

Improving the Performance of an Integer Linear Programming Community Detection Algorithm Through Clique Filtering

Luiz Henrique Nogueira Lorena[1], Marcos Gonçalves Quiles[1(✉)], and Luiz Antonio Nogueira Lorena[2]

[1] Institute of Science and Technology, Federal University of São Paulo (UNIFESP), São José dos Campos, SP, Brazil
luiz.lorena@hotmail.com, quiles@unifesp.br

[2] National Institute for Space Research (INPE), São José dos Campos, SP, Brazil
lorena@lac.inpe.br

Abstract. Different fields of science use network representation as a framework to model their systems. The analysis of network structure can give us essential information about the system. However, the size of such a network can limit the applicability of some fundamental techniques like mathematical programming. Thus, here we propose a novel network size reduction technique based on a clique filtering approach. Our goal is twofold: (1) reduce the network size and speed up the community detection process, and (2) preserve the modularity of the original partition in the context of the exact model. Conducted experiments show the feasibility and correctness of the proposed technique.

Keywords: Community detection · Integer linear programming · Preprocessing technique

1 Introduction

Network representation is being used to model systems in different fields of science, like computer science, physics, chemistry, biology, and sociology [2,13,18]. The goal is to uncover meaningful information about the system by analyzing the structure and dynamic of the network.

One particular network structure, relevant in different contexts of science, is the community or modular structure. A community can be defined as a group of nodes densely connected. Finding the communities of a network is an actively studied problem, and several techniques have been proposed in the literature, from heuristic to exact methods [5,6].

A great number of techniques are based on the optimization of a quality function. Newman's modularity [12] is the most popular quality function used

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11619, pp. 757–769, 2019.
https://doi.org/10.1007/978-3-030-24289-3_56

to estimate the goodness of a partition obtained by a community detection technique. High values of modularity indicate good partitions. Hence, modularity-based techniques try to obtain partitions that maximize such quality function.

This work focus on solving the community detection problem via the modularity-based Linear Programming (ILP) formulation proposed by Argawal et al. [1]. This approach, however, may create models that are large and time-consuming to solve. The size of the models may restrict the use of such technique. Hence, here we are interested in strategies that can reduce such computational burden.

Recent research aims to reduce the number of edges and/or nodes of a network while maintaining some of its structural properties. Such properties may include shortest paths and connectivity, cuts, source-to-sink flow, spectral properties and modularity [3,4,7,8,10,17,20]. The work of Arenas et al. [3] is particularly interesting in the context of modularity maximization. It shows that some peripheral cliques can be filtered from the network, creating a smaller network representation. By using this reduced network, they achieved a speedup in their community detection heuristic while preserving the quality of the obtained solutions.

Arenas et al. [3] technique, however, is limited to cliques composed by two or three nodes. Hence, here, we waive this restriction by proposing a technique capable of filtering larger cliques. Our goal is to speed up the community detection process and to preserve the quality of the obtained solutions, in the context of the exact model.

The computational experiments shows that, by reducing the size of the network, we are able to obtain ILP models that have a reduced number of variables and constraints. Such models require less memory and computation time to solve.

This work is organized as follows. First, community detection via ILP is presented in Sect. 2, highlighting its properties and limitations. In Sect. 3 the clique filtering approach proposed by Arenas et al. [3] is presented, and a relevant property discovered in their work is discussed. Section 4 introduces the new clique filtering algorithm. The computational experiments are reported in Sect. 5. Finally, conclusions are discussed in Sect. 6, along with future work directions.

2 Modularity Maximization via ILP

The following notation will be used throughout this paper. A network is represented by an undirected weighted graph $G = (V, E)$, consisting of $n = |V|$ nodes and $m = |E|$ edges. The edge weights are stored in an adjacency matrix A composed by $|V| \times |V|$ elements.

Newman [14] proposed modularity as a measure to evaluate the quality of a partition obtained by a community detection method. It takes values in the interval $[-\frac{1}{2}, 1]$. High values of modularity indicate good partitions. To calculate modularity, we assume $\mathcal{P} = \{C_1, C_2, \ldots, C_p\}$ as the partition obtained by a community detection method. $\mathcal{P}$ represents a collection of p disjoint subsets of nodes C (communities). In the context of G it is defined as follows

$$Q(\mathcal{P}) = \frac{1}{2s} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(A_{ij} - \frac{s_i s_j}{2s} \right) \delta_{ij} \tag{1}$$

where

$$x_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are in the same community} \\ 0 & \text{otherwise} \end{cases}$$

and A_{ij} is the weight of the edge $\{i, j\}$. Expressions s_i and s_j represents the strength of node i and j, respectively. The strength of a node is the sum of the edge weights incident to it, and it is calculated as $s_i = \sum_{j=1}^{n} A_{ij}$. Finally, s represents the sum of all edge weights in A, and it is calculated as $s = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij}$.

Modularity maximization can be formulated as an ILP by following the steps provided by Agarwal et al. [1]. The objective function is derived from modularity formula presented in Eq. 1. Binary variables x_{ij} are introduced, assuming a value equal to 1 if i and j belong to the same community and 0 otherwise.

$$\frac{1}{2s} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(A_{ij} - \frac{s_i s_j}{2s} \right) x_{ij}$$

The expression inside the parentheses is the modularity matrix M_{ij} defined by Newman et al. [15]. Consequently, the objective function can be simplified to

$$\frac{1}{2s} \sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij} x_{ij}$$

Variables x_{ii} can be replaced by a constant in the objective function, since $x_{ii} = 1$ for all nodes i:

$$C = \frac{1}{2s} \sum_{i=1}^{n} M_{ii}$$

Since $x_{ij} = x_{ji}$ for all nodes i and j, variables x_{ij} where $i > j$ can be removed from the model by doubling the objective function. Hence, the following ILP model is obtained:

$$Maximize \ \frac{1}{s} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} M_{ij} x_{ij} + C$$

$$subject \ to$$

$$x_{ij} + x_{jk} - x_{ik} \geq 0, \quad M_{ij} \geq 0 \vee M_{jk} \geq 0 \tag{2}$$

$$x_{ij} - x_{jk} + x_{ik} \geq 0, \quad M_{ij} \geq 0 \vee M_{ik} \geq 0 \tag{3}$$

$$-x_{ij} + x_{jk} + x_{ik} \geq 0, \quad M_{jk} \geq 0 \vee M_{ik} \geq 0 \tag{4}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j \in [1..n]$$

where

$$x_{ij} = \begin{cases} 0 & \text{if } i \text{ and } j \text{ are in the same community} \\ 1 & \text{otherwise.} \end{cases}$$

The transitivity constraints (2–4) assures that: if node i is in the same community as j, and j and k are in the same community, then i must be in the same community as k.

The original model by Agarwal et al. [1] has $O(n^2)$ variables and $O(n^3)$ transitivity constraints, and tends to grow fast as the size of the network increases. A recent work by Miyauchi et al. [11] showed that some of the transitivity constraints are redundant and can be eliminated during model construction. They provided conditional clauses that check for redundant constraints and prevent its creation. We use such strategy here to extend original ILP model. This explains the conditional $M_{ij} \geq 0 \vee M_{jk} \geq 0$ introduced in constraint (2). If the conditional is not respected the constraint is not inserted in the model.

The major drawback of the ILP approach is the size of its model. It is heavily affected by the number of nodes of the network being modeled. Consequently, any technique that could reduce the number of nodes in the network while maintaining some of its structural properties would be of great value. In the next section, a technique with such capabilities is discussed. This technique was used as the base to create the new filtering algorithm proposed in this work.

3 Clique Filtering

Some basic concepts are important in the context of the topic discussed in this section. A clique is a subset of V, such that every two distinct nodes are adjacent. A node whose removal disconnects a connected graph is called an articulation point.

In this work we focus on a particular type of clique, denoted here as K_s, where $s = |V_s|$ represents the number of elements in the clique. This clique is peripheral and have one specific node, denoted here as v^*, that is an articulation point. Figure 1 show examples of cliques K_2 and K_3.



(a) Clique K_2. (b) Clique K_3.

Fig. 1. Type of cliques filtered by Arenas et al. [3].

Arenas et al. [3] proposed a filtering technique for reducing the size of a network while preserving modularity. It is able to filter cliques K_2 and K_3 (Fig. 1) by collapsing such structures into a smaller network representation.

Their experimental results shows that their technique can reduce the size of networks, speed up the community detection heuristic and preserve the quality of the obtained solution.

In [3] a relevant property of modularity is used to reduce the size of the network: nodes forming a community in the optimal partition can be represented by a unique node in the reduced network. The basic idea is that each node and edge removed during the filtering process contributed to the original graph modularity value. Hence, to preserve this value, such contribution needs to be taken into account while calculating the reduced graph modularity. Figure 2 shows how this property can be used to create a smaller network representation.

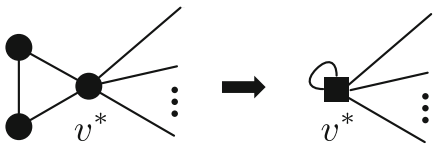


Fig. 2. Reducing a network size by collapsing nodes and edges.

In Fig. 2, node v^* will store in the weight of its self-loop the info about the filtering process that will be relevant for the modularity computation. The self-loop weight will accumulate the weight of the collapsed edges that were incident to v^*. The remaining $s-1$ clique nodes and respective edges are collapsed into a smaller network representation.

The following section presents the algorithm to identify and filter larger cliques K_s. The strategy to reduce the size of the network presented in this section will be employed to create a smaller network representation.

4 Clique Filter (CF) Algorithm

The proposed filter is based on the identification of cliques K_s (Fig. 3). Cliques K_2 are easily identified by the degree of the node incident to the articulation point v^*. However, to identify cliques with $s \geq 3$ we follow a different approach based on the local clustering coefficient of a node.

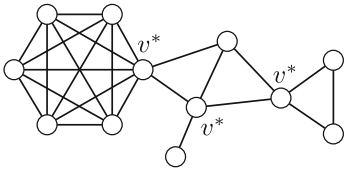


Fig. 3. Example with three cliques K_s.

The local clustering coefficient (LCC) [19] of a node quantifies how close its neighbors are to being a complete graph (clique). This measure assumes its maximum value (1.0) when every neighbor of i is connected (clique), and its minimum (0.0) when there are no connections between the neighbors of i.

By using the LCC, we can observe in Fig. 4 that cliques with $s \geq 3$ are composed by $s-1$ nodes with LCC equal to 1.0. The only node with a different LCC is the articulation point v^*. This property was used as a starting point to create the new clique filtering algorithm.

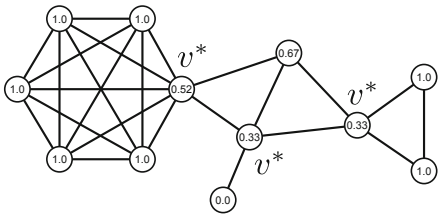


Fig. 4. Identifying larger cliques K_s.

Algorithm 1 presents the proposed clique filtering algorithm (CF). The objective is to receive a network as input G and output a reduced network representation G'. The algorithm is composed of four steps. Figure 5 depicts the behavior of each step of Algorithm 1 on a sample network. In the first step of the algorithm all nodes of G are assigned to its own group (Fig. 5a). The group ownership will guide the construction of the reduced network in the final step of the algorithm.

Algorithm 1. Clique Filter (CF).

input : A weighted graph $G = (V, E)$
output: A weighted graph $G' = (V', E')$

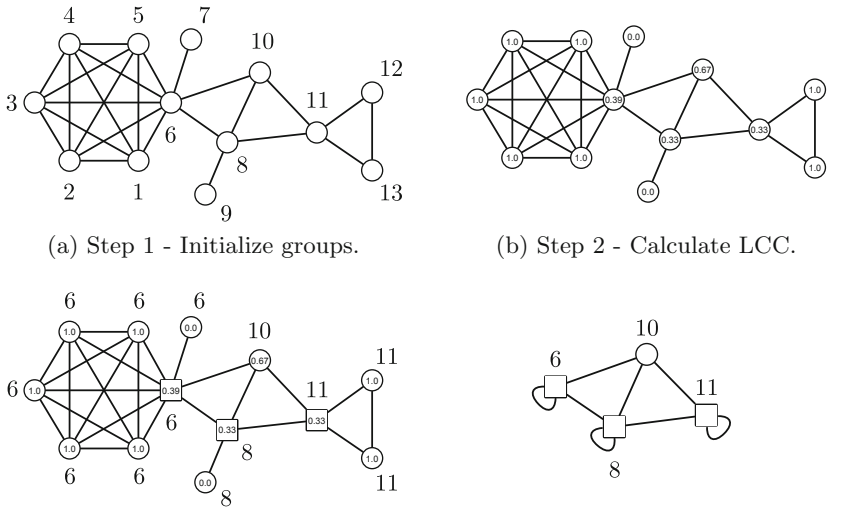
1 Assign each node of G to its own group
2 Calculate the LCC of each node in G
3 For all nodes in G, check if it has degree 1 or all its neighbors except one has LCC equal to 1.0. If so, assign the node and its neighbors to the same group of the articulation point v^*
4 Create a smaller network representation G' based on the group ownership of G

In the second step, the LCC of the nodes in G is calculated (Fig. 5b). Step three test if each node in G belongs to a clique K_s (Fig. 5c). Nodes with degree one are participants of cliques K_2 and nodes where all its neighbors except one has LCC equal to 1.0 are participant of a larger clique K_s. Such nodes and all its neighbors are assigned to the group of the articulation point (v^*) of K_s.

Finally, at step 4, the strategy discussed in Sect. 3 is used to create a reduced network representation G' (Fig. 5d). Nodes that belongs to the same group in G are collapsed into a single node in G', the edges connecting nodes in the same group are collapsed into a self-loop whose weight will be the sum of the weight of the collapsed edges. Groups composed by individual nodes in G remains unchanged, preserving their structural properties in the reduced network G'.

The worst-case time complexity for Algorithm 1 is $O(m^{\frac{3}{2}})$. The local clustering coefficient algorithm used on step 2 is the dominant factor of this time complexity. The remaining steps have linear time complexity.

The reduced network representation obtained by CF algorithm can be used directly in the ILP formulation described in Sect. 2. In the following section, the experimental results show how the reduced network can benefit the exact model.



(a) Step 1 - Initialize groups.

(b) Step 2 - Calculate LCC.

(c) Step 3 - Identify cliques K_s.

(d) Step 4 - Create a reduced network.

Fig. 5. Example illustrating each step of Algorithm 1.

5 Experimental Results

In this section, experiments were conducted in order to verify the validity of the CF algorithm. The experiments and algorithms were coded in C++14 and executed on a computer with the following configuration: Intel Core i7-6770HQ (3,5 GHz) with 32 GB RAM running Windows 10 64-Bit. The commercial solver IBM ILOG CPLEX [9] 12.7.1 was used to solve the ILP models.

Table 1 presents the datasets used in this work. Only the giant components of the networks were used in the experiments. All datasets were downloaded from the Network Repository website [16]. Each dataset received a unique id (first column). The second column presents the dataset name in the Network

Table 1. Benchmark datasets used in the experiments.

ID	Dataset	n	m
1	lemis	77	254
2	GD00-A	83	125
3	ca-sandi-auths	86	124
4	rt-retweet	96	177
5	netscience	379	914
6	bio-DM-LC	483	997
7	power-494-bus	494	586
8	bio-diseasome	516	1188
9	bio-grid-mouse	791	1098
10	ca-CSphd	1025	1043

Repository website, the remaining columns in this table represents the number of nodes (n) and the number of edges (m) of the network.

The first experiment evaluates the performance of the CF algorithm against the benchmark network datasets in terms of computational time and network size reduction. The results of this experiment are presented in Tables 2 and 3.

Table 2 reports the performance of the CF algorithm in ten independent runs and the total cliques filtered. It can be observed, from Table 2, that the computational time of CF algorithm is low, specially when compared to the computational time we expect to achieve by solving the ILP model. The algorithm is able to filter cliques with more than 3 elements in the datasets with ID 3, 5 and 8.

Table 2. Computational time and cliques filtered by CF algorithm.

ID	Time	Cliques
1	4.8×10^{-5} (1.3×10^{-6})	$K_2(17)$ $K_3(1)$
2	3.9×10^{-5} (4.7×10^{-7})	$K_2(19)$
3	4.2×10^{-5} (1.7×10^{-6})	$K_2(24)$ $K_3(5)$ $K_4(1)$
4	4.1×10^{-5} (1.8×10^{-6})	$K_2(53)$
5	6.5×10^{-5} (9.9×10^{-7})	$K_2(27)$ $K_3(16)$ $K_4(6)$ $K_5(4)$
6	5.8×10^{-5} (2.5×10^{-6})	$K_2(143)$
7	5.3×10^{-5} (1.6×10^{-6})	$K_2(146)$ $K_3(2)$
8	8.2×10^{-5} (1.4×10^{-6})	$K_2(90)$ $K_3(25)$ $K_4(8)$ $K_5(4)$ $K_6(2)$
9	6.8×10^{-5} (1.4×10^{-6})	$K_2(409)$ $K_3(1)$
10	1.2×10^{-4} (4.1×10^{-6})	$K_2(693)$

Table 3. Size of the original and reduced network.

ID	G		G'		%Reduction	
	n	m	n'	m'	$\%n$	$\%m$
1	77	254	58	242	24.68	4.72
2	83	125	64	118	22.89	5.60
3	86	124	57	98	43.02	20.97
4	96	177	43	95	55.21	46.33
5	379	914	286	803	24.54	12.14
6	483	997	340	935	29.61	6.22
7	494	586	344	544	30.36	7.17
8	516	1188	326	992	36.82	16.50
9	791	1098	380	843	51.96	23.22
10	1025	1043	332	588	67.61	43.62

Table 3 presents a comparison between the original (G) and filtered network (G') regarding its size. The percentage of network size reduction is presented in column *%Reduction*. It can be observed, from Table 3, that there is a reduction in the number of nodes in G' that goes from 22.89% up to 67.61%. The reduction in the number of edges goes from 4.72% up to 46.33%.

The next experiment was conducted to test the main hypothesis of this work, which states that the ILP technique can obtain a performance improvement and retain its quality by using the smaller network obtained by the CF algorithm. The results of this experiment are presented in Tables 4 and 5.

Table 4. Optimal solution and size of the original and filtered network ILP models.

ID	Q	#Variables		#Constraints	
		G	G′	G	G′
1	0.56000	2926	**1653**	34685	**23110**
2	0.53091	3403	**2016**	18434	**11581**
3	0.73683	3655	**1176**	20369	**7085**
4	0.67966	4560	**903**	21511	**4954**
5	0.84858	71631	**40755**	682739	**428071**
6	0.77869	116403	**57630**	946606	**566353**
7	0.85999	121771	**58996**	575323	**295959**
8	0.83199	132870	**52975**	1211786	**578838**
9	0.80101	312445	**72010**	1716725	**510436**
10	0.92558	524800	**54946**	2128786	**230136**

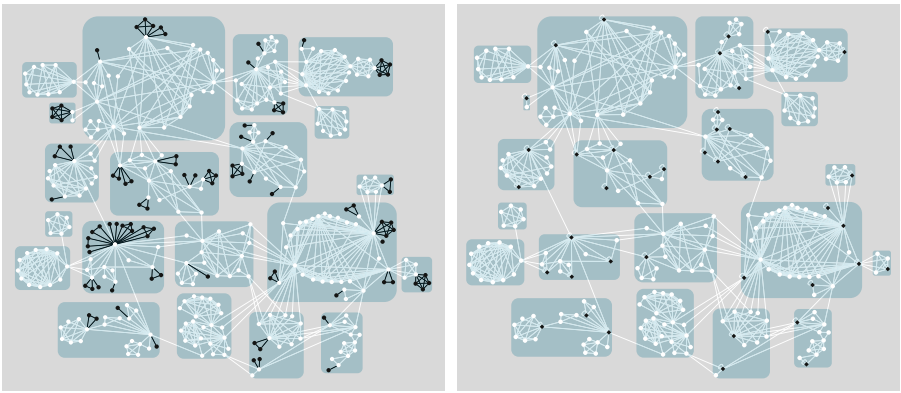
Table 5. Computational time and memory consumption of the original and filtered network ILP models.

ID	Time		Memory		%Reduction	
	G	G'	G	G'	%Time	%Memory
1	0.61	**0.42**	23.44	**15.88**	31.15	32.24
2	2.23	**1.46**	13.72	**8.90**	34.53	35.11
3	0.41	**0.15**	14.59	**6.18**	63.41	57.68
4	0.64	**0.18**	16.54	**4.42**	71.88	73.28
5	43.40	**20.67**	415.71	**269.53**	52.37	35.16
6	5315.57	**1692.86**	636.14	**361.01**	68.15	43.25
7	90419.51	**17586.84**	392.28	**199.97**	80.55	49.02
8	22966.99	**3046.39**	761.88	**364.17**	86.74	52.20
9	26551.02	**390.40**	1139.08	**341.81**	98.53	69.99
10	2512.05	**86.39**	1522.37	**169.82**	96.56	88.85

Table 4 presents the modularity (Q) and the size of the ILP models of the original (G) and reduced network (G'). It can be observed, from Table 4, that the modularity of the obtained solutions is preserved for both ILP models. However, the ILP models created for the reduced networks (G') are smaller in terms of the number of variables and constraints.

Table 5 reports the computational time and the memory consumption of the ILP models of the original (G) and reduced network (G'). It can be observed that the size reduction granted an improvement in performance of the ILP technique. There is a considerable percentage of reduction in terms of computational time and memory consumption. The percentage of reduction in time ranged from 31.15% up to 98.53%. For the datasets 6, 7, 8 and 9 it was possible to reduce the computational time from hours to minutes. For dataset 7, it was required one day to solve the ILP model for G, whereas the G' ILP model was solved in approximately five hours. The percentage of reduction in memory consumption was above 32.24% for all datasets. It achieved its highest value of 88.85% for the dataset 10, reducing the memory consumption from 1.5 GB to 169 MB.

Finally, Fig. 6 presents a comparison between the partitions obtained for the dataset with ID 5 (netscience). Figure 6a shows the partition obtained by solving the ILP model based on the original network (G), cliques filtered by the CF algorithm are highlighted in black. Figure 6b presents the partition obtained in the context of the reduced network (G'), the black color highlight nodes that received the cliques collapsed by the filtering algorithm. It can be observed that the groups obtained for the original network are preserved.



(a) Original network optimal solution. (b) Filtered network optimal solution.

Fig. 6. Comparing the optimal solution obtained for the original and filtered network of the dataset with ID 5.

6 Conclusions

Finding communities in networks is an actively studied topic, and several techniques were proposed in the literature. Modularity maximization via ILP is an interesting approach to obtain exact solutions for this problem. However, this technique is heavily affected by the size of the network because the resulting ILP models tends to grow fast as the number of nodes in the network increases.

Hence, this work proposed an algorithm, denoted here as CF, capable of reducing the size of the network by filtering peripheral cliques. The aim is to reduce the size of the network so it increases the performance of the ILP technique and the quality of its solution. The algorithm is based on the work by Arenas et al. [3], which showed that some peripheral cliques can be filtered from the graph, creating a smaller network representation without losing information relevant in the context of community detection. Their technique, however, is limited to cliques composed of two or three nodes.

The CF algorithm extends [3] work, by providing an approach to identify larger peripheral cliques via the local clustering coefficient of their members. The worst-case time complexity for CF is dominated by the local clustering coefficient calculus, however, the computational effort is relatively small when compared with computational time required to solve the ILP model based on the reduced network obtained as an output from the CF algorithm.

The proposed algorithm was validated in benchmark network datasets used in scientific literature. The experimental results shows that CF was capable of reducing the size of all the networks in a small computational time. The reduction in network size granted a considerable improvement in the performance of the ILP technique in terms of computational time and memory consumption. The quality of the solution obtained, measured via modularity, was preserved as well.

In this work we focused on solving community detection via an exact method, however, the gain in reduction provided by the CF algorithm can be useful to accelerate community detection heuristics. Therefore, as future work, we plan to adapt some community detection heuristics to work with the reductions introduced in this paper. The CF algorithm has its worst-case time complexity limited by the clustering coefficient algorithm. So, we plan to improve the algorithm used to identify the peripheral cliques.

Acknowledgment. M.G.Q. acknowledges the support by São Paulo Research Foundation (FAPESP, Proc. 2011/18496-7 & 2015/50122-0) and by the the Brazilian National Research Council (CNPq Proc. 310908/2015-9 & 434886/2018-1). L.A.N.L. acknowledges the support by (CNPq Proc. 301836/2014-0) and L.H.N.L. acknowledges the support by Coordination of Superior Level Staff Improvement (CAPES).

References

1. Agarwal, G., Kempe, D.: Modularity-maximizing graph communities via mathematical programming. Eur. Phys. J. B **66**(3), 409–418 (2008)
2. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Rev. Modern Phys. **74**(1), 47 (2002). https://doi.org/10.1103/RevModPhys.74.47
3. Arenas, A., Duch, J., Fernández, A., Gómez, S.: Size reduction of complex networks preserving modularity. New J. Phys. **9**(6), 176 (2007). https://doi.org/10.1088/1367-2630/9/6/176
4. Bonchi, F., Morales, G.D.F., Gionis, A., Ukkonen, A.: Activity preserving graph simplification. Data Min. Knowl. Discov. **27**(3), 321–343 (2013)
5. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010)
6. Fortunato, S., Hric, D.: Community detection in networks: a user guide. Phys. Rep. **659**, 1–44 (2016)
7. Gemmetto, V., Cardillo, A., Garlaschelli, D.: Irreducible network backbones: unbiased graph filtering via maximum entropy. arXiv preprint arXiv:1706.00230 (2017)
8. Gionis, A., Rozenshtein, P., Tatti, N., Terzi, E.: Community-aware network sparsification. In: Proceedings of the 2017 SIAM International Conference on Data Mining, pp. 426–434. SIAM (2017)
9. IBM: IBM ILOG CPLEX 12.7.1 (1987–2017)
10. Kim, J.R., Kim, J., Kwon, Y.K., Lee, H.Y., Heslop-Harrison, P., Cho, K.H.: Reduction of complex signaling networks to a representative kernel. Sci. Signal. **4**(175), ra35–ra35 (2011)
11. Miyauchi, A., Sukegawa, N.: Redundant constraints in the standard formulation for the clique partitioning problem. Optim. Lett. **9**(1), 199–207 (2015)
12. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**, 026113 (2004). https://doi.org/10.1103/PhysRevE.69.026113
13. Newman, M.E.: The structure and function of complex networks. SIAM Rev. **45**(2), 167–256 (2003). https://doi.org/10.1137/S003614450342480
14. Newman, M.E.: Analysis of weighted networks. Phys. Rev. E **70**(5), 056131 (2004)
15. Newman, M.E.: Modularity and community structure in networks. Proc. Natl. Acad. Sci. **103**(23), 8577–8582 (2006). https://doi.org/10.1073/pnas.0601602103

16. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (2015). http://networkrepository.com
17. Stanley, N., Kwitt, R., Niethammer, M., Mucha, P.J.: Compressing networks with super nodes. CoRR abs/1706.04110 (2017). http://arxiv.org/abs/1706.04110
18. Strogatz, S.H.: Exploring complex networks. Nature **410**(6825), 268–276 (2001). https://doi.org/10.1038/35065725
19. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature **393**(6684), 440–442 (1998). https://doi.org/10.1038/30918
20. Xiao, Y., MacArthur, B.D., Wang, H., Xiong, M., Wang, W.: Network quotients: structural skeletons of complex systems. Phys. Rev. E **78**(4), 046102 (2008)