Les Cahiers du GERAD ISSN: 0711–2440

A variable neighborhood search heuristic for semi-supervised minimum sum-of-squares clustering

M. H. Midingoyi, A. Meneses, D. Aloise

G–2026–09

March 2026

A variable neighborhood search heuristic for semi-supervised minimum sum-of-squares clustering

Mahuton Hugues Midingoyi [a]

André Meneses [a]

Daniel Aloise [a]

[a] *GERAD & Département de génie informatique et génie logiciel, Polytechnique Montréal, Montréal (Qc), Canada, H3T 1J4*

daniel.aloise@gerad.ca

March 2026
Les Cahiers du GERAD
G–2026–09

Abstract : Semi-supervised clustering is a learning approach that primarily relies on unlabeled data but incorporates some prior information to improve the clustering results. Among various clustering objectives, the minimum sum-of-squares clustering (MSSC) is widely used to partition data by minimizing intra-cluster variances. In our work, we propose a Variable Neighborhood Search (VNS) heuristic for semi-supervised MSSC, where prior information is given in the form of pairwise must-link and cannot-link constraints. Our approach reformulates the optimization problem by representing must-link constraints through the construction of super-points, which implicitly satisfy these constraints, while cannot-link constraints are incorporated as penalties in the objective function. Computational experiments indicate that, in the majority of tested cases, our proposed VNS heuristic outperforms the solutions obtained by the state-of-the-art heuristic algorithm found in the literature within the same computational time.

Keywords : Semi-supervised clustering; pairwise constraints; variable neighborhood search

Acknowledgements: The authors gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC) (grant number 2023–04466). The authors have no competing interests to declare that are relevant to the content of this article.

1 Introduction

The Minimum Sum-of-Squares Clustering (MSSC) problem is central in data analysis. It consists of partitioning a set of data points into a fixed number of clusters by minimizing intra-cluster variances. The problem is NP-hard in general dimension for $k \geq 2$ [3], being the k-means algorithm [13] the most popular heuristic to approach the problem.

Clustering is conventionally an unsupervised learning process. Without guidance from supervision, clustering algorithms may produce results that do not reflect the underlying structure of the data. Consequently, researchers have explored incorporating prior knowledge into the learning process to improve clustering quality. This approach is commonly referred to as semi-supervised clustering or constrained clustering.

Clustering problems can incorporate various forms of prior knowledge (see Cai et al. [8] for a detailed review). A common example is pairwise constraints, which include must-link and cannot-link constraints. Must-link constraints specify pairs of points that must belong to the same cluster, while cannot-link constraints specify pairs that must belong to different clusters.

Formally, given a set $P = \{p_1, p_2, \dots p_n\}$ of n data points in $\mathbb{R}^d$, let

$$ML = \{(i,j) \mid p_i \text{ and } p_j \text{ must be assigned to the same cluster}\},$$

$$CL = \{(i,j) \mid p_i \text{ and } p_j \text{ must be assigned to different clusters}\}.$$

denote the sets of must-link and cannot-link pairs, respectively. Then, the semi-supervised MSSC by pairwise constraints (MSSC) can be formulated as the following mixed-integer nonlinear programming (MINLP) model:

$$\min_{x} \quad \sum_{\ell=1}^{k} \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{i\ell} x_{j\ell} \|p_i - p_j\|_2^2}{\sum_{i=1}^{n} x_{i\ell}} \tag{1a}$$

$$\text{s.t.} \quad \sum_{\ell=1}^{k} x_{i\ell} = 1, \qquad \forall i = 1, \dots, n, \tag{1b}$$

$$x_{i\ell} = x_{j\ell}, \qquad \forall \ell = 1, \dots, k, \ \forall (i,j) \in ML, \tag{1c}$$

$$x_{i\ell} + x_{j\ell} \leq 1, \qquad \forall \ell = 1, \dots, k, \ \forall (i,j) \in CL, \tag{1d}$$

$$x_{i\ell} \in \{0,1\}, \qquad \forall i = 1, \dots, n, \ \forall \ell = 1, \dots, k, \tag{1e}$$

where the binary variables $x_{i\ell}$ indicate the assignment of point p_i to cluster ℓ, with $x_{i\ell} = 1$ meaning that p_i belongs to cluster ℓ, and $x_{i\ell} = 0$ otherwise. Constraints (1b) force each data point to be assigned exactly to one of the k clusters, whereas constraints (1c) and (1d).

In this paper, we propose an efficient Variable Neighborhood Search (VNS) [20] heuristic for the semi-supervised MSSC problem by pairwise constraints. Our approach is based on reformulating (1) so that must-link constraints are satisfied implicitly through the construction of super-points, while cannot-link constraints are incorporated as penalty terms in the objective function. Our main contributions can be summarized as follows:

- We propose a new formulation of the semi-supervised MSSC problem in which must-link constraints are implicitly satisfied through the creation of super-points. This transformation significantly reduces the dimensionality of the problem. The cannot-link constraints are incorporated as penalty terms in the objective function, leading to a penalized optimization model that maintains flexibility while encouraging constraint satisfaction.
- We develop a specialized Variable Neighborhood Search (VNS) heuristic tailored to the penalized semi-supervised MSSC formulation. The method systematically explores multiple neighborhood structures and applies an efficient local search procedure based on incremental cost updates, allowing for rapid move evaluations.

- We conduct extensive experiments on benchmark data instances used in the literature. Our results show that the proposed VNS consistently achieves optimal or near-optimal solutions, outperforming or matching the state-of-the-art heuristic within the same computing times.

The remainder of the paper is organized as follows. Section 2 presents a review of related works for solving the semi-supervised MSSC problem. Section 3 presents our methodology composed by two principal components: a reformulation of the problem described in Section 3.1, and our developed VNS method detailed in Section 3.2. Computational experiments are reported in Section 4, where our method is compared with the current state-of-the-art heuristic found in the literature. Finally, conclusions are given in section 5.

2 Related works

Semi-supervised MSSC by pairwise constraints has been studied by many authors in the literature. Based on different non-linear mixed-integer programming models, exact methods are developed to guarantee the optimality of the obtained solutions. Among them, we can enumerate the constraint programming approach proposed by Dao et al. [11], the column generation approach developed by Babaki et al. [4] and the branch-and-bound approach of Guns et al. [16]. A Branch-and-Cut algorithm proposed by Piccialli et al. [21] for the semi-supervised MSSC by pairwise constraints is considered to be the state-of-the-art method. This algorithm enables the resolution of problem instances with up to 800 data points in high dimensions, and for 8 clusters within a reasonable running time.

Due to the large number of data points in practice, heuristic algorithms have been designed for the problem, such that local or near optimal solutions are obtained. To the best of our knowledge, the first work to propose a heuristic (Cop-KMeans) for the problem was presented in [25] where the authors modified the classical k-means algorithm for (unsupervised) MSSC in order to make use of prior information. Later, other heuristic methods proposed enhancements to the Cop-Kmeans heuristic (e.g. [18, 23, 24, 27]).

Using a Lagrangian framework, Ganji et al. [14] address the semi-supervised MSSC problem by aggregating points connected by must-link constraints into super-points, thereby ensuring that all must-link constraints are satisfied. Their algorithm is based on an assignment approach and, in each iteration, includes a procedure that moves points between clusters with the goal of minimizing violations of cannot-link constraints. Regarding metaheuristics for the semi-supervised MSSC by pairwise constraints, González-Almagro et al. [15] proposed a dual Iterative Local Search (ILS), which introduces diversity to basic ILS framework.

More recently, Baumann et al. [6] proposed a heuristic called PCCC for semi-supervised MSSC, which allows pairwise constraints to be treated as either *hard*, i.e., requiring strict satisfaction, or *soft*, i.e., permitting violations that incur a penalty. The algorithm iteratively assigns data points to cluster centers while respecting the pairwise constraints and updates the cluster centers at each iteration. The assignment step of data points to centers is performed by solving a binary program. PCCC is capable of producing optimal or near-optimal solutions for large benchmark instances and is considered state-of-the-art for the problem.

In this work, we choose to implement a VNS heuristic for the problem, given its previous success in various other clustering problems found in the literature [7, 9, 10, 17, 26]. In particular, our method reformulates the semi-supervised MSSC with pairwise constraints (1) as an unconstrained problem, where must-link constraints are enforced through variable elimination, and cannot-link constraints are incorporated via large penalty coefficients in the objective function. The details of our approach are presented in the next section.

3 Methodology

The proposed methodology consists of two principal components. The first involves the reformulation of model (1) as an equivalent MINLP without explicit pairwise constraints, whereas the second implements a variable neighborhood search heuristic to address the problem.

3.1 Problem reformulation

Our problem reformulation is structured around two main steps: problem size reduction and penalization, which respectively address the treatment of must-link and cannot-link constraints.

3.1.1 Problem size reduction

The must-link constraints enable a reduction in the number of variables in (1), thereby eliminating constraints (1c) from the formulation.

More in detail, we construct an undirected graph from the obtained set of must-link constraints where each node is a data point, and there is an edge between two nodes if the corresponding data points are involved in a must-link constraint, i.e., belong to the same cluster. Next, we compute the transitive closure of the graph leading to a partition $B_1, \dots, B_{\bar{n}}$ of P into $\bar{n} < n$ components for ML non-empty. We represent these components as super-points $\{\bar{p}_i\}_{i=1}^{\bar{n}}$, where each super-points $\bar{p}_i$ is associated to a weight w_i equal to the number of points in the component. These super-points can be used to reconstruct the original data points in P, and conversely, if the set ML is known.

To perform data clustering with these super-points, we define a dissimilarity measure D_{ij} between the super-points $\bar{p}_i$ and $\bar{p}_j$ as follows:

$$D_{ij}^2 = \begin{cases} \sum\limits_{(s,t) \in B_i \times B_j} \|p_s - p_t\|^2, & \text{if } i \neq j, \\ \sum\limits_{\substack{s,t \in B_i \\ s<t}} \|p_s - p_t\|^2, & \text{if } i = j. \end{cases} \tag{2}$$

Therefore, problem (1) can be reformulated as:

$$\min_{\bar{x}} \quad \sum_{\ell=1}^{k} \frac{\sum_{i=1}^{\bar{n}} \sum_{j=i}^{\bar{n}} D_{ij}^2 \bar{x}_{i\ell} \bar{x}_{j\ell}}{\sum_{i=1}^{\bar{n}} w_i \bar{x}_{i\ell}} \tag{3a}$$

$$\text{s.t.} \quad \sum_{\ell=1}^{k} \bar{x}_{i\ell} = 1, \qquad \forall i = 1, \dots, \bar{n}, \tag{3b}$$

$$\bar{x}_{i\ell} + \bar{x}_{j\ell} \leq 1, \quad \forall \ell = 1, \dots, k, \ \forall (i,j) \in \overline{CL}, \tag{3c}$$

$$\bar{x}_{i\ell} \in \{0,1\}, \qquad \forall i = 1, \dots, \bar{n}, \ \forall \ell = 1, \dots, k, \tag{3d}$$

where, $\bar{x}_{i\ell} = 1$ indicates that super-point $\bar{p}_i$ is assigned to cluster ℓ, $\bar{x}_{i\ell} = 0$ otherwise. Note that constraints (1c) no longer appears explicitly in model (3), as it is now implicitly satisfied.

The idea of using must-link constraints to reduce the dimensionality of clustering problems is not new in the literature and has been used by diverse methods that optimize the MSSC criterion under pairwise constraints (see, e.g., [12, 14, 16, 21, 22]).

3.1.2 Penalization

The set of cannot-link constraints must be adjusted for the super-points. We denote by $\overline{CL}$ the set of index pairs of super-points linked by cannot-link constraints. The set $\overline{CL}$ consists of all pairs of super-point indices (i,j) such that there exists (s,t) in CL with $s \in B_i$ and $t \in B_j$.

We opted to incorporate the cannot-link constraints as a penalty in the objective function, allowing us to account for them without explicitly including them in the problem formulation. This approach gives our heuristic method greater flexibility by eliminating the need to check constraint satisfaction during the search. Thus, each cannot-link constraint violation is penalized in the objective function using the following penalized metric $\bar{D}$, where:

$$\bar{D}_{ij} = \begin{cases} \infty & \text{if } (i,j) \in \overline{CL}, \\ D_{ij} & \text{otherwise.} \end{cases} \tag{4}$$

In practical implementation, the infinity value is replaced by $M = (n-1) \times d_{max}$, where d_{max} corresponds to the maximum distance between a pair of points in the dataset. This choice of M ensures that it exceeds the cost of any feasible solution of the problem.

Finally, the following reformulation is obtained for the semi-supervised MSSC by pairwise constraints:

$$\min_{\bar{x}} \quad \sum_{\ell=1}^{k} \frac{\sum_{i=1}^{\bar{n}} \sum_{j=i}^{\bar{n}} \bar{D}_{ij}^2 \bar{x}_{i\ell} \bar{x}_{j\ell}}{\sum_{i=1}^{\bar{n}} w_i \bar{x}_{i\ell}} \tag{5a}$$

$$\text{s.t.} \quad \sum_{\ell=1}^{k} \bar{x}_{i\ell} = 1, \quad \forall i = 1, \dots, \bar{n}, \tag{5b}$$

$$\bar{x}_{i\ell} \in \{0,1\}, \quad \forall i = 1, \dots, \bar{n}, \ \forall \ell = 1, \dots, k \tag{5c}$$

We can notice that pairwise constraints (1c) and (1d) are not explicitly enforced in (5), which is the underlying optimization problem approached by our VNS heuristic explained in the next section.

3.2 VNS algorithm for pairwise constrained MSSC

In this section, we present the key aspects of our VNS heuristic for semi-supervised MSSC by pairwise constraints. Our VNS leverages multiple neighborhood structures and a local search to explore the solution space efficiently.

VNS is a metaheuristic for combinatorial optimization that systematically changes neighborhoods to escape local optima and enhance global exploration [20]. It has been successfully applied to several clustering problems [1, 2, 7, 10, 17, 22].

Starting from an initial solution x, the algorithm iteratively performs the following steps. First, a solution x' is generated by perturbing x within the current neighborhood (*shaking*). A local search is then applied to x', yielding a locally optimal solution x''. If x'' improves upon x, the incumbent solution is updated ($x \leftarrow x''$) and the search restarts from the first neighborhood; otherwise, the algorithm proceeds to the next neighborhood in the sequence. The process continues until a stopping criterion, such as a time limit or absence of improvement, is met.

3.2.1 Neighborhood evaluation

Our VNS heuristic works in its shaking by relocating t points from their current cluster to a different one. Accordingly, the neighborhood structure $\mathcal{N}_t$ of a solution consists of all configurations obtained by moving t points to alternative clusters.

Consider super-points partitioned into k clusters $C_1, C_2, \dots, C_k$, forming a solution $\mathcal{C}$. The contribution of cluster C_ℓ to the objective function of (5) can be expressed as:

$$\text{Cost}(C_\ell) = \frac{\sum_{i=1}^{\bar{n}} \sum_{j=i}^{\bar{n}} \bar{D}_{ij}^2 \bar{x}_{i\ell} \bar{x}_{j\ell}}{\sum_{i=1}^{\bar{n}} w_i \bar{x}_{i\ell}} \tag{6}$$

or equivalently by:

$$\mathrm{Cost}(C_\ell) = \frac{\bar{\mathcal{D}}_\ell}{s_\ell} \tag{7}$$

where $s_\ell = \sum_{i:\bar{p}_i \in C_\ell} w_i$ denotes the number of data points in cluster C_ℓ, and $\bar{\mathcal{D}}_\ell = \sum_{i<j:\bar{p}_i,\bar{p}_j \in C_\ell} \bar{D}_{ij}^2$ corresponds to the sum of the dissimilarities between the super-points in cluster C_ℓ.

Note that the numerator of the cluster cost is a quadratic term. This observation allows for an efficient computation of the cost variation in (6) when adding or removing a super-point to or from a cluster in the partition. The variation γ_ℓ^i in the numerator of the cost of C_ℓ resulting from moving the super-point $\bar{p}_i$ to it is given by:

$$\gamma_\ell^i = \sum_{j:\bar{p}_j \in C_\ell} \bar{D}_{ij}^2, \tag{8}$$

while the change in the denominator corresponds to the weight of the point being added or removed. Thus, the cost of adding super-point $\bar{p}_i$ to cluster C_ℓ is computed as:

$$\frac{\bar{\mathcal{D}}_\ell + \gamma_\ell^i}{s_\ell + w_i}, \tag{9}$$

while the cost of removing a super-point $\bar{p}_i$ from cluster C_r is computed as:

$$\frac{\bar{\mathcal{D}}_r - \gamma_r^i}{s_r - w_i}. \tag{10}$$

Once a super-point $\bar{p}_i$ is moved in solution $\mathcal{C}$ from cluster C_r to another cluster C_ℓ, all variation values γ_ℓ^j,γ_r^j for every $j = 1, \ldots, \bar{n}, j \neq i$ are directly updated as:

$$\gamma_r^j \leftarrow \gamma_r^j - \bar{D}_{ij}^2 \text{ and } \gamma_\ell^j \leftarrow \gamma_\ell^j + \bar{D}_{ij}^2. \tag{11}$$

Consequently, move evaluations in our neighborhood are performed in constant time, while updates of auxiliary structures γ are performed in $O(\bar{n})$.

3.2.2 Local search

Our local search is based on analyzing the possibilities of moving one point from one cluster to another, i.e., $\mathcal{N}_1$. Since identifying the best improving move can be computationally expensive, our local search adopts a compromise between the best- and first-improvement strategies. It evaluates each pair of clusters to find the best improving move. Let us denote:

- $\mathcal{K}$: the set of all combinations of two distinct cluster indices, defined as:

$$\mathcal{K} = \{(i,j) \mid 1 \leq i < j \leq k\}; \tag{12}$$

- $\mathcal{K}_l$: the subset of all pairs in $\mathcal{K}$ that include index l, for $l = 1, \ldots, k$. It is given by:

$$\mathcal{K}_l = \{(i,j) \mid (i,l) \in \mathcal{K} \vee (l,j) \in \mathcal{K}\}. \tag{13}$$

The pseudo-code presented in Algorithm 1 describes our local search procedure.

Algorithm 1 Local Search

```
1: Input: Initial solution $\mathcal{C}^0$
2: Output: Local solution $\mathcal{C}$
3: Initialize $\mathcal{C} \leftarrow \mathcal{C}^0$
4: Initialize next_$\mathcal{K} \leftarrow \mathcal{K}$
5: while True do
6:     current_$\mathcal{K} \leftarrow$ next_$\mathcal{K}$
7:     next_$\mathcal{K} \leftarrow \emptyset$
8:     for each $(i,j)$ in current_$\mathcal{K}$ do
9:         Find the best move of a super-point between the clusters $C_i$ and $C_j$
10:        if the found move between $C_i$ and $C_j$ improves the objective value then
11:            Apply the move and update $\mathcal{C}$
12:            next_$\mathcal{K} \leftarrow$ next_$\mathcal{K} \cup \mathcal{K}_i \cup \mathcal{K}_j$
13:        end if
14:    end for
15:    if next_$\mathcal{K}$ is empty then
16:        return the solution $\mathcal{C}$
17:    end if
18: end while
```

Algorithm 1 starts with a current solution $\mathcal{C}$, initialized from an initial solution $\mathcal{C}^0$ (line 3). The algorithm initializes a set `next_`$\mathcal{K}$ at line 4 containing all possible pairs of cluster indices for exploration. Then, the algorithm starts an iterative loop (lines 5–18) that continues as long as improving moves of super-points can be found. At each iteration, the set `current_`$\mathcal{K}$ receives the cluster index pairs to be tested from the previous iteration (line 6). An initially empty set `next_`$\mathcal{K}$ is prepared then in line 7 to store the pairs to be reconsidered in the next iteration. In the sequel, each pair (i,j) of clusters in `current_`$\mathcal{K}$ is examined in lines 8–14. The best potential move of a super-point in clusters C_i or C_j is found in 9. If several moves have the best objective value, only one move is considered among them. If it is detected that the considered super-point move between clusters C_i and C_j improve the objective value in line 10, the move is performed in line 11. Updating the current solution involves updating the size and cost of each affected cluster, as well as the variation values, using (8)–(11). Additionally, all cluster index pairs involving one of the two modified clusters are added to `next_`$\mathcal{K}$ in line 12 to be reconsidered in the next iteration. This mechanism avoids a full exploration of all cluster pairs at each iteration, focusing the search only on solution components directly affected by the solution modifications. The main loop continues until no cluster pairs are left to explore in the next iteration (lines 15–17). In that case, a local minimum is returned in line 16.

3.3 Feasibility step

The VNS process does not guarantee that the final solution satisfies all cannot-link constraints. Therefore, a post-processing step is applied whenever the method produces an infeasible solution for the semi-supervised MSSC.

Let $\mu_1, \dots, \mu_k$ denote the centroids of the final solution $\mathcal{C} = \{C_1, \dots, C_k\}$ returned by the VNS heuristic. The feasibility step consists in solving the following MIP:

$$\min_{\bar{x}} \quad \sum_{\ell=1}^{k} \sum_{i=1}^{\bar{n}} \bar{x}_{i\ell} \|\bar{z}_i - \mu_\ell\|^2 \tag{14a}$$

$$\text{s.t.} \quad \sum_{\ell=1}^{k} \bar{x}_{i\ell} = 1, \qquad \forall i = 1, \dots, n, \tag{14b}$$

$$\bar{x}_{i\ell} + \bar{x}_{j\ell} \le 1, \quad \forall \ell = 1, \dots, k, \ \forall (i,j) \in \overline{CL}, \tag{14c}$$

$$\bar{x}_{i\ell} \in \{0,1\}, \qquad \forall i = 1, \dots, n, \ \forall \ell = 1, \dots, k. \tag{14d}$$

where $\bar{z}_i$ is the centroid of the points merged at super-point $\bar{p}_i$, for $i = 1, \dots, \bar{n}$.

4 Computational experiments

To evaluate the performance of our VNS heuristic,[1] we used the COL1 benchmark suite of 25 datasets from [6]. These datasets contain between 47 and 846 samples, span 2 to 90 dimensions, and include 2 to 15 clusters. The corresponding sets of pairwise constraints were also obtained from [6], with each set defining a distinct instance of the semi-supervised MSSC problem. For each dataset, the authors constructed four different configurations of pairwise constraints, resulting in a total of $25 \times 4 = 100$ data instances.

The computational experiments were performed on a Intel 2.5Ghz processor with 24GB of RAM memory and compiled by gcc 11.4. Both algorithms were executed 10 times for the same amount of CPU time (determined by the average convergence time of the PCCC implementation available in [5]). We note that, although PCCC is implemented in Python, it primarily relies on Gurobi (version 12.0.3) to iteratively solve its auxiliary assignment steps.

The ten executions of the VNS and PCCC algorithms are paired, as each begins from the same initial solution generated by the classical k-means algorithm [19]. This initialization is performed on the super-points described in Section 3.1, where each super-point is represented by the average coordinates of its component data points, thereby respecting the must-link constraints. The VNS parameters were set to $t_{max} = \min\{200, 0.75 \times \bar{n}\}$ with step increments of $\min\{20, 0.1 \times t_{max}\}$.

We present the results for the COL1 data instances in two separate tables. Table 1 reports the cases for which optimal solutions are known using the algorithm of [21]. For these instances, we provide the average solution gaps relative to the corresponding optimal values. The table also includes, for each data instance, the number of original data points (n), the number of clusters (k), the number of must-link (ML) and cannot-link (CL) constraints, along with the average CPU time (in seconds) allocated for each algorithm run.

Table 1: Comparison between our VNS and PCCC in COL1 instances with known optimal values.

Dataset	n	k	ML	CL	VNS		PCCC		time
					cost	gap (%)	cost	gap (%)	
Appendicitis	106	2	13	2	491.80	**0.00**	493.91	0.43	0.09
			39	16	544.18	**0.55**	545.14	0.73	0.04
			71	49	612.93	**0.00**	612.93	**0.00**	0.04
			164	67	612.93	**0.00**	612.93	**0.00**	0.01
Breast Cancer	5300	2	216	190	12139.45	0.46	12084.17	**0.00**	0.20
			876	720	12185.03	**0.00**	12185.03	**0.00**	0.02
Bupa	345	2	79	74	1853.13	**0.17**	1853.79	0.21	0.21
			323	272	2041.64	**0.00**	2041.93	0.01	0.04
Circles	300	2	50	55	473.20	**0.01**	480.76	1.61	0.23
			208	227	598.80	**0.00**	599.08	0.05	0.05
Ecoli	336	8	357	918	1027.69	**0.00**	1034.12	0.63	1.00
			609	1669	1111.78	**0.00**	1120.20	0.76	0.26
Glass	214	6	11	44	812.41	**0.00**	842.19	3.67	0.57
			139	389	1266.22	**0.00**	1314.49	3.81	0.66
			259	644	1409.89	**0.00**	1416.29	0.45	0.36
Haberman	306	2	76	44	799.45	**0.03**	802.68	0.43	0.24
			304	161	889.43	**0.00**	894.97	0.62	0.05
			634	401	891.42	**0.00**	891.42	**0.00**	0.02
Hayes-roth	160	3	12	16	446.67	**0.00**	453.33	1.49	0.22
			102	174	553.10	**0.00**	553.48	0.07	0.13

Continued on next page

[1]available in C++ at https://github.com/andre-meneses/Constraint_Cluster\protect\discretionary{\char\hyphenchar\font}{}{}ing_VNS

Dataset	n	k	ML	CL	VNS		PCCC		time
					cost	gap (%)	cost	gap (%)	
			177	319	551.07	**0.00**	552.83	0.32	0.08
Heart	270	2	41	50	3044.98	**0.00**	3118.20	2.40	0.30
			178	173	3085.12	**0.00**	3287.93	6.57	0.06
Ionosphere	351	2	92	61	10265.88	0.07	10260.34	**0.01**	0.24
			330	300	10944.90	**0.00**	10946.22	0.01	0.06
Iris	150	3	6	22	146.30	**0.00**	146.30	**0.00**	0.15
			25	80	146.15	**0.00**	146.28	0.09	0.18
			57	196	140.97	**0.00**	141.22	0.18	0.25
			94	341	145.01	**0.00**	145.11	0.07	0.14
Led7digit	500	10	267	2508	1449.42	**0.00**	1449.99	0.04	6.43
			460	4490	1511.03	**0.00**	1511.19	0.01	2.12
Monk-2	432	2	101	130	2389.91	1.32	2358.87	**0.00**	0.41
			473	473	2382.69	**0.00**	2382.69	**0.00**	0.06
Moons	300	2	55	50	283.74	**0.00**	283.75	**0.00**	0.16
			200	235	322.11	**0.00**	322.27	0.05	0.07
			494	496	322.93	**0.00**	322.93	**0.00**	0.02
Mov. Libras	360	15	6	147	10597.90	**0.00**	11016.47	3.95	2.21
Newthyroid	215	3	25	30	506.06	**0.00**	506.06	**0.00**	0.24
			108	123	536.00	**0.00**	536.31	0.06	0.18
			270	258	550.93	**0.00**	550.93	**0.00**	0.08
			449	454	550.36	**0.00**	550.76	0.07	0.05
Saheart	462	2	152	124	3795.75	1.04	3757.36	**0.02**	0.28
			595	486	3924.31	**0.00**	3926.08	0.05	0.04
Sonar	208	2	29	26	11289.27	**0.01**	11308.61	0.18	0.14
			100	110	11873.80	**0.00**	11873.92	**0.00**	0.10
			245	251	11962.93	**0.00**	11962.93	**0.00**	0.03
Soybean	47	4	0	3	367.14	**0.00**	372.77	1.53	0.05
			4	6	367.14	**0.00**	369.16	0.55	0.05
			6	22	367.14	**0.00**	367.14	**0.00**	0.07
			12	33	367.14	**0.00**	367.14	**0.00**	0.06
Spectfheart	267	2	56	35	10467.75	**0.00**	10472.92	0.05	0.22
			233	118	11210.50	**0.00**	11488.33	2.48	0.08
			543	277	11268.30	**0.00**	11268.30	**0.00**	0.03
Spiral	300	2	52	53	462.24	**0.15**	462.51	0.21	0.20
			224	211	558.25	**0.00**	558.92	0.12	0.05
Tae	151	3	8	20	478.85	**0.00**	490.89	2.51	0.22
			82	171	684.30	**0.00**	685.93	0.24	0.15
			151	314	711.83	**0.00**	711.83	**0.00**	0.07
Vehicle	846	4	874	2696	13303.38	0.40	13291.72	**0.31**	0.33
			1955	6046	13334.20	**0.00**	13334.20	**0.00**	0.06
Wine	178	3	8	28	1284.69	**0.00**	1284.87	0.01	0.13
			31	122	1285.13	**0.00**	1285.13	**0.00**	0.21
			70	281	1285.13	**0.00**	1285.13	**0.00**	0.22
			143	487	1290.66	**0.00**	1291.36	0.05	0.11
Zoo	101	7	7	8	538.71	**0.00**	555.24	3.07	0.18
			21	34	545.96	**0.00**	565.45	3.57	0.23
			29	91	551.82	**0.00**	567.44	2.83	0.27
			41	169	552.75	**0.00**	560.39	1.38	0.33

We observe from Table 1 that our VNS obtained the optimal solutions in 57 out of 68 instances (approximately 84%). In the remaining 11 instances, the observed gaps were very small, ranging from 0.01% to 1.32%, with an average of 0.38%, thereby confirming the high effectiveness and stability of the heuristic. For comparison, PCCC reached the optimal value in 21 out of 68 instances (approximately 31%), while the remaining cases showed small deviations with an average gap of approximately 1.02%.

Table 2 reports average solution values for the remaining COL1 data instances. We note that, for some instances, the number of super-points $\bar{n}$ was equal to the number of clusters k after processing the must-link constraints. Since these instances are trivially solved by assigning each super-point to a singleton cluster, we excluded them from our comparison with the PCCC algorithm.

Table 2: Comparison between VNS and PCCC in COL1 instances when optimal solution values are not known.

Dataset	n	k	ML	CL	VNS	PCCC	time
Breast Cancer	5300	2	1965	1690	**12214.60**	**12214.60**	0.01
Bupa	345	2	699	627	**2047.30**	**2047.30**	0.01
			1201	1145	**2047.30**	**2047.30**	0.01
Circles	300		502	488	**599.99**	**599.99**	0.01
Ecoli	336	8	30	106	**791.15**	826.26	1.97
			163	398	**944.88**	960.64	1.42
Glass	214	6	52	179	**1085.60**	1110.41	0.80
Haberman	306	2	1135	756	**891.42**	**891.42**	0.01
Hayes-roth	160	3	39	81	**513.81**	516.21	0.31
Heart	270	2	396	424	**3120.52**	3302.53	0.02
Ionosphere	351	2	732	646	**10971.50**	**10971.50**	0.02
Led7digit	500	10	25	275	**1139.61**	1183.66	1.54
			126	1099	**1352.09**	1381.96	4.58
Monk-2	432	2	979	1101	**2384.29**	**2384.29**	0.01
Moons	300	2	900	870	**322.93**	**322.93**	0.01
Movement Libras	360	15	27	603	**11508.30**	11746.60	5.95
			112	1319	**15216.75**	15548.58	10.48
			158	2398	**17956.65**	18306.36	19.39
Saheart	462	2	1292	1123	**3927.71**	**3927.71**	0.02
Sonar	208	2	436	425	**11962.93**	**11962.93**	0.01
Spectfheart	267	2	965	466	**11268.30**	**11268.30**	0.01
Spiral	300	2	487	503	**564.47**	**564.47**	0.01
			918	852	**564.47**	**564.47**	0.01
Tae	151	3	40	80	**594.48**	596.27	0.33
Vehicle	846	4	221	682	9957.60	**9879.45**	2.36

We note from Table 2 that, overall, our VNS consistently produced equal or lower average costs than PCCC, except for the Vehicle instance. Across all 25 instances, VNS achieved an average relative improvement of approximately 1.02% over PCCC. The largest relative improvement, 5.51%, occurred on the Heart dataset instance.

Finally, it is worth noting that our VNS consistently produced final solutions that satisfied all cannot-link constraints, except for one instance of the Vehicle dataset (ML=874, CL=2696). For this instance, the feasibility step described in Section 3.3 was performed. The time spent by Gurobi on solving the MIP (14) was 0.06 seconds in average.

5 Conclusions

In this work, we proposed a Variable Neighborhood Search (VNS) heuristic for the semi-supervised Minimum Sum-of-Squares Clustering (MSSC) problem with pairwise must-link and cannot-link constraints. The proposed method addresses the limitations of existing approaches by reformulating the problem to reduce its size and simplify constraint handling. Specifically, must-link relationships are implicitly satisfied through the construction of super-points, thereby reducing the dimensionality of the problem, while cannot-link relationships are incorporated into the objective function as penalty terms. This leads to a penalized optimization formulation in which cannot-link relationships are handled implicitly within the objective function, providing a more tractable structure for metaheuristic search.

The VNS heuristic was designed with multiple complementary neighborhood structures and an efficient local search mechanism that exploits incremental cost updates for constant-time move evaluation. Additionally, a feasibility restoration step based on mixed-integer programming was integrated to guarantee full satisfaction of cannot-link constraints in the final solution when necessary. Together, these design elements enable the algorithm to efficiently explore the search space while maintaining solution feasibility. Computational experiments on benchmark data instances demonstrated that the proposed VNS heuristic consistently delivers high-quality clustering results. Across 68 benchmark instances with known optimal solutions, the algorithm attained the optimum in approximately 84% of cases and showed very small average deviations (below 0.4%) on the remaining instances. Overall, when compared to the state-of-the-art PCCC algorithm, our method achieved comparable or superior performance.

References

[1] Abdulrahman Alguwaizani, Pierre Hansen, Nenad Mladenović, and Eric Ngai. Variable neighborhood search for harmonic means clustering. Applied Mathematical Modelling, 35(6):2688–2694, 2011.

[2] Daniel Aloise, Gilles Caporossi, Pierre Hansen, Leo Liberti, Sylvain Perron, and Manuel Ruiz. Modularity maximization in networks by variable neighborhood search. Graph Partitioning and Graph Clustering, 588(113), 2012.

[3] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. Machine learning, 75(2):245–248, 2009.

[4] Behrouz Babaki, Tias Guns, and Siegfried Nijssen. Constrained clustering using column generation. In Helmut Simonis, editor, Integration of AI and OR Techniques in Constraint Programming: 11th International Conference, CPAIOR 2014, volume 8451 of Lecture Notes in Computer Science, pages 438–454, Cham, 2014. Springer.

[5] P. Baumann and D.S. Hochbaum. An algorithm for clustering with confidence-based must-link and cannot-link constraints, 2024.

[6] Philipp Baumann and Dorit S Hochbaum. An algorithm for clustering with confidence-based must-link and cannot-link constraints. INFORMS Journal on Computing, 37(4):1044–1068, 2025.

[7] Jack Brimberg, Nenad Mladenović, Raca Todosijević, and Dragan Urošević. Solving the capacitated clustering problem with variable neighborhood search. Annals of Operations Research, 272(1):289–321, January 2019.

[8] Jianghui Cai, Lifang He, Xia Hu, Meng Jiang, and Philip S. Yu. A review on semi-supervised clustering. IEEE Transactions on Knowledge and Data Engineering, 33(6):2210–2229, 2021.

[9] Emilio Carrizosa, Nenad Mladenović, and Raca Todosijević. Variable neighborhood search for minimum sum-of-squares clustering on networks. European Journal of Operational Research, 230(2):356–363, 2013.

[10] Leandro R. Costa, Daniel Aloise, and Nenad Mladenović. Less is more: basic variable neighborhood search heuristic for balanced minimum sum-of-squares clustering. Information Sciences, 415:247–253, 2017.

[11] Thi-Bich-Hanh Dao, Khac-Chinh Duong, and Christel Vrain. A declarative framework for constrained clustering. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný, editors, Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, volume 8190 of Lecture Notes in Computer Science, pages 419–434, Berlin, Heidelberg, 2013. Springer.

[12] Olivier Du Merle, Pierre Hansen, Brigitte Jaumard, and Nenad Mladenovic. An interior point algorithm for minimum sum-of-squares clustering. SIAM Journal on Scientific Computing, 21(4):1485–1505, 1999.

[13] Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. biometrics, 21:768–769, 1965.

[14] Mehran Ganji, James Bailey, and Peter J. Stuckey. Lagrangian constrained clustering. In Proceedings of the 2016 SIAM International Conference on Data Mining (SDM), pages 288–296. SIAM, 2016.

[15] Germán González-Almagro, Julián Luengo, José-Ramón Cano, and Salvador García. Dils: Constrained clustering through dual iterative local search. Computers & Operations Research, 121:104979, 2020.

[16] T. Guns, T.-B. H. Dao, C. Vrain, and K.-D. C. Duong. Repetitive branch-and-bound using constraint programming for constrained minimum sum-of-squares clustering. Constraints, 21(3):364–392, 2016.

[17] Pierre Hansen, Manuel Ruiz, and Daniel Aloise. A vns heuristic for escaping local extrema entrapment in normalized cut clustering. Pattern Recognition, 45(12):4337–4345, 2012.

[18] Haichao Huang, Yong Cheng, and Ruilian Zhao. A semi-supervised clustering algorithm based on must-link set. In Changjie Tang, Charles X. Ling, Xiaofang Zhou, Nick J. Cercone, and Xue Li, editors, Advanced Data Mining and Applications – ADMA 2008, volume 5139 of Lecture Notes in Computer Science, pages 492–499. Springer, Berlin, Heidelberg, 2008.

[19] Stuart P. Lloyd. Least squares quantization in pcm. IEEE Transactions on Information Theory, 28(2):129–137, 1982.

[20] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. Computers & Operations Research, 24(11):1097–1100, 1997.

[21] Veronica Piccialli, Anna Russo, and Antonio M. Sudoso. An exact algorithm for semi-supervised minimum sum-of-squares clustering. Computers & Operations Research, 147:105958, 2022.

[22] Rodrigo Randel, Daniel Aloise, Nenad Mladenović, and Pierre Hansen. On the k-medoids model for semi-supervised clustering. In International Conference on Variable Neighborhood Search, pages 13–27. Springer, 2018.

[23] Tonny Rutayisire, Yan Yang, Chao Lin, and Jinyuan Zhang. A modified cop-kmeans algorithm based on sequenced cannot-link set. In JingTao Yao, Sheela Ramanna, Guoyin Wang, and Zbigniew Suraj, editors, Rough Sets and Knowledge Technology — 6th International Conference, RSKT 2011, Banff, Canada, October 9-12, 2011, Proceedings, volume 6954 of Lecture Notes in Computer Science, pages 217–225. Springer, Berlin, Heidelberg, 2011.

[24] Wei Tan, Yan Yang, and Tien Li. An improved cop-kmeans algorithm for solving constraint violation. In Computational Intelligence: Foundations and Applications, volume 4 of World Scientific Proceedings Series on Computer Engineering and Information Science, pages 690–696. World Scientific Publishing, 2010.

[25] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In Proc. 18th Int. Conf. Mach. Learn. (ICML), pages 577–584, San Francisco, CA, 2001. Morgan Kaufmann.

[26] Yiyong Xiao, Changhao Huang, Jiaoying Huang, Ikou Kaku, and Yuchun Xu. Optimal mathematical programming and variable neighborhood search for k-modes categorical data clustering. Pattern Recognition, 90:183–195, June 2019.

[27] Yan Yang, Wei Tan, Tianrui Li, and Da Ruan. Consensus clustering based on constrained self-organizing map and improved cop-kmeans ensemble in intelligent decision support systems. Knowledge-Based Systems, 32:101–115, 2012.