Using mathematical programming to refine heuristic solutions for network clustering

Sonia Cafieri, Pierre Hansen

Using mathematical programming to refine heuristic solutions for network clustering

Sonia Cafieri[1] and Pierre Hansen[2]

[1]ENAC, MAIAA, F-31055 Toulouse, France and
University of Toulouse, IMT, F-31400 Toulouse, France
[2]GERAD, HEC Montréal, Canada

Abstract We propose mathematical programming based aproaches to refine graph clustering solutions computed by heuristics. Clustering partitions are refined by applying cluster splitting and a combination of merging and splitting actions. A refinement scheme based on iteratively fixing and releasing integer variables of a mixed-integer quadratic optimization formulation appears to be particularly efficient. Computational experiments show the effectiveness and efficiency of the proposed approaches.

1 Introduction

Networks, or graphs, provide very useful tools for modelling complex systems [33]. They consist of a set V of vertices associated to the entities under study and a set E of edges each of which joins two vertices and corresponds to relationships among the entities. For instance, in sociology vertices are associated with people and edges with relationships like friendship, communication or collaboration between them. In biology, vertices are associated for instance to proteins and the edges to their interactions. Some topological features of networks are studied to better understand the underlying complex systems, as they may reveal the organizational principles of the system components. The structure of complex systems can in fact be understood by identifying the way the nodes of the corresponding networks are connected to each other. A modular structure characterizes many complex systems, meaning that they contain subgroups of entities sharing some common properties. A topic of particular interest in the study of complex networks is therefore the identification of modules, also called *clusters* or *communities*. Given a graph $G = (V,E)$, roughly speaking one seeks subgraphs induced by sets of vertices $S_i \subseteq V$ which contain more inner edges (with both vertices in the same subset) than cut edges (with vertices in different subsets). In the last decade the problem of finding clusters in complex networks has been very extensively studied, see Fortunato [15] for an in-deep survey.

Many definitions of network modules have been proposed as well as criteria to evaluate partitions of vertices in modules. Maximizing any such criterion over the set of all partitions is a combinatorial optimization problem. The most popular criterion, despite some recent criticism [16, 8], is the *modularity* of a subnetwork [32]. The modularity of a module is defined as the difference of the fraction of the edges that it contains and the expected number of such edges in a network where edges are distributed at random while keeping the degree distribution of vertices constant, according to the so-called configuration model. Modularity of a partition is the sum of modularities of its clusters. So modularity of a network is a criterion whose maximization provides both the optimal number of clusters and an estimate of the amount of modularity of the network. Numerous heuristics have been proposed for maximizing modularity of a network. They include applications of simulated annealing [20, 28, 29], mean field annealing [26], genetic search [36], extremal optimization [14], variable neighborood search [3], spectral clustering [31], linear programming followed by randomized rounding [1], dynamical clustering [5], multilevel partitioning [13], contraction-dilation [30], divisive [31, 9] or agglomerative [11, 4] hierarchical clustering, and several other approaches.
Mathematical programming allows us rigorous formulations and solutions for the maximizing modularity optimization problem. Nevertheless, it is rarely used. There are two approaches to use mathematical programming formulations which can be solved to global optimality. Grötschel and Wakabayashi's [18, 19] model for clique partitioning can be immediately applied, replacing the original graph by a complete weighted graph. A closed model is used by Brandes et al. [6]. The second approach was proposed by Xu et al. [39], who express modularity maximization as a mixed-integer quadratic programming problem with a continuous convex relaxation. Column generation can be applied to solve both models [2]. In these models, modularity is the objective function to be maximized and constraints are used to impose conditions defining a partition of the vertex set.
The obtained optimization problems are generally difficult to solve and only small or medium-scale problems can be easily treated . The situation is more favorable when subgraphs of an original graph are handled, as they are more likely to have smaller size (possibly, medium-scale) than the original graph. Given a partition found by a heuristic, one can attempt to refine the result to obtain a new better partition, acting on subnetworks induced by the clusters of the original partition. The purpose of the present paper is to discuss and advance the use of mathematical programming to refine heuristic solutions for network clustering. Two approaches are discussed and compared, one of which is new. The first one was proposed in [10] and is based on splitting clusters using an exact algorithm for bipartitioning and merging pairs of clusters. The new one is inspired by the approach in [38] and is based on iteratively fixing integer variables and solving the corresponding problem.

The paper is structured as follows. In Sect. 2 we describe the proposed mathematical programming based approaches to refine heuristic partition. In particular, a mixed-integer quadratic model for modularity-maximizing clustering is recalled and the two strategies to refine partitions, that use such a model, are presented. In Sect. 3

a computational analysis and comparison, on a set of instances from the literature, is presented and discussed. Sect. 4 concludes the paper.

2 Mathematical programming based clustering refinement

Let us consider a partition found by a heuristic for network clustering. It is constituted by subnetworks induced by the clusters found. As a heuristic has been applied, there is no guarantee that the partition given by these subnetworks represents the optimal solution. Thus, one can seek an improved solution by applying a refinement technique.
We propose in this section mathematical programming-based refinement techniques, to be employed as post-processing of heuristics for modularity maximization. First, we recall the main elements of a mixed-integer quadratic model for modularity maximization which is used in these refinement techniques.

2.1 A MIQP mathematical programming model

Let $G = (V,E)$ be an undirected unweighted graph, with set of vertices V of order $n = |V|$ and set of edges E of size $m = |E|$. Modularity Q of G can be expressed as the sum of modularities of clusters, each one being a a function of its number of inner edges and of the sum of degrees of its vertices:

$$Q = \sum_s \left[\frac{m_s}{m} - \left(\frac{D_s}{2m} \right)^2 \right], \tag{1}$$

where m_s denotes the number of edges in cluster s, and D_s denotes the sum of degrees k_i of the vertices of cluster s.
In [39] a mixed-integer quadratic formulation is proposed, where (1) is the objective function to be maximized and binary variables are used to identify to which cluster each vertex and each edge belongs. Sets of allocation constraints, and constraints used to express that each vertex belongs to exactly one module, to impose lower and upper bounds on the cardinality of the modules and to break symmetries, fully define the model. In [10] this model is specialized to the case of two clusters only, i.e., a bipartition of the graph. Such a model for bipartitioning is recalled below. Notice that it has been also successfully used to build a hierarchical divisive clustering algorithm, see [9, 7].

First observe that in the case of bipartitioning the objective function (1) can be rewritten in a simpler form, expressing the sum of degrees of vertices belonging to one of the two clusters, say D_2, as a function of the sum of degrees D_1 of vertices belonging to the other one: $D_2 = D_c - D_1$, where D_c denotes the sum of degrees in the cluster c to be bipartitioned. The objective function to split cluster c can then be

written as the following quadratic function:

$$Q_c = \frac{m_1 + m_2}{m} - \frac{{D_1}^2}{2m^2} - \frac{{D_c}^2}{4m^2} + \frac{D_1 D_c}{2m^2}. \tag{2}$$

where m_1 and m_2 are respectively the number of edges inside the two clusters.
Decision variables are variables $X_{i,j,s}$ for each edge (v_i, v_j) and $s = 1,2$, with $X_{i,j,s}$ equal to 1 if the edge (v_i, v_j) is inside cluster s and 0 otherwise, and variables $Y_{i,1}$ for $i = 1,2,\ldots n$, equal to 1 if the vertex v_i is inside cluster 1 and 0 otherwise. Constraints on the problem are allocation constraints, used to impose that any edge (v_i, v_j) can belong to cluster s if and only if both of its end vertices i and j also belong to that cluster:

$$\forall (v_i, v_j) \in E_c \quad X_{i,j,1} \leq Y_{i,1} \tag{3}$$
$$\forall (v_i, v_j) \in E_c \quad X_{i,j,1} \leq Y_{j,1} \tag{4}$$
$$\forall (v_i, v_j) \in E_c \quad X_{i,j,2} \leq 1 - Y_{i,1} \tag{5}$$
$$\forall (v_i, v_j) \in E_c \quad X_{i,j,2} \leq 1 - Y_{j,1} \tag{6}$$

Further constraints express the number of edges of each of the two clusters and the sum of vertex degrees of the first cluster in terms of the decision variables X and Y, and finally integrality constraints are imposed on variables Y. Notice that integrality of variables X is implied by constraints (3)-(6), as well as integrality of D_1 follows by its defining constraint. The following mixed-integer quadratic (MIQP) model, that has a continuous convex relaxation, is finally obtained [10]:

$$(\mathscr{B}) \begin{cases} \max & Q_c & \\ \text{s.t.} & \forall (v_i, v_j) \in E_c & X_{i,j,1} \leq Y_{i,1} \\ & \forall (v_i, v_j) \in E_c & X_{i,j,1} \leq Y_{j,1} \\ & \forall (v_i, v_j) \in E_c & X_{i,j,2} \leq 1 - Y_{i,1} \\ & \forall (v_i, v_j) \in E_c & X_{i,j,2} \leq 1 - Y_{j,1} \\ & \forall s \in \{1,2\} & m_s = \sum_{(v_i,v_j) \in E_c} X_{i,j,s} \\ & & D_1 = \sum_{v_i \in V_c} k_i Y_{i,1} \\ & \forall s \in \{1,2\} & m_s \in \mathbb{R} \\ & D_1 \in \mathbb{R} & \\ & \forall v_i \in V_c & Y_{i,1} \in \{0,1\} \\ & \forall (v_i, v_j) \in E_c & \forall s \in \{1,2\} \quad X_{i,j,s} \in \mathbb{R}_0^+. \end{cases}$$

2.2 Splitting and merging clusters

In [10] we proposed a refinement technique for clustering results that is built on the mathematical programming formulation ($\mathscr{B}$) recalled above. First, clusters are considered one at a time and the bipartitioning problem ($\mathscr{B}$) is solved exactly, then pairs of clusters are merged and the exact bipartitioning is applied again. More pre-

cisely, in a sequence of steps, starting from the original partition obtained applying a heuristic, each cluster is first bipartitioned using an exact algorithm. Notice that ($\mathscr{B}$) is a MIQP with a continuous convex relaxation, that can be solved to global optimality by any standard solver for MIQP problems through the standard branch-and-bound method. If the modularity value corresponding to the obtained bipartition is higher than the one of the original cluster, then such original cluster is replaced by the new ones obtained by bipartition, otherwise the original cluster is kept. This sequence of bipartition attempts leads to a new, refined partition.
This new partition is furtherly refined by a new sequence of steps, where pairs of clusters, sorted by decreasing number of joining links, are provisionally merged and modularity of the merged cluster is compared to the sum of modularities of the two original clusters. In the case of improvement of the objective function value, the merged cluster is kept at the place of the two original ones. When merging is not beneficial in terms of improvement of the solution, the merged cluster is attempted to be splitted into two parts, according to the procedure applied in the first sequence of refining steps, exactly solving the bipartition problem. The two new clusters are possibly different from the original ones that have been merged, and can potentially correspond to an improved solution.

2.3 Fixing integer variables

We now present a novel mathematical programming- based approach to refine heuristic partitions. It is inspired by the methodology proposed by Xu et al. [38] for community detection in networks. In [38], the authors propose a two-stage procedure, where first a mixed-integer nonlinear problem (similar to that of [39] for a number of clusters generally greater than two, but where the only decision variables are binary variables Y expressing allocation of vertices to modules) is approximately solved to get an initial partition, and then a fixing and releasing scheme is applied. In this second stage, the authors consider the MIQP model in [39] and solve it, by standard solvers, iteratively fixing a certain number of variables Y to their value 1 and releasing the other variables, that are so free to take a value 1 or 0 depending on the way vertices are re-allocated in the current solution. Fixing integer variables gives a mathematical programming formulation with a reduced number of variables, and so more tractable.

We build upon the idea of fixing binary variables, though developing a different approach. Our approach is devised to refine approximate clustering solutions, so we start from the partition provided by a clustering heuristic, that replaces the first stage of the procedure in [38]. Then, we attempt to improve the original partition by acting on modules through a new heuristic based on variable fixing. Starting from an assignment of vertices to modules, i.e., from an assignment of 0-1 values to variables Y, we fix $nfix$ variables to their value 1 and compute a new value for the remaining variables, that is, we re-allocate the corresponding vertices. For each cluster, the vertices that are reallocated are chosen on the basis of their inner degree (the number

of neighbors of a vertex inside the cluster), moving first vertices that have a small inner degree and so are likely to have more connections inside a different cluster than the one they are assigned. A given number of (outer) iterations is performed, each one acting on a set Fix, containing variables whose value has to be fixed, and a set $Unfix$, containing variables to be released. To avoid using the same sets Fix-$Unfix$ in successive iterations, random perturbations are applied to these sets.

As acting on the whole graph requires to solve a mixed-integer nonlinear problem that may be quite large even with a number of variables that are fixed, and splitting and merging clusters appears to be an effective strategy for refinements [10], we integrate our fixing variables-based strategy in the procedure above based on splitting and merging clusters. To refine a given partition, again we implement the two consecutive steps performing respectively bipartitioning of each cluster and merging mixed to bipartitioning on pairs of clusters. Thus, we consider the MIQP formulation ($\mathscr{B}$), but in place of solving exactly the bipartitioning problem by standard branch-and-bound for MIQP, we apply our fixing variables-based strategy.

Thus, our refinement procedure works as follows.
First, each cluster of the original partition is splitted into two sets. To that effect, an initial approximate solution for the bipartition is computed and the above fixing variables-based approach is applied. If the modularity value corresponding to the obtained bipartition is higher than the one of the original cluster, then the original cluster is replaced by the new ones obtained by bipartition, otherwise the original cluster is kept. Once all clusters of the original partition have been examined, the merging-and-splitting procedure is applied. Pairs of clusters, sorted by decreasing number of joining links, are provisionally merged. If merging improves the objective function value, then the merged cluster is kept, otherwise it is splitted into two subsets again applying the fixing variables-based approach.

3 Computational results

In this section, we apply the proposed clustering refinement techniques to the partitions found by two known and heuristics for graph modularity maximization. The first one was proposed by Noack and Rotta [34] and is based on a single-step coarsening with a multi-level refinement. The second one was proposed in 2011 by Cafieri et al. [9] and is a hierarchical divisive heuristic that is locally optimal in the sense that bipartitions are computed by an exact optimization algorithm.

The first refinement technique (subsection 2.2) is implemented solving the mixed-integer quadratic bipartition problem ($\mathscr{B}$) using `CPLEX 12.2` [22], setting its parameters in such a way that the MIP cutting plane generation is disabled, the branching variable selection strategy is based on reduced pseudo costs, the number of nodes in the Branch-and-Bound tree is limited to 40000, and 1 only thread is used.
The fixing variables-based technique (subsection 2.3) is implemented using as a strarting guess an (approximate) affectation of variables provided by `CPLEX 12.2` limited to the solution at the root node, and then iterating the fixing variables scheme

over 100 iterations. At each iteration, the number of fixed variables is setted to half the cardinality of the current subgraph.

We test the proposed refinement algorithms on datasets in the literature, which correspond to networks modelling various real-life applications. Specifically, we consider a social network of dolphins [27], a network describing interactions among the characters of Hugo's novel *Les Misérables* [23], a biological network of protein-protein interactions [12], a network recording co-purchasing of political books on `Amazon.com` [24], a representation of the schedule of games between American college football teams in the Fall of 2000 [17], a network of connections between US airports [35], a network describing electronic circuits [25], e-mail interchanges between members of a university [21], a network giving the topology of the Western States Power Grid of the United States [37] and authors collaborations [35]. The considered datasets are listed in Table 1 together with their number of vertices n and number of edges m. Solutions have been obtained on a 2.4 GHz Intel Xeon CPU of a computer with 8GB RAM shared by three other similar CPU running Linux.

In Table 2 and Table 3 we report the results of the refinements of clustering results obtained using the Noack and Rotta's [34] (*NR*) heuristic and the Cafieri et al.'s [9] (*CHL*) heuristic respectively. We compare the results of the mathematical programming-based refinements described in Sect. 2, showing the original modularity value computed by the heuristic under consideration (*NR* or *CHL*), the intermediate result obtained by cluster splitting only and the final result after sequentially applying the splitting step and the merging step mixed to splitting, for the first refinement technique ($split$ and $mrg + spl$) (also in [10]) and respectively the new one based on fixing variables ($split\_fix$ and $mrg + spl\_fix$). We are able to obtain improved results for all the tested cases out of one (`political books`) refined with the fixing variables technique. Comparing the refined results with optimal modularity maximization solutions, when available in the literature [2], we remark that in some cases we get the optimal partitions, and in general very good quality solutions. The results obtained applying the two proposed refinements are generally comparable, and often we get the same modularity value (up to 5 decimal digits) in the two cases. When this is not the case, the values coincide up to 2 or 3 decimal digits.

In Table 4 we compare the two proposed approaches in terms of computing time. Very short times are spent in both cases on small-scale networks. For larger networks, it appears that the proposed approach based on fixing integer variables reduces sometimes significantly the time needed to refine the initial partition. This happens, as expected, especially for networks for which exact bipartitioning takes time because of the exploration of a large Branch-and-Bound tree. For example, improving the *NR* heuristic, time is reduced from 334.72 to 8.96 seconds for the 6-th dataset and from 919.74 to 241.29 seconds for the last dataset, and, improving the *CHL* heuristic, the reduction is from 454.64 to 16.86 seconds, again for the 6-th dataset.

Figure 1 illustrates the clustering of a network for which the optimal modularity-maximizing partition is obtained refining the *NR* heuristic result.

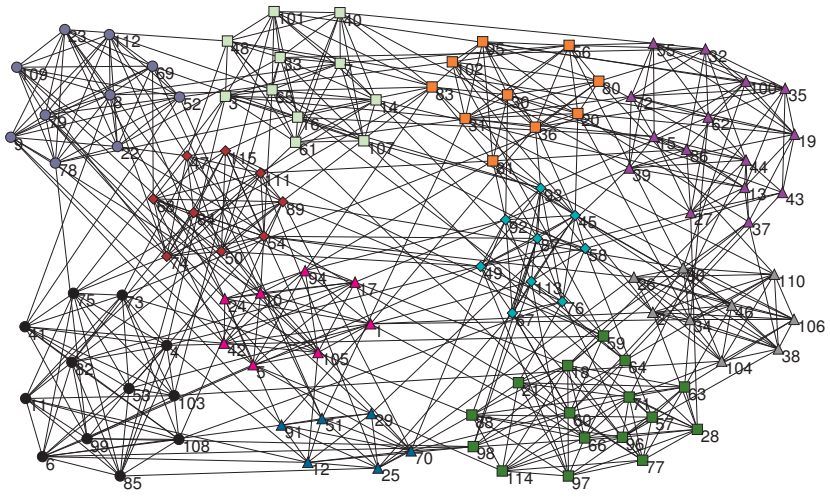

Fig. 1 Optimal clustering of network `football` obtained refining the *NR* heuristic result

Table 1 Datasets in the literature, whith their number of vertices n and number of edges m.

dataset	n	m
`dolphins`	62	159
`les miserables`	77	254
`p53_protein`	104	226
`political books`	105	441
`football`	115	613
`usair97`	332	2126
`netscience_main`	379	914
`s838`	512	819
`email`	1133	5452
`power`	4941	6594
`erdos02`	6927	11850

4 Conclusions

We proposed mathematical programming based approaches to refine graph clustering solutions. In particular we discussed and compared two approaches, the one in [10] based on splitting clusters and a combination of merging and splitting clusters, where bipartitions are computed exactly solving a MIQP problem, and a new one, based on iteratively fixing and releasing integer variables, again integrated in a splitting and merging-splitting scheme. We employ our approach as post-processing of some known heuristics for modularity maximization, obtaining improved solutions and, for some datasets, the optimal partition. The proposed approach based on fix-

Table 2 Modularity values corresponding to the partition found by the Noack and Rotta's heuristic [34] (Q_{NR}), by our first approach for refinement after the splitting step only (Q^{NR}_{split}) and after the merging and splitting step ($Q^{NR}_{mrg+spl}$), and by our fixing variables- based approach after the splitting step only ($Q^{NR}_{split\_fix}$) and after the merging and splitting step ($Q^{NR}_{mrg+spl\_fix}$). In the last column, the optimal modularity value Q_{opt} is reported, when available in the literature [2].

dataset	Q_{NR}	Q^{NR}_{split}	$Q^{NR}_{mrg+spl}$	$Q^{NR}_{split\_fix}$	$Q^{NR}_{mrg+spl\_fix}$	Q_{opt} [2]
dolphins	0.52377	0.52773	0.52852	0.52508	0.52646	0.52852
les miserables	0.56001	0.56001	0.56001	0.56001	0.56001	0.56001
p53_protein	0.53216	0.53216	0.53502	0.53216	0.53502	0.53513
political books	0.52694	0.52724	0.52724	0.52694	0.52694	0.52724
football	0.60028	0.60237	0.60457	0.60237	0.60457	0.60457
usair97	0.36577	0.36577	0.36808	0.36577	0.36808	0.3682
netscience_main	0.84745	0.84828	0.84842	0.84828	0.84842	0.8486
s838	0.81624	0.81624	0.81656	0.81624	0.81656	0.8194
email	0.57740	0.57741	0.57776	0.57741	0.57768	–
power	0.93854	0.93867	0.93873	0.93854	0.93858	–
erdos02	0.75926	0.75926	0.76958	0.75926	0.78952	–

Table 3 Modularity values corresponding to the partition found by the Cafieri et al.'s heuristic [9] (Q_{CHL}), by our first approach for refinement after the splitting step only (Q^{CHL}_{split}) and after the merging and splitting step ($Q^{CHL}_{mrg+spl}$), and by our fixing variables- based approach after the splitting step only ($Q^{CHL}_{split\_fix}$) and after the merging and splitting step ($Q^{CHL}_{mrg+spl\_fix}$). In the last column, the optimal modularity value Q_{opt} is reported, when available in the literature [2].

dataset	Q_{CHL}	Q^{CHL}_{split}	$Q^{CHL}_{mrg+spl}$	$Q^{CHL}_{split\_fix}$	$Q^{CHL}_{mrg+spl\_fix}$	Q_{opt} [2]
dolphins	0.52646	0.52646	0.52680	0.52646	0.52680	0.52852
les miserables	0.54676	0.54676	0.55351	0.54676	0.55351	0.56001
p53_protein	0.53000	0.53000	0.53004	0.53000	0.53145	0.53513
political books	0.52629	0.52629	0.52678	0.52629	0.52678	0.52724
football	0.60091	0.60091	0.60112	0.60091	0.60112	0.60457
usair97	0.35959	0.35959	0.35975	0.35959	0.35960	0.3682
netscience_main	0.84702	0.84702	0.84703	0.84702	0.84703	0.8486
s838	0.81663	0.81663	0.81675	0.81663	0.81667	0.8194
email	–	–	–			–
power	0.93937	0.93937	0.93941	0.93937	0.93941	–
erdos02	–	–	–			–

ing integer variables allow us to significantly reduce the computing time needed to provide an improved clustering solution.

Acknowledgements The first author has been supported by French National Research Agency (ANR) through grant ANR 12-JS02-009-01 "ATOMIC".

Table 4 Computing time (seconds) required by the proposed approaches applied as post-processing to Noack and Rotta's heuristic ($time^{NR}$) and Cafieri et al.'s heuristic ($time^{CHL}$). Solutions have been obtained on a 2.4 GHz Intel Xeon CPU of a computer with 8GB RAM shared by three other similar CPU running Linux.

dataset	$time^{NR}_{mrg+spl}$	$time^{NR}_{mrg+spl\_fix}$	$time^{CHL}_{mrg+spl}$	$time^{CHL}_{mrg+spl\_fix}$
dolphins	0.20	0.39	0.26	0.20
les miserables	0.67	0.71	0.35	0.30
p53_protein	1.02	1.23	0.26	0.49
political books	5.10	1.66	3.41	1.21
football	3.26	3.16	0.99	0.83
usair97	334.72	8.96	454.64	16.86
netscience_main	1.38	1.67	0.77	0.85
s838	1.20	1.40	1.06	1.16
email	57.80	56.02	–	–
power	18.62	15.81	17.50	15.42
erdos02	919.74	241.29	–	–

References

1. G. Agarwal and D. Kempe. Modularity-maximizing graph communities via mathematical programming. *The European Physical Journal B*, 66(3):409–418, 2008.
2. D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, L. Liberti, and S. Perron. Column generation algorithms for exact modularity maximization in networks. *Physical Review E*, 82(4):046112, 2010.
3. D. Aloise, G. Caporossi, P. Hansen, L. Liberti, S. Perron, and M. Ruiz. Contemporary Mathematics 588.
4. V.D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal Statistical Mechanics: Theory and Experiment*, page P10008, 2008.
5. S. Boccaletti, M. Ivanchenko, V. Latora, A. Pluchino, and A. Rapisarda. Detecting complex network modularity by dynamical clustering. *Physical Review E*, 75:045102, 2007.
6. U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008.
7. S. Cafieri, A. Costa, and P. Hansen. Reformulation of a model for hierarchical divisive graph modularity maximization. *Annals of Operations Research*, 2012. DOI 10.1007/s10479-012-1286-z (in press).
8. S. Cafieri, P. Hansen, and L. Liberti. Loops and multiple edges in modularity maximization of networks. *Physical Review E*, 81(4):046102, 2010.
9. S. Cafieri, P. Hansen, and L. Liberti. Locally optimal heuristic for modularity maximization of networks. *Physical Review E*, 83(5):056105, 2011.
10. S. Cafieri, P. Hansen, and L. Liberti. Improving heuristics for network modularity maximization using an exact algorithm. *Discrete Applied Mathematics*, 163(1):65–72, 2014.
11. A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70:066111, 2004.
12. L. Dartnell, E. Simeonidis, M. Hubank, S. Tsoka, I.D.L. Bogle, and L.G. Papageorgiou. Self-similar community structure in a network of human interactions. *FEBS Letters*, 579:3037–3042, 2005.
13. H.N. Djidjev. A scalable multilevel algorithm for graph clustering and community structure detection. *Lecture Notes in Computer Science*, 4936:117–128, 2008.

14. J. Duch and A. Arenas. Community identification using extremal optimization. *Physical Review E*, 72(2):027104, 2005.
15. S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.
16. S. Fortunato and M. Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences, USA*, 104(1):36–41, 2007.
17. M. Girvan and M. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences, USA*, 99(12):7821–7826, 2002.
18. M. Grötschel and Y. Wakabayashi. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45:59–96, 1989.
19. M. Grötschel and Y. Wakabayashi. Facets of the clique partitioning polytope. *Mathematical Programming*, 47:367–387, 1990.
20. R. Guimerà and A.N. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433:895–900, 2005.
21. R. Guimerà, L. Danon, A. Diaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Physical Review E*, 68:065103, 2003.
22. IBM. *ILOG CPLEX 12.2 User's Manual*. IBM, 2010.
23. D.E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, Reading, MA, 1993.
24. V. Krebs. `http://www.orgnet.com/` (unpublished).
25. Uri Alon Lab. `http://www.weizmann.ac.il/mcb/UriAlon/`.
26. S. Lehmann and L.K. Hansen. Deterministic modularity optimization. *European Physical Journal B*, 60:83–88, 2007.
27. 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. can geographic isolation explain this unique trait? *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.
28. C.P. Massen and J.P.K. Doye. Identifying communities within energy landscapes. *Physical Review E*, 71:046101, 2005.
29. A. Medus, G. Acuna, and C.O. Dorso. Detection of community structures in networks via global optimization. *Physica A*, 358:593–604, 2005.
30. J. Mei, S. He, G. Shi, Z. Wang, and W. Li. Revealing network communities through modularity maximization by a contraction-dilation method. *New Journal of Physics*, 11:043025, 2009.
31. M. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences, USA*, 103(23):8577–8582, 2006.
32. M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.
33. M. E. J. Newman. *Networks: an introduction*. Oxford University Press, Oxford, 2010.
34. A. Noack and R. Rotta. Multi-level algorithms for modularity clustering. *Lecture Notes in Computer Science*, 5526:257–268, 2009.
35. `http://vlado.fmf.uni-lj.si/pub/networks/data/`.
36. M. Tasgin, A. Herdagdelen, and H. Bingol. Community detection in complex networks using genetic algorithms. *arXiv:0711.0491*, 2007.
37. D.S. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):409–410, 1998.
38. G. Xu, Bennett, L.G. Papageorgiou, and S. Tsoka. Module detection in complex networks using integer optimisation. *Algorithms for Molecular Biology*, 5(36), 2010. DOI:10.1186/1748-7188-5-36.
39. G. Xu, S. Tsoka, and L.G. Papageorgiou. Finding community structures in complex networks using mixed integer optimization. *European Physical Journal B*, 60:231–239, 2007.