European Journal of Operational Research 230 (2013) 356–363

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Decision Support

Variable neighborhood search for minimum sum-of-squares clustering on networks

Emilio Carrizosa [a,*], Nenad Mladenović [b], Raca Todosijević [c]

[a] Faculdad de Matemáticas, Universidad de Sevilla, Spain
[b] School of Mathematics, Brunel University-West London, UK
[c] Mathematical Institute, Serbian Academy of Science and Arts, Serbia

ARTICLE INFO

Article history:
Received 1 May 2012
Accepted 16 April 2013
Available online 26 April 2013

Keywords:
Minimum sum-of-squares clustering
Location on networks
Variable neighborhood search

ABSTRACT

Euclidean Minimum Sum-of-Squares Clustering amounts to finding p prototypes by minimizing the sum of the squared Euclidean distances from a set of points to their closest prototype. In recent years related clustering problems have been extensively analyzed under the assumption that the space is a network, and not any more the Euclidean space. This allows one to properly address community detection problems, of significant relevance in diverse phenomena in biological, technological and social systems. However, the problem of minimizing the sum of squared distances on networks have not yet been addressed. Two versions of the problem are possible: either the p prototypes are sought among the set of nodes of the network, or also points along edges are taken into account as possible prototypes. While the first problem is transformed into a classical discrete p-median problem, the latter is new in the literature, and solved in this paper with the Variable Neighborhood Search heuristic. The solutions of the two problems are compared in a series of test examples.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Cluster Analysis is a popular and powerful tool in Data Analysis [24,33] that provides a rich variety of challenging optimization problems [16]. The aim of Cluster Analysis is to partition a set of entities into clusters, so that entities within the same cluster are similar and entities in different clusters are different. The most well-known model is *Minimum Sum-of-Squares Clustering* (MSSC). In its basic form, MSSC assumes the entities to be points in $\mathbb{R}^n$; p points (called prototypes) are sought by minimizing the sum of the squares of the Euclidean distances separating the entities from their closest prototypes. MSSC, recently shown to be NP-hard in [2], can be solved exactly for data sets of moderate sizes (over 2300 entities) by column generation [3]. For larger data sets, heuristics are used, see [18,22] and the references therein.

Whereas many clustering problems in Data Analysis are properly accommodated within such Euclidean framework (entities identified with points in the Euclidean space, entities closeness measured by the Euclidean distance), many data sets in complex systems in domains such as Sociology, Biology or Computer Science may be more naturally modeled with networks [1,6,7,14,31,34], where the entities are seen as the nodes, and the edges of the network model entities interaction. In such cases the *length* is usually called *weight*, in order to avoid confusion between some metric and nonmetric properties of the data. Each edge is assumed to have a positive weight, which in the simplest case always takes the value 1, indicating only that the two nodes associated with edges are linked. In many complex systems, however, the edges may have different cost (weight). Transportation and mobility networks, Internet, mobile phone networks, power grids, social and contact networks, and neural networks, are mentioned in [7] as contexts in which the network topology alone does not contain all the information, and community detection should take into account such different edges weights.

In this paper we assume that the set V of entities to be clustered is the set of nodes of a connected and undirected network $N = (V, E)$, where E is a collection of pairs of nodes, and each $e \in E$ has positive length l_e. The set V of nodes is to be split into p disjoint subsets (clusters, communities), and p prototypes (one per cluster) are to be chosen so that all entities in a given cluster are close to their associated prototype. We propose to use a sum-of-squared-distance criterion, which leads us to address two versions of the problem, called V-MSSC and E-MSSC. The V-MSSC (*vertex*-MSSC) consists of selecting a subset V^* of p entities (prototypes) within the set of vertices of the network, so that the sum of squares of the distances from each entity $v \in V$ to its closest prototype is minimized. Closeness between any pair of entities u, $v \in V$ is measured by the length $d(u, v)$ of the shortest path connecting u and v. The E-MSSC (*edge*-MSSC), [12], has the same objective, but the prototypes are sought not only at vertices, but also at points along the edges of the network. This way, one may obtain with the E-MSSC clusters configurations which are impossible if only vertices are allowed

* Corresponding author. Tel.: +34 954557943; fax: +34 954622800.
E-mail addresses: ecarrizosa@us.es (E. Carrizosa), Nenad.Mladenovic@brunel.ac.uk (N. Mladenović), racatodosijevic@gmail.com (R. Todosijević).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.04.027

to be prototypes. In other words, the clusters class which can be obtained by solving E-MSSC is richer than the one obtained by solving V-MSSC problems, and this may lead to more accurate clusters. We emphasize that we use prototypes as a means to cluster entities, though in some cases there is no direct physical meaning of the prototype locations: when applied to nonmetric data such as social networks, the prototype position on the edge that connects two entities, say Peter and Paul, has no physical meaning, but is used here as a tool to identify whether the two entities, say Peter and Paul, belong to the same cluster or not. It is interesting to note that the difference between vertex minimum sum-of-distances and edge minimum sum-of-distances (not squared distances) does not exist, since the optimal solutions of both are equivalent [15].

The purpose of this paper is to introduce these two clustering models on networks, analyzing some structural properties, comparing them, and developing algorithmic tools to cope with data sets of nontrivial size. The rest of the paper is organized as follows. In Section 2 the V-MSSC and E-MSSC problems are formally stated. Some structural properties are presented in Section 3. In Section 4 we describe how VNS can be customized to address E-MSSC. The paper ends with a battery of numerical experiments in Section 5, comparing the output of both problems and of different variants of the VNS designed to solve them. Some concluding remarks and possible lines of future research are given in Section 6.

2. Problems statement

A connected and undirected network $N = (V,E)$, with node set V and edge set E, is given. Each edge $e \in E$ has length $l_e > 0$. The lengths are assumed to satisfy the triangle inequality, and thus they induce a metric on the set of nodes, namely, the shortest-path distance. We will consider the network N to be a spatial network: for any edge $e = (u, v) \in E$, and any $x \in [0,1]$, the pair (e,x) will be identified with the *point* in edge e at a distance xl_e from u, and at distance $(1 - x)l_e$ from v. Let us denote with $P(N)$ the set of all pairs (e,x), i.e., the set of points of the network N. A metric on the set $P(N)$ is defined: for any two points x, $y \in P(N)$, $d(x,y)$ is the length of a shortest path connecting x and y. See [26] for details.

The *V-MSSC* problem is defined as the problem of finding p prototypes *from the set V of vertices* such that the weighted sum of squares of distances from the nodes to their closest prototype is minimized. V-MSSC is easily formulated as a linear integer program. Indeed,

- for each pair $(u, v) \in V \times V$, define the binary variable $x_{u,v}$, which takes the value 1 if entity u is allocated to prototype v, and zero otherwise,
- for each $v \in V$, define the binary variable y_v, which takes the value 1 if entity v is chosen to be one of the prototypes, and zero otherwise.

With this notation, V-MSSC is written as the Integer Program

$$\begin{aligned}
\min \quad & \sum_{u,v \in V} d^2(u,v) x_{uv} \\
\text{s.t.} \quad & \sum_{v \in V} x_{uv} = 1 \quad \forall\, u \in V \\
& x_{uv} \leqslant y_v \quad \forall\, u, v \in V \\
& \sum_{v \in V} y_v = p \\
& x_{uv} \in \{0,1\} \quad \forall\, u, v \in V \\
& y_v \in \{0,1\} \quad \forall\, v \in V.
\end{aligned} \tag{1}$$

Obviously, Problem (1) is the classical p-median problem with the finite set V both as set of users and candidate sites for the facilities, and $d^2(u, v)$ as the distance from the user $u \in V$ to the facility at $v \in V$. Hence, exact and heuristic algorithms as those described in [4,5,17,28] can be used to successfully address V-MSSC.

The *E-MSSC* problem is analogous to the V-MSSC, but prototypes are sought along the edges: we seek a set of p prototypes *from the set P(N) of points on edges* such that the weighted sum of squares of distances from the nodes to their closest prototype is minimized:

$$\begin{aligned}
\min \quad & \sum_{v \in V} \omega_v d^2(v; \{x_1, \ldots, x_p\}) \\
\text{s.t.} \quad & x_1, \ldots, x_p \in P(N),
\end{aligned} \tag{2}$$

where non-negative weights $\omega_v > 0$ are given for any $v \in V$. The distance $d(v,X)$ from v to a non-empty finite set X of points of the network is defined as the distance to the closest point in X,

$$d(v, \{x_1, \ldots, x_p\}) = \min_{1 \leqslant j \leqslant p} d(v, x_j). \tag{3}$$

Once prototypes $x_1^*, \ldots, x_p^*$ have been obtained, solving either (1) or (2), a partition $\{C_1, C_2, \ldots, C_p\}$ of the set V of entities is defined by allocating each entity $v \in V$ to its closest prototype:

$$d(x_i^*, v) \leqslant d\left(x_j^*, v\right) \qquad \forall\, v \in C_i, \quad i = 1, 2, \ldots, p. \tag{4}$$

3. Structural properties

V-MSSC and E-MSSC differ in the space in which prototypes are sought, and thus they may yield different clusters. In [12], 1000 test instances were generated, each having 10 entities uniformly distributed on the line segment [0,1], and $p = 2$ prototypes were sought. In 160 cases, the clusters obtained by the V-MSSC and E-MSSC models were different. It will be shown in Section 5 that this percentage is larger in general graphs, in which the E-MSSC yields clusters which may not be discovered by solving the V-MSSC.

As discussed in Section 2, V-MSSC is a p-median problem with d^2 as distance measure. We refer the reader to [4,5,17,27,32] for an analysis of p-median problems and algorithmic approaches. We present now some properties of the E-MSSC, extending the results given by the authors in [12]. These will be helpful in the design of the heuristic algorithm described in Section 4.

We assume in what follows that the number p of prototypes to be located is strictly smaller than the number of nodes. Otherwise, the problem is solved in a straightforward manner, since locating prototypes at all nodes would yield objective value of zero, which is optimal.

First we show that any optimal solution to E-MSSC yields p nonempty clusters, since, any prototype has at least one entity which is strictly closer to it than to any other prototype.

Property 3.1. *Let $\left(x_1^*, \ldots, x_p^*\right)$ be an optimal solution to (2). Then, for any x_j^* there exists $v \in V$ such that*

$$d\left(v, x_j^*\right) < d(v, x_i^*) \quad \forall\, i \neq j.$$

Proof. Suppose that, for any v there exists $x_{i(v)}^*$ with $d\left(v, x_j^*\right) \geqslant d\left(v, x_{i(v)}^*\right)$ such that $i(v)$ is not equal to j. Take an arbitrary $v_0 \in V$ such that $v_0 \notin \left\{x_1^*, \ldots, x_p^*\right\}$. It is clear that the collection of prototypes replacing x_j^* by v_0 will yield a strictly lower objective value, which contradicts the optimality of $\left(x_1^*, \ldots, x_p^*\right)$. □

Since any optimal solution minimizes the sum of increasing functions of the distance (point,prototype), one has the following.

Property 3.2. *At any optimal solution $\left(x_1^*, \ldots, x_p^*\right)$, the shortest path from any node $v \in V$ to its closest prototype cannot pass through any other prototype.*

Whereas prototypes are allowed to be located at the interior of edges, they cannot be concentrated on a given edge, since each edge can contain at most one optimal prototype in its interior, as stated in the following.

Property 3.3. *Let $\left(x_1^*,\ldots,x_p^*\right)$ be an optimal solution to (2) and let $e=(u,v)\in E$. The interior of e contains at most one optimal prototype. If it contains one optimal prototype x_j^*, then both endpoints u and v have x_j^* as the closest prototype.*

Proof. By contradiction, suppose that the interior of e contains two optimal prototypes, $x_i^*, x_j^*, x_i^*\neq x_j^*$. Without loss of generality, we assume that x_i^* is between u and x_j^*, which is between x_i^* and v. By Property 3.1, there exist $v_i, v_j \in V$ such that

$$d(v_i,x_i^*)\leqslant d(v_i,x_k^*)\quad \forall k$$
$$d\left(v_j,x_j^*\right)\leqslant d(v_j,x_k^*)\quad \forall k.$$

By Property 3.2, u,v, the ending nodes of e are not optimal prototypes since the shortest path from v_i and v_j to their closest facility would otherwise pass through another prototype.

For all nodes v^* that have x_i^* as their closest facility, the shortest path from v^* to x_i^* passes through the end point u (otherwise, it would also pass through x_j^*, contradicting Property 3.2). Hence, if we replace x_i^* by u in the set of prototypes, the objective function would be strictly decreased, contradicting the optimality of $\left(x_1^*,\ldots,x_p^*\right)$. Thus, it is not possible to have two different prototypes in the interior of the edge e.

Now, suppose that only one prototype x_i^* belongs to the interior of the edge e. In order to show that both u and v have x_i^* as their closest prototype, suppose, by contradiction, that x_j^* exists with $d(u,x_i^*)>d\left(u,x_j^*\right)$. For any node $v^*\in V$, having x_i^* as its closest prototype, it follows that the shortest path from v^* to x_i^* must pass through v and not through u, since otherwise x_j^* would be closer than x_i^* to v^*. Hence, replacing x_i^* with v in the set of prototypes, we would obtain another feasible solution with a strictly smaller objective value, contradicting the optimality of $\left(x_1^*,\ldots,x_p^*\right)$. This shows that both u and v have x_i^* as their closest prototype. □

By Property 3.3, given a node v, if an edge e adjacent to v contains in its interior some optimal prototype x_j^*, then both endpoints of e, including v, must have x_j^* as their closest prototype, and thus v cannot be an optimal prototype. This implies the following.

Property 3.4. *If an optimal prototype is located at a node $v\in V$, then the interior of all edges adjacent to v contains no optimal prototypes.*

We end this section studying in more detail the case $p=1$, i.e., one single prototype is to be chosen, which can be seen as the centroid of the network. Of course this case has no direct application for clustering (just one cluster, namely, V, will be obtained), but it will be useful to design the algorithm for the general case.

For $p=1$, it is easy to construct a finite dominating set, [25], i.e., a set known to contain an optimal solution, and thus global optimization of Problem (2) is reduced to inspecting a finite set of candidates. To construct such finite dominating set, it is important to recall that, given $v\in V$, the distance from v to a point x in a given edge $e\in E$ with endpoints u_1 and u_2 is given by

$$\begin{aligned} d(v,x) &= \min\{d(v,u_1)+d(u_1,x), d(v,u_2)+d(u_2,x)\}\\ &= \min\{d(v,u_1)+d(u_1,x), d(v,u_2)+l_e-d(u_1,x)\}. \end{aligned}$$

Whether the minimum above is attained at the first or the second term depends on the relative position of point x with respect to the so-called *bottleneck point*, [26], $z(v)$, defined as

$$z(v)=\frac{1}{2}(l_e-d(v,u_1)+d(v,u_2)). \tag{5}$$

Formula (5) allows us to compute the distance from node v to any point in the edge e:

- If $z(v)\leqslant 0$, then the shortest path from v to any point $x\in e$ passes through u_2.
- If $z(v)\geqslant l_e$, then the shortest path from v to any point $x\in e$ passes through u_1.
- If $0<z(v)<l_e$, then the shortest path from v to $x\in e$ passes through u_1 if x belongs to the sub-edge with endpoints u_1 and $z(v)$, and it passes through u_2 if x belongs to the sub-edge with endpoints $z(v)$ and u_2.

In the latter case, such $z(v)$ will be called a bottleneck point. By definition, for such v, the distance from v to $z(v)$ via u_1 is equal to the distance from v to $z(v)$ via u_2, and then two shortest-paths exist from v to $z(v)$.

Given an edge $e\in E$, and $v\in V$ the distance to v is an affine function on the two subintervals (possibly degenerate) in which the bottleneck $z(v)$ splits e. Hence, within each such subinterval, the squared distance is a quadratic polynomial in the variable x. This process can be done on any given edge e for all nodes v, calculating all bottleneck points $z(v)$, and splitting e into $O(|V|)$ subintervals, such that, within each subinterval, each squared distance is a quadratic polynomial function, and thus the sum of the squared distances is also a quadratic polynomial function. In other words, the objective function of Problem (2) is a second-degree polynomial function in one variable on each such interval. Thus, the derivative of its minimum point should be equal to zero, if it belongs to the considered interval; otherwise its minimum is achieved at one of endpoints of the considered interval. Therefore, at each edge e, the objective function can have at most $|V|-1$ local minima, which are obtained analytically. By doing this process for all edges, we obtain a finite dominating set D for Problem (2). The pseudo-code for finding dominating set D is given in Algorithm 1.

Algorithm 1. Finite Dominating Set

Function FDS(N,E));
1 $D=\emptyset$;
2 for each $e\in E$ **do**
3 **for** each $v\in N$ **do**
4 Calculate bottleneck $z(v)$ according to (5)
end
5 Sort all bottleneck values in nondecreasing order;
6 Form set S as a set of sub-edges obtained by splitting edge e by bottlenecks;
7 **for** each sub-edge e_ℓ from S **do**
8 Find the minimum objective value; denote the corresponding point with y_ℓ;
9 $D=D\cup\{y_\ell\}$
end
end

Since the optimal point has to belong to the finite dominating set D, it can easily be obtained by its inspection.

4. Variable neighborhood search for solving E-MSSC

E-MSSC is a nonlinear optimization problem defined on a network. Finding a globally optimal solution may be done by inspecting, for all possible partitions $C_1,\ldots,C_p$ of V, the objective value at

$\left(x_1^*,\ldots,x_p^*\right)$, where each x_j^* is the optimal solution to E-MSSC for $p = 1$. Such x_j^* can be obtained, as described in Section 3, by inspecting a finite set of points. However, this approach is only applicable for networks of very small size.

For large data sets, heuristic methods seem to be the only option. For that purpose, we propose a heuristic based on Variable Neighborhood Search metaheuristic (VNS) [29,20,30], although some other metaheuristics can be customized for this problem as well.

VNS is a flexible framework for building heuristics to solve approximately combinatorial and global optimization problems. It exploits systematically the possibility of changing the definition of neighborhood structures within the search for a globally optimal (or near-optimal) solution. VNS is based on the following simple observations: (i) An optimum for one neighborhood structure is not necessarily optimal for another neighborhood structure; (ii) a global optimum is a local optimum with respect to all neighborhood structures; and (iii) empirical evidence shows that for many problems all local optima are relatively close to each other. The first property is exploited by using increasingly complex moves in so-called Variable Neighborhood Descent (VND) in order to find local optima. The second property suggests using more neighborhoods if the local optima found are of poor quality. Finally, the third property allows, once a local optimum is reached, to exploit this information to find a better local optimum in its vicinity. See [19,20,30] for further details and applications.

Three important choices for the implementation of VNS are how a starting solution is generated, how the shaking is performed, and how local searches are implemented. We describe now such issues.

4.1. Building an initial solution

The simplest way to build an initial solution is to follow an iterate process by randomly selecting points which do not violate the structural properties described in Section 3. Given a k-uple ($k < p$) of protopypes already selected, we say that a point of the network is feasible if, together with those prototypes previously chosen, Properties 3.3 and 3.4 are satisfied. We say an edge is feasible if it contains feasible points. In our randomized procedure, at each step one feasible edge is chosen at random, and then one feasible point in such edge is chosen at random. The process is repeated until p points are obtained.

After calculating, as preprocessing, all-pairs shortest path distances, the set of feasible edges E_1 initially contains all edges, and the set of feasible points N_1 contains all points of the network. Once a prototype has been chosen, we fathom points and edges according to Properties 3.2 and 3.4. If the chosen prototype belongs to the interior of one edge, then we eliminate all points of such edge from the set of feasible points. If the prototype is equal to one endpoint, we eliminate such endpoint and all interior points of each edge adjacent with such prototype as well. After that, each edge that does not contain feasible points is excluded from the set of feasible edges. So, a random initial solution X is generated by the procedure described as Algorithm 2.

Algorithm 2. Find initial solution at random

Function `RIS` (N,E,p,X);
1 Calculate all-pairs shortest distances;
2 $X = \emptyset$; $E_1 = E$; $N_1 = N$;
3 for $i := 1,\ldots,p$ **do**
4 choose an edge e from the set E_1 at random;
5 choose a feasible point y on the edge e at random;
6 $X = X \cup \{y\}$;
7 reduce sets E_1 and N_1;
end

Since using a good starting solution may be crucial to speed up the convergence of the procedure, we can enhance the quality of the solution obtained in Algorithm 2 by running a heuristic for the V-MSSC instead. Both strategies will be analyzed in the computational results reported in Section 5.

4.2. Shaking

Feasible solutions of E-MSSC are identified with sets $X \subset P(N)$ of cardinality p. The distance between two solutions X_1, X_2 is equal to k if and only if the sets X_1 and X_2 differ exactly in k locations. A (symmetric) distance function ρ can be defined on the set of solutions as

$$\rho(X_1,X_2) = |X_1 \setminus X_2| = |X_2 \setminus X_1| \quad \forall X_1,X_2.$$

Neighborhood structures are induced by the metric ρ, i.e., k locations of facilities ($k \leqslant p$) from the current solution are replaced with k locations that are not in the current solution. We denote by N_k, $k = 1,\ldots,k_{max}$ ($k_{max} \leqslant p$) the set of such neighborhood structures, and by $N_k(X)$ we denote the set of solutions forming neighborhood N_k of a current solution X. In our implementation k_{max}, the highest radius considered, is set to p.

4.3. Local search

The most popular local-search approach for the *Euclidean* MSSC is the so-called *k-means* algorithm, [23]. The k-means is a location–allocation procedure, in which location and allocation steps are repeated until convergence: in the location step (the allocations are assumed to be given), the optimal locations for the p prototypes are obtained; later, in the allocation step (the prototype locations are assumed to be given), all entities are allocated to their closest prototype. The process is repeated until convergence is reached. The key property is that, in the location step, since the allocations are assumed to be fixed, the problem is split into p independent problems, namely, finding one optimal prototype for each cluster.

Here we follow the very same strategy: we propose a location–allocation heuristic, Algorithm 3, in which we exploit the fact that, in the location step, the p independent subproblems to be solved are easy, since, as discussed in Section 3, they can be solved by inspection of the low-cardinality set of candidate points constructed as in Algorithm 1.

Algorithm 3. K-Net-Means algorithm (`Net-KM`) for the NMSSC problem

Function `NetKM` (n,p,X);
1 $C_j = \emptyset$, $j = 1,\ldots,p$;
2 repeat
3 **for** $i := 1,\ldots,n$ **do**
4 $m(u_i) \leftarrow \arg\min_{x_j \in X} d^2(u_i,x_j); m(u_i) \in \{1,\ldots,p\}$
5 $C_{m(u_i)} = C_{m(u_i)} \cup \{u_i\}$
end
6 `RemoveDeg`(n,p,C);
7 **for** $j := 1,\ldots,p$ **do**
8 calculate prototype x_j
end
until m does not change or $\ell = Maxit$;

Starting from a set of p initial prototypes (e.g. taken at random, as explained in Algorithm 2), users are assigned to their closest prototype (steps 3–5). $m(u_i)$ denotes the membership index of a user u_i. Each user u_i is assigned to the cluster $C_{m(u_i)}$, where $m(u_1)$ is the index of a prototype closest to u_i. In the case of ties, i.e., if

there are more than one prototype with the same distance to u_i, the one with the smallest index is chosen. Steps 7 and 8 are location steps, where prototypes x_j, $j = 1, \ldots, p$ are found for a given clusters C_j. More precisely, for each cluster a 1-prototype problem is solved Allocation step of `K-Net-Means` is repeated with the new locations of the prototypes. These steps are repeated until no more changes in assignments occur.

When using this local-search procedure, we may face *degeneracy* [10,13] problems: when customers are allocated to prototypes, some prototypes may remain with no customers assigned. Obviously, if we move one of such prototypes to any node, we will strictly improve the objective value. So, if a degenerate configuration is obtained during this local-search procedure, all prototypes without nodes allocated can be moved randomly to remaining nodes that are not used already as a prototype, improving the objective value. The algorithm for removing degeneracy, as used in Algorithm 3, is described as Algorithm 4.

Algorithm 4. Removing degeneracy

Function `RemoveDeg`(n,p,C);
1 Find prototypes without users, i.e., find indices ℓ with $C_\ell = \emptyset$, $\ell = 1, \ldots, q$;
2 if $q > 0$ **then**
3 **for** $\ell = 1, \ldots, q$ **do**
4 Relocate prototype x_ℓ to random non-occupied node u.
end
end

4.4. A VNS heuristic for E-MSSC

The basic VNS rules, as described above, for solving the E-MSSC problem, lead to Algorithm 5.

Algorithm 5. Basic VNS for E-MSSC

Function `Net-VNS` (x, k_{max}, t_{max});
1 Get an initial solution X;
2 repeat
3 $k \leftarrow 1$;
4 **repeat**
5 $X' \leftarrow$ `Shake`(X,k) /* `Shaking` */;
6 $X'' \leftarrow$ `NetKM`(n,p,X') /* `Local search` */;
7 **if** $f(X'') < f(X)$ **then**
8 $X \leftarrow X'$; $k \leftarrow 1$ /* `Make a move` */;
else
9 $k \leftarrow k + 1$ /* `Next neighborhood`*/;
end
10 $t \leftarrow$ `CpuTime ()`
until $k = k_{max}$;
until $t > t_{max}$;

5. Computational results

The aim of this section is twofold: first, we want to explore whether the new model, E-MSSC, is essentially different from the V-MSSC, by checking if the clusters obtained are the same or not to those given when only entities (nodes) are allowed to be prototypes. In [12] some experiments were performed on line segment networks, showing that different clusters are obtained by V-MSSC and E-MSSC in around 20% of cases. We will show that, for more complex and larger networks, the differences are larger. Second, we want to explore the influence of the starting solution strategy, as described in Section 4.1.

For solving V-MSSC we use a VNS based heuristics described in [17], called here `VNS-0`. As an initial solution, p nodes are selected at random. Then, the algorithm explores neighborhood structures, induced by metric ρ, (see Section 4.2) using Interchange (or vertex substitution) heuristic as a local search.

For solving E-MSSC, three different VNS-based heuristic, called `VNS-1`, `VNS-2` and `VNS-3`, are applied. Algorithm `VNS-1` starts with the solution obtained by `VNS-0`, and then the local-search described in Algorithm 3 is performed. Algorithm `VNS-2` uses as starting solution the one obtained by `VNS-1`, and then our Network VNS, `Net-VNS` (X, k_{max}, t_{max}), explained in Algorithm 5, is run. Finally, `VNS-3` starts with a random initial solution, obtained by `RIS`(N,E,p,X), followed by our `Net-VNS`(X, k_{max}, t_{max}).

All algorithms described in the previous paragraph have been tested on 40 p-median instances taken from the OR-Lib [8]. Each algorithm has been run on each instance 10 times for different choices of the random seed, with the time limit of 10 seconds. The results, obtained with a personal computer with a 2.53 gigahertz CPU and 3 gigabytes of RAM, are reported in Tables 1 and 2. The first three columns are common to both tables. The first column, `Instance`, gives the name of the OR-Lib instances. Instances parameters, namely, the number n of entities, and the number p of prototypes sought, are given in columns 2 and 3 respectively.

Effect of the initial solution. In Table 1 we provide numerical results which give us more insight into the behavior of the VNS algorithm `VNS-0` for solving V-MSSC, as well as into the behavior of VNS algorithms for solving E-MSSC. More specifically, we were interested in exploring the ability of Algorithm 3 to improve a solution found by `VNS-0` as well as the ability of `VNS-2` to improve the output solution of `VNS-1`. Also, we wanted to check the influence of the initial solution on our Network VNS. For this purpose, for each chosen random seed, we compared the objective values of the solutions found by `VNS-0`, `VNS-1`, `VNS-2`, `VNS-3`, i.e. $f_{\texttt{VNS-0}}$, $f_{\texttt{VNS-1}}, f_{\texttt{VNS-2}}, f_{\texttt{VNS-3}}$, by calculating % deviations using the following formula:

$$dev(a,b) = \frac{f_a - f_b}{f_a} \cdot 100.$$

More precisely, for each chosen seed we calculated the following % deviations: dev (`VNS-0`,`VNS-1`), dev (`VNS-1`,`VNS-2`) and dev (`VNS-3`,`VNS-2`). The average values as well as standard deviations for each of these % deviations regarding ten different seeds are reported in Table 1.

From Table 1 the following observations may be derived.

1. `VNS-2`, which uses the most sophisticated strategy for building its starting solution (it solves heuristically the V-MSSC and then runs Algorithm 3) clearly outperforms `VNS-3`, which is initialized with a random solution. Indeed, only for instance `pmed3` `VNS-3` behaves better (average % deviation dev (`VNS-3`,`VNS-2`) is equal to-0.384). On the other hand, the overall average % deviation is 13.394, thus favorable for `VNS-2`. Therefore, the best known solutions for all instances, except one, were found by `VNS-2`. This stresses the importance of having a good initial solution for solving E-MSSC, in accordance with previous observations on solving the continuous p-median problem (also called multi-source Weber problem) [9,21] and MSSC on $\mathbb{R}^n$ [18]. It is apparent that the best results for these problems are obtained if the algorithm takes its time to find the V-MSSC solution first, to be used as starting solution.

Table 1
VNS comparison.

Instance	n	p	Average dev (VNS-0,VNS-1)	St. dev σ	Average dev (VNS-1,VNS-2)	St. dev. σ	Average dev (VNS-3,VNS-2)	St. dev. σ
pmed1	100	5	0.000	0.000	0.042	0.000	0.000	0.000
pmed2	100	10	0.818	0.000	0.175	0.217	0.027	0.364
pmed3	100	10	0.131	0.000	0.913	0.464	−0.384	0.470
pmed4	100	20	1.667	0.000	1.503	0.990	1.190	2.005
pmed5	100	33	5.378	0.549	0.212	0.260	5.522	3.810
pmed6	200	5	0.000	0.000	0.000	0.000	2.306	1.583
pmed7	200	10	0.013	0.000	0.000	0.000	3.004	1.548
pmed8	200	20	0.256	0.000	0.010	0.020	3.093	1.530
pmed9	200	40	4.607	0.000	0.211	0.258	9.129	1.810
pmed10	200	67	5.154	0.015	0.243	0.489	18.162	5.043
pmed11	300	5	0.000	0.000	0.001	0.002	6.089	2.900
pmed12	300	10	0.000	0.000	0.000	0.000	6.783	2.590
pmed13	300	30	0.731	0.020	0.000	0.000	10.062	1.464
pmed14	300	60	1.649	0.078	0.000	0.000	15.313	1.927
pmed15	300	100	7.387	0.368	0.000	0.000	21.353	2.444
pmed16	400	5	0.000	0.000	0.000	0.000	10.670	6.125
pmed17	400	10	0.000	0.000	0.000	0.000	8.344	2.484
pmed18	400	40	0.666	0.176	0.000	0.000	10.610	1.540
pmed19	400	80	4.035	0.335	0.000	0.000	17.134	2.596
pmed20	400	133	6.394	0.197	0.029	0.087	20.576	3.095
pmed21	500	5	0.000	0.000	0.000	0.000	15.474	5.457
pmed22	500	10	0.000	0.000	0.000	0.000	13.806	5.127
pmed23	500	50	0.649	0.024	0.000	0.000	11.703	1.529
pmed24	500	100	3.077	0.135	0.000	0.000	19.524	2.076
pmed25	500	167	6.300	0.192	0.000	0.000	26.485	4.165
pmed26	600	5	0.000	0.000	0.000	0.000	15.628	6.088
pmed27	600	10	0.000	0.000	0.000	0.000	18.697	4.768
pmed28	600	60	0.295	0.100	0.000	0.000	15.623	2.170
pmed29	600	120	1.864	0.291	0.000	0.000	21.985	3.296
pmed30	600	200	6.697	0.433	0.000	0.000	26.962	2.910
pmed31	700	5	0.000	0.000	0.000	0.000	16.115	4.773
pmed32	700	10	0.000	0.000	0.000	0.000	21.323	4.163
pmed33	700	70	0.187	0.049	0.000	0.000	14.340	0.815
pmed34	700	140	2.298	0.123	0.000	0.000	21.700	2.842
pmed35	800	5	0.000	0.000	0.000	0.000	15.701	3.852
pmed36	800	10	0.000	0.000	0.000	0.000	22.618	4.696
pmed37	800	80	0.394	0.053	0.000	0.000	14.431	2.048
pmed38	900	5	0.000	0.000	0.000	0.000	14.346	6.649
pmed39	900	10	0.000	0.000	0.000	0.000	23.625	6.168
pmed40	900	90	0.402	0.112	0.000	0.000	16.700	2.165
Average			1.526	0.081	0.083	0.070	13.394	3.027

2. VNS-2 does not significantly improve VNS-1: on 30 instances out of 40 the average % deviation dev (VNS-1,VNS-2) is equal to 0, while on the others the average % deviation is between 1.503 and 0.001. Therefore, the overall % deviation is very close to 0, i.e. 0.083. This means that the V-MSSC solution obtained with a standard VNS, followed by a local search, already yields excellent results.
3. The average % deviation dev (VNS-0,VNS-1) on all instances is between 7.387 and 0, while the overall average % deviation is equal to 1.526. Furthermore, just on 16 instances out of 40 the average ratio is equal to 0. Therefore, we may conclude that the local search (Algorithm 3) is usually capable to improve the V-MSSC solution.

5.1. Comparison of V-MSSC vs. E-MSSC

Table 2 presents a comparison of V-MSSC and E-MSSC model. The first three columns are the same as those in the Table 1. Columns 4 and 5 report the results of two different heuristics: in column V-MS we have chosen as prototypes those p entities minimizing the sum of distances from the entities to the closest prototype, i.e., the optimal solution to the p-median problem, while in column V-MSS we report the value of the best solution of V-MSSC obtained in ten runs. In both cases, the problem is not solved exactly, but using the version of the VNS for solving p-median described in [17], i.e. VNS-0. The best objective values f_{V-MS} and f_{V-MSS} of the prototypes obtained when solving both problems are compared: column dev reports the % deviation between these two values (i.e., dev (V-MS,V-MSS)). The next column reports the best solution value (column E-MSSC) obtained by one of three different variants of VNS applied to the E-MSSC problem (VNS-1, VNS-2, VNS-3) in ten runs. The deviation of this value from the value reported in column V-MSS is reported in column 8. It should be emphasized that almost all values reported in column E-MSSC were found by VNS-2. The only exception is instance pmed2, on which VNS-3 found the best value. Finally, the last two columns analyze whether E-MSSC yields solutions which are not obtained when only entities are considered as prototypes. Column Node indicates whether the set of optimal prototypes, which corresponds to the value reported in column E-MSSC only contains nodes (and coincides with the optimal prototypes for the V-MSSC problem). Even of the set of optimal prototypes of E-MSSC and V-MSSC do not coincide, it may be the case that they yield identical clusters. This is reported in the last column, Same.

From Table 2 the following observations may be derived.

1. Comparing the values reported in columns V-MSS and E-MSS, one can observe that the difference between these values mostly depend on the value of p. For almost all instances with $p = 5$ or 10, especially those with large n, the best obtained values for V-MSSC and E-MSSC are the same, as well as yielded

Table 2
Computational results.

Instance	n	p	V-MS	V-MSS	Dev. (%)	E-MSS	Dev. (%)	Node	Same
pmed1	100	5	450,233	450,233	0.00	450043.94	0.04	No	No
pmed2	100	10	271,829	256,874	5.50	253067.60	1.48	No	No
pmed3	100	10	295,752	263,385	10.94	259643.17	1.42	No	No
pmed4	100	20	159,678	153,963	3.58	147685.50	4.08	No	No
pmed5	100	33	45,055	42,671	5.29	40066.36	6.10	No	No
pmed6	200	5	410,360	406,195	1.01	386642.24	4.81	No	No
pmed7	200	10	222,901	221,631	0.57	221602.83	0.01	No	Yes
pmed8	200	20	157,807	151,558	3.96	151094.71	0.31	No	No
pmed9	200	40	68,886	66,525	3.43	63126.34	5.11	No	No
pmed10	200	67	16,199	15,938	1.61	14917.01	6.41	No	No
pmed11	300	5	256,532	256,532	0.00	256512.74	0.01	No	No
pmed12	300	10	197,970	197,814	0.08	197814.00	0.00	Yes	Yes
pmed13	300	30	100,398	99,210	1.18	98471.40	0.74	No	Yes
pmed14	300	60	53,604	49,977	6.77	49152.57	1.65	No	No
pmed15	300	100	20,593	20,213	1.85	18653.68	7.71	No	Yes
pmed16	400	5	210,452	209,886	0.27	209886.00	0.00	Yes	Yes
pmed17	400	10	160,401	160,401	0.00	160401.00	0.00	Yes	Yes
pmed18	400	40	92,325	88,234	4.43	87499.01	0.83	No	No
pmed19	400	80	35,678	33,782	5.31	32292.46	4.41	No	No
pmed20	400	133	16,769	16,032	4.40	14930.55	6.87	No	No
pmed21	500	5	203,552	203,552	0.00	203552.00	0.00	Yes	Yes
pmed22	500	10	189,091	188,857	0.12	188857.00	0.00	Yes	Yes
pmed23	500	50	67,359	66,257	1.64	65834.72	0.64	No	No
pmed24	500	100	30,715	29,478	4.03	28533.80	3.20	No	No
pmed25	500	167	13,736	13,377	2.61	12502.12	6.54	No	No
pmed26	600	5	199,503	199,503	0.00	199503.00	0.00	Yes	Yes
pmed27	600	10	147,401	147,096	0.21	147096.00	0.00	Yes	Yes
pmed28	600	60	52,546	51,239	2.49	51030.45	0.41	No	Yes
pmed29	600	120	27,143	25,848	4.77	25335.36	1.98	No	No
pmed30	600	200	12,755	12,533	1.74	11671.78	6.87	No	No
pmed31	700	5	172,938	171,963	0.56	171963.00	0.00	Yes	Yes
pmed32	700	10	157,283	157,177	0.07	157177.00	0.00	Yes	Yes
pmed33	700	70	49,432	47,255	4.40	47188.77	0.14	No	Yes
pmed34	700	140	22,807	21,981	3.62	21461.21	2.36	No	Yes
pmed35	800	5	160,564	160,564	0.00	160541.91	0.01	No	No
pmed36	800	10	153,164	152,914	0.16	152914.00	0.00	Yes	Yes
pmed37	800	80	50,665	48,246	4.77	48195.16	0.11	No	Yes
pmed38	900	5	161,102	161,102	0.00	161102.00	0.00	Yes	Yes
pmed39	900	10	126,553	125,175	1.09	125175.00	0.00	Yes	Yes
pmed40	900	90	44,596	43,035	3.50	42877.82	0.37	No	No

clusters. On the other hand, for larger values of p, a significant lower value for E-MSSC is given, yielding different clusters than those obtained by V-MSSC.

2. In 12 out of 40 instances, the prototypes of E-MSSC and V-MSSC coincide (see column `Node`). On the other hand, in 19 instances out of 40 (47.5%, see column `Same`), the clusters obtained by the two methods are the same. This ratio is much lower than the one reported by the authors in [12], whose preliminary results on clustering on the line showed that 80% of the cases considered gave the same partitions.
3. The classical p-median problem usually yields very good solutions for V-MSSC, as seen when comparing columns `V-MS` and `V-MSS`, so they could be used almost indifferently to build the starting solution of `VNS-2`.

6. Concluding remarks

Minimum Sum-of-Squares Clustering problem (MSSC) in $\mathbb{R}^n$ is probably the most studied clustering problem in the literature. Only recently such MSSC model has been extended to networks, [12], which is a more natural framework for clustering problems from complex systems. In this paper we suggest a basic Variable Neighborhood Search approach for solving E-MSSC, namely, the MSSC when prototypes can be located at vertices or on edges of the network.

Perturbations of the incumbent solution are obtained by using the symmetric difference between two sets (solutions) of common cardinality p, i.e., a random solution at distance k from the incumbent solution is the one that has k different elements. The well-known k-means algorithm is adapted to be used as a local-search routine.

Different VNS-based heuristics developed differ in the way initial solutions are generated. From the computational analysis performed on test instances, several conclusions can be obtained. First, it appears that approximately in 52% of the cases, clusters obtained by E-MSSC and V-MSSC (when prototypes are restricted to be nodes of the network) are different. In addition, finding a good initial solution is shown to be crucial for getting a final solution of good quality.

Several possible directions for future work exist. First, new VNS-based heuristics for solving E-MSSC problem, such as General VNS or Decomposition VNS can be developed; second, the proposed methodology can be extended to similar and more complex clustering models on networks, such as those including additional constraints; third, further testing of our VNS-based heuristic on larger and real world test instances would allow to support strongly our conclusions; finally, while the paper assumes the number p of clusters to be fixed, one may consider p as a decision variable, as done in [11] for the p-median problem.

Acknowledgements

This research is partially supported by the bilateral Serbian–Spanish project AIB2010SE-00318, and projects MTM2009-14039,

MTM2012-36163 (Ministry of Science and Innovation, Spain), FQM-329 (Junta de Andalucia, Spain) and EU European Regional Development Funds. The last two authors are also partially supported by Project #172010, financed by Serbian Ministry of Sciences.

References

[1] Y.-Y. Ahn, J.P. Bagrow, S. Lehmann, Link communities reveal multiscale complexity in networks, Nature 466 (2010) 761–764.
[2] D. Aloise, A. Deshpande, P. Hansen, P. Popat, NP-hardness of Euclidean sum-of-squares clustering, Machine Learning 75 (2009) 245–248.
[3] D. Aloise, P. Hansen, L. Liberti, An improved column generation algorithm for minimum sum-of-squares clustering, Mathematical Programming 131 (2012) 195–220.
[4] P. Avella, A. Sassano, On the p-median polytope, Mathematical Programming 89 (2001) 395–411.
[5] P. Avella, A. Sassano, I. Vasil'ev, Computational study of large-scale p-median problems, Mathematical Programming 109 (2007) 89–114.
[6] A. Barrat, M. Barthélemy, R. Pastor-Satorras, A. Vespignani, The architecture of complex weighted networks, Proceedings of the National Academy of Sciences of the United States of America 101 (2004) 3747–3754.
[7] M. Barthélemy, Spatial networks, Physics Reports 499 (2011) 1–101.
[8] J.E. Beasley, A note on solving large p-median problems, European Journal of Operational Research 21 (1985) 270–273.
[9] J. Brimberg, P. Hansen, N. Mladenović, E. Taillard, Improvements and comparison of heuristics for solving the multisource Weber problem, Operations Research 48 (2000) 444–460.
[10] J. Brimberg, N. Mladenović, Degeneracy in the multi-source Weber problem, Mathematical Programming 85 (1999) 13–220.
[11] E. Carrizosa, A. Ushakov, I. Vasilyev, A computational study of a nonlinear minsum facility location problem, Computers and Operations Research 39 (2012) 2625–2633.
[12] E. Carrizosa, N. Mladenović, R. Todosijević, Sum-of-squares clustering on networks, Yugoslav Journal of Operations research 21 (2011) 157–161.
[13] E. Carrizosa, A. Al-Guwaizani, P. Hansen, N. Mladenović, Degeneracy of Harmonic Means Clustering. Working paper, 2013.
[14] S. Fortunato, Community detection in graphs, Physics Reports 486 (2010) 75–174.
[15] S.L. Hakimi, Optimum distribution of switching centers in a communication network and some related graph theoretic problems, Operations Research 13 (1965) 462–475.
[16] P. Hansen, B. Jaumard, Cluster Analysis and Mathematical Programming, Mathematical Programming. 79 (1997) 191–215.
[17] P. Hansen, N. Mladenović, Variable neighborhood search for the p-median, Location Science 5 (1997) 207–226.
[18] P. Hansen, N. Mladenović, J-Means: a new local search heuristic for minimum sum-of-squares clustering, Pattern Recognition 34 (2001) 405–413.
[19] P. Hansen, N. Mladenovic, J.A. Moreno-Pérez, Variable neighbourhood search: methods and applications, 4OR 6 (2008) 319–360.
[20] P. Hansen, N. Mladenovic, J.A. Moreno-Pérez, Variable neighbourhood search: methods and applications, Annals of Operation Research 175 (2010) 367–407.
[21] P. Hansen, N. Mladenović, E. Taillard, Heuristic solution of the multisource Weber problem as a p-median problem, Operations Research Letters 22 (1998) 55–62.
[22] P. Hansen, E. Ngai, B. Cheung, N. Mladenović, Analysis of global k-means, an incremental heuristic for minimum sum-of-squares clustering, Journal of Classification 22 (2005) 287–310.
[23] J.A. Hartigan, Clustering Algorithms, John Wiley & Sons Inc., New York, 1975.
[24] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, second ed., Springer, New York, 2009.
[25] J.N. Hooker, R.S. Garfinkel, C.K. Chen, Finite dominating sets for network location problems, Operations Research 39 (2001) 100–118.
[26] M. Labbé, D. Peeters, J.F. Thisse, Location on networks, in: M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser (Eds.), Handbooks in OR & MS, Elsevier, Amsterdam, 1995, pp. 551–624.
[27] P. Mirchandani, R. Francis, Discrete Location Theory, Wiley-Interscience, New York, 1990.
[28] N. Mladenović, J. Brimberg, P. Hansen, J.A. Moreno-Pérez, The p-median problem survey of metaheuristic approaches, European Journal of Operational Research 179 (2007) 927–939.
[29] N. Mladenović, P. Hansen, Variable neighborhood search, Computers and Operations Research 24 (1997) 1097–1100.
[30] N. Mladenović, R. Todosijević, D. Urošević, An efficient General variable neighborhood search for large TSP problem with time windows, Yugoslav Journal of Operations Research 23 (in press), http://yujor.fon.bg.ac.rs/index.php/journal/article/view/1008/501.
[31] M.C.V. Nascimento, A. de Carvalho, Spectral methods for graph clustering-A survey, European Journal of Operational Research 211 (2) (2011) 221–231.
[32] J. Reese, Solution methods for the p-median problem: an annotated bibliography, Networks 48 (2006) 125–142.
[33] H. Späth, Cluster Analysis Algorithms for Data Reduction and Classification of Objects, Ellis Horwood, Chichester, 1980.
[34] A. Veremyev, V. Boginski, Identifying large robust network clusters via new compact formulations of maximum k-club problems, European Journal of Operational Research 218 (2) (2012) 316–326.