Improving heuristics for network modularity maximization using an exact algorithm

Sonia Cafieri, Pierre Hansen, Leo Liberti

Improving heuristics for network modularity maximization using an exact algorithm

Sonia Cafieri[a], Pierre Hansen[b,c], Leo Liberti[c]

[a]Laboratoire MAIAA, École Nationale de l'Aviation Civile, 7 Av. E. Belin, 31055 Toulouse, France
[b]GERAD and HEC Montréal, 3000 Chemin de la Côte-Sainte-Catherine, Montréal, Canada, H3T 2A7
[c]LIX, École Polytechnique, 91128 Palaiseau, France

Abstract

Heuristics are widely applied to modularity maximization models for the identification of communities in complex networks. We present an approach to be applied as a post-processing to heuristic methods in order to improve their performances. Starting from a given partition, we test with an exact algorithm for bipartitioning if it is worthwhile to split some communities or to merge two of them. A combination of merge and split actions is also performed. Computational experiments show that the proposed approach is effective in improving heuristic results.

Keywords: clustering, bipartition, network, graph, community, modularity, heuristic, exact algorithm, matheuristic

1. Introduction

The identification of communities in complex networks has become in recent years a very active research domain [1, 2] because of the common representation of complex real-world systems arising in a variety of fields as networks. One then aims to find *communities*, or *clusters*, of entities grouped on the basis of some relationship holding among them. Telecommunication

Email addresses: sonia.cafieri@enac.fr (Sonia Cafieri), pierre.hansen@gerad.ca (Pierre Hansen), liberti@lix.polytechnique.fr (Leo Liberti)

networks such as the World Wide Web, biological networks representing interactions between proteins and social networks representing collaborations or conflicts between people or countries are some examples of real-life applications (see [1] for a detailed introduction).

Intuitively, one would say that a set of vertices of a network form a community if edges joining two vertices of that set are frequent and edges joining a vertex of that set to a vertex outside are not. This concept has been refined in many ways, leading to the introduction of concepts of *modularity* [3], *modularity density* [4], *min-max cut* [5], *normalized cut* [6] and others. Among these concepts, modularity is by far the most popular.

Modularity of a community is defined in [3] as the difference between the fraction of edges it contains and the expected fraction of edges it would contain if they were placed at random, keeping the same degree distribution. Then, modularity of a partition of a network into communities is defined as the sum of the modularities of these communities. Modularity expresses not only that a community contains a large fraction of the edges, but also that it contains a larger fraction of the edges than would be expected. It can be viewed as a measure of the extent to which the classes of a partition of a graph can be considered to be communities. Alternatively, modularity can be maximized to find an optimal partition of a network.

Modularity maximization has spawned in recent years numerous methods for cluster identification in networks. Despite its popularity, the accuracy and the significance of modularity maximizing modules are not well understood for real-world networks [7]. Furthermore, some criticism have been raised in recent literature, see, e.g., [7, 8, 9, 10, 11]. The two main concerns are the existence of a resolution limit and the fact that modularity function exhibits a degeneracy. The resolution limit, identified by Fortunato and Barthelemy [8], implies that, in the presence of large clusters, some clusters smaller than a certain size which depends on the number of edges of the network can be undetectable. As a consequence, modular structures like small cliques can be hidden in larger clusters. This effect appears to be driven primarly by the assumption that inter-module connectivity follows a random graph model [7]. Degeneracy (see [7]) implies that there can be a large number of partitions, even very different from each other, having high modularity values. This makes it easy to find high-scoring partitions but difficult to identify the global optimum. To address these criticisms a few approaches have been proposed. Sales-Pardo et al. [12] address the problem of degenearcy combining information from many distinct partitions with high modularity. Multiresolution methods [13, 14, 15] allow to specify a target resolution limit and identify clusters on such given scale, though they do not solve the problem in a fully satisfactory manner. Despite these criticisms, modularity maximization still appears to be a very popular technique for network clustering. It exhibits, in fact, some clear advantages: modularity function has a clear and simple mathematical description and does not depend on parameters being decided arbitrarily (as an example, maximizing the number of intracluster edges requires some other parameter, e.g. the minimum cluster size); modularity maximization gives an optimal partition together with the number of clusters not to be specified in advance. Interestingly, one can use mathematical programming to model the community detection problem and, using modularity maximization, the splitting of a cluster into two can be expressed as a quadratic programming problem. This paper discusses such a formulations and exploits it within a procedure used as a refinement of previously computed partitions.

Numerous heuristics and a few algorithms have been proposed to find near optimal or optimal partitions respectively for the maximum modularity criterion. Heuristics are either partitioning methods or hierarchical divisive or agglomerative ones. Partitioning heuristics are based on simulated annealing [16, 10, 17], mean field annealing [18], genetic search [19], extremal optimization [20], spectral clustering [21], linear programming followed by randomized rounding [22], dynamical clustering [23], multilevel partitioning [24], contraction-dilation [25], quantum mechanics [26] and several other approaches [27, 28**?** , 29, 30, 13]. Agglomerative hierarchical clustering [31, 32, 33, 34, 27] proceeds from an initial partitions into communities each containing a single vertex to merging sequentially vertices or sets of vertices corresponding to communities. In [35] this approach is combined with a vertex mover routine which improves the partitions by changing the community of a vertex to that of one of its adjacent vertices. Divisive hierarchical clustering proceeds from an initial trivial partition in one community containing all vertices and sequentially selects a community and proceeds to its bipartitioning. Divisive heuristics are much less frequent than agglomerative ones. The best known of them is Newman's spectral heuristic [21], which uses the signs of the first eigenvector of the modularity matrix to perform successive bipartitions. In a companion paper [36], we propose a hierarchical divisive heuristic which is locally optimal, i.e., in which all successive bipartitions are done in an optimal way.

These heuristics are able to solve large instances with up to thousand or tens of thousands of vertices (and sometimes over a million) and therefore are often preferred to exact algorithms, even though they do not have a guarantee of optimality. Only a few papers propose exact algorithms for maximizing modularity. The first one, due to Xu et al. [37], uses quadratic mixed-integer programming with a convex relaxation. Networks with up to 104 vertices were addressed successfully. Brandes et al. [38] have shown that modularity maximization is NP-hard, even if there are only two communities. In addition, they propose to express modularity maximization as a clique partitioning problem. They maximize modularity of networks with up to 105 vertices. Their algorithm is close to that one of Grötschel and Wakabayashi [39, 40]. Aloise et al. [41] apply column generation to modularity maximization and solve exactly instances with up to 512 vertices.

Given a partition found by a heuristic, one can apply another heuristic or an exact algorithm to the subnetworks induced by the communities found. This will eventually lead to a new, better, partition. Moreover, this refinement can be based on splitting a community or merging a pair of communities. In the spirit of *matheuristics*, an exact algorithm for bipartition is applied in our approach first to the communities considered one at a time, then merging pairs of communities and applying again the bipartition algorithm.

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 paper is organized as follows. In the next section, the proposed approach to improve heuristic results for modularity maximization is described, presenting in particular an exact algorithm for bipartition. Section 3 presents the results of computational experiments carried out applying the proposed approach as post-processing to three of the best heuristics available for modularity maximization of networks, i.e., the agglomerative hierarchical heuristic of Clauset et al. [32], the partitioning heuristic of Noack and Rotta [42] and the multistep greedy with vertex move heuristic of Schuetz and Caflisch [35]. We also apply this approach to the locally optimal divisive hierarchical heuristic of [36]. Conclusions are given in Section 4.

2. Improving heuristics for modularity maximization

2.1. An exact algorithm for bipartition

We present in this section an exact algorithm for modularity maximizing bipartition of networks. Although it can be applied in full generality to any graph, we specifically apply it in the role of post-processing step to heuristics for the identification of communities in networks.

We model this bipartitioning problem using binary variables to identify to which community each vertex and each edge belongs. In this respect, our model is similar to that of Xu et al. [37]. These authors proposed in 2007 [37] a model for modularity maximization of networks which leads to an optimal partition generally with more than two communities. Their model is a mixed-integer quadratic program with a convex relaxation.

Let $G = (V, E)$ be a graph, or network, with vertex set V of cardinality n and edge set E of of cardinality m. First, we recall the definition of modularity Q as a sum over communities of their modularities [3]:

$$Q = \sum_s [a_s - e_s],$$

where a_s is the fraction of all edges that lie within community s, and e_s is the expected value of the same quantity in a graph in which the vertices have the same degrees but edges are placed at random. Modularity can then be written equivalently as:

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

where m_s denotes the number of edges in community s, i.e., which belong to the subgraph induced by the vertex set V_s of that community, and d_s denotes the sum of degrees k_i of the vertices of community s. Since we aim to find a bipartition, only two sub-modules of the original community have to be considered, i.e. $s \in \{1, 2\}$. We can express the sum of degrees d_2 of vertices belonging to the second community as a function of the sum of degrees d_1 of vertices belonging to the first one:

$$d_2 = d_t - d_1, \tag{2}$$

where d_t is the sum of degrees in the community to be bipartitioned. It is equal to $2m$ at the first iteration. We rewrite (1) for $s \in \{1, 2\}$, using (2):

$$\begin{aligned} Q &= \frac{m_1 + m_2}{m} - \frac{d_1^2}{4m^2} - \frac{d_2^2}{4m^2} = \\ &= \frac{m_1 + m_2}{m} - \frac{d_1^2}{4m^2} - \frac{d_t^2 + d_1^2 - 2d_t d_1}{4m^2} = \\ &= \frac{m_1 + m_2}{m} - \frac{d_1^2}{2m^2} - \frac{d_t^2}{4m^2} + \frac{d_t d_1}{2m^2}. \end{aligned} \tag{3}$$

We then introduce binary variables X_{r1}, X_{r2} and Y_{i1} to model assignment of vertices and edges to the two communities of the bipartition. These variables are defined as follows:

$$X_{rs} = \begin{cases} 1 & \text{if edge } r \text{ belongs to community } s \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

for $r = 1, 2, \ldots m$ and $s = 1, 2$ and

$$Y_{i1} = \begin{cases} 1 & \text{if vertex } i \text{ belongs to community 1} \\ 0 & \text{otherwise, i.e., vertex } i \text{ belongs to community 2} \end{cases} \tag{5}$$

for $i = 1, 2, \ldots n$.
We impose that any edge $r = \{v_i, v_j\}$ with end vertices indiced by i and j can only belong to community s if both of its end vertices belong also to that community:

$$\begin{aligned} X_{r1} &\leq Y_{i1} \quad \forall r = \{v_i, v_j\} \in E \\ X_{r1} &\leq Y_{j1} \quad \forall r = \{v_i, v_j\} \in E \end{aligned} \tag{6}$$

and

$$\begin{aligned} X_{r2} \leq 1 - Y_{i1} \quad \forall r = \{v_i, v_j\} \in E \\ X_{r2} \leq 1 - Y_{j1} \quad \forall r = \{v_i, v_j\} \in E \quad . \end{aligned} \tag{7}$$

Furthermore, we exploit the following expressions in terms of variables X_{r1}, X_{r2}, $r = 1, 2, \ldots m$, and Y_{i1}, $i = 1, 2, \ldots n$, for the number of edges of each of the two communities and the sum of vertex degrees of the first one:

$$m_s = \sum_r X_{rs} \quad \forall s \in \{1, 2\}, \tag{8}$$

$$d_1 = \sum_{i \in V_1} k_i Y_{i1}. \tag{9}$$

Only the sum of vertex degrees of the first community is exploited, because of expression (2).

We then have the following integrality constraints on the variables:

$$\begin{array}{ll} X_{rs} \in \{0,1\} & \forall r = \{v_i, v_j\} \in E,\ \forall s \in \{1,2\} \\ Y_{i1} \in \{0,1\} & \forall i \in \{1,\dots,n\} \\ m_1, m_2, d_1 \in \mathbb{N}^+. & \end{array} \tag{10}$$

Maximizing modularity (3) subject to constraints (6)-(7) and (8)-(9) and replacing the integrality constraints (10) by range constraints gives a quadratic mixed-integer program with a convex relaxation which can be solved by recent versions of `CPLEX` [43]. This model has $2m+n+3$ variables and $4m+3$ constraints. For sparse networks, as is the case in many applications, these sizes are reasonable.

2.2. Improving a partition by merging and splitting

The proposed post-processing heuristic aims at improving the modularity of a given partition obtained with some heuristic. A new partition is obtained in a sequence of steps, which act on the current communities by splitting and merging.

First, we split each community of the original partition into two subcommunities by applying the exact algorithm for bipartition described in subsection 2.1. We then check if the modularity value corresponding to the obtained bipartition is higher than the one of the original community. This comparison is justified by the definition of modularity of a partition as sum of modularities of its communities. If the new modularity value is higher than that one of the original community, this community is replaced by the two new communities. Otherwise the two obtained communities are discarded and the original one is kept. When all the original communities have been checked, a new partition is obtained with a higher modularity than before if at least one bipartition has been accepted.

Second, we merge provisionally pairs of communities and check if this induces an increased value for modularity. For each pair of communities, we consider the new community containing all vertices of this pair and check if the larger community has a modularity value higher than the sum of the modularities of the two original communities. If this is the case, the new large community is kept in place of the other two. Otherwise, if merge is not beneficial, we try to split the merged community using again the exact algorithm presented in subsection 2.1. As before, the two communities resulting from the bipartition are kept if the sum of their modularities is higher than the modularity of the splitted community. Obviously, pairs of clusters to be merged can be selected according to different criteria. We compute the number of edges joining pairs of clusters, that is the number of edges joining vertices belonging to the first cluster of the pair with vertices belonging to the second cluster. Then, the pairs are sorted by decreasing number of joining links. This gives the list of pairs of clusters to be considered for merging. In this way, we first attempt to improve the current partition by merging clusters which are more strongly connected than others, so that a merging can be expected to be beneficial.

A sketch of our algorithm is given in Alg. 1.

3. Computational results

We apply our approach as a post-processing heuristic to three known heuristics due to Clauset, Newman and Moore [32], Noack and Rotta [42] and Schuetz and Caflisch [35]. We also apply it to the locally optimal divisive hierarchical heuristic of [36]. Clauset et al. [32] proposed in 2004 an efficient implementation of an agglomerative hierarchical scheme, that for sparse networks has a very low complexity and is considerably faster than previously proposed methods. Noack and Rotta [42] presented in 2008 a comparison of heuristics for modularity maximization and proposed a heuristic based on a single-step coarsening with a multi-level refinement, which is competitive with other methods in the literature. Schuetz and Caflisch [35] introduced in 2008 a multistep extension of the greedy heuristic and combined it with a vertex-by-vertex refinement procedure, called vertex mover. Their main idea is to promote simultaneous merging of several pairs of communities. Moreover, the vertex mover acts as an efficient ascent heuristic, used repetitively. The present authors proposed in 2011 a hierarchical divisive heuristic where bipartitions are done exactly using the model of Section 2.

Our computational results have been obtained on some datasets that are often used to evaluate heuristics and algorithms for identification of communities in networks. These datasets correspond to various real-life applications: a social network of dolphins described by Lusseau [44], a network describing interactions among the characters of Hugo's novel *Les Misérables* [45], a network dealing with protein-protein interactions [46], a network recording co-purchasing of political books on `Amazon.com` [47], a network representing the schedule of games between American college football teams in the Fall of 2000 [48], a network dealing with connections between US airports [49], a network describing electronic circuits [50], a network representing e-mail interchanges between members of a university [51], a network giving the topology of the Western States Power Grid of the United States [52] and a network of authors collaborations [49].

In our implementation, the quadratic mixed-integer program with a convex relaxation which models the modularity maximizing bipartition problem is solved using `CPLEX 12.2` [43], with the following parameters: 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.

In Table 1 we report, for each dataset, the values of modularity computed by the four considered heuristics and by the proposed approach when applied as post-processing to the partitions obtained with these heuristics, together with the optimal value of modularity, when available in the literature. The number of vertices n and the number of edges m of the datasets are also reported.

It appears that:

- the best result obtained with the four heuristics and our proposed post-processing approach is optimal 4 cases out of the 8 for which an optimal value is known.

- However, for the four cases in which the optimal solution could not be found, the error between the optimal value and the best value found by one of the heuristics appears to be very moderate, i.e., 0.00011 or 0.021% for `p53_protein` and 0.000016 or 0.043% for `usair97`, 0.000022 or 0.026% for `netscience_main` and 0.00265 or 0.32% for `s838`.

- The proposed approach is very efficient in the sense that it improved the values given by the heuristics in all cases for all of them, except for `les_miserables` for which the optimal solution was already obtained by Noack and Rotta's and Schuetz and Caflisch's heuristics.

- After post-processing, the Noack and Rotta's heuristic gives the best results in 8 cases over 11, the Schuetz and Caflish's heuristic in 4 cases over 11, which are a subset of the 8 cases solved by the Noack and Rotta's heuristic, the Clauset et al.'s heuristic found the best solution in

2 cases out of 11, i.e., `political_books` (for which it was also obtained by the Noack and Rotta's and the Schuetz and Caflish's heuristics) and `erdos02`. Finally, the best solution after post-processing was found by the locally optimal divisive heuristic in 2 cases, i.e., `s838` and `power`.

- The average value of modularity for the Clauset et al.'s heuristic over 11 problems is 0.616975 before post-processing and 0.634178 after post-processing, the average improvement is 0.0172027 and the corresponding percentage of increase in modularity is 2.78824%. The average value of modularity for the Noack and Rotta's heuristic over 11 problems is 0.640711 before post-processing and 0.643135 after post-processing, the average improvement is 0.0024245 and the corresponding percentage of increase in modularity is 0.378415%. The average value of modularity for the Schuetz and Caflisch's heuristic over 11 problems is 0.640084 before post-processing and 0.643521 after post-processing, the average improvement is 0.0034373 and the corresponding percentage of increase in modularity is 0.537004%. Average value of modularity for the Cafieri et al.'s heuristic over 9 problems is 0.632559 before post-processing and 0.633466 after post-processing, the average improvement is 0.000906667 and the corresponding percentage of increase in modularity is 0.143333%.

The approach proposed in the present paper is based on two main steps, which are applied sequentially. We call these steps `split` and `merge+split` for short. In order to evaluate the impact of the two steps, we report in Tables 2 and 3 the modularity values obtained applying `split` and `merge+split` starting from Clauset et al.'s (*CNM*) solution and Noack-Rotta's (*NR*) solution for the first table and starting from Schuetz and Caflisch's (*SC*) solution and from Cafieri et al.'s (*CHL*) solution for the second table respectively. Note that modularity values for `merge+split` are the final results provided by our moves, already shown in Table 1. These results show that the splitting step provides in most cases a significant improvement of the original partition. Examples are given by `dolphin`, `political_books`, `football`, `usair97`, `netscience_main` and `email` datasets (that is, 6 cases out of 11) for *CNM* and by `p53_protein` dataset for *SC*, where an improvement on the second decimal digit of modularity value is obtained. Furthermore, the splitting step provides the optimal solution of `political_books` dataset for *NR* and of `political_books` and `football` datasets for *SC*. By contrast, this step does not provide for some instances a better partition

dataset	n	m	Q_{CNM}	Q'	Q_{NR}	Q''	Q_{SC}	Q'''	Q_{CHL}	Q^{iv}	Q_{opt}
`dolphin`	62	159	0.49549	0.52011	0.52377	**0.52852**	0.52456	**0.52852**	0.52646	0.52680	0.52852
`les miserables`	77	254	0.50060	0.52438	0.56001	**0.56001**	0.56001	**0.56001**	0.54676	0.55351	0.56001
`p53_protein`	104	226	0.52052	0.52621	0.53216	**0.53502**	0.51825	0.52910	0.53000	0.53004	0.53513
`political_books`	105	441	0.50197	**0.52724**	0.52694	**0.52724**	0.52694	**0.52724**	0.52629	0.52678	0.52724
`football`	115	613	0.57728	0.59121	0.60028	**0.60457**	0.60316	**0.60457**	0.60091	0.60112	0.60457
`usair97`	332	2126	0.32039	0.36221	0.36577	**0.36808**	0.36374	0.36458	0.35959	0.35975	0.3682
`netscience_main`	379	914	0.83829	0.84551	0.84745	**0.84842**	0.84567	0.84754	0.84702	0.84703	0.8486
`s838`	512	819	0.80556	0.80666	0.81624	0.81656	0.81274	0.81364	0.81663	**0.81675**	0.8194
`email`	1133	5452	0.51169	0.54736	0.57740	**0.57776**	0.57425	0.57660	–	–	–
`power`	4941	6594	0.93402	0.93658	0.93854	0.93873	0.93679	0.93752	0.93937	**0.93941**	–
`erdos02`	6927	11850	0.78092	**0.79543**	0.75926	0.76958	0.77481	0.78941	–	–	–

Table 1: Results on real-world datasets: comparison between the modularity values found by heuristics and by the proposed approach applied as post-processing. Q_{CNM}, Q_{NR}, Q_{SC} and Q_{CHL} are modularities computed using heuristics by Clauset et al. [32], Noack and Rotta [42], Schuetz and Caflisch [35] and Cafieri et al. [36]. Q', Q'', Q''', Q^{iv} are modularities computed applying the proposed approach to the partitions obtained by these heuristics. Q_{opt} are the optimal modularity values as reported in the literature. n and m are the number of vertices and the number of edges of the networks.

than the original one, leading to an unchanged modularity value. Examples are given by `erdos02` dataset for *CNM*, by `p53_protein`, `usair97`, `s838` and `erdos02` datasets for *NR*, and by `usair97` and `erdos02` datasets for *SC*. The splitting step never improves solutions found by the fourth heuristic *CHL*, as expected being the splitting step of that divisive heuristic already performed by using the exact algorithm of Section 2.1. This behavior shows the importance of a combined use of both splitting and merging steps in the proposed approach to obtain eventually a new, better, partition.

Table 4 shows computing time required by our post-processing strategy applied to the four considered heuristics to get an improved solution. Results 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. As expected, times are roughly increasing with network dimension, even though they depend mostly on the quality of the initial partition and the cardinality of its communities to be handled. Times are in general reasonably moderate, and very short times are spent on most of the tested networks. The optimal partition is found in less than 1 second for `dolphin` dataset and in less than 4 seconds for `football` dataset starting from *NR* and *SC* solutions and in less than 4.30 seconds and slightly more than 5 seconds respectively for `political_books` starting from *CNM*, *NR* and *SC* solutions.

dataset	*CNM*			*NR*		
	Q_{CNM}	Q'_{split}	$Q'_{merge+split}$	Q_{NR}	Q''_{split}	$Q''_{merge+split}$
`dolphin`	0.49549	0.51693	0.52011	0.52377	0.52773	0.52852
`les miserables`	0.50060	0.50732	0.52438	0.56001	0.56001	0.56001
`p53_protein`	0.52052	0.52518	0.52621	0.53216	0.53216	0.53502
`political_books`	0.50197	0.52708	0.52724	0.52694	0.52724	0.52724
`football`	0.57728	0.58232	0.59121	0.60028	0.60237	0.60457
`usair97`	0.32039	0.36157	0.36221	0.36577	0.36577	0.36808
`netscience_main`	0.83829	0.84537	0.84551	0.84745	0.84828	0.84842
`s838`	0.80556	0.80639	0.80666	0.81624	0.81624	0.81656
`email`	0.51169	0.53939	0.54736	0.57740	0.57741	0.57776
`power`	0.93402	0.93605	0.93658	0.93854	0.93867	0.93873
`erdos02`	0.78092	0.78092	0.79543	0.75926	0.75926	0.76958

Table 2: Modularity values corresponding to the partition found by the heuristic and by our approach after the splitting step only (Q'_{split}, Q''_{split}) and after the successive application of the merging and splitting step ($Q'_{merge+split}, Q''_{merge+split}$) for Clauset et al.'s heuristic (*CNM*) and Noack and Rotta's heuristic (*NR*).

dataset	*SC*			*CHL*		
	Q_{SC}	Q'''_{split}	$Q'''_{merge+split}$	Q_{CHL}	Q^{iv}_{split}	$Q^{iv}_{merge+split}$
`dolphin`	0.52456	0.52852	0.52852	0.52646	0.52646	0.52680
`les miserables`	0.56001	0.56001	0.56001	0.54676	0.54676	0.55351
`p53_protein`	0.51825	0.52663	0.52910	0.53000	0.53000	0.53004
`political_books`	0.52693	0.52724	0.52724	0.52629	0.52629	0.52678
`football`	0.60316	0.60457	0.60457	0.60091	0.60091	0.60112
`usair97`	0.36374	0.36374	0.36458	0.35959	0.35959	0.35975
`netscience_main`	0.84566	0.84587	0.84754	0.84702	0.84702	0.84703
`s838`	0.81274	0.81338	0.81364	0.81663	0.81663	0.81675
`email`	0.57425	0.57557	0.57660	–	–	–
`power`	0.93678	0.93718	0.93752	0.93937	0.93937	0.93941
`erdos02`	0.77481	0.77481	0.78941	–	–	–

Table 3: Modularity values corresponding to the partition found by the heuristic and by our approach after the splitting step only ($Q'''_{split}, Q^{iv}_{split}$) and after the successive application of the merging and splitting step ($Q'''_{merge+split}, Q^{iv}_{merge+split}$) for Schuetz and Caflisch's heuristic (*SC*) and Cafieri et al.'s heuristic (*CHL*).

4. Conclusion

This paper describes the application of an approach based on an exact algorithm for bipartitioning a network, in the framework of split and merge

dataset	$time_{CNM}$	$time_{NR}$	$time_{SC}$	$time_{CHL}$
dolphin	0.58	0.20	0.38	0.26
les miserables	0.32	0.67	0.68	0.35
p53 protein	0.18	1.02	0.15	0.26
political books	4.25	5.10	5.10	3.41
football	0.95	3.26	2.51	0.99
usair97	1000.96	334.72	525.91	454.64
netscience_main	1.02	1.38	0.75	0.77
s838	0.73	1.20	0.63	1.06
email	15591.40	57.80	1196.91	–
power	21.63	18.62	20.53	17.50
erdos02	525.37	919.74	15043.85	–

Table 4: Computing time (seconds) required by the proposed approach applied as post-processing to Clauset at al.'s heuristic ($time_{CNM}$), Noack and Rotta's heuristic ($time_{NR}$), Schuetz and Caflisch's heuristic ($time_{SC}$) 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.

movements on communities of a network partition. Computational results obtained on a set of examples from the literature, applying the proposed approach as post-processing to four heuristics for modularity maximization of networks, show the impact of an exact approach on the improvement of heuristic results.

The presented approach can be easily applied in full generality to any modularity maximization based heuristic to improve the quality of the partition provided by the heuristic.

It has been succesfully exploited to develop a hierarchical divisive clustering heuristic which is locally optimal [36] and may be further developed including the described moves directly in a local search heuristic.

References

[1] M. E. J. Newman, Networks: an introduction, Oxford University Press, Oxford, 2010.

[2] S. Fortunato, Community detection in graphs, Physics Reports 486 (3-5) (2010) 75–174.

[3] M. Newman, M. Girvan, Finding and evaluating community structure in networks, Physical Review E 69 (2004) 026133.

[4] Z. Li, S. Zhang, R.-S. Wang, X.-S. Zhang, L. Chen, Quantitative function for community detection, Physical Review E 77 (3) (2008) 036109.

[5] C. H. Q. Ding, X. He, H. Zha, M. Gu, H. D. Simon, A min-max cut algorithm for graph partitioning and data clustering, in: International Conference on Data Mining, IEEE Computer Society, Los Alamitos, CA, USA, 2001, pp. 107–114.

[6] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (8) (2000) 888–905.

[7] B. H. Good, Y.-A. de Montjoye, A. Clauset, The performance of modularity maximization in practical contexts, Physical Review E 81 (2010) 046106.

[8] S. Fortunato, M. Barthelemy, Resolution limit in community detection, Proceedings of the National Academy of Sciences, USA 104 (1) (2007) 36–41.

[9] S. Cafieri, P. Hansen, L. Liberti, Loops and multiple edges in modularity maximization of networks, Physical Review E 81 (4) (2010) 046102.

[10] C. Massen, J. Doye, Identifying communities within energy landscapes, Physical Review E 71 (2005) 046101.

[11] A. Lancichinetti, S. Fortunato, Limits of modularity maximization in community detection, Physical Review E 84 (2011) 066122.

[12] M. Sales-Pardo, R. Guimera, A. Moreira, L. Amaral, Extracting the hierarchical organization of complex systems, Proc. Natl. Acad. Sci. USA 104 (2007) 15224.

[13] J. Kumpula, J. Saramaki, K. Kaski, J. Kertesz, Limited resolution and multiresolution methods in complex network community detection, Fluctuation and Noise Letters 7 (3) (2007) L209–L214.

[14] A. Arenas, A. Fernandez, S. Gomez, Analysis of the structure of complex networks at different resolution levels, New Journal of Physics 10 (2008) 053039.

[15] J. Reichardt, S. Bornholdt, Statistical mechanics of community detection, Physical Review E 74 (2006) 016110.

[16] R. Guimerà, A. Amaral, Functional cartography of complex metabolic networks, Nature 433 (2005) 895–900.

[17] A. Medus, G. Acuna, C. Dorso, Detection of community structures in networks via global optimization, Physica A 358 (2005) 593–604.

[18] S. Lehmann, L. Hansen, Deterministic modularity optimization, European Physical Journal B 60 (2007) 83–88.

[19] M. Tasgin, A. Herdagdelen, H. Bingol, Community detection in complex networks using genetic algorithms, arXiv:0711.0491 (2007).

[20] J. Duch, A. Arenas, Community identification using extremal optimization, Physical Review E 72 (2) (2005) 027104.

[21] M. Newman, Modularity and community structure in networks, Proceedings of the National Academy of Sciences, USA 103 (23) (2006) 8577–8582.

[22] G. Agarwal, D. Kempe, Modularity-maximizing graph communities via mathematical programming, The European Physical Journal B 66 (3) (2008) 409–418.

[23] S. Boccaletti, M. Ivanchenko, V. Latora, A. Pluchino, A. Rapisarda, Detecting complex network modularity by dynamical clustering, Physical Review E 75 (2007) 045102.

[24] H. Djidjev, A scalable multilevel algorithm for graph clustering and community structure detection, Lecture Notes in Computer Science 4936 (2008) 117–128.

[25] J. Mei, S. He, G. Shi, Z. Wang, W. Li, Revealing network communities through modularity maximization by a contraction-dilation method, New Journal of Physics 11 (2009) 043025.

[26] Y. Niu, B. Hu, W. Zhang, M. Wang, Detecting the community structure in complex networks based on quantum mechanics, Physica A 387 (24) (2008) 6215–6224.

[27] V. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics (2008) P10008.

[28] D. Chen, Y. Fu, M. Shang, A fast and efficient heuristic algorithm for detecting community structures in complex networks, Physica A 388 (13) (2009) 2741–2749.

[29] J. Ruan, W. Zhang, Identifying network communities with a high resolution, Physical Review E 77 (2008) 016104.

[30] Y. Fan, M. Li, P. Zhang, J. Wu, Z. Di, Accuracy and precision of methods for community identification in weighted networks, Physica A 377 (1) (2007) 363–372.

[31] M. Newman, Fast algorithm for detecting community structure in networks, Physical Review E 69 (2004) 066133.

[32] A. Clauset, M. Newman, C. Moore, Finding community structure in very large networks, Physical Review E 70 (2004) 066111.

[33] L. Danon, A. Diaz-Guilera, A. Arenas, The effect of size heterogeneity on community identification in complex networks, Journal of Statistical Mechanics 2006 (11) (2006) P11010.

[34] K. Wakita, T. Tsurumi, Finding community structure in mega-scale social networks, Tech. Rep. 0702048v1, arXiv (2007).

[35] P. Schuetz, A. Caflisch, Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement, Physical Review E 77 (2008) 046112.

[36] S. Cafieri, P. Hansen, L. Liberti, Locally optimal heuristic for modularity maximization of networks, Physical Review E 83 (5) (2011) 056105.

[37] G. Xu, S. Tsoka, L. Papageorgiou, Finding community structures in complex networks using mixed integer optimization, European Physical Journal B 60 (2007) 231–239.

[38] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, D. Wagner, On modularity clustering, IEEE Transactions on Knowledge and Data Engineering 20 (2) (2008) 172–188.

[39] M. Grötschel, Y. Wakabayashi, A cutting plane algorithm for a clustering problem, Mathematical Programming 45 (1989) 59–96.

[40] M. Grötschel, Y. Wakabayashi, Facets of the clique partitioning polytope, Mathematical Programming 47 (1990) 367–387.

[41] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, L. Liberti, S. Perron, Column generation algorithms for exact modularity maximization in networks, Physical Review E 82 (4) (2010) 046112.

[42] A. Noack, R. Rotta, Multi-level algorithms for modularity clustering, Lecture Notes in Computer Science 5526 (2009) 257–268.

[43] IBM, ILOG CPLEX 12.2 User's Manual, IBM (2010).

[44] D. Lusseau, K. Schneider, O. Boisseau, P. Haase, E. Slooten, S. 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) (2003) 396–405.

[45] D. Knuth, The Stanford GraphBase: A Platform for Combinatorial Computing, Addison-Wesley, Reading, MA, 1993.

[46] L. Dartnell, E. Simeonidis, M. Hubank, S. Tsoka, I. Bogle, L. Papageorgiou, Self-similar community structure in a network of human interactions, FEBS Letters 579 (2005) 3037–3042.

[47] V. Krebs, `http://www.orgnet.com/` (unpublished).

[48] M. Girvan, M. Newman, Community structure in social and biological networks, Proceedings of the National Academy of Sciences, USA 99 (12) (2002) 7821–7826.

[49] `http://vlado.fmf.uni-lj.si/pub/networks/data/`.

[50] U. A. Lab, `http://www.weizmann.ac.il/mcb/UriAlon/`.

[51] R. Guimerà, L. Danon, A. Diaz-Guilera, F. Giralt, A. Arenas, Self-similar community structure in a network of human interactions, Physical Review E 68 (2003) 065103.

[52] D. Watts, S. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 409–410.

Algorithm 1

/* ncl = number of communities of the partition found by a heuristic */

/* CL_i = community of the partition found by a heuristic, $\forall i = \{1, \ldots, ncl\}$ */

Require: $V, E, ncl, CL_i \, \forall i = \{1, \ldots, ncl\}$

$ncl_{split} \leftarrow 0$

for all $i \leq ncl$ **do**

  split CL_i into CL_1, CL_2 using algorithm in subsection 2.1

  if $Q(CL_1) + Q(CL_2) > Q(CL_i)$ **then**

    replace CL_i with CL_1, CL_2

  else

    keep CL_i

  end if

  $ncl_{split} \leftarrow$ number of communities of the new partition

end for

$ncl_{merge+split} \leftarrow ncl_{split}$

for all $i \leq ncl_{split}$ **do**

  $listcl \leftarrow$ list of pairs of communities $(CL_j, CL_k),\ j, k \in \{1, \ldots, ncl_{split}\}$

  while $listcl \neq \emptyset$ **do**

    select a pair of communities CL_j, CL_k from $listcl$

    merge CL_j and CL_k into CL_m

    if $Q(CL_m) > Q(CL_j) + Q(CL_k)$ **then**

      replace CL_j, CL_k with $CL_m = CL_j \cup CL_k$

    else

      split CL_m into CL_{m1}, CL_{m2}

      if $Q(CL_{m1}) + Q(CL_{m2}) > Q(CL_m)$ **then**

        replace CL_m with CL_{m1}, CL_{m2}

      else

        keep CL_m

      end if

    end if

    update $listcl$

  end while

end for

$ncl_{merge+split} \leftarrow$ number of communities of the new partition

compute modularity $Q = \sum_{i=1}^{i=ncl_{merge+split}} Q(CL_i)$

return final partition, Q