Les Cahiers du GERAD ISSN: 0711–2440

On the k-medoids model for semi-supervised clustering

R. Randel, D. Aloise,
N. Mladenović, P. Hansen

G–2018–23

March 2018

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2018
– Bibliothèque et Archives Canada, 2018

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2018
– Library and Archives Canada, 2018

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

On the k-medoids model for semi-supervised clustering

Rodrigo Randel [a]
Daniel Aloise [a]
Nenad Mladenović, [b]
Pierre Hansen [c]

[a] *GERAD & Département de Génie Informatique et Génie Logiciel, École Polytechnique Montréal, Montréal (Québec) Canada*

[b] *GERAD & Mathematical Institute, Serbian Academy of Science and Arts, Belgrade, Serbia*

[c] *GERAD & HEC Montréal, Montréal (Québec), Canada*

rodrigo.randel@gerad.ca
daniel.aloise@gerad.ca
nenad.mladenovic@gerad.ca
pierre.hansen@gerad.ca

March 2018
Les Cahiers du GERAD
G–2018–23

Abstract: Clustering is an automated and powerful technique for data analysis. It aims to divide a given set of data points into clusters which are homogeneous and/or well separated. The biggest challenge of data clustering is indeed to find a clustering criterion to express good separation of data into homogeneous groups so that they bring useful information to the user. To overcome this issue, it is suggested that the user provides a priori information about the data set. Clustering under this assumption is often called semi-supervised clustering. This work explores semi-supervised clustering through the k-medoids model. Results obtained by a Variable Neighborhood Search (VNS) heuristic show that the k-medoids model presents classification accuracy compared to that of the typical k-means approach. Furthermore, the model demonstrates high flexibility and performance by combining kernel projections with clustering constraints.

Keywords: k-medoids, semi-supervised clustering, variable neighborhood search

1 Introduction

In unsupervised machine learning, no information is known in advance about the input data. In this learning category, the objective is usually to provide the best description of the input data by looking at the similarities/dissimilarities between its elements. Clustering is one of the main unsupervised machine learning techniques. It addresses the following general problem: given a set of data objects $O = \{o_1, \ldots, o_n\}$, find subsets, namely *clusters*, which are homogeneous and/or well separated [1]. Homogeneity means that objects in the same cluster must be similar and separation means that objects in different clusters must differ one from another. The dissimilarity (or similarity) d_{ij} between a pair of objects (o_i, o_j) is usually computed as a function of the objects' attributes, such that d values (usually) satisfy: (i) $d_{ij} = d_{ji} \geq 0$, and (ii) $d_{ii} = 0$. Note that dissimilarities do not need to satisfy triangle inequalities, i.e., to be distances.

In spite of its concise definition, the clustering problem can have significant variations, depending on the specific model used and the type of data to be clustered. The clustering criterion used plays a key role in the clustering obtained. For example, the homogeneity of a particular cluster can be expressed by its *diameter* defined as the maximum dissimilarity between two objects within the same cluster, while the separation of a cluster can be expressed by the *split* or the minimum dissimilarity between an object inside the cluster and another outside.

When considering dissimilarity measures, the definitions above yield two families of clustering criteria: those to be *maximized* for separation and those to be *minimized* for homogeneity. In general, these criteria are expressed in the form of thresholds, min-sum or max-sum for a set of clusters. Thus, for instance, the diameter minimization problem corresponds to minimizing for a set of clusters the maximum diameter found among them, while in the split maximization, one seeks to maximize the minimum split found in the clustering partition. The clustering criterion used is also determinant to the computational complexity of the associated clustering problem. For example, split maximization is polynomially solvable in time $O(n^2)$, while diameter minimization is NP-hard already in the plane for more than two clusters [2].

In order to overcome this difficulty and improve the result of the data clustering, it has been suggested that the domain expert could provide, whenever possible, auxiliary information regarding the data distribution, thus leading to better clustering solutions more in accordance to his knowledge, beliefs, and expectations. The clustering process driven by this side-information is called *Semi-Supervised Clustering* (SSC). SSC has become an important tool in data mining due to the continuous increase in the volume of generated data [3].

The most common types of side-information are pairwise constraints such as *must-link* and *cannot-link* [4]. A *must-link* constraint between two objects implies that they must be assigned to the same cluster, whereas a *cannot-link* constraint that they must be allocated in different clusters. In this paper, we make an in-depth analysis of the use of the k-medoids model for the SSC problem. We also propose a new Variable Neighborhood Search (VNS) [5] algorithm that uses a location-allocation heuristic and takes into consideration pairwise constraints.

The paper is organized as follows. The next section presents the related works to this research. Section 3 describes the k-medoids model for the SSC problem. Section 4 describes the two-stage local descent algorithm proposed, and in Section 5 a VNS algorithm is presented for optimizing the described model. Computational experiments that demonstrate the effectiveness of our methodology in a set of benchmark data sets are reported in Section 6. Finally, the conclusions are presented in Section 7.

2 Related works

Algorithms that make use of constraints as *must-link* and *cannot-link* in clustering became widely studied and developed after the COP-Kmeans algorithm of Wagstaff and Cardie's work [6]. The algorithm is based on modifying the unsupervised original k-means algorithm by adding a routine to prevent an object from changing cluster if any of the *must-link* or *cannot-link* constraints is violated.

The model optimized by COP-Kmeans consider that objects $o_i \in O$ correspond to points p_i of a s-dimensional Euclidean space, for $i = 1, \ldots, n$. The objective is to find k clusters such that the sum of

squared Euclidean distances from each point to the centroid of the cluster to which it belongs is minimized while respecting a set of pairwise constraints. The set $\mathcal{ML}$ is formed by the pairs of points (p_i, p_j) such that p_i and p_j must be clustered together, whereas the set $\mathcal{CL}$ contains the pair of points (p_i, p_j) such that p_i and p_j must be assigned to different clusters.

The *semi-supervised minimum sum-of-squared clustering* (SSMSSC) model is mathematically expressed by:

$$\min_{x,y} \quad \sum_{i=1}^{n}\sum_{j=1}^{k} x_{ij}\|p_i - y_j\|^2 \tag{1}$$

subject to

$$\sum_{j=1}^{k} x_{ij} = 1, \quad \forall i = 1, ..., n \tag{2}$$

$$x_{ij} - x_{wj} = 0, \quad \forall (p_i, p_w) \in \mathcal{ML}, \quad \forall j = 1, ..., k \tag{3}$$

$$x_{ij} + x_{wj} \leq 1, \quad \forall (p_i, p_w) \in \mathcal{CL}, \quad \forall j = 1, ..., k \tag{4}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i = 1, ..., n; \quad \forall j = 1, ..., k. \tag{5}$$

The binary decision variables x_{ij} express the assignment of point p_i to the cluster j whose centroid is located at $y_j \in \mathbb{R}^s$. Constraints (2) guarantee that each data point is assigned to exactly one cluster. Constraints (3) refer to the must-link constraints, and constraints (4) to the cannot-link ones.

The simplicity and pioneering of COP-Kmeans have made it a basic algorithm for many later works. Some examples are: semi-supervised clustering using combinatorial Markov random fields [7]; adaptive kernel method [8]; clustering by probabilistic constraints [9]; and density-based clustering [10].

A relevant work involving clustering under pairwise constraints was conducted by Xia [11]. The global optimization method proposed in that work is an adaptation of the Tuy's cutting planes method [12]. The algorithm is proved to obtain optimal solutions in exponential time in the worst case, and hence, it cannot be used for practical purpose for larger data mining tasks. Xia [11] reported a series of experiments where the algorithm is halted before convergence. The obtained clustering results were superior other algorithms based on COP-Kmeans.

Restricting the solution space through the explict use of pairwise constraints is not the only possible approach for SSC. Many works have been published to propose mechanisms using distance metric learning to explore these side-information. Among them, a well-known algorithm is the *Semi-Supervised-Kernel-kmeans* [13] that enhances the similarity matrix obtained from the application of a kernel function by adding a term that brings closer together *must-link* objects while driving away *cannot-link* objects. The algorithm defines a similarity matrix $\mathbf{S} = \mathcal{K} + W + \sigma I$, where $\mathcal{K}$ is a kernel matrix, W is the matrix responsible to include the pairwise constraints into the distance metric, and σ is the term that multiplies an identity matrix I to ensure that $\mathbf{S}$ is semi-definite positive. The kernel-k-means algorithm [14] is then executed over $\mathbf{S}$ in an unsupervised manner (see [13] for details).

3 Proposed model

Another classical representative-based clustering model is the k-medoids whose objective is to partition the points into exactly k clusters so that the sum of distances between each point and the central object (i.e., the *medoid*) of their respective cluster is minimized.

The input of the k-medoids model is a distance matrix, D, with each entry d_{ij} providing the dissimilarity between points p_i and p_j. It can be mathematically formulated in its semi-supervised version as:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} d_{ij} \tag{6}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i = 1, ..., n \tag{7}$$

$$x_{ij} - x_{wj} = 0 \quad \forall (p_i, p_w) \in \mathcal{ML}, \quad \forall j = 1, ..., n \tag{8}$$

$$x_{ij} + x_{wj} \leq 1 \quad \forall (p_i, p_w) \in \mathcal{CL}, \quad \forall j = 1, ..., n \tag{9}$$

$$x_{ij} \leq y_j \quad \forall i = 1, \ldots, n, \forall j = 1, ..., n \tag{10}$$

$$\sum_{j=1}^{n} y_j = k \tag{11}$$

$$x_{ij} \in \{0, 1\} \quad \forall i = 1, \ldots, n, \forall j = 1, \ldots, n, \tag{12}$$

$$y_j \in \{0, 1\} \quad \forall j = 1, \ldots, n, \tag{13}$$

where y_j is equal to 1 if p_j is selected as the medoid of cluster j, and 0 otherwise. Constraints (10) assure that points can only be assigned to selected medoids, and constraint (11) defines that k medoids must be selected. The resulting model (6)–(13) is named thereafter the **Semi-Supervised K-Medoids Problem** (SSKMP).

The possibility of defining the matrix D allows the objective function of the model to be flexible to use different measures to express the dissimilarities between points and medoids. The k-medoids model can be used to cluster metric data, as well more generic data with notions of similarity/dissimilarity. For this reason, one of the main features of k-medoids is its vast list of applications [15].

When comparing the k-means model with the k-medoids model, Steinley [16] listed three relevant advantages in using the later for clustering:

1. Although both models work with a center-based approach, the k-means model defines the central element as the centroid of the cluster, while in the k-medoids this element is taken directly from the data set. This feature allows, for example, to identify which is the most representative element of each cluster.
2. The k-medoids, in its formal definition, usually consider the Euclidean distance to measure the dissimilarity between points and medoids, instead of the quadratic one considered in k-means. As consequence, the k-medoids is generally more robust to outliers and noise present in the data [17].
3. While k-means only uses quadratic distance and may need to constantly recompute the distances between points and centroids every time centroids are updated, the k-medoids run over any distance matrix, even those for which there exist triangle inequality violations and which are not symmetric.

4 Local descent algorithm for SSKMP

Several heuristics methods have already been proposed to solve the original k-medoids problem. A very popular one is the *interchange* heuristic introduced in [18]. This local descent method searches, in each iteration, for the best pair of medoids (one to be inserted in the current solution, and another to be removed) that leads to the best-improving solution if swapped. If such pair exists, the swap is performed and the procedure is repeated. Otherwise, the algorithm stops and the best solution found during this descent path is returned. An efficient implementation of this procedure, called *fast-interchange*, was proposed by Whitaker [19]. However, this method was not widely used (possibly due to an error in the article) until Hansen and Mladenović [20] corrected it and successfully applied it as a subroutine of a VNS heuristic. After, Resende and Werneck [21] proposed an even more efficient implementation by replacing one of the data structures present in the implementation of Whitaker [19] by two new data structures. Although the implementation suggested in [21] has the same worst-case complexity, $O(n^2)$, it is significantly faster and, to the best of the authors' knowledge, is the best implementation for the heuristic *interchange* already published.

In this paper, the method proposed in [21] is used as local descent procedure for our algorithm in order to refine a given SSKMP solution, but with a slight modification to ensure that pairwise constraints are respected.

4.1 Handling must-link constraints

The following strategy is proposed to respect *must-link* constraints:

If a set of points is connected by must-link constraints, they can all be merged into a single point, which is enough to represent them all.

This assumption relies on the fact that all these points need to be together in the final partition, and aggregating them is just an efficient shortcut for assigning them to the same cluster repeatedly times.

Figure 1 illustrates this process on a set of *must-link* constraints given by $\mathcal{ML} = \{(p_1, p_2), (p_4, p_6), (p_2, p_6)\}$. It is possible to replace the set $\mathcal{ML}$ by an equivalent set $\mathcal{ML}' = \{(p_1, p_2), (p_1, p_4), (p_1, p_6)\}$, with p_1 as the root point for all other linked points p_2, p_4 and p_6 (Figure 1b). This aggregation creates a so-called *super-point* and is showed in Figure 1c where all points involved in that *must-link* constraint are all represented by the super-point p_1. Note that the super-point could have been aggregated over p_2, p_4 or p_6 instead of p_1 without prejudice.

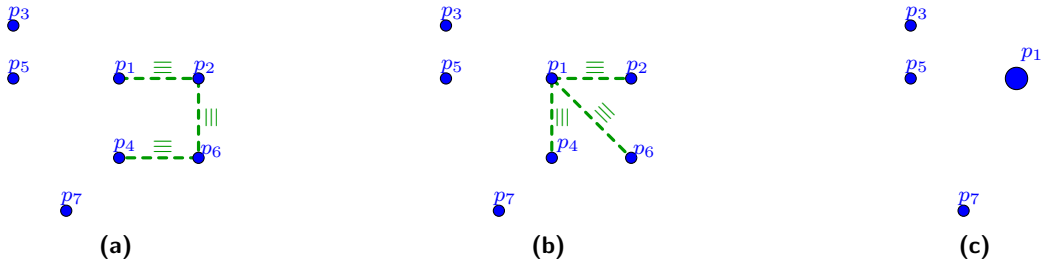

Figure 1: Illustration of a super-point aggregation.

However, since the points involved in *must-link* constraints can be aggregated and viewed as a single point, then it is also necessary to update the dissimilarity d_{ij} of a super-point p_i, and all medoids $j = 1, \ldots, n$, as the sum of dissimilarities of all points that compose it. Let $H(p_i) = \{p_h \in P \mid (p_i, p_h) \in \mathcal{ML}\}$ be the set of points that are part of the super-point p_i. The cost $d_{ij} = d_{ji}$, for each $j = 1, \ldots, n$ is then calculated as the sum of dissimilarities considering all aggregated points, i.e.,

$$d_{ij} = \sum_{h:p_h \in H(p_i)} d_{hj} \quad j = 1, \ldots, n \tag{14}$$

For the example in Figure 1, d_{13} is updated as: $d_{13} = d_{13} + d_{23} + d_{43} + d_{63}$.

The super-point aggregation is a quick step that can be entirely performed during the preprocessing stage of the algorithm. It also helps to reduce the dimension of the original data set once the points are merged. Consequently, the more *must-link* constraints are provided by the expert, the best is the performance of our algorithmic approach.

4.2 Handling cannot-link constraints

Once all *must-link* constraints are respected, the local descent algorithm only concerns violated *cannot-link* constraints. A solution is said to be infeasible if there exists any pair $(p_i, p_w) \in \mathcal{CL}$ such that p_i and p_w are assigned to the same cluster. In order to avoid that, the algorithm is divided into two stages:

Stage 1. In this first stage, the *cannot-link* constraints are temporarily neglected and the local descent algorithm proceeds to improve the current best solution.

Stage 2. For each new improved solution found in stage 1, there is a chance of this solution be infeasible, so the algorithm invokes a routine able to restore its feasibility (with respect to the *cannot-link* constraints).

In summary, the approach of our local descent algorithm is to allow an efficient search to be executed in the direction of the best possible solution (regardless of the *cannot-link* constraints), whereas the solutions

obtained during the descent search path are turned into feasible solutions. Thus, the algorithm relies upon the possibility of restoring the feasibility of solutions generated in the first stage of the algorithm. The key point of our strategy is to guide the search exclusively by the gradient of the objective function, disregarding the *cannot-link* constraints.

Let $\mathbf{s}$ be a solution for the problem with its k selected medoids, i.e., $\mathbf{s} = \{j|y_j = 1\}$. Let us denote $X(i) = \{j|x_{ij} = 1\}$ the cluster of point p_i. We also define the set $E(i) = \{h|(p_i, p_h) \in \mathcal{CL}\}$ as the set of points that cannot be clustered together with p_i, and $B(i) = \{j \in \mathbf{s}|\exists h \in E(i), X(h) = j\}$ as the set of clusters in $\mathbf{s}$ that are *blocked* to p_i since it contains at least one point from $E(i)$.

The feasibility routine is presented in Algorithm 1. It is called whenever a new infeasible solution $\mathbf{s}$ is obtained by the algorithm. Let $\phi_1(i) \in \mathbf{s}$ be the closest medoid in $\mathbf{s}$ from point p_i. Remark that after stage 1, since *cannot-link* constraints are not considered, every point p_i, for $i = 1, \ldots, n$, is assigned to its closest medoid, i.e., $X(i) = \phi_1(i)$.

Algorithm 1 Restore feasibility function

```
1:  R ← ∅
2:  for i = 1, ..., n do
3:     if E(i) ≠ ∅ then
4:        R ← R ∪ {i}
5:     end if
6:  end for
7:  repeat
8:     shuffle(R)
9:     for all i ∈ R do
10:       if X(i) ∈ B(i) or X(i) ≠ φ1(i) then
11:          Assign p_i to the closest medoid j ∈ s such that j ∉ B(i)
12:       end if
13:    end for
14: until no assignment is made
```

The restore function works as follows: first, between lines 1-6, a new set R is built to contain all points p_i that are involved in *cannot-link* constraints (i.e., $E(i) \neq \emptyset$). The loop of lines 7-14 proceeds by removing the cannot-link violations. The order in which the points are examined determines the solution obtained or even if the method is able to restore feasibility. Therefore, set R is shuffled at the beginning of that loop at line 8. Then, the algorithm iterates in the loop of lines 9-13 searching for assignments that can make the solution feasible (condition $X(i) \in B(i)$) or that can improve its cost (condition $X(i) \neq \phi_1(i)$). The rationale behind the second condition is that p_i might have been allocated to a farther cluster in a previous iteration of the restoration routine because its closest medoid was not available for assignment due to a cannot-link constraint. Note that algorithm 1 needs to keep that B updated. This is performed every time after a point p_i is assigned from a medoid p_ℓ to medoid p_j, blocking this medoid for each point in $h \in E(i)$, and maybe removing ℓ from their sets $B(h)$, depending on the presence of any other point in $E(h)$ assigned to medoid p_ℓ.

Algorithm 1 is assured to finish although a feasible solution is not guaranteed. Indeed, the decision problem of whether a clustering problem is feasible given a set $\mathcal{CL}$ of cannot-link constraints is NP-complete [22]. In that case, the obtained solution is simply discarded.

5 Variable Neighborhood Search for SSKMP

Variable Neighborhood Search (VNS) metaheuristic [23] has been successfully applied to many clustering problems (e.g. [24, 25, 26, 27]). The neighborhood structure adopted in our VNS algorithm is based on *swapping* selected medoids of a solution $\mathbf{s}$ by others non-selected medoids outside $\mathbf{s}$. In this sense, v_{max} neighborhoods are defined, where the v-th neighborhood of $\mathbf{s}$, $N_v(\mathbf{s})$, contains all solutions obtained after replacing v medoids $j \in \mathbf{s}$ with others v not-selected medoids $l \notin \mathbf{s}$.

The Algorithm 2 presents the complete framework of our VNS algorithm. It starts by preprocessing the *must-link* constraints via the *super-point* concept (lines 1). Following that, the algorithm constructs an initial feasible solution (line 2) obtained in a series of three steps: (i) an initial solution $\mathbf{s}_b$ is built by randomly

selecting k initial medoids and assigning each point to its closest medoid; (ii) the restore feasibility function is applied for $\mathbf{s}_b$; (iii) if $\mathbf{s}_b$ is still infeasible, the algorithm proceeds and replace $\mathbf{s}_b$ by the first feasible solution found during the VNS. We assume that the problem is always feasible, i.e., the sets $\mathcal{ML}$ and $\mathcal{CL}$ allows to obtain a feasible solution for the SSC under consideration. The algorithm considers that infeasible solutions have infinity cost.

Next, the algorithm starts the VNS block (loop 3–15) that chooses a random neighbor solution (line 6) and applies the two-stage local descent method described in Section 4 to possibly improve it (line 10). However, notice that after a random solution $\mathbf{s}^r$ is chosen in the neighborhood $N_v(\mathbf{s})$ of our VNS algorithm, an allocation step must follow to re-assign the points that were allocated to the replaced medoids (removed medoids of $\mathbf{s}$) to their new closest medoid. However, this process does not take into consideration the *cannot-link* constraints, and then, $\mathbf{s}^r$ might be infeasible. To overcome this situation, we also invoke the restore function for $\mathbf{s}^r$ before proceeding to the local search procedure (line 8). If the best feasible solution found in the descent path has a better cost than $\mathbf{s}_b$, then it is stored in $\mathbf{s}_b$ (line 12). The algorithm repeats this process until a defined stopping criterion is met.

Algorithm 2 VNS for SSKMP

```
1: Apply the super-point concept, merging points interconnected by must-link constraints into super-points;
2: Find an initial feasible solution s_b (if possible);
3: repeat
4:     v ← 1;
5:     repeat
6:         Choose a random neighbor solution s^r ∈ N_v(s);
7:         if s^r is infeasible then
8:             Call the restore feasibility function for s^r.
9:         end if
10:        Apply the local descent method from s^r, obtaining a local minimum s_f
11:        if cost of s_b > cost of s_f then
12:            s_b ← s_f; v ← 1;
13:        end if
14:        v ← v + 1;
15:    until v = v_max
16: until a stopping criterion is met
```

6 Experiments

This work explores the results under three different perspectives. First, model SSKMP is analyzed with respect to the clustering classification when compared with the popular used SSC model, i.e. SSMSSC. Next, the VNS performance is tested using a set of benchmark data sets for SSC problem. Third, the flexibility of SSKMP is explored in combination with distance metric learning.

Computational experiments were performed on an Intel i7-6700 CPU with a 3.4GHz clock and 16 Gigabytes of RAM memory. The algorithms were implemented in C++ and compiled by gcc 6.3.

6.1 Model accuracy

First of all, it is important to keep in mind that it is impossible to determine whether a model is better than another with respect to all possible data sets (see Kleinberg's impossibility theorem [28]). The SSMSSC and SSKMP are comparable models given that (i) both are representative-based; and (ii) the data sets used in the experiments are considered as points in the Euclidean space. It was decided to compare the models regarding accuracy using the *Adjusted Rand Index* (ARI) [29], which can measure how close the clustering result is to the ground-truth classification obtained in the UCI repository [30].

We first compare the models using the ARI results reported by Xia [11]. As done in her work, we ran the VNS algorithm 100 times and reported the average ARI value. We also defined the stop criterion as the average CPU time used by Xia's algorithm. In all experiments we used the v_{max} parameter equal to 10. Table 1 presents this the of these two models for 12 benchmark data sets. For each of them, column n

indicates the number of points and k the number of clusters. In the following, we present results for two configurations of $\mathcal{ML}$ and $\mathcal{CL}$ used in [11]. The first two columns refer to the number of *must-link* and *cannot-link* constraints, and the last two refer to the ARI index values obtained by each model with respect to the ground-truth partition.

Table 1: Datasets configurations and ARI results for SSMSSC and SSKMP.

			Configuration 1				Configuration 2			
Instance	n	k	$\lvert\mathcal{ML}\rvert$	$\lvert\mathcal{CL}\rvert$	**ssmssc**	**sskmp**	$\lvert\mathcal{ML}\rvert$	$\lvert\mathcal{CL}\rvert$	**ssmssc**	**sskmp**
Soybean	47	4	4	24	0.55	**0.60**	8	4	0.62	0.62
Protein	116	6	18	12	**0.31**	0.25	26	18	**0.32**	0.25
Iris	150	3	12	12	0.74	**0.75**	16	8	0.75	**0.76**
Wine	178	3	44	26	0.44	**0.45**	72	44	0.45	0.45
Ionosphere	351	2	52	36	0.16	0.16	122	64	0.14	**0.15**
Control	600	6	60	30	**0.54**	0.50	90	60	**0.53**	0.51
Balance	625	3	156	94	**0.32**	0.24	218	126	**0.43**	0.25
Yeast	1484	10	296	178	0.16	0.16	520	296	0.17	0.17
Optical	3823	10	496	306	**0.70**	0.68	689	420	**0.71**	0.69
Statlog	4435	6	444	222	0.53	0.53	666	444	**0.54**	0.53
Page	5473	5	548	274	0.01	**0.03**	1024	820	0.01	**0.03**
Magic	19020	2	1902	952	0.05	**0.18**	2854	1902	0.04	**0.16**

We note from Table 1 that both models present quite similar results and comparable clustering performances. For the 24 tests cases, each model had nine times each the best ARI, and for six data sets, they had the same ARI value. Moreover, even when the ARI indices were not equal, the difference in values was marginal.

6.2 VNS performance

This section is dedicated to evaluating the VNS performance for optimizing the SSKMP model. In order to obtain the optimal solution for the tested data sets, we used the solver CPLEX 12.6. This restricted our sample in this experiment because CPLEX was not able to solve data sets `Optical`, `Statlog`, `Page` and `Magic` in a reasonable amount of time (less than 50 hours).

Table 2 shows the results of our computational experiments. For each configuration, we executed the algorithm 10 times using 300 seconds as time limit. Columns f_{opt} and t_{opt} provide optimal solution values and the time needed by CPLEX to obtain it, respectively. The column `vns` reports the gap between the optimal solution and the best solution found by the VNS from the 10 distinct executions. In the sequel, columns $\overline{\texttt{vns}}$ and $\overline{t_{vns}}$ report the average values for the same 10 execution of the algorithm. The column `restore` presents the average percentage of time required by the restore feasibility function during the execution.

Table 2: Performance results for VNS and CPLEX.

Instance	Configuration	f_{opt}	t_{opt}	`vns`	$\overline{\texttt{vns}}$	$\overline{t_{vns}}$	`restore`
Soybean	1	1.138047e+02	0.12	**0%**	**0%**	0.00	11%
	2	1.156629e+02	0.16	**0%**	**0%**	0.00	10%
Protein	1	1.269331e+03	2.46	**0%**	**0%**	0.01	10%
	2	1.262633e+03	0.93	**0%**	**0%**	0.00	15%
Iris	1	9.835843e+01	8.85	**0%**	**0%**	0.01	7%
	2	9.962796e+01	7.94	**0%**	**0%**	0.00	6%
Wine	1	1.749303e+04	10.07	**0%**	**0%**	0.01	7%
	2	1.907746e+04	17.16	**0%**	**0%**	7.08	7%
Ionosphere	1	8.172423e+02	61.44	**0%**	**0%**	5.13	9%
	2	8.384550e+02	53.58	**0%**	0.02%	65.04	9%
Control	1	2.693438e+04	135.20	**0%**	**0%**	0.13	5%
	2	2.693937e+04	124.34	**0%**	**0%**	0.12	5%
Balance	1	1.466425e+03	881.94	**0%**	0.002%	110.42	4%
	2	1.471803e+03	816.61	**0%**	**0%**	91.51	4%
Yeast	1	2.523202e+02	166622.71	**0%**	**0%**	97.78	3%
	2	2.605097e+02	42557.74	**0%**	0.004%	124.44	3%

Firstly, we justify the importance of having a heuristic approach to the problem since the time to optimally solve it increases exponentially as the number of points scales (t_{opt}). In the other hand, for all the 16 test cases, the VNS was able to find the optimal solutions using much less time. For the test where CPLEX took the longest time to solve, 46h for `Yeast` configuration 1, the VNS only needed, on average, 98 seconds to obtain a solution with the same cost. Furthermore, only in 3 scenarios, the VNS was not able to obtain the best solution in all 10 executions, but still, the gaps are tiny.

From the results reported in column `restore`, we verify that the restore feasibility function does not require much computational time (7% on average) for the instances used in [11]. Besides, the amount of time is reduced for larger data sets, which is expected as the local descent procedure starts to demand more computational resources to perform the search.

6.3 Model flexibility

One of the main advantages of using SSKMP is the ability to work with a general dissimilarity matrix D as input. This feature not only allows the model to work with many different metric systems but also provides a great flexibility to define a clustering criterion. For example, it is possible to use the *distance metric learning* technique without a single modification with our the algorithm. Take for instance the *Semi-Supervised-Kernel-Kmeans* (SS-Kernel-k-means) algorithm [13], which defines the similarity matrix $\mathbf{S} = \mathcal{K} + W + \sigma I$, aggregating the kernel matrix $\mathcal{K}$ and constraints matrix W (metric learning). Then, we can easily transform $\mathbf{s}$ into a dissimilarity matrix D (e.g. subtract each entry by the maximum element in $\mathbf{s}$) and use it as input for SSKMP. Furthermore, having the distance metric modification in the input does not preclude the use of pairwise constraints in combination, which has already been proven to be a good approach [31].

Consider the synthetic data set **Two Circles** showed in Figure 2, which presents 200 points in the Euclidean plane, with 100 points in each class. This data set has an inner circle and a surrounding outer circle.

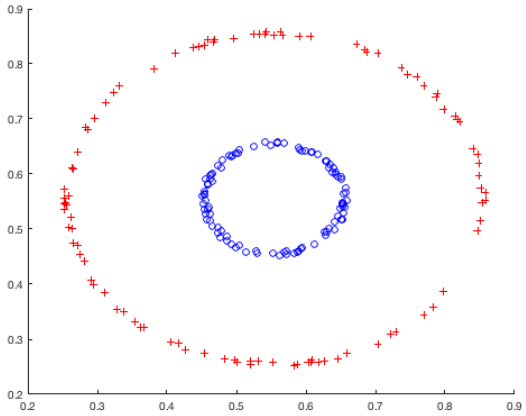

Figure 2: Two Circles synthetic data set.

The Figure 3 presents the ARI results for our proposed VNS and the SS-Kernel-k-means algorithm using the two circles instance. Both algorithms were executed 100 times starting from a random initial solution, and the average ARI was reported. As suggested in [13], we used an exponential kernel ($exp(\|xy\|^2/2\sigma)$ for SS-Kernel-k-means in order to linearly separate the two classes in the mapped space. We also included the algorithm `vns+` which combines the distance metric learning and explicit pairwise constraints into the model optimized by our proposed VNS. The time limit used for both VNS algorithms was the average time needed by SS-Kernel-k-means to finish one execution.

We note from Figure 3 that the VNS algorithms based on model SSKMP outperformed the typical kernel approach, both reaching the maximum ARI value. We highlight that the VNS algorithm improved

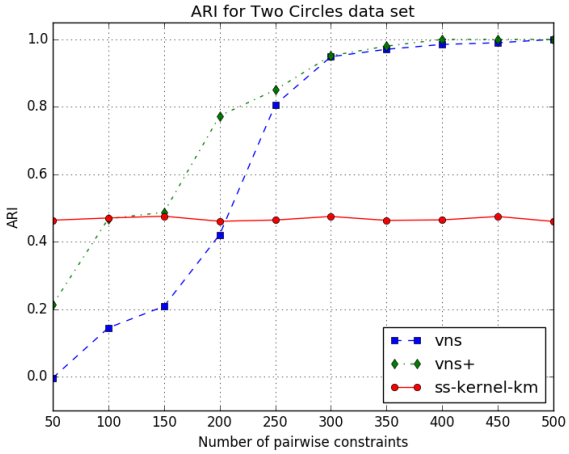

Figure 3: ARI performance for Two Circles data set.

its accuracy performance by adding the distance metric learning mechanism, reaching the maximum ARI value with 20% less constraints than the VNS algorithm that uses only the pairwise constraints. We also observed that the SS-Kernel-k-means algorithm was not able to improve ARI as the number of pairwise constraints increased. We believe that the kernel-based algorithm is more sensitive to initialization besides not being able to escape from local optima. In contrast, the VNS was proved robust, making powerful use of a priori information.

7 Conclusion

This paper proposed a VNS heuristic for assessing the performance of the k-medoids model for semi-supervised clustering. Experiments showed that the new model had similar classification performance with respect to the typically used model based on k-means. The VNS algorithm was validated in a series of comparative experiments against CPLEX, presenting solutions very close to the optimal ones (never exceeding 0.02% in average) using much less CPU time. Moreover, the flexibility of the k-medoids model was tested regarding the addition of a dissimilarity matrix generated by a kernel function with distance metric learning. The VNS that combined the kernel trick with the explicit use of pairwise constraints presented the best accuracy performance among the algorithms compared.

References

[1] Hansen, P., Jaumard, B.: Cluster analysis and mathematical programming. Mathematical programming 79(1–3) (1997) 191–215

[2] Delattre, M., Hansen, P.: Bicriterion cluster analysis. IEEE TPAMI 4 (1980) 277–291

[3] Aggarwal, C.C., Reddy, C.K.: Data Clustering: Algorithms and Applications. Chapman & Hall/CRC (2013)

[4] Basu, S., Davidson, I., Wagstaff, K.: Constrained Clustering: Advances in Algorithms, Theory, and Applications. 1 edn. Chapman & Hall/CRC (2008)

[5] Hansen, P., Mladenović, N.: Variable neighborhood search: Principles and applications. EJOR 130(3) (2001) 449–467

[6] Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: ICML. (2001) 577–584

[7] Bekkerman, R., Sahami, M.: Semi-supervised clustering using combinatorial MRFs. In: ICML. (2006)

[8] Yan, B., Domeniconi, C.: An adaptive kernel method for semi-supervised clustering. In: ECML, Springer (2006) 521–532

[9] Law, M.H.C., Topchy, A., Jain, A.K.: Model-based Clustering with Probabilistic Constraints. In: SIAM-SDM. (2005) 641–645

[10] Ruiz, C., Spiliopoulou, M., Menasalvas, E.: Density-based semi-supervised clustering. Data Mining and Knowledge Discovery 21(3) (2010) 345–370

[11] Xia, Y.: A global optimization method for semi-supervised clustering. Data Mining and Knowledge Discovery 18(2) (2009) 214–256

[12] Tuy, H.: Concave programming under linear constraints. Soviet Math 5 (1964) 1437–1440

[13] Kulis, B., Basu, S., Dhillon, I., Mooney, R.: Semi-supervised graph clustering: a kernel approach. Machine learning 74(1) (2009) 1–22

[14] Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Comp. 10(5) (1998) 1299–1319

[15] Christofides, N.: Graph Theory: An Algorithmic Approach (Computer Science and Applied Mathematics). Academic Press, Inc., Orlando, FL, USA (1975)

[16] Steinley, D.: K-medoids and Other Criteria for Crisp Clustering. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. In: Handbook of cluster analysis. CRC Press (2015)

[17] Kaufman, L., Rousseeuw, P.J.: Partitioning around medoids (program PAM). Finding groups in data: An introduction to cluster analysis (1990) 68–125

[18] Teitz, M.B., Bart, P.: Heuristic methods for estimating the generalized vertex median of a weighted graph. Oper. Res. 16(5) (October 1968) 955–961

[19] Whitaker, R.: A fast algorithm for the greedy interchange for large-scale clustering and median location problems. INFOR 21(2) (1983) 95–108

[20] Hansen, P., Mladenović, N.: Variable neighborhood search for the p-median. Location Science 5(4) (1997) 207–226

[21] Resende, M.G.C., Werneck, R.F.: A fast swap-based local search procedure for location problems. ANOR 150(1) (2007) 205–230

[22] Davidson, I., Ravi, S.: Clustering with constraints: Feasibility issues and the k-means algorithm. In: SIAM-SDM. (2005) 138–149

[23] Mladenovi, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research 24(11) (1997) 1097–1100

[24] R. Costa, L., Aloise, D., Mladenović, N.: Less is more: basic variable neighborhood search heuristic for balanced minimum sum-of-squares clustering. Information Sciences 415 (2017) 247–253

[25] Hansen, P., Ruiz, M., Aloise, D.: A vns heuristic for escaping local extrema entrapment in normalized cut clustering. Pattern Recog. 45(12) (December 2012) 4337–4345

[26] Hansen, P., Mladenović, N.: J-means: a new local search heuristic for minimum sum of squares clustering. Pattern Recog. 34(2) (2001) 405–413

[27] Santi, É., Aloise, D., Blanchard, S.J.: A model for clustering data from heterogeneous dissimilarities. EJOR 253(3) (2016) 659–672

[28] Kleinberg, J.: An impossibility theorem for clustering. Advances in neural information processing systems (2003) 463–470

[29] Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification 2(1) (1985) 193–218

[30] Lichman, M.: UCI machine learning repository (2013)

[31] Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: Proceedings of the Twenty-first International Conference on Machine Learning. ICML '04, New York, NY, USA, ACM (2004) 81–88