Computers & Operations Research 71 (2016) 100–109

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

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

ARTICLE INFO

Available online 22 January 2016

Keywords:
Clustering
Modularity density maximization
Multilinear terms
Reformulation
Heuristic

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.

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) 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 [5,11–20] and some exact methods [21–23] 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]

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

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

[*] Corresponding author.
E-mail addresses: costa@lix.polytechnique.fr (A. Costa), sergey_kushnarev@sutd.edu.sg (S. Kushnarev), liberti@lix.polytechnique.fr (L. Liberti), zeyu_sun@mymail.sutd.edu.sg (Z. Sun).

http://dx.doi.org/10.1016/j.cor.2016.01.009
0305-0548/© 2016 Elsevier Ltd. All rights reserved.

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 Eq. (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 Eq. (2), the modularity maximization problem can be formulated as a convex Mixed Integer Quadratic Programming problem [21]. Notice that in this case the cardinality of C, that is not known a priori, must be determined. This information is not needed if using Eq. (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

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

where for each cluster $c \in C$ containing n_c vertices, m_c is the number of inner edges, $\overline{m}_c$ is the number of cut edges, and D_c is its modularity density. Notice that the structure of this formulation is similar to Eq. (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 Eq. (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. [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 Costa [31] in Section 5. Finally, we draw conclusions in Section 6.

2. Divisive hierarchical heuristic

The main idea of the heuristic, which was first employed 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 by Costa [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.

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     $(c_{2i}, c_{2i+1}) \leftarrow \mathrm{MDS}(c_i)$
7     /* Modularity density $D$ is computed according to Eq. (3) */
8     if $D(c_{2i}) + D(c_{2i+1}) \geq D(c_i)$ then
9        $C \leftarrow (C \setminus \{c_i\}) \cup \{c_{2i}\} \cup \{c_{2i+1}\}$
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
```