Variable Neighborhood Search for edge-ratio network clustering

Sonia Cafieri[1], Pierre Hansen[2], Nenad Mladenović[3]

[1]ENAC, MAIAA, F-31055 Toulouse, France and
University of Toulouse, IMT, F-31400 Toulouse, France
[2]GERAD, HEC Montréal, Canada
[3]School of Mathematics, Brunel University, United Kingdom

Abstract

Edge-ratio clustering was introduced in [Cafieri et al., Phys.Rev. E 81(2):026105, 2010], as a criterion for optimal graph bipartitioning in hierarchical divisive algorithms for cluster identification in networks. Exact algorithms to perform bipartitioning maximizing the edge-ratio were shown to be too time consuming to be applied to large datasets. In this paper, we present a Variable Neighborood Search (VNS)-based heuristic for hierarchical divisive edge ratio network clustering. We give a full description including the structure of some algorithmic procedures which are used to implement the main steps of the heuristic. Computational results show that the proposed algorithm is very efficient in terms of quality of the bipartitions, moreover the computing time is much smaller than that one for exact algorithms.

1 Introduction

Network analysis is successfully used in the study of complex systems in a variety of domains, where a network representation and the study of its topological features help to better understand some characteristics of the system under consideration. Prominent examples include social networks, describing individuals and their interactions and relationships, telecommunication networks, such as the World Wide Web, transportation networks, biological networks, and many more. A detailed introduction to networks has recently been given by Newman [18]. A mathematical representation of a network is obtained using a graph $G = (V, E)$, where vertices (in the set V) are associated to the entities under study and the edges (in the set E) joining pairs of vertices correspond to relationships among the entities.

Network clustering represents a topic of particular interest in network analysis. It consists in detecting subsets of vertices which are more densely linked compared to the rest of the graph, called modules or *clusters* or *communities*. This is of much interest in the study of complex systems as many of such systems are characterized by a modular structure. For example, a community in a social network can be constituted by individuals sharing a common interest or location, in a biological network by entities with a common function. The reader is referred to Fortunato [4] for a recent survey of this research domain.

Network clustering is generally based on a definition of community or on a criterion to evaluate a partition found. A clustering criterion can be used as an objective function in a combinatorial optimization problem, whose solution gives an optimal partition for the considered network. Several alternatives have been proposed. Among clustering criteria, the most used is *modularity* [16], based on the idea of comparing the fraction of edges falling within communities to the expected fraction of such edges. Two of the most known definitions of community were proposed in [20] and give conditions defining communities in a strong and a weak sense, respectively. Building on the most significant condition, the *weak* condition, in [1] a new criterion was introduced for a bipartition to be optimal, called the *edge-ratio* criterion. This criterion, as well as the weak condition, are recalled below. The edge-ratio criterion is maximized to obtain successive bipartitions in a hierarchical divisive algorithm. In [1] the optimal bipartitioning problem was solved exactly at each iteration of the divisive algorithm. It turns out that this exact algorithm is very time consuming even for small or medium size datasets (i.e., with a few hundred of vertices). Therefore, heuristic approaches appear to be the most adequate to solve in reasonable time the bipartitioning problems. In this paper, we propose a Variable Neighborhood Search heuristic to solve the bipartitioning problems. Such heuristic was presented in [3] somewhat informally. In the present paper we give a full mathematical description of this heuristic, including the structure of some algorithmic procedures which are used to implement the main steps of the heuristic.

Let us consider a graph $G = (V, E)$ with V set of vertices, with cardinality n, and E set of edges, with cardinality m. Let $S \subseteq V$ be a subset of vertices. Then the degree k_i of a vertex i belonging to V (i.e., the number of its neighbors) can be separated into two components $k_i^{in}(S)$ and $k_i^{out}(S)$, which represent respectively the number of neighbors of i inside S and the number of neighbors of i outside S.
Radicchi et al. [20] give the following definition for a set of vertices S forming a community in the *weak sense*:

$$\sum_{i \in S} k_i^{in}(S) > \sum_{i \in S} k_i^{out}(S)$$

That is, S is a community in the weak sense if and only if the sum of all degrees within S is larger than the sum of all degrees joining S to the rest of the graph. As this condition can be used ([20]) as a local stopping criterion in hierarchical clustering, it has been used by Wang et al. [21] to define a community S *indivisible* if there is no bipartition, (S_1, S_2) of S, such that both S_1 and S_2 satisfy the weak condition.
In [1] the definition of community in the weak sense is extended into a criterion for a bipartition to be optimal: one seeks to maximize the minimum, for both classes S_1 and S_2 of the bipartition of S (such that $S_1 \cup S_2 = S$, $S_1 \cap S_2 = \emptyset$, $S_1, S_2 \neq \emptyset$), of the ratio of inner edges to cut edges:

$$f(S_1, S_2) = \max_{S_1, S_2 \subset V} \min\left(r(S_1), r(S_2)\right),$$

where for a subgraph S the ratio $r(S)$ is given by:

$$r(S) = \sum_{i \in S} k_i^{in}(S) / \sum_{i \in S} k_i^{out}(S).$$

This definition is strenghten to be used in a hierarchical divisive clustering algorithm, by introducing a parameter α and by quantifying how much the number of inner edges is larger

than the number of cut edges. Thus, the condition becomes:

$$\sum_{i \in S} k_i^{in}(S) \geq \alpha \sum_{i \in S} k_i^{out}(S). \tag{1}$$

In case of equality, the coefficient α is equal to the ratio of twice the number of edges within the community S divided by the number of edges within the cut of that community. It is called the *edge ratio* [1]. Then, hierarchical divisive clustering algorithm, where at each step a bipartitioning of the current subgraph is done, one can then seek the maximum value of α for which the subgraph to be bipartitioned will be divisible. So, the problem of detecting indivisible communities is also solved. The crucial step of bipartitioning in a hierarchical divisive clustering algorithm is also the most computationally demanding. The VNS based heuristic presented in this paper is aimed at performing efficiently this algorithmic step.

The paper is organized as follows. In Sect. 2 we recall the structure of a hierarchical divisive clustering algorithm and introduce VNS to perform its main computational step. Sect. 3 presents the details of the algorithmic structure of some subroutines which are used to implement the main steps of the heuristic. A few results about algorithmic complexity are also discussed. In Sect. 4 the results of computational experiments performed on a set of instances from the literature are presented and discussed. Sect. 5 concludes the paper.

2 Divisive hierarchical clustering using VNS

2.1 Basic notation

Let us consider an unweighted graph $G(V, E)$ with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set $E = \{e_1, \ldots, e_m\}$. Let k_i denotes the degree of vertex $v_i \in V$. Let $A = a_{ij}$ be the corresponding adjacency matrix ($a_{ij} \in \{0, 1\}$) and $H = h_{ij}$ be the adjacency list, i.e., raw i contains the list $h_{ij}, j = 1, \ldots, d_i$ of vertices that are adjacent to vertex v_i.

2.2 Divisive hierarchical clustering

Hierarchical divisive clustering heuristics (see, e.g. [2]) proceed, in a top-to-bottom approach, from an initial partition of the graph containing all its n vertices, iteratively bipartitioning a subgraph, until a partition into n clusters is obtained, or the value of a chosen objective function is not improved anymore. At each step of a divisive clustering algorithm, one must select a cluster among the ones obtained at the previous hierarchy levels and divide it into two. At the first step, the first bipartition is done on V. Let ℓ in $\{1, \ldots, n\}$ be the index of the subgraph S^ℓ selected to be bipartitioned during a step of the divisive algorithm. After a bipartition into subsets of vertices S_1 and S_2, we obtain $S^\ell = S_1$ and $S^{\ell+1} = S_2$. Then ℓ is increased by one and all process repeated. In the present study, the objective function is based on the edge-ratio criterion (1). So, for each bipartition, the objective function value $f(S_1, S_2)$ is computed. If it is greater than or equal to one, i.e., the subgraph that one seeks to bipartition is divisible, then the obtained subgraphs are added to the list P of subgraphs to be considered for bipartition.
The corresponding pseudo-code is given in Algorithm 1. Notice that bipartitioning is done

by Variable Neighbourhood Search, so an appropriate function `BVNS` is called to perform the bipartitioning steps. It is described in details in Sect. 3.

Function `DHC-VNS` (m, n, A, P)
$\ell = 1, S^\ell = \{v_1, \ldots, v_n\}, P = \emptyset$
while $|S^\ell| > 1$ **do**
  $S_1, S_2 \leftarrow$ `BVNS` (S^ℓ) // Find bipartition of S^ℓ according to $f(S_1, S_2)$;
  if $f \geq 1$ **then**
    $P = P \cup \arg\min f(S_1, S_2)$
  $S^\ell \leftarrow S_1$; $S^{\ell+1} \leftarrow S_2$
  $\ell \leftarrow \ell + 1$
  Among $\{S^1, \ldots, S^\ell\}$, select cluster with the largest cardinality
  Exchange its position with S^ℓ

Algorithm 1: Divisive hierarchical clustering algorithm with VNS for bipartitioning

3 Variable Neighborhood Search for edge-ratio clustering

Variable Neighbourhood Search (VNS) is a metaheuristic aimed at solving combinatorial and global optimization problems. It is based on the idea of a systematic change of neighbourhood combined with a local search to escape the current local optimum. The main ingredients of such heuristic are thus given by a local search procedure, shaking, i.e., a procedure to perturb the current solution, and a procedure to update the current solution. The reader is referred to [15, 7, 8, 9, 10] for an introduction of VNS and its main applications.

In this section, a detailed description of the proposed Variable Neighborhood Search is given, together with pseudo-codes for its main procedures.

3.1 Initialization

The VNS for bipartitioning starts with a random partition of the current vertex set $S = S_1 \cup S_2$, $S_1 \cap S_2 = \emptyset$, where the cardinality of the first set is $\frac{n}{2}$ ($|S_1| = [\frac{n}{2}]$). In that way, two subgraphs $G_1 = (S_1, E_1)$ and $G_2 = (S_2, E_2)$ are obtained. A third set of vertices $S_3 = V \setminus \{S_1 \cup S_2\}$ is also handled through the execution of the algorithm. Obviously, at the first iteration $S_3 = \emptyset$. Let $\mathcal{P}_\ell$ denotes all possible partitions of vertex set S_ℓ, where ℓ represents the hierarchy level of the divisive hierarchical clustering algorithm. In order to simplify notation, in the following we will omit the index ℓ.

3.2 Data structure and objective function

Let S be a set of vertices to be bipartitioned into S_1 and S_2. A bipartition is represented in the computer memory in the following way:

$x = \{x_1, \ldots, x_{|S|}\}$ – indices of vertices that belong to S

$$\delta_{x_j} = \begin{cases} 1 & \text{if} \quad x_j \in S_1 \\ 0 & \text{if} \quad x_j \in S_2 \end{cases}$$

and the following variables are used to update the objective function values:

n_1 – number of vertices in set S_1;

n_2 – number of vertices in set S_2;

n_3 – number of vertices in set S_3;

m_1 – number of edges in subgraph G_1;

m_2 – number of edges in subgraph G_2;

c_1 – number of cut edges with one end point in S_1 and another in $V \setminus S_1 = S_2 \cup S_3$;

c_2 – number of cut edges with one end point in S_2 and another in $V \setminus S_2 = S_1 \cup S_3$;

k_i^{in} – inner degree of vertex v_i (both end points belong to either S_1 or S_2);

k_i^{cut} – cut (outer) degree of vertex v_i (number of links of v_i with vertices from different sets);

$k_i^{rest} = k_i - k_i(S), v_i \in S$ – "rest" degree of vertex v_i, where $k_i(S)$ denotes the degree of the vertex in the subgraph of G induced from vertices of S;

$f_1 = 2 \cdot \frac{m_1}{c_1}$ – ratio for subgraph G_1;

$f_2 = 2 \cdot \frac{m_2}{c_2}$ – ratio for subgraph G_2;

$f = \min\{f_1, f_2\}$ – objective function value.

Note that all values from above can be computed if a partition (S_1, S_2, S_3) is known $(S_j \in \mathcal{P})$. Therefore, instead of n_1 we could in fact put $n_1(S_1)$, instead of m_2 we could put $m_2(S_2)$, etc. Also note that the degree of each vertex v can be presented as

$$k_v = k_v^{in} + k_v^{cut} + k_v^{rest}$$

with respect to the current partition of V in three subsets.

At each step of the divisive hierarchical clustering algorithm, the problem is to find a bipartition of the current S into S_1 and S_2 such that the largest value between the objective functions associated to the corresponding subgraphs, f_1 and f_2, is as small as possible. The objective function of the edge-ratio clustering at each iteration of the divisive algorithm can be presented as follows

$$f(S_1, S_2) = \max_{S_1, S_2 \in \mathcal{P}} \min\{f_1(S_1), f_2(S_2)\} \tag{2}$$

where $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \emptyset$.

Proposition 1 *Given a graph $G(V, E)$ and the bipartition of its vertex set V into sets S_1 and S_2, computing the objective function value (2) is in $O(|E|)$.*

Proof. In order to compute f, we need to find m_1, m_2, c_1 and c_2. For each vertex v_i we check if h_{ij} belongs to S_1 or S_2 (i.e., we check if $\delta(h_{ij}) = \texttt{true}$ or $\texttt{false}$). If both δ_i and $\delta(h_{i,j})$ have the same value, the corresponding inner degree (m_1 or m_2) is increased by one. Otherwise, cut degree (c_1 or c_2) is increased by one. Since the number of elements in H is $m = |E|$ we need $O(m)$ calculations. ◇

3.3 Allocation local search

Local search in the proposed VNS is based on the selection of the entity (vertex) whose associated local change (move) gives the best improvement of the objective function value, and on the update of all entities involved in the problem after that move.
Assume that any solution (S_1, S_2, S_3) is known. We use an *allocation move* as a local change. It consists in taking some entity v from $S = S_1 \cup S_2$ and changing its allocation: if $v \in S_1$ then, after the allocation move, it will belong to S_2 and vice-versa.
The vertex whose allocation change improves the most the objective function (2) is selected and all variables updated accordingly. The datailed algorithms are given below.

3.3.1 Move.

In Algorithm $\texttt{Move}(v, f, \delta)$ input values are the index of entity v that will change its membership, the current objective function value f and the current solution δ. We also assume that values of m_1, m_2, c_1 and c_2 are known as well.

Function $\texttt{Move}(v, f, \delta)$
if δ_v **then**
  $mm_1 \leftarrow m_1 - k_v^{in}$; $mm_2 \leftarrow m_2 + k_v^{cut}$
  $cc_1 \leftarrow c_1 + k_v^{in} - k_v^{cut} - k_v^{rest}$
  $cc_2 \leftarrow c_2 + k_v^{in} - k_v^{cut} + k_v^{rest}$
else
  $mm_1 \leftarrow m_1 + k_v^{cut}$; $mm_2 \leftarrow m_2 - k_v^{in}$
  $cc_1 \leftarrow c_1 + k_v^{in} - k_v^{cut} + k_v^{rest}$
  $cc_2 \leftarrow c_2 + k_v^{in} - k_v^{cut} - k_v^{rest}$
$f_1 \leftarrow \frac{2 \cdot mm_1}{cc_1}$; $f_2 \leftarrow \frac{2 \cdot mm_2}{cc_2}$; $f \leftarrow \min\{f_1, f_2\}$

Algorithm 2: Allocation move

Notice that mm_1, mm_2, cc_1 and cc_2 are used to temporally denote the values of m_1, m_2, c_1 and c_2; these values will be actually changed once the move to be performed is selected.

Proposition 2 *Let v be a vertex that moves from one subset to another. If $v \in S_1$ then the new objective function value is*

$$f^{new} = \min\{\frac{2 \cdot (m_1 - k_v^{in})}{c_1 + k_v^{in} - k_v^{cut} - k_v^{rest}}, \frac{2 \cdot (m_2 + k_v^{cut})}{c_2 + k_v^{in} - k_v^{cut} + k_v^{rest}}\}.$$

If $v \in S_2$ then

$$f^{new} = \min\{\frac{2 \cdot (m_1 + k_v^{cut})}{c_1 + k_v^{in} - k_v^{cut} + k_v^{rest}}, \frac{2 \cdot (m_2 - k_v^{in})}{c_2 + k_v^{in} - k_v^{cut} - k_v^{rest}}\}.$$

Proof. A new value of the objective function, by definition is

$$f^{new} = \min\{f_1^{new}, f_2^{new}\} = \min\{\frac{2m_1^{new}}{c_1^{new}}, \frac{2m_2^{new}}{c_2^{new}}\}. \tag{3}$$

If $v \in S_1$ changes its membership to S_2, then we have:

$$m_1^{new} = m_1 - k_v^{in} \tag{4}$$
$$m_2^{new} = m_2 + k_v^{cut} \tag{5}$$

because the new number of inner edges m_1 in S_1 is reduced by k_v^{in} (the inner degree of v) (see (4)), and the number of edges in S_2 is augmented by the cut degree of v (see (5)). The numbers of cut edges c_1 and c_2 of S_1 and S_2 are updated as follows:

$$c_1^{new} = c_1 + k_v^{in} - k_v^{cut} - k_v^{rest} \tag{6}$$
$$c_2^{new} = c_2 + k_v^{in} - k_v^{cut} + k_v^{rest} \tag{7}$$

Indeed, since v moves to S_2, the number of cut edges c_1^{new} of S_1 is increased by its inner degree in S_1 ($+k_v^{in}$); also, it is reduced by k_v^{cut}, since its cut degree contributes to inner degree of S_2; finally, all vertices from $S \setminus (S_1 \cup S_2)$ connected with v after the move to S_2 do not produce cut edges of S_1 anymore ($-k_v^{rest}$).
Observe that $c_1^{new} = c_2^{new}$ only if $S = V$, i.e., at the first hierarchy level $\ell = 1$, where subset $S_3 = \emptyset$ and therefore the *rest* degrees are equal to 0. Similarly for (7).

If vertex v belongs to S_2 and moves to S_1, then new numbers of inner and cut edges in S_1 and S_2 are

$$m_1^{new} = m_1 + k_v^{cut} \tag{8}$$
$$m_2^{new} = m_2 - k_v^{in} \tag{9}$$
$$c_1^{new} = c_1 + k_v^{in} - k_v^{cut} + k_v^{rest} \tag{10}$$
$$c_2^{new} = c_2 + k_v^{in} - k_v^{cut} - k_v^{rest} \tag{11}$$

Substituting (6) - (11) into equations (3) we get the result. ◇

3.3.2 Update.

Once the entity v improving the most the objective function value is selected, a move is done and the current solution as well as all the variables that we keep during the search are updated. In order to get a new value improving the objective function, both f_1^{new} and f_2^{new} have to be larger than f, since the minimum of these two values should be larger than f. Let us first assume that $v \in S_1$ is moved to S_2. Besides updating values with formulas (4) - (11), we

also need to update the number of vertices of S_1 (reduced by 1), and the cardinality of S_2 (increased by 1):

$$n_1^{new} = n_1 - 1,\ n_2^{new} = n_2 + 1. \tag{12}$$

In addition, inner and cut degrees of vertices that are adjacent to the moved vertex v have to be updated.

Proposition 3 *Let v be a vertex that changes its membership and let u be any adjacent vertex to v ($(v,u) \in E$). Then values of new inner and cut degrees of u are:*

(i) $\bar{k}_u^{in} = k_u^{in} - 1,\ \ \bar{k}_u^{cut} = k_u^{cut} + 1$, if both v and u belong to the same subset;

(ii) $\bar{k}_u^{in} = k_u^{in} + 1,\ \ \bar{k}_u^{cut} = k_u^{cut} - 1$, if v and u are in different subsets.

Proof. As mentioned earlier, $u_j = h_{vj}$, $j = 1, \ldots, k_v$ indicate all adjacent vertices of v. There are k_v of them since k_v is the degree of v. Let us denote with $\bar{k}_u^{in}$ and $\bar{k}_u^{cut}$ a new values of inner and cut degrees (after the vertex v is moved from one to another subset of vertices), respectively. Then four different cases can occur.

Case 1. $v \in S_1$ and $u_j \in S_1$. Then $\bar{k}_{u_j}^{in} = k_{u_j}^{in} - 1$ and $\bar{k}_{u_j}^{cut} = k_{u_j}^{cut} + 1$;

Case 2. $v \in S_1$ and $u_j \in S_2$. Then $\bar{k}_{u_j}^{in} = k_{u_j}^{in} + 1$ and $\bar{k}_{u_j}^{cut} = k_{u_j}^{cut} - 1$;

Case 3. $v \in S_2$ and $u_j \in S_1$. Then $\bar{k}_{u_j}^{in} = k_{u_j}^{in} + 1$ and $\bar{k}_{u_j}^{cut} = k_{u_j}^{cut} - 1$;

Case 4. $v \in S_2$ and $u_j \in S_2$. Then $\bar{k}_{u_j}^{in} = k_{u_j}^{in} - 1$ and $\bar{k}_{u_j}^{cut} = k_{u_j}^{cut} + 1$.

So, cases 1 and 4 give the same outcome in updating degrees of adjacent vertices to v, and the same holds for cases 2 and 3. Thus, we get the result. ◇

Algorithm `Update`(i, f, δ) summarizes the above updating procedure. Input values are given by vertex v (that changes its membership), a solution δ and the objective function value f. Notice that all other values mentioned before (n_1, n_2, m_1, etc.) are also input values, however we do not put them in the input list for the sake of readability.

Function `Update`(v, f, δ)

1 **if** δ_v **then**
2 $\quad n_1 \leftarrow n_1 - 1$; $n_2 \leftarrow n_2 + 1$
3 $\quad m_1 \leftarrow m_1 - k_v^{in}$; $m_2 \leftarrow m_2 + k_v^{cut}$
4 $\quad c_1 \leftarrow c_1 + k_v^{in} - k_v^{cut} - k_v^{rest}$
5 $\quad c_2 \leftarrow c_2 + k_v^{in} - k_v^{cut} + k_v^{rest}$
else
6 $\quad n_2 \leftarrow n_2 - 1$; $n_1 \leftarrow n_1 + 1$
7 $\quad m_1 \leftarrow m_1 + k_v^{cut}$; $m_2 \leftarrow m_2 - k_v^{in}$
8 $\quad c_1 \leftarrow c_1 + k_v^{in} - k_v^{cut} + k_v^{rest}$
9 $\quad c_2 \leftarrow c_2 + k_v^{in} - k_v^{cut} - k_v^{rest}$
10 $y \leftarrow k_v^{cut}$; $k_v^{cut} \leftarrow k_v^{in}$; $k_v^{in} \leftarrow y$
11 **for** $j = 1, k_v$ **do**
12 $\quad r \leftarrow h_{ij}$
13 $\quad$ **if** $(\delta_r \wedge \delta_v) \vee (\neg\delta_r \wedge \neg\delta_v)$ **then**
14 $\quad\quad k_r^{cut} \leftarrow k_r^{cut} + 1; k_r^{in} \leftarrow k_r^{in} - 1$
$\quad$ **else**
15 $\quad\quad k_r^{cut} \leftarrow k_r^{cut} - 1; k_r^{in} \leftarrow k_r^{in} + 1$
16 $f_1 \leftarrow \frac{2 \cdot m_1}{c_1}$; $f_2 \leftarrow \frac{2 \cdot m_2}{c_2}$; $f \leftarrow \min\{f_1, f_2\}$
17 $\delta_v \leftarrow \neg\delta_v$;

Algorithm 3: Updating values after allocation move

In line 1 we check if $v \in S_1$. If so, all variables are updated: in line 2 the cardinality of S_1 is reduced by one ($n_1 \leftarrow n_1 - 1$), and cardinality of S_2 increased by one ($n_2 \leftarrow n_2 + 1$). The total number of inner and cut degrees of both sets (S_1 and S_2) are updated in lines 3, 4 and 5 respectively. From step 11 to 15 the degrees of vertices adjacent to v are updated. Similar steps are repeated if $v \in S_2$. Finally, the new objective function is computed, and membership of vertex v flipped.

The following proposition can be easily deduced looking at the given pseudo-code.

Proposition 4 *The worst case complexity of algorithm* `Update` *is in* $O(k_{max})$*, where* k_{max} *is the maximum degree of* G*.*

Note that, comparing the result of Proposition 2 with the result of Proposition 1, we see that the number of operations is decreased by order of $m - k_{max}$ by updating.

3.3.3 Local search algorithm.

A local search, or steepest descent procedure, is built on the idea of moving vertices from one subset to another until an improvement in the objective function value, i.e., until $f^{new} > f$, is obtained. The vertex whose change of membership produces the maximum improvement is chosen to make a new partition. Such partition corresponds to a local maximum of the optimization problem.

Algorithm `Best-Impr-LS`(x, f_{opt}, δ) describes the steps of the proposed Allocation local search.

```
Function Best-Impr-LS(x, f_opt, δ)
1  f_opt ← ∞; improve ← true
2  while Improve do
3      Improve ← false
4      for i = 1, n do
5          v ← x_i
6          if k_v^cut > 0 then
7              Move(v, f, δ)
8              if f < f_opt then
9                  f_opt ← f; w ← v
10                 improve ← true
11     if ¬improve return
12     Update(w, f, δ)
```

Algorithm 4: Best improvement Allocation (re-assignment) local search

The following property holds.

Proposition 5 *The time complexity of one iteration of the local search algorithm is* $O(n)$.

Proof. The proposed local search is based on the move procedure, which is repeated n times. The results follows from the observation that the procedure move is $O(1)$. ◇

3.4 Shaking

Since the solution of bipartitioning S is represented with a binary vector $\delta(x_j)$, the natural way to introduce a distance into the solution space is to use the Hamming distance. That is, we say that the two solutions x and y are at distance k if their Hamming distance is equal to k, i.e., the minimum number of changes of zeros to ones (or reciprocally) required to change x into y is equal to k. We build the shaking procedure on this concept, perturbing the current solution x to a new vector which has Hamming distance k from it. This means that k zeros or ones are switched to their opposite value. In other words, k entities change their original cluster.
Therefore, our `Shake`(k, δ) procedure works as described in Algorithm 5.

```
Function Shake(k, δ)
1 ℓ ← 1
2 while ℓ < k do
3 |   r ← 1 + (|S| − ℓ) · Rand
4 |   v ← x_r
5 |   Update(v, f, δ)
6 |   ℓ ← ℓ + 1;
```

Algorithm 5: Steps of Shaking operator

3.5 Basic VNS for edge-ratio clustering

The basic VNS heuristic for edge-ratio clustering is straightforward, once the local search and the shaking procedures have been introduced. It is summarized in Algorithm 6. Note that `BVNS` is called at each iteration of the divisive hierarchical clustering Algorithm 1 to perform bipartitioning.

```
   Function BVNS (δ, k_max, t_max)
1  Initialization(δ)
2  repeat
3  |   k ← 1
4  |   repeat
5  |   |   δ' ← Shake(k, δ)                    /* Shaking */
6  |   |   δ'' ← Best-Impr-LS(δ', f)          /* Local search */
7  |   |   k ← k + 1                          /* Next neighborhood */
8  |   |   if f(δ'') < f(δ) then
9  |   |   |   δ ← δ''; k ← 1                 /* Make a move */
   |   until k = k_max
10 |   t ← CpuTime()
   until t > t_max
```

Algorithm 6: Basic VNS for edge-ratio clustering

In Algorithm 6, k_{max} and t_{max} represent respectively the maximum number of explored neighborhoods and the maximum cpu time used in the stopping criterion.

4 Computational results and analysis

The proposed VNS heuristic for edge-ratio clustering is validated on a set of 11 problems from the literature, corresponding to various real-world applications. They are: Zachary's karate club dataset [23], describing friendship relationships in a karate club; Lusseau's dolphins dataset [14], describing communications between dolphins; Hugo's *Les Misérables* dataset [11], describing characters in Victor Hugo's masterpiece and their interactions; Krebs' political

books dataset [12]; a dataset representing the schedule of football games between American college teams [5]; a dataset dealing with connections between US airports [19]; a dataset on a coauthorship network of scientists working on network theory and experiment [17]; a network describing electronic circuits [13]; a network representing e-mail interchanges between members of a university [6]; a network giving the topology of the Western States Power Grid of the United States [22] and a network of authors collaborations [19]. All considered networks are undirected and unweighted, without loops.

We compare the results obtained using the VNS proposed in this paper for edge-ratio bipartitioning in a divisive hierarchical algorithm (Algorithm 1) with the results obtained solving, at each iteration of the divisive algorithm, the bipartition problem by an exact algorithm. An exact solution of the bipartitioning problem based on the edge-ratio criterion was proposed in [1]. In that paper, a mathematical programming formulation is proposed where the edge-ratio α is maximized subject to non-linear and non-convex constraints, coming from the products of α with binary variables. An exact algorithm is then obtained fixing the value of α to solve a linear program in 0-1 variables, and then proceeding with a dichotomous search on the values of α. Specifically, one strarts with a value of α equal to 1. If there is no feasible solution for that value, the network is indivisible and one stops, otherwise the value of α is doubled and feasibility checked until a value is attained for which the weak condition cannot be satisfied. This gives an upper bound $\bar{\alpha}$ and the previous value of α gives a lower bound $\underline{\alpha}$. Then the dichotomous search proceeds by considering the mid value of the interval $[\underline{\alpha}, \bar{\alpha}]$ (see [1]). It turns out that this exact algorithm is very time consuming even for small or medium size datasets (i.e., with a few hundred of vertices).

We implemented `BVNS` with the value of k_{max} set to $\min\{50, n/2\}$. Tests were run on AMD Opteron 2 GHz CPU, 128 GB RAM. Results are shown in Table 1.
Networks are listed together with their number of vertices n and number of edges m. We report the cpu time (seconds) to obtain clustering solutions by a divisive hierarchical algorithm using the VNS and the above exact algorithm for the bipartitioning problem solved at each hierarchical level.

dataset	n	m	Time VNS (sec.)	Time exact (sec.)
`karate`	34	78	0.0	62.10
`dolphins`	62	159	9.99e-3	172.2
`les miserables`	77	254	1.99e-2	283.49
`political books`	105	441	1.99e-2	716.45
`football`	115	613	3.99e-2	11780.79
`Usair97`	332	2126	0.62	3752906.94
`netscience_main`	379	914	1.14	2347.24
`s838`	512	819	0.64	12612.53
`email`	1133	5452	6.84	–
`power`	4941	6594	29.80	–
`erdos02`	6927	11850	35.50	–

Table 1: Comparison of results obtained using VNS and an exact algorithm for bipartition according to the edge-ratio criterion, on datasets from the literature

It appears that the proposed VNS for edge-ratio is very efficient in terms of quality of ob-

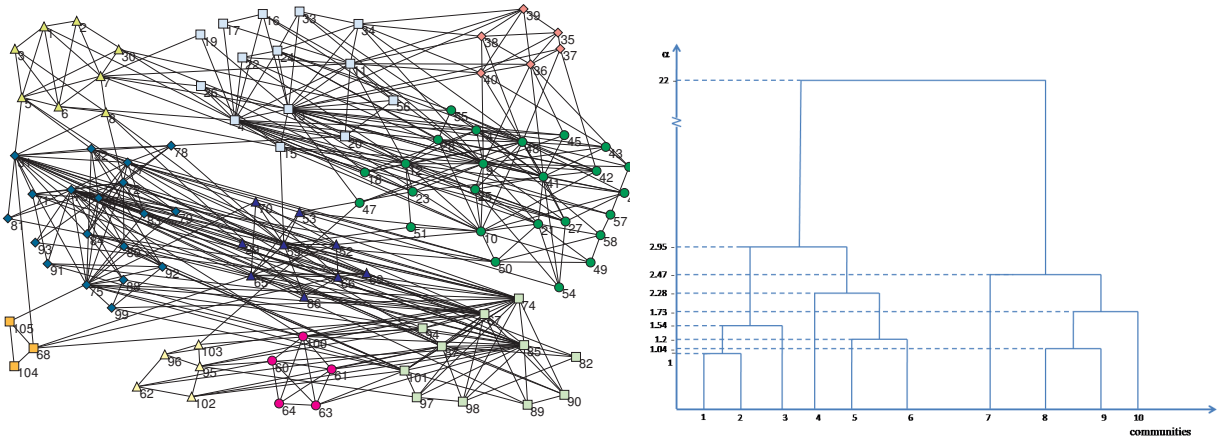

Figure 1: Partition and dendrogram obtained for dataset `polbooks`

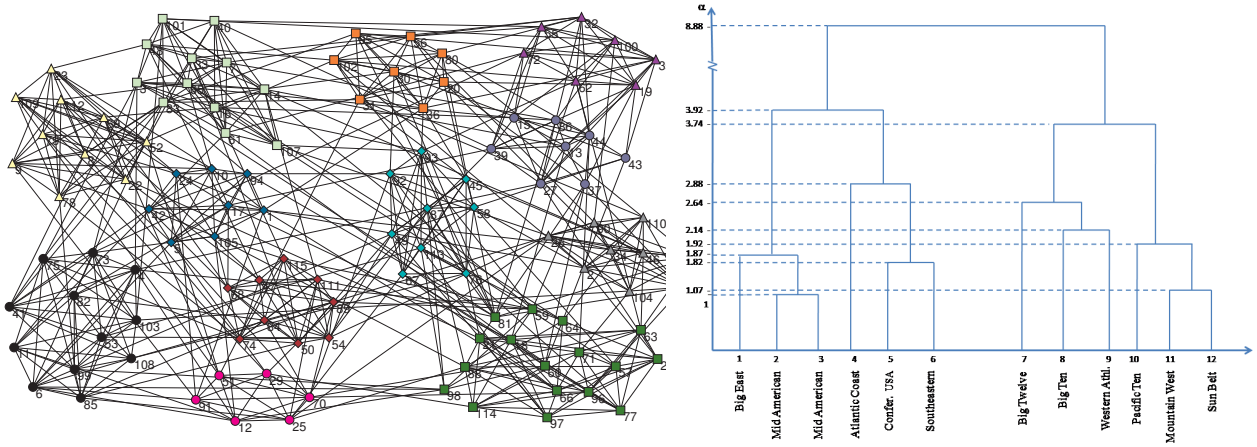

Figure 2: Partition and dendrogram obtained for dataset `football`

tained bipartitions and in terms of computational time. It allows us to solve the bipartitioning problem at each hierarchical level significantly faster than exact bipartitioning, reducing the computational time up to 7 orders of magnitude. This allows us to solve larger problems, increasing the number of problems that we can solve using the exact method. In terms of quality of the solution, the proposed VNS also appears to be efficient. In the most of the cases, we obtain the same partition using VNS and the exact bipartitioning method. We give in Figures 1 and 2 two examples of obtained partitions, and the corresponding dendrogram illustrating the hierarchical divisive algorithm.

5 Conclusions

We proposed a Variable Neighborhood Search heuristic for solving the bipartitioning problem in a hierarchical divisive graph clustering algorithm, according to the recently introduced edge-ratio criterion. Neighborhoods to be used in the VNS are defined using the Hamming distance. We describe the proposed local search and shaking procedures, the latter used to perform systematic perturbations of the current incumbent solution. We give a full description including the pseudo-codes of the procedures used to implement the main steps of the heuristic.

Computational results show that VNS, compared to exact bipartitioning, gives good quality results significantly reducing the computational time.

References

[1] S. Cafieri, P. Hansen, and L. Liberti. Edge ratio and community structure in networks. *Physical Review E*, 81(2):026105, 2010.

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

[3] S. Cafieri, P. Hansen, and N. Mladenović. Edge-ratio network clustering by variable neighborhood search. 2013. submitted.

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

[5] 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.

[6] 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.

[7] P. Hansen and N. Mladenović. Variable neighbourhood search: Principles and applications. *European Journal of Operations Research*, 130:449–467, 2001.

[8] P. Hansen and N. Mladenović. *Variable neighbourhood search*. Oxford University Press, Oxford, 2002.

[9] P. Hansen and N. Mladenović. *Variable neighbourhood search*. Kluwer, Dordrecht, 2003.

[10] P. Hansen, N. Mladenović, and J.A.M. Pérez. Variable neighbourhood search: methods and applications. *4OR*, 6:319–360, 2008.

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

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

[13] Uri Alon Lab. `http://www.weizmann.ac.il/mcb/UriAlon/`.

[14] 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.

[15] N. Mladenović and P. Hansen. Variable neighbourhood search. *Computers and Operations Research*, 24:1097–1100, 1997.

[16] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026133, 2004.

[17] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74:036104, 2006.

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

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

[20] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying comminities in networks. *Proceedings of the National Academy of Sciences, USA*, 101(9):2658–2663, 2004.

[21] J. Wang, Y. Qui, R. Wang, and X. Zhang. Remarks on network community properties. *Journal of Systems Science and Complexity*, 21(4):637–644, 2008.

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

[23] W.W. Zachary. An information flow model for conflict and fission in small group. *Journal of Anthropological Research*, 33:452–473, 1977.