Divisive heuristic for modularity density maximization

Alberto Costa[a], Sergey Kushnarev[b], Leo Liberti[c], Zeyu Sun[b]

[a] *National University of Singapore and ETH Zurich, Future Resilient Systems Program, Singapore*
[b] *Singapore University of Technology and Design, Singapore*
[c] *CNRS LIX, École Polytechnique, Palaiseau, France*

Abstract

In this paper we consider a particular method of clustering for graphs, namely the modularity density maximization. We propose a hierarchical divisive heuristic which works by splitting recursively a cluster into two new clusters by maximizing the modularity density, and we derive four reformulations for the mathematical programming model used to obtain the optimal splitting. We report computational results of the eight algorithms (four reformulations with two different symmetry breaking strategies) obtained on some instances from the literature. Statistical tests show that the best model in terms of computational time is the one that is obtained with a dual reformulation of the bilinear terms arising in the objective function. Moreover, the hierarchical divisive heuristic provides generally near-optimal solutions in terms of modularity density.

Keywords: Clustering, Modularity density maximization, Multilinear terms, Reformulation, Heuristic

---

1. Introduction

Given a graph, cluster analysis aims to find subsets of vertices, called clusters, where inner edges (i.e., edges connecting vertices in the same cluster) are dense and cut edges (i.e., edges connecting vertices in different clusters)

Email addresses: `costa@lix.polytechnique.fr` (Alberto Costa), `sergey_kushnarev@sutd.edu.sg` (Sergey Kushnarev), `liberti@lix.polytechnique.fr` (Leo Liberti), `zeyu_sun@mymail.sutd.edu.sg` (Zeyu Sun)

are sparse [1, 2]. This problem has many applications, e.g., recommender systems [3], social networks [4], biology and bioinformatics [5, 6].

Many methods have been proposed to tackle this problem, and they can be divided into three main classes:

- Heuristics without a function to optimize, for example the Girvan and Newman's heuristic [4] where the edge with highest "betweenness" (i.e., the number of shortest paths between pairs of nodes that run along that edge) is iteratively removed.

- Certain rules which must be respected by each cluster. Examples are the *strong* and *weak* definitions of Radicchi *et al.* [7] which impose, for the vertices in a cluster, some conditions on the number of neighbors inside and outside that cluster. Other examples are the *semi-strong* and *extra-weak* definitions by Hu *et al.* [8], and the *almost-strong* definition [9].

- A certain function that needs to be optimized. The most famous is the maximization of the modularity, which represents the fraction of edges within clusters minus the expected fraction of such edges in a random graph with the same degree distribution [4, 10]. Many heuristics [5, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20] and some exact methods [21, 22, 23] have been proposed to solve the modularity maximization problem, which has been proved to be **NP**-hard by Brandes *et al.* [22]. A study on the impact of the definitions presented above when applied to the modularity maximization problem is presented by Cafieri *et al.* [24].

Of particular interest from a mathematical programming point of view is the method of modularity maximization. Given an unweighted graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, its modularity Q is defined as [2, 4]:

$$Q = \frac{1}{2m} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(a_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j), \tag{1}$$

where m is the total number of edges of G (i.e., $m = |E|$), n is the number of vertices of G (i.e., $n = |V|$), a_{ij} is an element of the adjacency matrix of G (i.e., $a_{ij} = 1$ if $\{v_i, v_j\} \in E$, $a_{ij} = 0$ otherwise), k_i is the degree (i.e., number of neighbors) of vertex v_i, c_i and c_j are the clusters to which the vertices

v_i and v_j belong, and $\delta(c_i, c_j)$ is the Kronecker symbol, which is equal to 1 if $c_i = c_j$, and it is equal to 0 otherwise. Another equivalent definition of modularity is the following [2, 10]:

$$Q = \sum_{c \in C} Q_c = \sum_{c \in C} \left(\frac{m_c}{m} - \frac{{K_c}^2}{4m^2} \right), \tag{2}$$

where C is the set of clusters, Q_c is the contribution to modularity of cluster c, m_c is the number of edges within cluster c, K_c is the sum of the degrees of the vertices which are inside the cluster c, $\frac{m_c}{m}$ is the fraction of edges in cluster c, and $\frac{K_c^2}{4m^2}$ is the expected number of edges in cluster c in a graph where vertices have the same distribution of degrees of G but edges are placed randomly.

Using Equation (1) the modularity maximization problem can be modeled by means of the clique partitioning formulation [22, 25]. Recently, improved versions of the model derived by Grötschel and Wakabayashi [25], having a smaller set of inequalities, have been proposed [26, 27]. On the other hand, using Equation (2), the modularity maximization problem can be formulated as a convex Mixed Integer Quadratic Programming problem [21]. Note that in this case the cardinality of C, that is not known a priori, must be determined. This information is not needed if using Equation (1).

It has been pointed out that modularity maximization presents some issues. One of them is the resolution limit, i.e., the difficulty to find small clusters which may remain hidden within larger clusters [28, 29]. To overcome this problem, some authors presented a new function to be maximized called modularity density [30]. The modularity density D of a graph G, whose original formulation is reported in Li *et al.* [30], can be defined as:

$$D = \sum_{c \in C} D_c = \sum_{c \in C} \left(\frac{2m_c - \bar{m}_c}{n_c} \right), \tag{3}$$

where for each cluster $c \in C$ containing n_c vertices, m_c is the number of inner edges, $\bar{m}_c$ is the number of cut edges, and D_c is its modularity density. Notice that the structure of this formulation is similar to Equation (2), and again the optimal number of clusters $|C|$ is not known a priori. The corresponding optimization problem is nonlinear because of the denominator in Equation (3). Some exact linearizations have been presented by Costa [31], but they require the computation of an upper bound (possibly tight) for the modularity density of a cluster, that is not an easy task in general.

It it still unclear whether modularity density maximization is **NP**-hard or not. The argument given by Li *et al.* in [30] is invalid (see Costa [32]). Indeed, the modularity density maximization problem is not easy to solve, as it is integer and nonlinear. Even though it was possible to obtain some exact Mixed Integer Linear Programming (MILP) reformulations (see Costa [31]), experiments performed therein showed that only small size instances can be solved (see the results reported in Table 5). The main motivation for the heuristic presented in this paper is to provide a method which can find good quality solutions and which is faster than the exact method. More precisely, we propose a hierarchical divisive heuristic which works by splitting recursively a cluster into two clusters by maximizing the modularity density. More details about this method are presented in Section 2. After that, we present in Section 3 some formulations for the problem of the optimal splitting of a cluster. Some strategies to break symmetries of the problem are introduced in Section 4. We then compare performances of different formulations, and also the quality of the results with respect to global optimal solutions of [31] in Section 5. Finally, we draw conclusions in Section 6.

2. Divisive hierarchical heuristic

The main idea of the heuristic, which was originally proposed for modularity maximization by Cafieri *et al.* [19, 20] and then extended to bipartite modularity maximization by Costa and Hansen [33], is to start from an initial cluster containing all the vertices and then recursively split it into two clusters by maximizing the modularity density. The splitting of a cluster is stopped when either the size of the cluster is too small or the current division produces two clusters whose total modularity density is lower than that of the original cluster. By "small sized cluster" we mean a cluster containing less than four vertices: it has been proven in [31] that a cluster with one vertex (if not isolated) does not belong to the optimal partition, as it can be merged with another cluster to increase the total modularity density. If a cluster has less than four vertices, the splitting would produce at least one subcluster with one vertex, therefore we do not consider such cases. A pseudo-code for the heuristic is shown in Algorithm 1.

The most important part of the algorithm is the procedure MDS (*Modularity Density Splitting*), which takes as an input the cluster c_i and gives as an output clusters c_{2i} and c_{2i+1}, which are obtained by maximizing the

Algorithm 1: Hierarchical divisive heuristic

Input: graph $G = (V, E)$
Output: a partition C of V

```
1   $c_1 \leftarrow V$
2   $C \leftarrow \{c_1\}$
3   $i \leftarrow 1$
4   while $i > 0$ do
5       /* Splitting $c_i$ into two clusters maximizing modularity density */
6       $(c_{2i}, c_{2i+1}) \leftarrow$ MDS$(c_i)$
7       /* Modularity density $D$ is computed according to Equation (3) */
8       if $D(c_{2i}) + D(c_{2i+1}) \geq D(c_i)$ then
9           $C \leftarrow (C \backslash \{c_i\}) \cup \{c_{2i}\} \cup \{c_{2i+1}\}$
10      end if
11      /* Record indices of the clusters that need to be visited */
12      $I \leftarrow \{j : (c_j \in C) \wedge (|c_j| > 3) \wedge (c_j \text{ not visited})\}$
13      if $I = \emptyset$ then
14          $i \leftarrow 0$
15      else
16          $i \leftarrow \min_{j \in I} j$
17      end if
18  end while
19  return $C$
```

modularity density. More details on the MDS procedure are provided in the next section.

3. Formulations for the optimal splitting problem

The MDS procedure computes the optimal splitting of a cluster into two clusters by maximizing the modularity density. To do that, an optimization problem need to be solved.

Let $c = (V_c, E_c)$ be the cluster to be split into new clusters c_a and c_b, Y_i the binary variable that equals to 1 if the vertex v_i belongs to c_a, and 0 if the vertex v_i belongs to c_b, and k_i is the degree of the vertex v_i. According to Li *et al.* [30] and Costa [31], the problem can be expressed as follows:

$$\max \left(\frac{4\sum\limits_{\{v_i,v_j\}\in E_c} Y_iY_j - \sum\limits_{v_i\in V_c} k_iY_i}{\sum\limits_{v_i\in V_c} Y_i} + \frac{4\sum\limits_{\{v_i,v_j\}\in E_c}(1-Y_i)(1-Y_j) - \sum\limits_{v_i\in V_c} k_i(1-Y_i)}{|V_c| - \sum\limits_{v_i\in V_c} Y_i} \right) \tag{4}$$

$$\text{s.t.}\quad 2 \leq \sum_{v_i\in V_c} Y_i \leq |V_c| - 2 \tag{5}$$

$$\forall v_i \in V_c \quad Y_i \in \{0,1\}. \tag{6}$$

Solving this problem can be computationally expensive if using nonlinear solvers such as COUENNE [34] or BARON [35]: some experiments performed by Costa [31] showed that using the exact nonlinear model on the *strike* instance (the smallest considered in this paper), the solver COUENNE was still running after 2000 seconds on a server more powerful that the laptop used for our experiments. We can linearize exactly the objective function as shown by Costa [31] and then use the solver CPLEX [36], but this requires the computation of an upper bound for the modularity density of each cluster, but this can be computationally expensive because the upper bound is found by solving an additional Nonlinear Program (NLP). This bound is required to define the McCormick's inequalities [37], used to linearize the fractions arising in the nonlinear formulation. The reader interested in finding more details about the exact MILP reformulations of the modularity density maximization problem can refer to [31]. Here we employ another strategy, which turns out to be more efficient in terms of computational time. In order to find the optimal solution of Problem (4)–(6), we fix the denominator of the first term of Equation (4) to $p \in \{2, \ldots, |V_c| - 2\}$. Then, we can solve a simpler problem for each value of p, and take the best solution found. Given the value of p, after some manipulations, the *Bipartition Modularity Density* (BMD) model can be expressed as follows:

$$\max \frac{4|V_c| \sum_{\{v_i,v_j\}\in E_c} Y_iY_j + (2p-|V_c|)\sum_{v_i\in V_c} k_iY_i - 4p \sum_{\{v_i,v_j\}\in E_c} (Y_i+Y_j) - p\sum_{v_i\in V_c} k_i + 4p|E_c|}{p\left(|V_c| - p\right)}$$

$$\text{s.t.} \sum_{v_i\in V_c} Y_i = p$$

$$\forall v_i \in V_c \quad Y_i \in \{0,1\}. \tag{BMD}$$

Notice that this problem, unlike (4)–(6), can be solved directly with CPLEX. We have now all the ingredients to define the MDS procedure. It is shown in Algorithm 2.

Algorithm 2: Modularity Density Splitting (MDS)

Input: cluster $c = (V_c, E_c)$
Output: two subclusters c_1, c_2 obtained from c by maximizing the modularity density

```
1   $p \leftarrow 2$
2   $D^* \leftarrow -\infty$
3   while $p \leq |V_c| - 2$ do
4       $(c_a, c_b) \leftarrow$ solve BMD for the given $p$
5       /* Modularity density $D$ is computed according to Equation (3) */
6       if $D(c_a) + D(c_b) \geq D^*$ then
7           $D^* \leftarrow D(c_a) + D(c_b)$
8           $(c_1, c_2) \leftarrow (c_a, c_b)$
9       end if
10      $p \leftarrow p + 1$
11  end while
12  return $(c_1, c_2)$
```

The code of line 4 in Algorithm 2 takes as an input the value of p, solves BMD, and returns the optimal splitting. One might wonder if it is possible to derive a more efficient formulation than BMD. This question will be addressed in following subsections, where we derive some reformulations of BMD.

3.1. Fortet reformulation

One of the ways to obtain an *exact* reformulation (i.e., using the terminology of Liberti [38], a formulation which preserves all the optimality information of the original model) of BMD is to employ the Fortet inequalities [39] to linearize the products between the binary variables Y_i and Y_j arising in the objective function of BMD (notice that Fortet inequalities are a special case of the McCormick's inequalities [37] for binary variables). This technique is often employed in clustering problems [20, 31, 33, 40]. More precisely, each product Y_iY_j can be replaced by a new variable W_{ij} and the following linear inequalities:

$$W_{ij} \le Y_i \tag{7}$$
$$W_{ij} \le Y_j \tag{8}$$
$$W_{ij} \ge 0 \tag{9}$$
$$W_{ij} \ge Y_i + Y_j - 1. \tag{10}$$

Since we are maximizing Y_iY_j in the objective function, we can discard (9) and (10), thus obtaining the following model, BMD-F:

$$\begin{aligned}
\max \quad & \frac{4|V_c| \sum\limits_{\{v_i,v_j\}\in E_c} W_{ij} + (2p - |V_c|) \sum\limits_{v_i\in V_c} k_i Y_i - 4p \sum\limits_{\{v_i,v_j\}\in E_c} (Y_i + Y_j) - p \sum\limits_{v_i\in V_c} k_i + 4\,p\,|E_c|}{p\,(|V_c| - p)} \\
\text{s.t.} \quad & \sum_{v_i\in V_c} Y_i = p, \\
& \forall \{v_i, v_j\} \in E_c \quad W_{ij} \le Y_i, \\
& \forall \{v_i, v_j\} \in E_c \quad W_{ij} \le Y_j, \\
& \forall \{v_i, v_j\} \in E_c \quad W_{ij} \in \mathbb{R}, \\
& \forall v_i \in V_c \quad Y_i \in \{0, 1\}.
\end{aligned} \tag{BMD-F}$$

3.2. Dual reformulation

It was shown by Rikun [41] that multilinear functions (i.e., functions $\mu(x) = \mu(x_1, \ldots, x_k)$ of the form $x_1 \cdots x_k$) defined on hyperrectangles $[x^L, x^U] \subseteq \mathbb{R}^k$ are vertex polyhedral, i.e., their convex envelopes are the convex envelopes of their restriction to the vertices of the corresponding box [42]. In this section we exploit the dual representation of polyhedra in terms of their vertices to provide another representation (different from Fortet's, see previous section) of the convex envelope of bilinear products occurring in Formulation BMD above. It turns out that the computational behaviour of Branch-and-Bound (BB) algorithm improves when using dual reformulations (see Section 5). This is consistent with the findings by Costa and Liberti [43].

Let $w(x)$ be the value of the convex envelope of $\mu(x)$ and denote by $p_1, \ldots, p_\nu \in \mathbb{R}^n$ the vertices of the box $[x^L, x^U]$. Since the box $[x^L, x^U]$ is a polyhedron, it is convex. Hence we can write any $x \in [x^L, x^U]$ as a convex combination of $p_1, \ldots, p_\nu$ as follows:

$$x = \sum_{\ell=1}^{\nu} \lambda_\ell p_\ell, \tag{11}$$

$$1 = \sum_{\ell=1}^{\nu} \lambda_\ell, \tag{12}$$

for some nonnegative vector $\lambda = (\lambda_\ell \mid \ell \le \nu) \ge 0$. Since $\mu(x) = x_1 \cdots x_k$ we have:

$$\mu(x) = \prod_{h=1}^{k} x_h = \prod_{h=1}^{k} \sum_{\ell=1}^{\nu} \lambda_\ell p_{\ell h}, \tag{13}$$

and by vertex polyhedrality we can exchange sum and product, yielding:

$$w(x) = \sum_{\ell=1}^{\nu} \lambda_\ell \prod_{h=1}^{k} p_{\ell h}. \tag{14}$$

If a polynomial program consists of many multilinear monomials, the dual relaxation can be carried out term by term, yielding an alternative relaxation of the whole program. This relaxation was discussed by Costa and Liberti [43] in an abstract setting. Although its tightness is exactly the same as the generalized Fortet relaxation (so the bounds it yields are the same), the computational performance of BB is improved, and it was empirically shown to be more stable across the whole Branch-and-Bound (BB) tree.

3.2.1. The dual reformulation of BMD

Formulation BMD is a Mixed-Integer Nonlinear Program (MINLP) with bilinear terms $W_{ij} = Y_i Y_j$, where $Y_i, Y_j \in \{0, 1\}$, which means that the continuous relaxation has $Y_i, Y_j \in [0,1]$. Since we have a bilinear term defined over a box, the convex envelope is vertex polyhedral, so we have $\nu = 4$,

$$p_1 = (0,0), \quad p_2 = (0,1), \quad p_3 = (1,0), \quad p_4 = (1,1),$$

and

$$\begin{aligned} Y_i &= \sum_{\ell=1}^{4} \lambda_{ij\ell} p_{\ell 1}, && (15)\\ Y_j &= \sum_{\ell=1}^{4} \lambda_{ij\ell} p_{\ell 2}, && (16)\\ W_{ij} &= \sum_{\ell=1}^{4} \lambda_{ij\ell} p_{\ell 1} p_{\ell 2}, && (17)\\ 1 &= \sum_{\ell=1}^{4} \lambda_{ij\ell}, && (18)\\ \forall \ell \in \{1,\ldots,4\} \quad \lambda_{ij\ell} &\geq 0. && (19) \end{aligned}$$

We observe further that since $p_1 = (0,0)$ we do not need λ_{ij1} for any i, j: in particular, this replaces the quantifier $\forall \ell \in \{1,\ldots,4\}$ by $\forall \ell \in \{2,\ldots,4\}$ throughout and Constraint (18) can be replaced with $\sum_{\ell=2}^{4} \lambda_{ij\ell} \leq 1$. Moreover, since $\lambda_{ij\ell} p_{\ell 1} p_{\ell 2}$ is only nonzero whenever $p_{\ell 1} p_{\ell 2} = 1$, which only happens whenever $\ell = 4$, we can replace Constraint (17) with $W_{ij} = \lambda_{ij4}$. In summary, we obtain the following reformulation:

$$\begin{aligned} \max \quad & \frac{4|V_c| \sum_{\{v_i,v_j\} \in E_c} \lambda_{ij4} + (2p - |V_c|) \sum_{v_i \in V_c} k_i Y_i - 4p \sum_{\{v_i,v_j\} \in E_c} (Y_i + Y_j) - p \sum_{v_i \in V_c} k_i + 4p|E_c|}{p\,(|V_c| - p)} \\ \text{s.t.} \quad & \sum_{v_i \in V_c} Y_i = p \\ & \forall \{v_i, v_j\} \in E_c \quad \lambda_{ij3} + \lambda_{ij4} = Y_i \\ & \forall \{v_i, v_j\} \in E_c \quad \lambda_{ij2} + \lambda_{ij4} = Y_j \\ & \forall \{v_i, v_j\} \in E_c \quad \lambda_{ij2} + \lambda_{ij3} + \lambda_{ij4} \leq 1 \\ & \forall \{v_i, v_j\} \in E_c, \forall \ell \in \{2,3,4\} \quad \lambda_{ij\ell} \geq 0 \\ & \forall \{v_i, v_j\} \in E_c, \forall \ell \in \{2,3,4\} \quad \lambda_{ij\ell} \in \mathbb{R} \\ & \forall v_i \in V_c \quad Y_i \in \{0,1\}. \end{aligned} \qquad \text{(BMD-}\lambda\text{)}$$

3.3. Square reformulation

Another way to reformulate the problem is to express the products Y_iY_j in terms of the sum $Y_i + Y_j$. When the variables are binary, the following relationship holds:

$$(Y_i + Y_j)^2 = Y_i^2 + Y_j^2 + 2Y_jY_j = (Y_i + Y_j) + 2Y_jY_j, \tag{20}$$

where we use the fact that $Y^2 = Y$ if Y is binary. From Equation (20) it turns out that we can replace each term Y_iY_j in the objective function of BMD with the expression $\frac{(Y_i+Y_j)^2-(Y_i+Y_j)}{2}$. After a few manipulations, we obtain the following formulation, BMD-S:

$$\begin{aligned}
\max \quad & \frac{2|V_c| \sum_{\{v_i,v_j\}\in E_c} (Y_i + Y_j)^2 + (2p - |V_c|) \sum_{v_i \in V_c} k_i Y_i - 2(|V_c| + 2p) \sum_{\{v_i,v_j\}\in E_c} (Y_i + Y_j)}{p\,(|V_c| - p)} + \\
& + \frac{-p \sum_{v_i \in V_c} k_i + 4p|E_c|}{p\,(|V_c| - p)} \\
\text{s.t.} \quad & \sum_{v_i \in V_c} Y_i = p \\
& \forall v_i \in V_c \quad Y_i \in \{0, 1\}.
\end{aligned} \tag{BMD-S}$$

4. Symmetry breaking strategies

There exist symmetric solutions for the modularity density splitting problem, for example one can swap the vertices of the two subclusters obtained after the splitting to get an equivalent solution. It has been shown by Costa *et al.* [44] that symmetric solutions can impair the performances of Branch-and-Bound algorithms. Hence, we propose two possible strategies to reduce the number of symmetric solutions.

The first way to do it, which was already exploited in Cafieri *et al.* [20], Costa and Hansen [33], is to fix the vertex with highest degree to belong to one cluster. In case there exist multiple vertices with the same highest degree, we choose the one with the smallest index:

$$Y_g = 0, \quad g = \min\Big\{ j \,|\, k_j = \max\{k_i,\, \forall v_i \in V_c\} \Big\}. \tag{21}$$

Another way to reduce the symmetries of the problem is to consider only half of the values of p in the procedure MDS in Algorithm 2. In fact, the solution obtained with $p = \hat{p}$ is the same obtained with $p = |V_c| - \hat{p}$. Therefore, we just need to consider the values of p in $\{2, \ldots, \left\lfloor \frac{|V_c|}{2} \right\rfloor\}$ in the procedure MDS (and the condition in line 3 of Algorithm 2 becomes $p \leq \left\lfloor \frac{|V_c|}{2} \right\rfloor$). This way, we have to solve approximately half of the subproblems.

Notice that in general we cannot combine this strategy with Constraint (21). Therefore, we study their impact separately.

5. Computational results

In this section we present the results obtained with some instances from the literature, whose details are reported in Table 1. The instances are available on the Pajek repository [45] (we consider the largest connected component if the graph is composed of more than one. Moreover, if present, we replace directed arcs with undirected edges and we remove duplicated edges). We also report the results, in terms of computational time, obtained with some artificially generated instances. Experiments were performed on a 2.8 GHz core i7 CPU with 8GB of RAM running Linux and the solver CPLEX 12.6 [36].

5.1. Quantitative results: comparison between the different formulations

We compare here the four formulations (i.e., BMD, BMD-F, BMD-λ, and BMD-S) in terms of the total number of Branch-and-Bound nodes and total computational time needed to find the optimal partition C of Algorithm 1. Table 2 reports the results when employing the first symmetry breaking strategy described in Section 4 (i.e., fixing the vertex with highest degree in one of the clusters), whereas Table 3 presents the results obtained with the second strategy (i.e., considering half of the values of p). The best results in terms of computational time are highlighted in bold.

Visual representation of the data from Tables 2, 3 can be found in Figure 1. We have performed regression to the origin, in other words the intercept

Table 1: Information about the instances.

ID	Graph	$\lvert V\rvert$	$\lvert E\rvert$
1	Strike	24	38
2	Galesburg F	31	63
3	Galesburg D	31	67
4	Karate	34	78
5	Korea 1	35	69
6	Korea 2	35	84
7	Mexico	35	117
8	Sawmill	36	62
9	Dolphins small	40	70
10	Journal index	40	189
11	Graph	60	114
12	Dolphins	62	159
13	Les Misérables	77	254
14	A00 main	83	135
15	Protein p53	104	226
16	Political books	105	441

is assumed to be zero (since it is reasonable to expect the algorithm to solve a problem of size zero in zero seconds). The best regression fit in terms of R^2 values was obtained when the regressor is the product $|V|\,|E|$. Adjusted R^2 values are more than 98% and p-values associated with slopes are less than $5 \cdot 10^{-15}$ in both cases. We can observe from Figure 1 that the slope of the regression line corresponding to the BMD-λ formulation is the smallest one, indicating that it outperforms the other formulations in terms of computational time with respect to the size of the instances. Furthermore, in Table 2 the slope for the BMD-λ formulation is $2.2 \cdot 10^{-3}$, whereas in Table 3 it is $1.4 \cdot 10^{-3}$, thus indicating that the second symmetry breaking strategy is more effective.

To further support our conclusions, we performed a non-parametric statistical test. We employ the Quade test [46] (that is an extension of the Wilcoxon signed-rank test) instead of the Friedman test as it seems to be more powerful when the number of groups of data is low (we compare four

Table 2: Comparison between the four models with the symmetry breaking strategy of fixing the vertex with the highest degree. The column "time" refers to the *total* running time of the algorithm for solving the corresponding instance.

	BMD		BMD-F		BMD-λ		BMD-S	
ID	nodes	time $[s]$	nodes	time $[s]$	nodes	time $[s]$	nodes	time $[s]$
1	0	4.38	0	5.81	0	4.69	0	**4.31**
2	149	5.51	0	6.57	0	5.15	0	**4.66**
3	111	7.36	0	6.26	0	**5.1**	0	5.85
4	0	**3.69**	0	4.6	0	4.44	0	5.97
5	0	**3.47**	0	4.38	0	5.45	0	5.2
6	17	7.78	0	9.35	0	**4.86**	0	7.18
7	0	7.09	0	8.4	239	**5.31**	0	9.15
8	5	**4.84**	0	6.11	0	13.47	0	8.62
9	0	**6.57**	0	6.77	0	8.06	0	8.42
10	570	16.32	117	19.85	2424	**10.5**	0	18.49
11	0	20.78	0	25.54	0	**15.56**	0	26.54
12	200	31.63	0	30.97	637	**19.53**	0	34.04
13	586	51.25	181	46.99	638	**40.68**	86	50.72
14	149	27.38	0	**25.73**	0	26.89	0	26.13
15	83	79.81	0	78.05	87	**46.3**	0	81.14
16	2819	140.23	1763	140.2	9053	**106.2**	2750	142.31

models and two symmetry breaking strategies). We use the computational time on each instance as block (rows), and the formulations as groups (columns). The null hypothesis of the test is that there is no difference among the groups. This allows us to assess if one of the formulations is consistently better than the others in the majority of instances. The comparisons are performed at the 95% significance level: if we obtain a p-value < 0.05 then we can conclude that one formulation is more efficient than the others. We report in Table 4 the results of the Quade test.

The Quade test confirms that the best formulation is BMD-λ, and that the best symmetry breaking strategy is to consider half of the values of p (Table 3).

It is also interesting to notice that BMD-λ seems to be associated to the

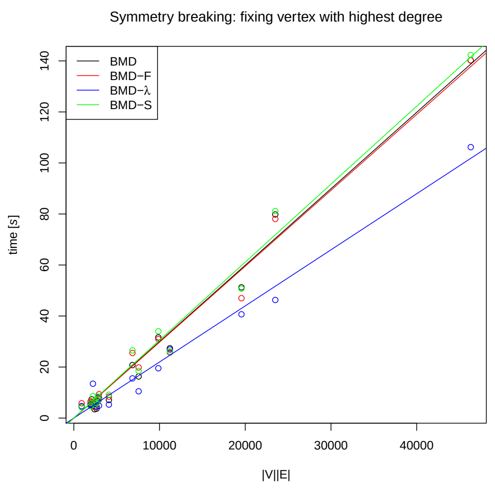
Table 3: Comparison between the four models with the symmetry breaking strategy of considering only half of the values of p. The column "time" refers to the *total* running time of the algorithm for solving the corresponding instance.

	BMD		BMD-F		BMD-λ		BMD-S	
ID	nodes	time $[s]$	nodes	time $[s]$	nodes	time $[s]$	nodes	time $[s]$
1	17	**2.26**	0	2.85	0	2.37	0	2.49
2	0	3.14	0	3.39	0	**2.65**	0	3.52
3	0	3.73	0	3.9	0	**2.99**	0	4.82
4	8	4.44	0	3.59	0	**2.48**	0	4.19
5	15	3.82	0	3.72	0	**2.81**	0	2.93
6	27	5.37	0	8.3	0	**2.8**	0	6.46
7	0	5.74	0	6.18	29	**3.35**	0	5.97
8	93	4	0	5.37	0	**1.89**	0	4.01
9	0	5	0	4.79	0	**2.58**	0	5.2
10	1710	10.02	0	11.77	1491	**6.92**	5	11.03
11	17	12.75	0	13.73	0	**7.37**	0	16.42
12	0	16.35	0	16.45	987	17.65	0	**15.44**
13	805	38.46	155	36.06	615	**30.04**	157	36.33
14	153	19.03	0	15.55	0	**14.59**	0	15.36
15	17	60.27	0	54.51	0	**30.49**	0	56.77
16	1200	93.49	1321	91.43	6294	**72.47**	1262	91.39

Table 4: Results of the Quade test.

Models tested	Best model	p-value
Table 2 (all)	BMD-λ	0.0025
Table 3 (all)	BMD-λ	$< 10^{-4}$
BMD-λ Tables 2, 3	BMD-λ Table 3	$< 10^{-5}$

larger total number of BB nodes. However, the computational times are lower, thus indicating that the subproblems solved by CPLEX, even though larger in number with respect to the other formulations, are easier to solve.



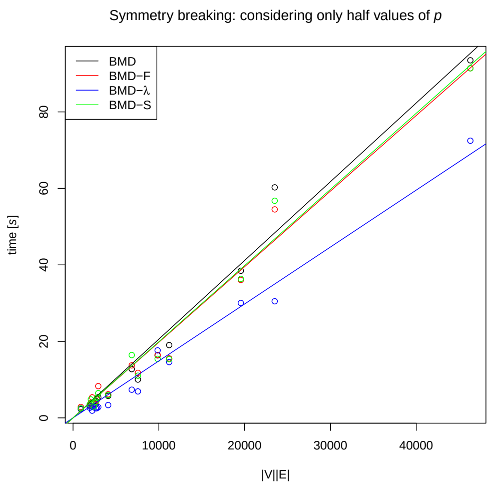


Figure 1: Data of Table 2 (left plot) and Table 3 (right plot). The x-axis is the size of the instance, i.e., the product $|V||E|$, and the y-axis is the computational time in seconds. Straight lines represent regression lines. The best formulation (i.e., the one associated with the regression line having the smallest slope) is BMD-λ in both cases.

5.2. Qualitative results: heuristic versus optimal solution

In this section we compare the partitions (number of clusters and modularity density value) obtained with the heuristic to the optimal solutions of Costa [31]. Those optimal solutions are known only for the first ten instances. The results are presented in Table 5, together with the gap between the heuristic modularity density (D_{heur}) and the optimal modularity density (D_{opt}). The gap is computed using the CPLEX definition:

$$\text{gap}_D = \left(\frac{100|D_{opt} - D_{heur}|}{|D_{heur}| + 10^{-10}}\right)\%.$$

Optimal solutions better than those of the divisive heuristic are highlighted in bold. In Table 5, $|C_{heur}|$ and $|C_{opt}|$ denote the number of clusters obtained with the divisive heuristic and the optimal one, respectively. It appears that the results of the heuristic are close, and often equal to the optimal ones. Note that the optimal solution times were obtained on a server more powerful than the PC used for the heuristic.

Table 5: Comparison of the results between the divisive heuristic proposed in this paper and the optimal solutions reported in [31].

	Heuristic			Optimal			gap$_D$
ID	$\|C_{heur}\|$	D_{heur}	time$_{heur}[s]$	$\|C_{opt}\|$	D_{opt}	time$_{opt}[s]$	%
1	4	8.86111	2.37	4	8.86111	**1.40**	0
2	3	8.28571	**2.65**	3	8.28571	3.36	0
3	3	6.92692	2.99	3	6.92692	**2.90**	0
4	3	7.84242	**2.48**	3	**7.8451**	4.51	0.03
5	5	10.9667	**2.81**	5	10.9667	8.75	0
6	5	11.143	**2.8**	5	11.143	15.55	0
7	2	8.55797	**3.35**	3	**8.71806**	58.02	1.87
8	5	8.52929	**1.89**	4	**8.62338**	10.11	1.10
9	7	12.8019	**2.58**	8	**13.0519**	121.31	1.95
10	4	17.8	**6.92**	4	17.8	67.92	0
11	7	9.57875	7.37	-	-	-	-
12	5	12.1252	17.65	-	-	-	-
13	9	24.5339	30.04	-	-	-	-
14	11	13.3731	14.59	-	-	-	-
15	8	12.9895	30.49	-	-	-	-
16	7	21.9652	72.47	-	-	-	-

5.3. Artificially generated instances

We report in this section the results obtained by the divisive heuristic using the best setting identified above (i.e., BMD-λ with the symmetry breaking strategy of considering half of the subproblems) on some artificial instances generated with the *Matlab Tools for Network Analysis* toolbox [47]. More precisely, for each combination of values of $|V|$ and $|E|$ reported in Table 6, we have generated 10 random instances and solved them using the divisive heuristic. Table 6 shows the mean ($\hat{\mu}$) and standard deviation ($\hat{\sigma}$) of the computational times for each case.

It is clear from the average times of Table 6 that the complexity of the problem increases significatively when the size of the instances grows. Moreover, the standard deviation of the computational time with respect to the average time increases as well, thus indicating that the particular instance

Table 6: Results obtained with some artificial instances. For each case we report the mean and standard deviation of the computational times obtained with the best algorithm (i.e., the formulation BMD-λ with the symmetry breaking strategy of considering half of the subproblems) on 10 random instances.

$\|V\|$	$\|E\|$	$\hat{\mu}$ time $[s]$	$\hat{\sigma}$ time $[s]$
50	75	6.68	1.22
50	100	9.02	2.71
50	150	28.94	14.22
100	150	37.87	13.01
100	200	119.00	22.16
150	225	179.79	39.47
200	300	1124.79	622.74
250	350	1984.58	957.56
300	400	4880.81	2349.77

considered can impact more on the complexity when the size increases.

6. Conclusions

In this paper we considered a problem of clustering for graphs via maximization of the modularity density. To simplify this nonlinear problem we proposed a hierarchical divisive heuristic, and used four reformulations with two symmetry breaking strategies. We performed the comparison of these eight algorithms on some instances of the literature.

Statistical tests showed that overall the best performing formulation (in terms of the computational time) is BMD-λ, wich runs up to twice faster than the other formulations in the worst case scenario. Between the two symmetry breaking strategies, i.e., fixing the vertex with the highest degree and solving only half of the subproblems, the latter is the most efficient. Therefore, the best performing algorithm is the dual reformulation BMD-λ with the symmetry breaking strategy of considering half of the subproblems. Moreover, the heuristic's solutions in terms of modularity density value coincide with the optimal ones in more than half of instances (for which the optimal solution is known, see Table 5), thus showing the quality of the proposed method.

Test performed on some larger artificially generated instances show that the heuristic can solve problems of size much bigger than that of the exact method presented in [31]. However, the computational complexity increases significantly. One of the future work directions is to address this point, in order to test the divisive heuristic on larger instances. To do so one could solve the problem within a column generation framework (as done for modularity maximization by Aloise *et al.* [23]). Moreover, a parallel implementation of the algorithm would improve the performance significantly. As example, this could be done by solving the subproblems of the procedure MDS (one for each value of p, see line 4 in Algorithm 2) in parallel.

Acknowledgements

The authors would like to thank the anonymous referees for their remarks, which helped to improve significantly the quality of the paper. A. C. has been partially supported by SUTD-MIT International Design Center under grant IDG21300102. His research was partially conducted at the Future Resilient Systems at the Singapore-ETH Centre (SEC). The SEC was established as a collaboration between ETH Zurich and National Research Foundation (NRF) Singapore (FI 370074011) under the auspices of the NRF's Campus for Research Excellence and Technological Enterprise (CREATE) programme.

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] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering 17 (6) (2005) 734–749.

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

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

[6] G. Palla, I. Derényi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (7043) (2005) 814–818.

[7] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, D. Parisi, Defining and identifying communities in networks, Proceedings of the National Academy of Sciences of the U.S.A. 101 (9) (2004) 2658–2663.

[8] Y. Hu, H. Chen, P. Zhang, M. Li, Z. Di, Y. Fan, Comparative definition of community and corresponding identifying algorithm, Physical Review E 78 (2) (2008) 026121.

[9] S. Cafieri, G. Caporossi, P. Hansen, S. Perron, A. Costa, Finding communities in networks in the strong and almost-strong sense, Physical Review E 85 (4) (2012) 046113.

[10] M. Newman, M. E. J. Girvan, Finding and evaluating community structure in networks, Physical Review E 69 (2) (2004) 026113.

[11] M. C. V. Nascimento, L. Pitsoulis, Community detection by modularity maximization using GRASP with path relinking, Computers & Operations Research 40 (12) (2013) 3121 – 3131.

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

[13] A. Medus, G. Acuna, C. Dorso, Detection of community structures in networks via global optimization, Physica A 358 (2-4) (2005) 593–604.

[14] S. Lehmann, L. Hansen, Deterministic modularity optimization, The European Physical Journal B 60 (1) (2007) 83–88.

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

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

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

[18] P. Schuetz, A. Caflisch, Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement, Physical Review E 77 (4) (2008) 046112.

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

[20] S. Cafieri, A. Costa, P. Hansen, Reformulation of a model for hierarchical divisive graph modularity maximization, Annals of Operations Research 222 (1) (2014) 213–226.

[21] G. Xu, S. Tsoka, L. G. Papageorgiou, Finding community structures in complex networks using mixed integer optimisation, The European Physical Journal B 60 (2) (2007) 231–239.

[22] 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.

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

[24] S. Cafieri, A. Costa, P. Hansen, Adding cohesion constraints to models for modularity maximization in networks, Journal of Complex Networks 3 (3) (2015) 388–410.

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

[26] T. N. Dinh, M. T. Thai, Finding community structure with performance guarantees in complex networks, Tech. Rep. 1108.4034, arXiv (2011).

[27] A. Miyauchi, N. Sukegawa, Redundant constraints in the standard formulation for the clique partitioning problem, Optimization Letters 9 (1) (2015) 199–207.

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

[29] B. H. Good, Y.-A. de Montjoye, A. Clauset, Performance of modularity maximization in practical contexts, Physical Review E 81 (4) (2010) 046106.

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

[31] A. Costa, MILP formulations for the modularity density maximization problem, European Journal of Operational Research 245 (1) (2015) 14 – 21.

[32] A. Costa, Some remarks on modularity density, Tech. Rep. 1409.4063, arXiv (2014).

[33] A. Costa, P. Hansen, A locally optimal hierarchical divisive heuristic for bipartite modularity maximization, Optimization Letters 8 (3) (2014) 903–917.

[34] P. Belotti, J. Lee, L. Liberti, F. Margot, A. Wächter, Branching and bounds tightening techniques for non-convex MINLP, Optimization Methods and Software 24 (4-5) (2009) 597–634.

[35] N. Sahinidis, M. Tawarmalani, BARON 7.2.5: Global Optimization of Mixed-Integer Nonlinear Programs, *User's Manual* (2005).

[36] IBM, ILOG CPLEX 12.6 User's Manual, IBM (2013).

[37] G. P. McCormick, Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems, Mathematical Programming 10 (1976) 146–175.

[38] L. Liberti, Reformulations in Mathematical Programming: Definitions and Systematics, RAIRO-OR 43 (1) (2009) 55–86.

[39] R. Fortet, Applications de l'algèbre de Boole en recherche opérationelle, Revue Française de Recherche Opérationelle 4 (14) (1960) 17–26.

[40] S. Benati, S. García, A mixed integer linear model for clustering with variable selection, Computers & Operations Research 43 (2014) 280 – 285.

[41] A. Rikun, A convex envelope formula for multilinear functions, Journal of Global Optimization 10 (4) (1997) 425–437.

[42] F. Tardella, Existence and sum decomposition of vertex polyhedral convex envelopes, Optimization Letters 2 (2008) 363–375.

[43] A. Costa, L. Liberti, Relaxations of multilinear convex envelopes: dual is better than primal, in: R. Klasing (Ed.), Experimental Algorithms, Vol. 7276 of LNCS, Springer, Heidelberg, 2012, pp. 87–98.

[44] A. Costa, P. Hansen, L. Liberti, On the impact of symmetry-breaking constraints on spatial branch-and-bound for circle packing in a square, Discrete Applied Mathematics 161 (1-2) (2013) 96 – 106.

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

[46] W. J. Conover, Practical nonparametric statistics, 3rd Edition, John Wiley & Sons, New York, 1999.

[47] G. Bounova, O. de Weck, Overview of metrics and their correlation patterns for multiple-metric topology analysis on heterogeneous graph ensembles, Physical Review E 85 (1) (2012) 016117.