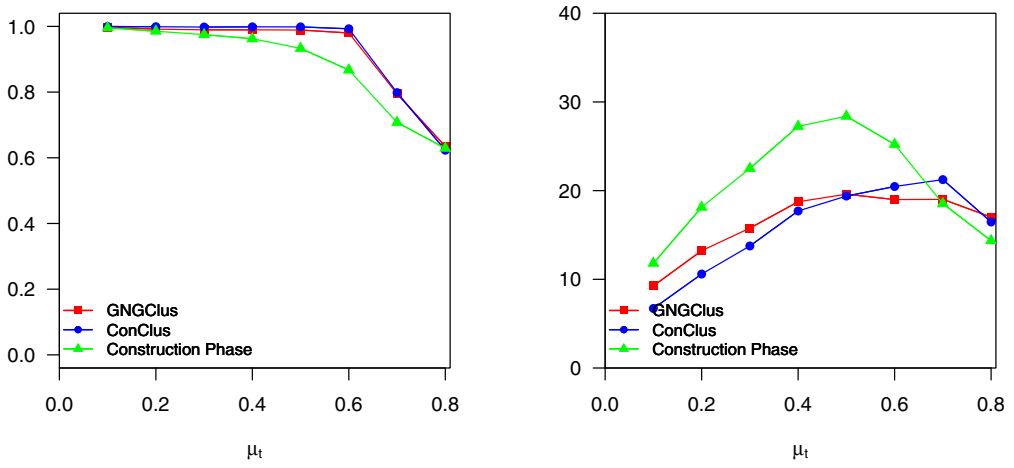
In this paper, θ was empirically defined as 5. The Growing Neural Phase asymptotic complexity is $O(|A||V|^2)$.

4. Computational Experiment

The proposed algorithm, named GNGClus, has a high resemblance with a strategy introduced in[4], called ConClus. For this reason, in the computational experiment, its results are compared with those achieved by ConClus. Additionally, this experiment presents the results achieved by the version of the construction phase without the contraction strategy and memory mechanism. The key difference between ConClus and GNGClus is in the choice of which pairs of vertices to contract. The former takes into account the pairs of vertices that are in the same community in 50% of an elite set of solutions. The latter considers the GNG network to perform these contractions, as presented in the previous section. The number of iterations of GNGClus was set as 30, the same number of iterations of ConClus and Construction Phase. This number was chosen after experiments performed in[4].

The experiment evaluates the performance of GNGClus, in a set of 80 artificial undirected networks, generated by the software introduced in[15]. In the literature, networks generated by this software are referred as LFR networks. The set of LFR networks has graphs with less fuzzy communities to more mixed communities. This feature is controlled by the mixture parameter, that the higher it is, the more overlapping communities are defined. This parameter, here referred as μ, ranges from 0.1 to 0.8 in the graphs considered in the experiments. All LFR networks in this experiment have 1000 vertices. The other parameters used to define these networks are the same as in[16]. For each mixture parameter, 10 networks are considered.

The strategy found in[8] , which identifies an interval for the adjustment parameter value, was used in the three algorithms of this experiment to provide the adjustment parameter of modularity. Alike the consensus strategy of[4], the algorithm under consideration is used to obtain a partition for *five* different values of the adjusted parameter



(a) Mean NMI values.

(b) Mean times for each mixture degree.

Fig. 1. Average results achieved by GNGClus, ConClus and Construction Phase.

interval. A final partition is constructed by assigning vertices that appear in the same cluster in more than 50% of the partitions obtained by the adjusted parameters.

As the LFR networks have the expected partitions, they are used as referential to assess the quality of the achieved results. For this, the results obtained by the algorithms are contrasted with the expected partitions using the well-known *Normalized Mutual Information* (NMI) metric [17]. The closer to 1 this value is, the more similar to the expected partitions the results are. Figure 1 displays the average results obtained by the three tested algorithms.

According to Figure1.a, GNGClus and ConClus had a very competitive behavior, significantly outperforming the Construction Phase in both time and NMI. The results achieved by GNGClus, according to these averages, apparently have a more competitive behavior with networks with mixture degree higher than 0.5. To present a more robust analysis of the results obtained by the three algorithms to investigate this hypothesis, we present the performance profiles of [5]. According to them, to assess how effective is an algorithm s that belongs to a set of algorithms S to solve a problem p from a set of problems P, the following ratio must be considered:

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,sof} : \forall sof \in S\}} \tag{3}$$

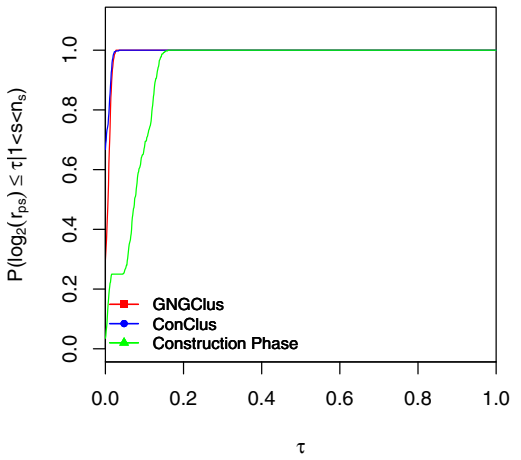
In this case, the metric to be analyzed must be minimized, as, for example, the time to solve a problem. The ratio is always greater than or equal to 1. To assess the effectiveness of an algorithm to solve the set P of problems, one must evaluate the percentage of problems that an algorithm solves within a factor τ, calculated as:

$$\rho(\tau) = \frac{1}{|P|}|\{p \in P : r_{p,s} \leq \tau\}|.$$

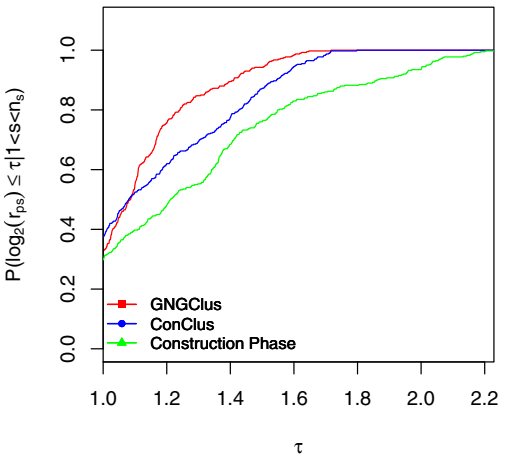
As in our experiment we analyze both time and NMI values, for the latter, we will have to adapt this ratio to take into account the goal of maximizing a metric. In this case, the ratio is defined to be the difference between the highest NMI value obtained to solve a problem p and the NMI value obtained by the algorithm s to solve p.

The performance profiles considering the networks with the parameter mixture degree higher than 0.5, 0.6, 0.7 and 0.8 are presented in Figure 3.

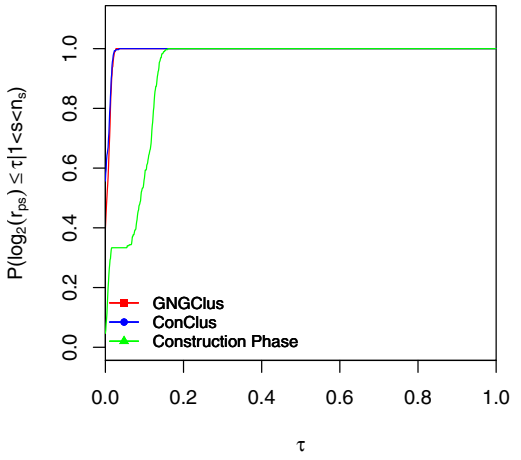
According to the performance profiles considering the NMI values, the major difference in the algorithms occurs for low values of τ. Additionally, the difference gets smaller as the mixture degree increases. Both GNGClus and ConClus clearly outperformed Construction Phase with respect to the NMI values. However, the Construction Phase



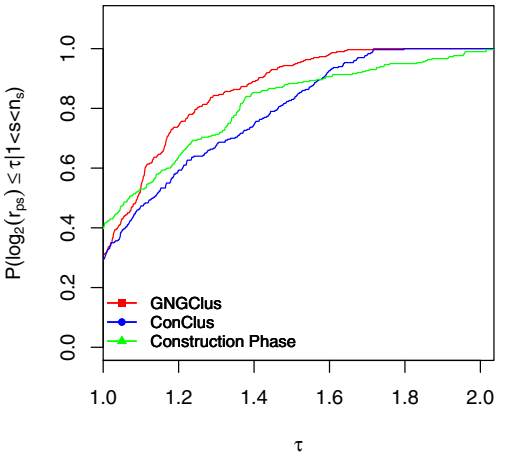
(a) Performance profiles considering the NMI values for networks with mixture degree higher than 0.4.



(b) Performance profiles considering the times of the algorithms to detect communities in networks with mixture degree higher than 0.5.



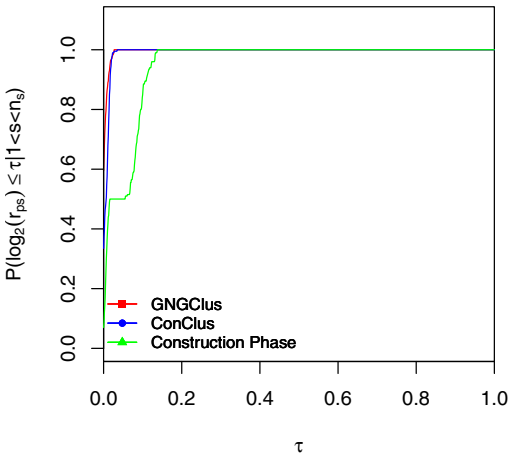
(c) Performance profiles considering the NMI values for networks with mixture degree higher than 0.6.



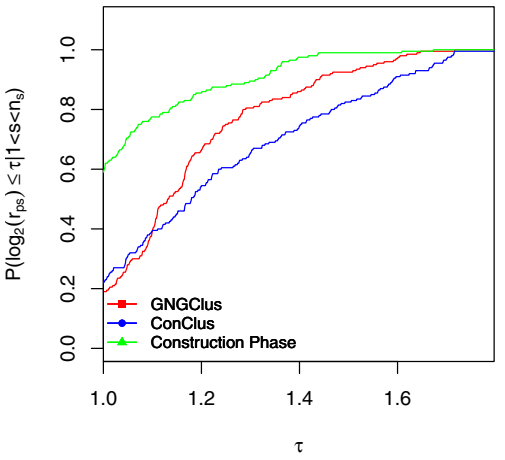
(d) Performance profiles considering the times of the algorithms to detect communities in networks with mixture degree higher than 0.6.

Fig. 2. Performance profiles of GNGClus, ConClus and Construction Phase considering the time to solve and the NMI valus, for networks with mixture degree parameter higher than 0.4.

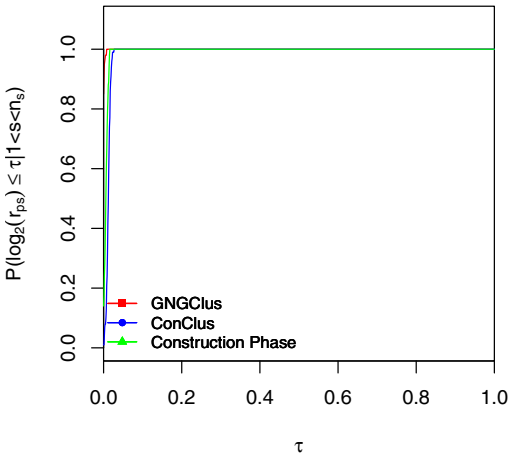
time performance was better than GNGClus and ConClus for networks with mixture degrees 0.7 and 0.8. In particular, Construction Phase were able to obtain an NMI performance very similar to GNGClus and ConClus for the mixture parameter 0.8 with a much better computational performance. The reason for this behavior might be explained by the difficult of GNGClus and ConClus in obtaining the slightly better NMI performances in networks with fuzzy communities.



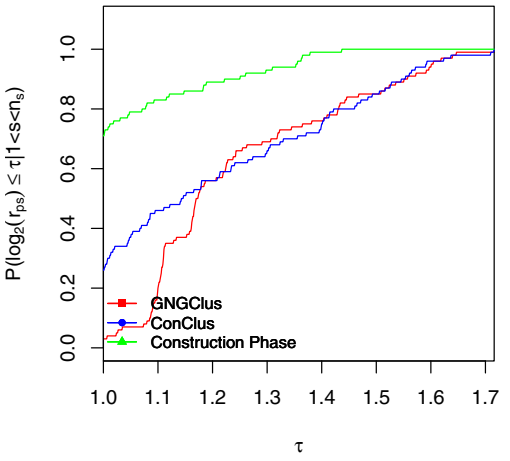
(a) Performance profiles considering the NMI values for networks with mixture degree higher than 0.7.



(b) Performance profiles considering the times of the algorithms to detect communities in networks with mixture degree higher than 0.7.



(c) Performance profiles considering the NMI values for networks with mixture degree higher than 0.8.



(d) Performance profiles considering the times of the algorithms to detect communities in networks with mixture degree higher than 0.8.

Fig. 3. Performance profiles of GNGClus, ConClus and Construction Phase considering the time to solve and the NMI valus, for networks with mixture degree parameter higher than 0.6.

Concerning GNGClus and ConClus, ConClus had a slightly better NMI performance considering small factors τ for the networks with mixture parameter higher than 0.5. ConClus solved the highest percentage of problems within the smallest values of τ and in the best times. However, for approximately $\tau > 1.1$, GNGClus time performance was significantly better. The NMI values of GNGClus and ConClus were equal in networks with mixture parameter higher than 0.6. In such cases, GNGClus presented a better performance on time.

GNGClus obtained the best NMI performance in the networks with mixture parameters higher than 0.7 and 0.8. For the mixture parameter 0.7, GNGClus achieved a time performance worse than ConClus within a factor of $\tau < 1.1$, approximately, and a better time performance for the remaining factors. In networks with the mixture parameter 0.8, the time performance of ConClus was considerably worse than of the other algorithms for $\tau < 1.2$ and comparable with the time performance of GNGClus for the remaining factors. These results attest the better performance of the proposed GNGClus with respect to ConClus in networks with high values of mixture degree.

5. Final Remarks

This paper presented a comparative analysis of three forms to construct a solution using a semi-greedy process. They are: using no memory mechanism; using memory through the storage of the best (elite) solutions; and employing a Self Organizing Map (SOM) neural network as memory mechanism to guide future decisions based on former solutions. The latter approach is introduced in this paper, whilst the former two are strategies found in the literature. In a computational experiment with 80 networks, it was clear that the two memory-based strategies significantly outperformed that without such mechanism. However, even though the two memory-based strategies were very competitive, the strategy based on elite solutions was slightly better than the introduced strategy, GNGClus. Nevertheless, when observing the performance profiles of every solution, it was possible to observe that the improvement was very small and by inspecting the individual solutions, we attested that there were cases in which the introduced strategy outperformed ConClus. As future work, we intend to hybridize the two strategies, as an attempt of significantly outperforming the two strategies. It is worth mentioning that, ConClus presented better performance than a number of strategies found in the literature, as Infomap.

Acknowledgements

The authors are grateful to *Fundação de Amparo à Pesquisa do Estado de São Paulo* (FAPESP Proc.: 2015/21660-4) and *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq id: 448614/2014-6) for the financial support.

References

1. Girvan, M., Newman, M.E.J.. Community structure in social and biological networks. *Proc Natl Acad Sci USA* 2002;**99**:7821–7826.
2. Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., et al. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering* 2008;**20**:172–188.
3. Fortunato, S., Barthelemy, M.. Resolution limit in community detection. *Proceedings of the National Academy of Sciences* 2007;**104**(1):36–41.
4. Santos, C., Carvalho, D.M., Nascimento, M.C.V.. A consensus graph clustering algorithm for directed networks. *Expert Systems with Applications* 2016;**54**:121–135.
5. Dolan, E.D., Moré, J.J.. Benchmarking optimization software with performance profiles. *Math Program, Ser A* 2002;**91**:201–213.
6. Rosvall, M., Bergstrom, C.T.. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences* 2008;**105**(4):1118–1123.
7. Reichardt, J., Bornholdt, S.. Statistical mechanics of community detection. *Physical Review E* 2006;**74**:016110.
8. Carvalho, D.M., Resende, H., Nascimento, M.C.V.. Modularity maximization adjusted by neural networks. *21st International Conference, ICONIP 2014* 2014;:287–294.
9. Fritzke, B.. A Growing Neural Gas network learns topologies. In: *Advances in Neural Information Processing Systems (NIPS) 7*. MIT Press; 1995, p. 625–632.
10. Martinetz, T., Schulten, K., et al. *A" neural-gas" network learns topologies*. University of Illinois at Urbana-Champaign; 1991.
11. Martinetz, T.. Competitive hebbian learning rule forms perfectly topology preserving maps. In: *ICANN'93*. Springer; 1993, p. 427–434.
12. Fišer, D., Faigl, J., Kulich, M.. Growing neural gas efficiently. *Neurocomputing* 2013;**104**:72–82.
13. Máximo, V.R., Nascimento, M.C.V., Carvalho, A.C.P.L.. Intelligent Guided Adaptive Search for the maximum covering location problem. *Submitted to Computers & Operations Research* 2016;.
14. Feo, T., Resende, M.. Greedy randomized adaptive search procedures. *Journal of Global Optimization* 1995;**6**:109–133.
15. Lancichinetti, A., Fortunato, S., Radicchi, F.. Benchmark graphs for testing community detection algorithms. *Physical Review E* 2008; **78**:046110–[5 pages].
16. Nascimento, M.C.V., Pitsoulis, L.. Community detection by modularity maximization using grasp with path relinking. *Computers & Operations Research* 2013;**40**(12):3121–3131.
17. Danon, L., Daz-Guilera, A., Duch, J., Arenas, A.. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment* 2005;**2005**(09):P09008–[11 pages].