Journal of Classification
https://doi.org/10.1007/s00357-025-09528-z

ORIGINAL RESEARCH

Efficiently Solving Semi-supervised Clustering Problems Through Differential Evolution and Local Optimization

Pierluigi Mansueto[1] · **Fabio Schoen**[1]

Received: 29 November 2024 / Accepted: 16 November 2025

Abstract
We propose an innovative algorithm for clustering problems with additional constraints specifying that selected pairs must or must not belong to the same cluster. Few methods have been proposed for this relevant extension. We propose, for the first time according to our knowledge, a memetic optimization framework exploiting a specialized local search algorithm. A clever use of local searches coupled with an exploration phase based on differential evolution makes our algorithm extremely efficient and effective in detecting very good feasible clusters. Our approach originates from a framework proposed in the unsupervised setting and presents several innovative extensions, like the definition of exact and greedy procedures to assign points to clusters (a nontrivial step in this problem) and the possibility of temporarily generating infeasible solutions. A wide set of numerical experiments confirms that mixing differential evolution with local optimization leads to significantly superior results with respect to the current literature.

Keywords Semi-supervised clustering · Minimum sum-of-squares clustering · Memetic differential evolution · Global optimization

1 Introduction

Clustering is one of the most relevant statistical techniques, as reported in various specialized books and surveys (see, e.g., Han et al., 2022; Jain, 2010). In this paper, we do not wish to present in great detail the ideas of clustering, for which we refer the reader to the cited references. Here, we would simply like to recall that clustering is a (usually) unsupervised methodology aimed at grouping together data records based on suitable similarity and dissimilarity criteria, so that similar observations are grouped together, while samples in different clusters are sufficiently dissimilar from one another.

The easiest and by far best known clustering model is the Euclidean Minimum Sum-of-Squares one (MSSC). In this scenario, observations are numerical arrays, each lying in

✉ Pierluigi Mansueto
pierluigi.mansueto@unifi.it

Fabio Schoen
fabio.schoen@unifi.it

[1] Global Optimization Laboratory - Department of Information Engineering (DINFO), University of Florence, Via di Santa Marta, 3, Florence 50139, Italy

the d-th dimensional Euclidean space $\mathbb{R}^d$, while similarity is based on Euclidean distance. Clustering is performed by choosing a set of centers, one for each cluster, so that the sum of squared distances from each sample point to its closest center is minimized (Hansen & Jaumard, 1997). Later, we will give a formal description of this basic model. Here, it suffices to recall that finding a suitable clustering according to this (but also to many other) similarity criterion requires the solution of a large-scale non-convex global optimization problem, usually characterized by a large number of local optima which are not global (Jain, 2010).

The problem itself is, in general, NP-hard (Aloise et al., 2009) and, thus, extremely difficult to solve, even in an approximate way. For this reason, a widespread use of fast heuristic techniques can be found in the clustering literature. For the Euclidean MSSC problem, one of the best known and used techniques is K-means (Hartigan & Wong, 1979; MacQueen, 1965). Given an initial position of the cluster centers, K-means iterates the two following steps until a stopping condition is satisfied: (1) each data point is assigned to its closest cluster center; (2) each center is moved to the arithmetic mean position of the points assigned to its cluster. The loop is usually stopped when cluster center positions are not updated for two consecutive iterations. Although being quite easy to implement and efficient, K-means usually returns only a local optimum, which might be very far from the global one. Moreover, its performance highly depends on the execution initial settings. In order to overcome the limitations of the approach, multiple variants of K-means have been explored over the years (e.g., Arthur and Vassilvitskii, 2007; Likas et al., 2003).

Much recent literature has been devoted to heuristics and metaheuristics which try to improve on the quality of K-means, without excessively sacrificing CPU time. As members of this class, we can cite the classical metaheuristic frameworks, such as simulated annealing (Selim & Alsultan, 1991), tabu search (Al-Sultan, 1995), and evolutionary algorithms (Maulik & Bandyopadhyay, 2000; Sarkar et al., 1997), as well as the more recent heuristic and convex optimization methodologies, such as the ones presented by Ordin and Bagirov (2015); Bagirov et al. (2016). Among the mentioned methods, the memetic procedure HG-means (Gribel & Vidal, 2019) certainly stands out. The latter is a population-based metaheuristic inspired by the process of natural selection typical of the larger class of evolutionary algorithms: at each iteration, the algorithm maintains and updates a population of candidate solutions; the evolution of the latter takes place through the repeated application of specific genetic operators such as crossover, mutation and selection; at the end of the approach, the best generated solution is returned. Unlike the (evolutionary) genetic algorithms, HG-means also integrates K-means as a local refiner for the solutions at hand in order to avoid premature convergence. Inspired by the state-of-the-art performance of HG-means, novel memetic strategies have been proposed by Mansueto and Schoen (2021). Here, the authors adapted the differential evolution paradigm (Storn & Price, 1997) to the MSSC case, and similar to HG-means, they employed K-means as a local search procedure to refine the population; the outcome, called MDEClust, is a novel memetic framework for the MSSC clustering, which, according to an extensive set of computational tests, is competitive w.r.t. HG-means.

In this paper, starting from the memetic approach MDEClust, we develop an innovative algorithm specialized for an even more challenging problem, the semi-supervised clustering problem. Successful application of such problem can be found in different domains, such as gene clustering (Pensa & Boulicaut, 2008; Tran et al., 2021) and document clustering (Huang & Mitchell, 2006; Hu et al., 2008). Different from the standard model, here, some information on the clusters is available a priori, and the algorithms need to take it into consideration. In particular, in this work, we consider instance-level constraints, i.e., constraints in the form of either must-link or cannot-link requirements. A must-link constraint imposes that two

specific data points need to be assigned to the same cluster. On the other hand, a cannot-link constraint requires two points to be assigned to different clusters. Thus, the basic objective of the semi-supervised MSSC problem remains the same, i.e., that of choosing the location of K centers in $\mathbb{R}^d$ so that the sum of squared distances from each point to its relevant center is minimized. However, now must-link and cannot-link constraints have to be considered, and thus, the closest center is no more guaranteed to be the relevant one. It goes without saying that, if the unsupervised MSSC problem is computationally NP-hard, this variant is, in general, at least as hard.

1.1 Related Works

Over the years, K-means variants capable of handling must-link and cannot-link constraints have been proposed. Usually, these approaches are local, since they are designed to return, at most, a locally optimal solution satisfying all the constraints. The first most notable exemplar is COP-K-means (Wagstaff et al., 2011), which aims to assign each point to the nearest cluster center such that no constraint is violated. The approach is not equipped with any backtracking procedure; thus, if none of the remaining observations can be associated with a center without violating the constraints, the algorithm stops, failing to return a feasible solution for the problem. In order to address this issue, modified versions of COP-K-means have been proposed in the literature: some examples are the procedures presented by Tan et al. (2010); Rutayisire et al. (2011). Nghiem et al. (2020) employ an integer programming-based post-processing procedure in solutions returned by unconstrained algorithms (K-means included). Moreover, K-means-like procedures are proposed by Baumann (2020); Baumann and Hochbaum (2024), with assignment steps based on binary programming problems to enforce the solution to be feasible at each iteration.

In fact, the approach proposed by Baumann and Hochbaum (2024) may also belong to the class of local, soft-constrained algorithms—methodologies which penalize constraint violations aiming to find a local optimal solution in a smoother way. Other representatives of this class are the following: Basu et al. (2004) consider a modified version of MSSC where penalties for the violation of must-link and cannot-link constraints take place; Davidson and Ravi (2025) introduce a constraint violation based on the Euclidean distance; finally, Ganji et al. (2016) propose an iterative approach optimizing a Lagrange relaxation, considering increasing penalties only for cannot-link violations, while must-link constraints are used to aggregate data points together a priori.

As already stated above, the local approaches are meant to return a locally optimal solution satisfying all constraints, at most. Thus, in order to approximate as accurately as possible a global solution, such local approaches need to be run in a multi-start fashion, that is, several times with different initial conditions. Still, in general, this is not enough to reach state-of-the-art quality (see, for instance, Gribel and Vidal, 2019). Then, global methodologies – aiming to find (approximately) global optimal solutions – have been proposed. Although the literature is relatively young compared to that related to local approaches, we can divide this class into two macro-groups: the exact and the heuristic algorithms. The first ones theoretically guarantee a globally optimal solution, but in practice may require great computational cost and thus are suitable for small datasets or are stopped before convergence. For instance, Xia (2009) proposes a cutting-plane algorithm, extending one for unsupervised MSSC; Aloise et al. (2012a, b) propose column-generation approaches for instances with fewer than 200 data points; Dao et al. (2017) define an exact approach for constrained clustering based on constraint programming (CP); Guns et al. (2016) integrate a branch-and-bound method with a

CP solver incorporating must-link and cannot-link constraints; Piccialli et al. (2022) develop a branch-and-cut algorithm based on a semi-definite programming (SDP) relaxation of the MSSC discrete optimization model with a cutting-plane procedure for bound strengthening. On the other hand, the heuristic approaches do not guarantee global optimality, but are lighter and more suitable for larger datasets. To the best of our knowledge, the majority of global heuristic approaches to semi-supervised clustering fall under the category of soft-constrained metaheuristics. For example, González-Almagro et al. (2020) propose an iterative procedure based on genetic operators, while Gribel et al. (2022) introduce a probability-based extension of HG-means tailored for semi-supervised clustering.

Finally, the semi-supervised MSSC clustering problem has been studied by Liberti and Manca (2022), in which several MINLP formulations, both exact and approximate, are presented.

1.2 Purpose of the Research

Our contribution consists on the proposal, for the first time in the literature to the best of our knowledge, of a novel global (heuristic) memetic methodology for semi-supervised MSSC problems to find good quality solutions satisfying all the must-link and cannot-link constraints. The new approach, called S-MDEClust, presents the following innovative extensions:

1. Two novel (exact and greedy) assignment methodologies for the memetic paradigm are proposed, which take into account the presence of must-link and cannot-link constraints.
2. The new specialized greedy assignment algorithm might possibly return a temporary infeasible solution; the advantages resulting from this methodology are, on the one hand, the computational speed and, on the other hand, the fact that infeasibility might drive the algorithm towards the exploration of a wider solution space.
3. A new mutation operator is proposed in order to take into account the link constraints as well as the possibility of managing a temporary solution which possibly violates some of them.
4. The inclusion of the optimization paradigm proposed by Baumann (2020) in a memetic scheme.

The remainder of the paper is structured as follows. In Section 2, we recall the definition of the semi-supervised MSSC problem and the main characteristics of the memetic framework MDEClust. In Section 3, we describe our novel approach, highlighting the new extensions necessary to handle the additional complexities of semi-supervised problems. In Section 4, we show the quality of the proposal, comparing it with some state-of-the-art local and global approaches on a vast set of well-known and synthetic datasets; we remark that all the performed experiments are reproducible as the source codes of both our approach and the competitors are available online. Finally, in Section 5, we provide some concluding remarks.

2 Preliminaries

2.1 Problem Statement

Given a dataset $\mathcal{D} = \{p_1, p_2, \ldots, p_N\}$ where p_i is a d-dimensional sample, i.e., $p_i \in \mathbb{R}^d$, for all $i \in \{1, \ldots, N\}$, solving a clustering problem means the partitioning of such samples in K disjoint sets $C_1, C_2, \ldots, C_K$, where, for all $k \in \{1, \ldots, K\}$, the cluster C_k is identified by a

center $y_k \in \mathbb{R}^d$. We denote the set of the cluster centers as $\mathcal{Y} = \{y_1, \ldots, y_K\}$. In general, the partition should lead to the grouping of points similar to each other (based on a specifically chosen distance metric) and to the separation of dissimilar samples. To this end, the semi-supervised MSSC model consists in associating each point p_i to a center y_k such that: the must-link and cannot-link constraints, whose sets are denoted by $\mathcal{M}_\mathcal{L}, \mathcal{C}_\mathcal{L} \subseteq \{1, \ldots, N\} \times \{1, \ldots, N\}$, respectively, are satisfied; the sum of the squared Euclidean distances of each point to its associated center is minimized.

A possible formulation for the semi-supervised MSSC problem is thus the following:

$$\begin{aligned} \min \ & \sum_{i=1}^{N} \sum_{k=1}^{K} x_{ik} \| p_i - y_k \|^2 \\ \text{s.t. } & x \in \mathcal{X} \cap \mathcal{L} \\ & y_1, \ldots, y_K \in \mathbb{R}^d \end{aligned}, \tag{1}$$

where each binary variable x_{ik} indicates whether the point p_i is assigned to the cluster C_k,

$$\mathcal{X} = \left\{ \begin{array}{l} x \in \{0,1\}^{N \times K} \text{ s.t.} \\ \quad \displaystyle\sum_{k=1}^{K} x_{ik} = 1, \ \forall i \in \{1, \ldots, N\} \\ \quad \displaystyle\sum_{i=1}^{N} x_{ik} \geq 1, \ \forall k \in \{1, \ldots, K\} \end{array} \right\}, \tag{2}$$

$$\mathcal{L} = \left\{ \begin{array}{l} x \in \{0,1\}^{N \times K} \text{ s.t.} \\ \quad x_{ik} = x_{jk}, \quad \forall (i,j) \in \mathcal{M}_\mathcal{L}, \ \forall k \in \{1, \ldots, K\} \\ \quad x_{ik} + x_{jk} \leq 1, \ \forall (i,j) \in \mathcal{C}_\mathcal{L}, \ \forall k \in \{1, \ldots, K\} \end{array} \right\} \tag{3}$$

and $\| \cdot \|$ indicates the Euclidean norm in $\mathbb{R}^d$. The constraints defining $\mathcal{X}$ ensure that each point p_i is associated with one and only one center y_k (first set of constraints) and that each cluster C_k has at least one sample (second set of constraints). On the other hand, the set $\mathcal{L}$ is defined by the must-link and cannot-link constraints: for each $(i_m, j_m) \in \mathcal{M}_\mathcal{L}$, we require that p_{i_m} and p_{j_m} belong to the same cluster, while, for each $(i_c, j_c) \in \mathcal{C}_\mathcal{L}$, the points p_{i_c} and p_{j_c} should be assigned to different clusters.

It is well known that, similar to the unconstrained MSSC model (Aloise et al., 2009), the considered problem is NP-hard (Davidson & Ravi, 2007).

The solutions of the presented semi-supervised MSSC model present an easy-to-prove property, fundamental in the mechanisms of MSSC algorithms.

Lemma 1 *Let $(x^\star, y^\star)$ be an optimal solution of problem (1). Then, for each $k \in \{1, \ldots, K\}$, the point $y_k^\star$ is the centroid (or geometric center) of the k-th cluster:*

$$y_k^\star = \frac{1}{\sum_{i=1}^{N} x_{ik}^\star} \sum_{i=1}^{N} x_{ik}^\star p_i . \tag{4}$$

Lemma 1 also holds for the solutions of the unsupervised MSSC problem (Lloyd, 1982). Thus, given the point assignment in the form of $x^\star$, the computation of the cluster centers is the same for both the unconstrained and the semi-supervised MSSC scenarios. On the other

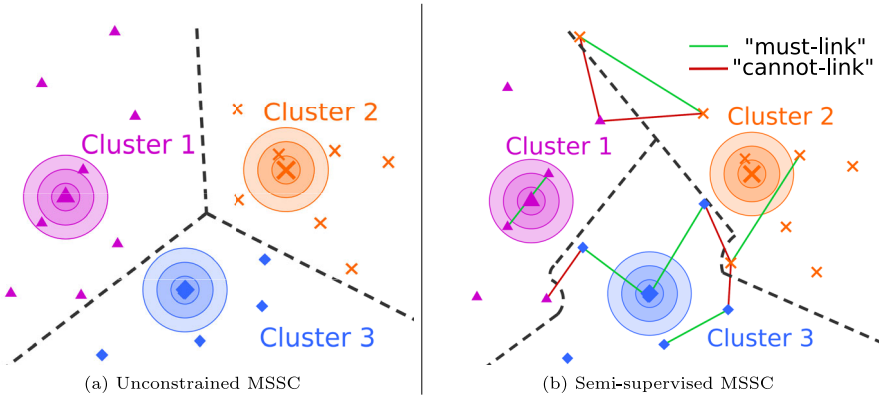
hand, the assignment step, i.e., the procedure by which we assign each point to a center having the coordinates $y^\star$, requires different effort. While in the unsupervised MSSC scenario, this step is indeed trivial to perform, that is, we assign each point to its closest cluster center. In the semi-supervised context, must-link and cannot-link constraints must be taken into account, and then, it is no more true that points are associated with their closest centroid. A graphical explanation of how the outcome of the assignment step can change depending on the considered scenario is shown in Fig. 1. As will be attested in the next sections, this difference usually leads to the need of solving an additional binary optimization problem, being then one of the main reasons for the higher consumption of computational resources by semi-supervised MSSC algorithms.

Finally, note that in problem (1), the number of clusters K is kept fixed. As discussed by Mansueto and Schoen (2021), this characteristic is beneficial in applications where K is known in advance, while it is detrimental in cases where K is not. In this last scenario, employing additional statistical techniques is necessary to find a suitable value for K before addressing the MSSC problem. Actually, in the semi-supervised case, the cannot-link constraints, if present, can give a hint about how large K should be at least. This a priori knowledge, combined with the mentioned statistical techniques, may make this preprocessing step easier to perform. We will not discuss this topic further, assuming in the following that K is fixed.

2.2 MDEClust for Unconstrained MSSC Problems

In this section, we report a brief description of MDEClust, the Memetic Differential Evolution (MDE) framework proposed by Mansueto and Schoen (2021) for unconstrained MSSC problems.

MDEClust initializes and keeps updated throughout the iterations a population of P MSSC solutions $\mathcal{P} = \{S_1, \ldots, S_P\}$. In particular, the initialization of the population is accomplished by executing P independent runs of the K-means algorithm (Lloyd, 1982), in each of which the initial centers are randomly selected among the points of the dataset $\mathcal{D}$ at hand. Then, at



(a) Unconstrained MSSC (b) Semi-supervised MSSC

Fig. 1 Two-dimensional examples of assignment step outcome with $N = 20$ points and $K = 3$ cluster centers. The dotted black lines indicate the separations between clusters, while the shaded areas just emphasize cluster centers. Note that these latter ones have not been relocated yet in order to mainly emphasize the assignment step

each iteration of MDEClust, the population is updated performing the following operations for each solution $S_s \in \mathcal{P}$, with $s \in \{1, \ldots, P\}$:

1. An offspring solution O_s is generated by the crossover operator (Section 2.2.1).
2. The (optional) mutation operator is performed, returning a modified offspring $\tilde{O}_s$ (Section 2.2.2).
3. K-means is executed as local search procedure to refine $\tilde{O}_s$.
4. The resulting solution O'_s is then compared with S_s in terms of MSSC objective function (see problem 1), and if it is better, it replaces S_s in the population.

Definition 1 Each solution $S \in \mathcal{P}$ is identified through the two following data structures.

1. Membership vector $\phi^S \in \mathbb{N}^N$: for all $i \in \{1, \ldots, N\}$, $\phi_i^S = k$, with $k \in \{1, \ldots, K\}$, if and only if $x_{ik} = 1$, i.e., the point p_i is associated to the cluster C_k.
2. Coordinate matrix $\psi^S = [y_1, y_2, \ldots, y_K]^\top \in \mathbb{R}^{K \times d}$: given $k \in \{1, \ldots, K\}$, the k-th row contains the coordinates of the k-th cluster center.

As remarked by Gribel and Vidal (2019); Mansueto and Schoen (2021), each of the two items is sufficient to completely characterize a solution. Indeed, having the membership vector ϕ^S, we can find each center as the centroid of its associated cluster (Lemma 1); on the other hand, we can derive ϕ^S by ψ^S performing an assignment step, i.e., assigning each point to its closest cluster center. Both operations can be performed in $O(NKd)$, albeit maintaining both may be preferred to reduce the CPU time at the expense of an increase in memory usage. In the remainder of the paper, we will use the following notation: a solution S will be denoted by $S = (\phi^S, \psi^S)$; the MSSC objective function value of a solution S will be indicated with

$$f(S) = \sum_{i=1}^{N} \left\| p_i - \psi^S_{\phi_i^S} \right\|^2.$$

The algorithm stops when one of the following stopping criteria is met, returning the best solution in the population in terms of the MSSC objective function:

1. In all, $N_{\max} > 0$ consecutive iterations have been performed without any best solution improvement.
2. The difference in terms of the MSSC objective function among the solutions in the population has fallen below a threshold, i.e.,

$$\sum_{s=1}^{P} \sum_{\bar{s}>s}^{P} |f(S_s) - f(S_{\bar{s}})| \leq \text{tol},$$

where tol > 0; in this scenario, we have that the population has collapsed to a single solution.

Other stopping criteria can also be possible.

In the next sections, we provide a brief description of the crossover and mutation operators employed in the MDEClust framework.

2.2.1 Crossover Operator

Given a solution $S_s \in \mathcal{P}$, with $s \in \{1, \ldots, P\}$, in MDEClust the crossover operator selects from the population three distinct random solutions S_1, S_2, S_3, all different from S_s, and it creates an offspring solution $O_s = (\phi^{O_s}, \psi^{O_s})$ combining them in the following way:

$$\psi^{O_s} = \psi^{S_1} + F(\psi^{S_2} - \psi^{S_3}), \tag{5}$$

with $F \in (0, 2)$. If $F \approx 0$, the cluster centers of the new offspring O_s are likely to be near to the ones of solution S_1, thus leading to a more intensive exploration of S_1 neighborhood; on the other hand, if $F \approx 2$, then the offspring will tend to be different from S_1, taking the exploration to new regions of the MSSC problem feasible set. The membership vector ϕ^{O_s} is then obtained performing an assignment step. Notice that, as it is standard in differential evolution approaches, S_s plays no role in the definition of the offspring solution; it will be considered only to compare function values and decide how to update the population.

Taking inspiration from Gribel and Vidal (2019), the linear combination is carefully designed based on center matching strategies. The necessity of a matching phase comes from the fact that, when dealing with operations that involve linear combinations of two or more clustering solutions (e.g., Eq. 5), any permutation of cluster centers gives rise to the same clustering solution, yet with a different analytic representation. Moreover, no ordering of the cluster centers is a priori provided. Thus, before performing any linear combination, a bi-partite matching strategy is employed. This strategy takes into account Euclidean distances between every pair of cluster centers in the two solutions, so that, before applying the linear combination, a re-labeling of cluster centers is implicitly performed. Thanks to this procedure, if two solutions, e.g., S_2 and S_3, are very similar to each other, their difference $\psi^{S_2} - \psi^{S_3}$ will be close to zero. Mansueto and Schoen (2021) have proposed two different center matching methodologies for MDEClust: an exact strategy based on the Hungarian algorithm (Kuhn, 1955), which is also employed by Gribel and Vidal (2019) and has a complexity of $O(K^3)$; a greedy and faster strategy, whose complexity is $O(K^2)$, more sensitive to the cluster centers permutation. For more information on these two methodologies, the reader is referred to Mansueto and Schoen (2021).

2.2.2 Mutation Operator

In general, the mutation operator aims to diversify the population, preventing a premature stop of the framework caused by a rapid collapse of the population into a single solution. In the MDEClust framework, the operator mainly consists of a random relocation of one center of the offspring solution O_s generated during the crossover phase. The modified solution $\tilde{O}_s$ will then be characterized by new features that are not inherited from either of the parents. The center relocation is performed through the following steps.

1. A center randomly selected with uniform probability is removed.
2. Each dataset point p_i, with $i \in \{1, \ldots, N\}$, is re-assigned to its closest center among the $K - 1$ remaining ones, returning a new temporary membership vector $\tilde{\phi}^{O_s}$.
3. The missing center is now restored as a point $p_{\bar{i}}$, with $\bar{i} \in \{1, \ldots, N\}$, randomly chosen by roulette wheel; the latter is a randomized operation applied in order to select potentially useful points, based, in this scenario, on the probability

$$P\left(\bar{i}\right) = \left(\frac{1-\alpha}{N}\right) + \left(\frac{\alpha \left\| p_{\bar{i}} - y_{\tilde{\phi}_{\bar{i}}^{O_s}} \right\|}{\sum_{j=1}^{N} \left\| p_j - y_{\tilde{\phi}_j^{O_s}} \right\|}\right). \tag{6}$$

   The value of $\alpha \in [0, 1]$ influences the selection of the new center: if $\alpha = 0$, every point has the same chances to be selected, thus potentially leading to the exploration of always new solutions; as α gets larger, the selection is biased towards points that are more distant from their closest centers.

The whole procedure requires $O(NKd)$ time to be performed.

3 A Novel Memetic Methodology for Semi-supervised MSSC

3.1 Algorithmic Scheme

In this section, we propose our novel memetic strategy, called S-MDEClust, for semi-supervised MSSC problems. We want to remark that, to the best of our knowledge, S-MDEClust represents the first attempt in the semi-supervised clustering literature to define a memetic approach designed to generate strictly feasible optimal solutions.

Algorithm 1 S-MDEClust framework.

Require: $\mathcal{D} = \{p_1, \ldots, p_N\} \subset \mathbb{R}^d$, $K \in \mathbb{N}_{>0}$ number of clusters, $\mathcal{M}_{\mathcal{L}}, \mathcal{C}_{\mathcal{L}} \subseteq \{1, \ldots, N\} \times \{1, \ldots, N\}$ sets of must-link and cannot-link constraints respectively, $P \in \mathbb{N}_{>0}$ population size, $N_{\max} \in \mathbb{N}_{>0}$, tol $\in \mathbb{R}_{>0}$.

1: Initialize population $\mathcal{P} = \{S_1, \ldots, S_P\}$
2: Let $S^\star \in \mathcal{P}$ be such that $f(S^\star) \leq f(S), \forall S \in \mathcal{P}$
3: Let $n_{\text{it}}^\star = 0$
4: **while** $(n_{\text{it}}^\star < N_{\max}) \wedge (\sum_{s=1}^{P} \sum_{\bar{s}>s}^{P} |f(S_s) - f(S_{\bar{s}})| > \text{tol})$ **do**
5: **for all** $S_s \in \mathcal{P}$ **do**
6: Randomly select $S_1, S_2, S_3 \in \mathcal{P}$, all different from each other and from S_s
7: Execute crossover with S_1, S_2, S_3 to generate an offspring O_s
8: (Optional) Execute semi-supervised mutation on O_s to get $\tilde{O}_s$
9: Apply constrained local search to $\tilde{O}_s$ to obtain a solution O_s'
10: **if** $f(O_s') < f(S_s)$ **then**
11: Set $S_s = O_s'$
12: **if** $f(O_s') < f(S^\star)$ **then**
13: Set $S^\star = O_s'$
14: Set $n_{\text{it}}^\star = 0$
15: **else**
16: Set $n_{\text{it}}^\star = n_{\text{it}}^\star + 1$
17: **end if**
18: **end if**
19: **end for**
20: **end while**
21: **return** $S^\star$

We begin showing the algorithmic scheme in Algorithm 1. The structure of the proposed framework derives from one of MDEClust, from which we inherit the stopping criteria (line 4, Section 2.2) and the crossover operator (line 7, Section 2.2.1). The novelty of the new framework lies in the modification of operators such as mutation and local search, which now heavily depend on the assignment step. This step is no longer trivial, as the nearest center is no longer the most relevant for a data point. Instead, the assignment must account for must-link and cannot-link constraints (see Section 2.1 for further details), introducing additional complexities that must be carefully managed.

Note that the new employed local search is also used in S-MDEClust for the initialization of the population (line 1). This operator should always return a feasible solution, provided that the problem itself is feasible, and, thus, by instructions of Algorithm 1, it is easy to observe that the framework always guarantees to insert feasible solutions into the population. We analyze this aspect, as well as the ones related to the other mentioned operators, in the next sections.

3.2 Assignment Step

As mentioned in Section 2.2, each solution S in the population can be represented by any of the following items: the membership vector ϕ^S or the coordinate matrix ψ^S. However, while in the unsupervised MSSC scenario each one can be simply derived from the other in $O(NKd)$, in the semi-supervised context, as anticipated in Section 2.1, the assignment step to obtain ϕ^S from the coordinates of the clusters centers stored in ψ^S may not be trivial since must-link and cannot-link constraints must be taken into account.

In order to deal with the must-link and cannot-link constraints, we propose an exact assignment step and a greedy one.

3.2.1 Exact Assignment Step

Inspired by Baumann (2020); Piccialli et al. (2022), the exact assignment step is based on solving the following optimization problem:

$$\begin{aligned} \min \ & \sum_{i=1}^{N} \sum_{k=1}^{K} x_{ik} \| p_i - \psi_k^S \|^2 \\ \text{s.t. } & x \in \mathcal{X} \cap \mathcal{L} \end{aligned}, \tag{7}$$

where the sets $\mathcal{X}$ and $\mathcal{L}$ are defined as in equations (2) and (3), respectively.

Solving an additional optimization problem each time an assignment step is required surely leads to a greater consumption of computational resources. Moreover, although it is evident that knowing only one of ϕ^S and ψ^S is sufficient to characterize a solution S, now, differently from the unconstrained MSSC scenario, deriving one of the two from the other requires a different effort. The greater computational complexity is justified as we deal with semi-supervised MSSC problems, which are known to be more complex than the already NP-hard MSSC model. It is worth observing however that the exact assignment step is itself computationally demanding. Indeed, the problem can be seen as a generalization of the minimum cost perfect matching problem with conflict pair constraints (Öncan et al., 2013), which is known to be NP-hard. Thus, in order to reduce the computational effort, we also propose a faster greedy assignment methodology, to be used in specific phases of the S-MDEClust framework.

3.2.2 Greedy Assignment Step

We report the scheme of the proposed greedy methodology in Algorithm 2.

First, all the points are collected in groups based on the must-link constraints (line 1). Then, for each group G, we try to find the cluster centers that can be assigned to the points of the group (line 4): such cluster centers should not be already associated with a group $\hat{G}$ that has a point involved in a cannot-link constraint with a point of G. Similarly to COP-K-means (Wagstaff et al., 2011), the methodology highly depends on the assignment order of the groups; thus, it could happen that, in presence of cannot-link constraints, finding a cluster center to associate with a group is not possible. In such cases, any cluster center can be assigned to the points of the group (line 6), without taking into account any possible constraint violation. Given the available cluster centers, we finally assign the group points to one of the nearest (lines from 8 to 11).

Algorithm 2 has no guarantee of returning a final assignment that does not violate the constraints. However, being less demanding in terms of computational costs than the exact

Algorithm 2 Greedy assignment step.

Require: $\mathcal{D} = \{p_1, \ldots, p_N\} \subset \mathbb{R}^d$, $K \in \mathbb{N}_{>0}$ number of clusters, $\mathcal{M}_{\mathcal{L}}, \mathcal{C}_{\mathcal{L}} \subseteq \{1, \ldots, N\} \times \{1, \ldots, N\}$ sets of must-link and cannot-link constraints respectively, $S = (\phi^S, \psi^S)$.

1: Let $\mathcal{G}_{\mathcal{M}_{\mathcal{L}}} = \left\{G \subseteq \{1, \ldots, N\} \mid \forall i, j \in G \text{ s.t. } i \neq j, (i, j) \in \mathcal{M}_{\mathcal{L}}\right\}$
2: Let $\mathcal{A} = \emptyset$
3: **for all** $G \in \mathcal{G}_{\mathcal{M}_{\mathcal{L}}}$ **do**
4: Let

$$\hat{K} := \left\{ k \in \{1, \ldots, K\} \,\middle|\, \begin{array}{l} \forall\, (\hat{G}, k) \in \mathcal{A}, \forall\, i_c \in G, j_c \in \hat{G} \implies \\ (i_c, j_c) \notin \mathcal{C}_{\mathcal{L}} \end{array} \right\}$$

5: **if** $\hat{K} = \emptyset$ **then**
6: Let $\hat{K} := \{1, \ldots, K\}$
7: **end if**
8: Let

$$k_G \in \arg\min_{k \in \hat{K}} \sum_{i \in G} \left\| p_i - \psi_k^S \right\|^2$$

9: Set $\mathcal{A} = \mathcal{A} \cup \{(G, k_G)\}$
10: **for all** $i \in G$ **do**
11: Set $\phi_i^S = k_G$
12: **end for**
13: **end for**
14: **return** S

assignment step (no binary programming problem is involved), the greedy methodology can be useful in phases of S-MDEClust (i.e., crossover and mutation) where having a solution satisfying all the constraints is not mandatory (see Algorithm 1). In this way, we can indeed reduce the computational costs of the entire framework, still returning a final feasible solution. More information on the topic can be found in the next section.

3.3 Mutation Operator

While the crossover operator (line 7 of Algorithm 1) is the same one employed in MDEClust (Section 2.2.1), the mutation operator described in Section 2.2.2 must be modified so as to take into account must-link and cannot-link constraints. As already explained in the referenced section, the operator consists of a random-based relocation of the center of the offspring solution O_s generated by the crossover operator. However, in the semi-supervised MSSC model, the following two issues may arise.

1. After removing a cluster center, if the exact assignment step (Section 3.2) is employed in step 2. of the mutation operator (Section 2.2.2), it may not be possible to assign each dataset sample to one of the remaining centers without violating any cannot-link constraint, if present.
2. Regardless of the outcome of the assignment phase, it is still required to choose a new center to get a mutated solution with exactly K clusters.

In Algorithm 3, we provide the algorithmic scheme of the new mutation operator.

1. The assignment steps (lines 3–12) can be performed through the exact or the greedy methodology proposed in Section 3.2. However, while in line 12 the two methodologies are carried out as indicated in Section 3.2, in line 3, we have $K - 1$ centers to which the points can be assigned, thus requiring the following modifications in the two assignment step versions.

Algorithm 3 Mutation operator.

Require: $\mathcal{D} = \{p_1, \ldots, p_N\} \subset \mathbb{R}^d$, $K \in \mathbb{N}_{>0}$ number of clusters, $\mathcal{M}_{\mathcal{L}}, \mathcal{C}_{\mathcal{L}} \subseteq \{1, \ldots, N\} \times \{1, \ldots, N\}$ sets of must-link and cannot-link constraints respectively, $O_s = (\phi^{O_s}, \psi^{O_s})$ offspring solution, $\mathcal{U}(\cdot)$ uniform random number generator, $\alpha \in [0, 1]$.

1: Let $\psi^{\tilde{O}_s} = \psi^{O_s}$
2: Let $\bar{c} = \mathcal{U}(1, \ldots, K)$
3: Perform assignment step with $\mathcal{Y} \setminus \{y_{\bar{c}}\}$
4: **if** assignment step has not failed **then**
5: Let $\tilde{\phi}^{O_s}$ be the temporary membership vector
6: **else**
7: $\alpha = 0$
8: **end if**
9: Let $P_r = [P(1), \ldots, P(N)]$ ($\forall i \in \{1, \ldots, N\}$, $P(i)$ defined as in Eq. 6)
10: Choose $\bar{i} \in \{1, \ldots, N\}$ by roulette wheel based on P_r
11: Set $\psi_{\bar{c}}^{\tilde{O}_s} = p_{\bar{i}}$
12: Perform assignment step to get $\phi^{\tilde{O}_s}$
13: **return** $\tilde{O}_s = (\phi^{\tilde{O}_s}, \psi^{\tilde{O}_s})$

(a) In the exact assignment step, problem (7) changes into

$$\begin{aligned} \min \ & \sum_{i=1}^{N} \sum_{k=1}^{K} x_{ik} \left\| p_i - \psi_k^{\tilde{O}_s} \right\|^2 \\ \text{s.t. } & x \in \mathcal{X}_{\bar{c}} \cap \mathcal{L} \end{aligned} \tag{8}$$

where

$$\mathcal{X}_{\bar{c}} = \left\{ \begin{array}{l} x \in \{0, 1\}^{N \times K} \text{ s.t. } \displaystyle\sum_{k=1}^{K} x_{ik} = 1 \ \ \forall i \in \{1, \ldots, N\}, \\ \displaystyle\sum_{i=1}^{N} x_{ik} \geq 1 \ \ \forall k \in \{1, \ldots, K\} \setminus \{\bar{c}\}, \qquad \displaystyle\sum_{i=1}^{N} x_{i\bar{c}} = 0 \end{array} \right\}.$$

As anticipated at the beginning of the section, if the semi-supervised MSSC problem presents some cannot-link constraints, problem (8) may not have a feasible solution, and thus, the exact assignment step may fail.

(b) In the greedy methodology, we only consider the groups whose points are assigned to the removed center $y_{\bar{c}}$. In detail, in line 3 of Algorithm 2, we only take into account the sets $G \in \mathcal{G}_{\mathcal{M}_{\mathcal{L}}}$ such that $\forall i \in G$, $\phi_i^{O_s} = \bar{c}$. Moreover, we add a constraint in line 8: the center chosen for a group cannot be the removed one, that is, $k_G \neq \bar{c}$. Unlike the exact assignment step, the greedy methodology always returns a solution, although it may not be feasible.

2. If the exact assignment step has failed, in line 7, we set $\alpha = 0$ for Eq. 6, and thus, we employ the roulette wheel operation with each dataset point having the same chances to be selected. In this scenario, a probability based on the point-center distances cannot indeed be defined, as we cannot exactly associate each dataset point to a cluster center without getting a violation on the cannot-link constraints.

Note that, although it is worth discussing how the two assignment steps could be integrated into the mutation operator, the greedy assignment step is likely more beneficial at this stage, as well as during the crossover phase, in reducing the computational burden of the entire

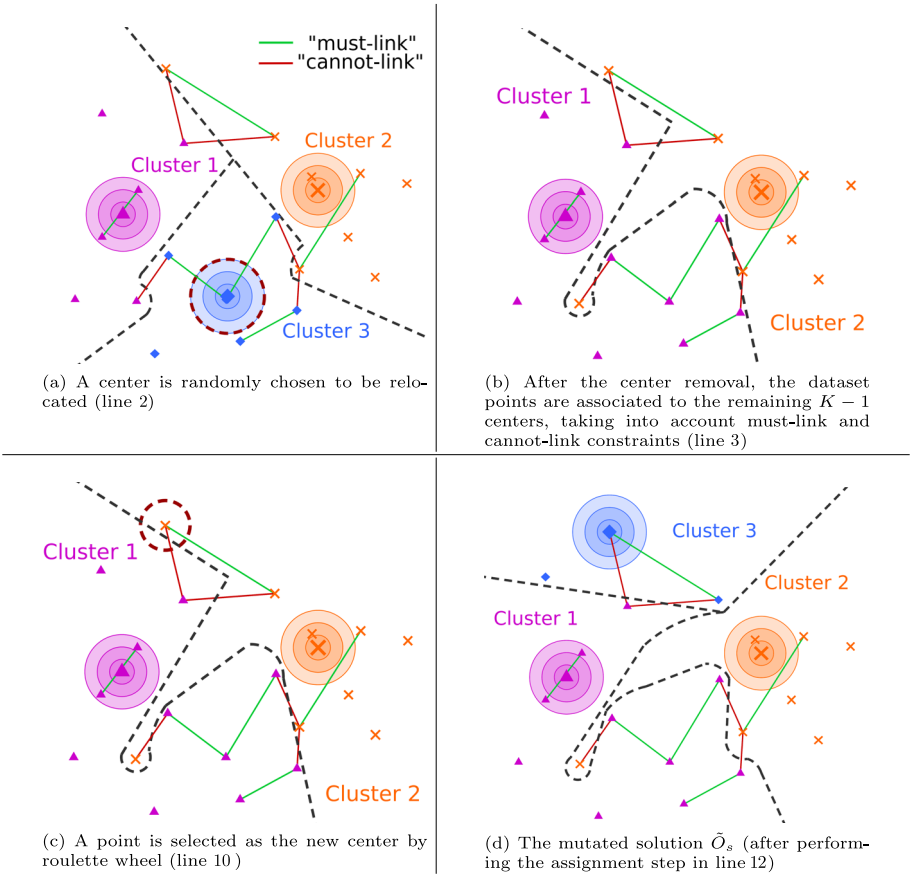
S-MDEClust framework. Indeed, in these phases, a feasible solution is not required, since it will be then used as a starting point for the local search procedure (Section 3.4), designed to generate a solution satisfying all the constraints regardless of the starting conditions. Moreover, performing local search steps from non-feasible solutions can be seen as a further mechanism to explore new regions of the feasible set. These insights are supported by the numerical experiments presented in Section 4.

In Fig. 2, we show a possible application of the mutation operator on a solution for the same two-dimensional problem of Fig. 1.

3.4 Semi-supervised Local Search Procedure

In the S-MDEClust framework, we employ the semi-supervised K-means variant proposed by Baumann (2020) as a local search procedure. Similar to the original approach, this methodology iterates two steps until convergence is reached:

1. An exact assignment step (Section 3.2) is performed.



(a) A center is randomly chosen to be relocated (line 2)

(b) After the center removal, the dataset points are associated to the remaining $K-1$ centers, taking into account must-link and cannot-link constraints (line 3)

(c) A point is selected as the new center by roulette wheel (line 10)

(d) The mutated solution $\tilde{O}_s$ (after performing the assignment step in line 12)

Fig. 2 Steps of the mutation operator (Algorithm 3) on a solution in the same setting as Fig. 1. Note that for the assignment steps, we employ the exact methodology

2. Each center y_k, with $k \in \{1, \ldots, K\}$, is moved to the geometric center of the k-th cluster (see Eq. 4).

The main difference with respect to the original approach is on the assignment step, where must-link and cannot-link constraints must be taken into account. In other words, the local search procedure is a standard K-means algorithm in which an exact binary linear optimization problem is solved in order to assign points to cluster centers taking into account every constraint.

By definition of the feasible set $\mathcal{X}$ (Eq. 2), a solution for problem (7) is feasible only if all the clusters have at least one assigned point. Thus, unlike K-means, the algorithm proposed by Baumann (2020) cannot produce degenerate solutions, i.e., instances where some clusters are left empty. This property makes useless the employment of solution repair operations, such as the one proposed by Mansueto and Schoen (2021) for MDEClust. Indeed, even if a degenerate solution is generated by either the crossover or mutation operator, that one is implicitly repaired after the local search procedure execution.

Finally, note that the local search procedure proposed by Baumann (2020) is further extended by Baumann and Hochbaum (2024). In particular, the extension presents a model size reduction technique based on kd-trees for problem (7) to improve scalability. Moreover, the possible management of must-link and cannot-link constraints as soft ones, for each of which the user can express a confidence value, is an additional feature which increases the number of clustering applications where the new approach can be used. We decided not to compare the performance of the two local search procedures within the S-MDEClust framework, since the main focus of the paper is the proposal of a novel global memetic approach for semi-supervised MSSC problems. In addition, we are confident that the experimental comparisons proposed in Section 4 would be similar regardless of the employed local search methodology.

4 Numerical Experiments

4.1 Experimental Settings

In this section, we report the results of thorough computational experiments, where we compared our framework S-MDEClust with state-of-the-art local and global methodologies from the semi-supervised clustering literature, i.e., COP-KM (Wagstaff et al., 2011), BLP-KM (Baumann, 2020), PC-SOS-SDP (Piccialli et al., 2022), and S-HG-means (Gribel et al., 2022) (more information on these competitors is provided in the next subsections). For the comparisons, S-MDEClust was implemented in Python (the implementation code can be found in the supplementary material). All the tests were run on a computer with the following characteristics: Intel Xeon processor E5-2430 v2, 6 cores, 12 threads, 2.50 GHz, 32 GB RAM. Finally, in order to solve instances of problems (7)-(8), the Gurobi optimizer (Gurobi Optimization, LLC, 2023) (Version 11) was employed.

The proposed framework S-MDEClust was tested in four different variants, with each of them differing from the others in the use of the mutation operator and/or the employment of the greedy assignment step in the crossover and mutation operations.

1. SG-MDE: no mutation operator and greedy assignment step
2. S-MDE: no mutation operator and exact assignment step
3. SMG-MDE: mutation operator and greedy assignment step
4. SM-MDE: mutation operator and exact assignment step

We remind that in the local search procedure (Section 3.4), the exact assignment step is always employed, in order to have at the end of each iteration a feasible solution for the semi-supervised MSSC problem. All the tested variants employed the exact matching for the crossover operator (Section 2.2.1).

For what concerns the parameters setting, we set the values according to Mansueto and Schoen (2021) and additional preliminary experiments, not reported here for the sake of brevity: $P = 20$; $N_{\max} = 500$, tol $= 10^{-4}$; $F \in [0.5, 0.8]$, randomly chosen at each crossover execution; $\alpha = 0.5$ for the mutation operator.

4.1.1 Datasets

The datasets used for experimentation are listed in Tables 1, 2, and 3.

As for the first set of datasets, all of them can be downloaded from the UCI Machine Learning Repository (Kelly et al., 2024, website: archive.ics.uci.edu) and from the UCR Time Series Classification Archive (Dau et al., 2018, website: cs.ucr.edu/%7Eeamonn/time_series_data_2018). The set is quite varied, being composed of datasets of different sizes and/or number of features. In particular, these datasets were part of the benchmark used to evaluate the quality of PC-SOS-SDP (Piccialli et al., 2022); moreover, some of them (Iris, Wine, and Ecoli) were also employed in the S-HG-means computational experimentation (Gribel et al., 2022), although with different (not available online) constraint configurations. As for the constraints definition, we considered the instances generated to test PC-SOS-SDP by Piccialli et al. (2022); such constraint configurations can be downloaded at github.com/antoniosudoso/pc-sos-sdp. The instances were composed of approximately $N/2$ and $N/4$ constraints, and they were generated taking into account the real dataset class partitioning. In Table 1, we list the various

Table 1 First set of datasets with the tested constraints configurations

Dataset	N	d	K	% Viol. (Mean)			$(\lvert\mathcal{M}_\mathcal{L}\rvert, \lvert\mathcal{C}_\mathcal{L}\rvert)$
				$\mathcal{M}_\mathcal{L}$	$\mathcal{M}_\mathcal{L}$ & $\mathcal{C}_\mathcal{L}$	$\mathcal{C}_\mathcal{L}$	
Iris	150	4	3	16.1	14.9	9.7	(50, 0), (100, 0), (25, 25), (50, 50), (0, 50), (0, 100)
Wine	178	13	3	41.3	32.8	20.8	
Connectionist	208	60	2	51.6	47.2	51.6	
Seeds	210	7	3	18.4	16.0	10.7	
Glass	214	9	6	57.7	40.9	24.0	
Accent	329	12	6	76.7	46.3	18.7	(100, 0), (150, 0), (50, 50), (75, 75), (0, 100), (0, 150)
Ecoli	336	7	8	57.9	31.9	3.4	
ECG5000	500	140	5	50.0	27.7	5.2	(150, 0), (250, 0), (75, 75), (125, 125), (0, 150), (0, 250)
Computers	500	720	2	46.6	49.8	52.3	
Gene	801	20531	5	1.3	0.9	0.5	(200, 0), (400, 0), (100, 100), (200, 200), (0, 200), (0, 400)

For each dataset and constraint type, the average percentage of violated pairwise constraints in the global minimum of the unconstrained MSSC problem is reported from Piccialli et al. (2022)

Table 2 Second set of datasets: two-dimensional synthetic instances generated by mixtures of spherical Gaussian distributions

N	K	$\lvert\mathcal{M}_{\mathcal{L}}\rvert/\lvert\mathcal{C}_{\mathcal{L}}\rvert$							%Viol. (Mean)						
		100	200	500	1000	2000	5000	10,000	100	200	500	1000	2000	5000	10,000
500	2	0.92	0.9	1.02	0.96	0.99	1.04	1	58	45.5	48.8	49.2	50.75	50.6	49.8
500	20	0.09	0.04	0.08	0.05	0.06	0.05	0.05	16.33	9.17	11.4	9.6	10.15	9.55	9.31
500	100	0.01	0.02	0	0.01	0.01	0.01	0.01	1.67	2.67	1.07	2	1.98	1.97	2.01
1000	2	1.44	0.96	1.07	0.97	0.94	1.04	1	41.33	45.17	45.67	40.53	43.6	41.87	42.31
1000	20	0.03	0.05	0.04	0.04	0.06	0.05	0.05	9.33	10.33	7.47	9.33	10.9	9.53	10.47
1000	100	0.02	0.01	0.01	0.01	0.01	0.01	0.01	3.67	0.83	2.53	2.53	1.82	2.03	2.06
2000	2	1.13	0.87	0.89	0.93	0.99	0.94	0.99	43.33	44.83	44.93	44.9	43.27	47.02	44.05
2000	20	0.02	0.03	0.08	0.05	0.05	0.05	0.05	4	6.83	13.2	9.6	9.92	10.32	10.14
2000	100	0.02	0.03	0.01	0.01	0.01	0.01	0.01	2.67	4.17	1.73	1.93	1.58	2.05	2.03
5000	2	1.22	1.02	1.07	1.13	1.01	1.01	0.98	45	46.5	42.2	45.37	46.08	44.79	44.71
5000	20	0.05	0.04	0.04	0.04	0.05	0.05	0.05	12.67	9.5	10.53	9.6	9.63	9.45	10.58
5000	100	0	0.02	0.01	0.01	0.01	0.01	0.01	0.33	3.83	1.27	1.7	1.98	1.96	2.15
10,000	2	0.79	0.75	1.2	0.98	1	1.03	1.06	43	44.83	46	46	46.08	45.76	45.87
10,000	20	0.06	0.09	0.05	0.05	0.05	0.05	0.05	12	12	10.33	9.93	10.4	10.23	10.16
10,000	100	0.01	0.02	0.01	0.01	0.01	0.01	0.01	2.67	2.5	1.4	2.13	1.97	2.32	2.35
20,000	2	0.92	1.33	1.11	1.02	0.96	1.03	1.03	39.67	43	43.47	42.73	44.12	43.91	45
20,000	20	0.06	0.05	0.05	0.05	0.05	0.05	0.05	9	11	10.87	8.53	10.7	10.29	10.4
20,000	100	0	0.02	0.01	0.01	0.01	0.01	0.01	0	2.17	2.27	1.6	2.35	2.45	2.11

For each dataset and number of constraints, we provide the ratio of must-link and cannot-link constraints, and the average percentage of violated pairwise constraints obtained by 3 runs of K-means solving the unconstrained MSSC problem

Table 3 Third set of datasets: large-size datasets

Dataset	N	d	K	$\lvert\mathcal{M}_{\mathcal{L}}\rvert/\lvert\mathcal{C}_{\mathcal{L}}\rvert$				%Viol. (Mean)			
				100	200	500	1000	100	200	500	1000
Banana	5300	2	2	1.08	1	1.16	1	43	55	50	50.1
Letter	20,000	16	26	0.04	0.02	0.05	0.04	7.67	4.83	7.2	6.1
Shuttle	57,999	9	7	1.17	1.9	2.03	1.9	26.67	32.33	37.33	37
Mnist	70,000	784	10	0.1	0.14	0.08	0.11	12.67	11.33	11.47	14.83
Cifar10	60,000	3072	10	0.16	0.12	0.12	0.11	17	18.5	15.93	17.6

For each dataset and number of constraints, we provide the ratio of must-link and cannot-link constraints, and the average percentage of violated pairwise constraints obtained by 3 runs of K-means solving the unconstrained MSSC problem

constraint set configurations used for each dataset: a set can contain either only must-link constraints, only cannot-link ones, or constraints of both types. Since for each configuration the constraint generation was made five times with different seeds, the total number of problems used for the comparisons was equal to 300. Finally, in Table 1, for each dataset and constraint typology, the average percentage of violated pairwise constraints in the global minimum of the unconstrained MSSC problem is reported from Piccialli et al. (2022): a configuration where several constraints are violated should ideally lead to a non-trivial semi-supervised clustering problem.

Regarding the second and third sets of datasets, we can find synthetic and large-size instances. The employment of heuristics in machine learning is usually carried out to handle (real-world) difficult problems; thus, we think that an efficacy and efficiency assessment of our proposal in semi-supervised clustering problems presenting thousands of samples/constraints and/or requiring a high number of partitions is necessary. In the synthetic datasets, we have two-dimensional samples generated by mixtures of spherical Gaussian distributions; the standard deviation of the resulting clusters was set to 10 so that they would not be well separated. As for the large-size datasets, we employed well-known clustering instances characterized by high values for N, d, and K. Each synthetic dataset, along with the tested constraints configurations, can be found in the supplementary material. The constraints for the large-size datasets are also provided, while these latter ones can be downloaded by executing the code in github.com/INFORMSJoC/2023.0419/tree/main.

For these second and third sets of datasets, predetermined numbers of constraints were taken into account; in order to define a constraint, we randomly selected a pair of samples, and based on the true class partitioning, we defined a must-link or a cannot-link constraint. Similarly to the first set of datasets, in Tables 2 and 3, we provide for each constraint configuration the ratio of must-link and cannot-link constraints, and the average percentage of violated pairwise constraints obtained by 3 runs of K-means solving the unconstrained MSSC problem.

4.1.2 Metrics

The metrics we employed to test the effectiveness and efficiency of the methodologies are the following: MSSC objective function (MSSC O.F.), the elapsed time in seconds (time), the number of calls to the local search procedure (N° L.S.), and the total number of iterations of the local search procedure (N° L.S. Iters), which is more suited than N° L.S. when comparing structurally different algorithms. In order to compare a variant of the S-MDEClust framework

and a competitor on a specific metric, we also made use of the relative percentage difference defined as

$$\Delta = \frac{100(m - m_R)}{m_R}, \tag{9}$$

where m is the metric value achieved by the variant and m_R is the one obtained by the competitor. A negative Δ means that the variant outperformed the competitor in terms of the considered metric.

Finally, in order to summarize some numerical results, we made use of the performance profiles (Dolan & Moré, 2002), a graphical tool showing the probability that a metric value achieved by a method in a problem is within a factor $\tau \in \mathbb{R}$ of the best value obtained by any of the algorithms in that problem. Formally, let $\mathcal{P}$ be a set of benchmark problems, $\mathcal{S}$ a set of solvers, and m a performance metric such that, for each $\pi \in \mathcal{P}$ and $\sigma \in \mathcal{S}$, the value $m_{\pi,\sigma}$ denotes the metric achieved by solver σ on problem π. The performance profile of solver σ is defined as follows:

$$p_\sigma(\tau) = \frac{1}{|\mathcal{P}|}\left|\left\{\pi \in \mathcal{P} \;\middle|\; \frac{m_{\pi,\sigma}}{\min\limits_{\tilde{\sigma}\in\mathcal{S}} m_{\pi,\tilde{\sigma}}} \leq \tau\right\}\right|,$$

which represents the estimated probability that the metric value achieved by solver σ on an arbitrary problem π is within a factor τ of the best observed value. In particular, $p_\sigma(1)$ gives the fraction of problems where σ achieved the best performance. Performance profiles are thus recognized as a powerful and compact way to assess the relative performance and robustness of the algorithms under consideration. For a more detailed technical explanation, we refer the reader to Dolan and Moré (2002).

4.2 Preliminary Assessment of the S-MDEClust Variants

Before proceeding with the comparisons w.r.t. the state-of-the-art approaches, we first compared the proposed variants of the S-MDEClust framework on a subset of datasets (Iris, Accent, ECG5000) in order to test the influence in each of them of the mutation operator presence and/or the use of the greedy assignment step. In Fig. 3, we report the performance profiles obtained by 5 runs of SG-MDE, S-MDE, SMG-MDE, and SM-MDE with five different seeds for the pseudo-random number generator.

First, we can observe that all the variants present similar effectiveness in terms of MSSC O.F.: the best results were obtained only employing the mutation in the framework, at the expense, however, of a greater number of local search executions and, as a consequence, of a greater elapsed time. The performance profiles w.r.t. time highlight the influence of the variants of the presence/absence of the greedy assignment step. We can indeed observe that this methodology allowed to save much computing time without degrading the solution quality in terms of MSSC O.F.: the differences between SG-MDE and S-MDE, which are not obvious in terms of N° L.S. and N° L.S. Iters, are sharp in terms of time; the same result holds for SMG-MDE and SM-MDE.

Taking into account these results, we decided to consider only the variants equipped with the greedy assignment step for the next comparisons, i.e., SG-MDE and SMG-MDE.

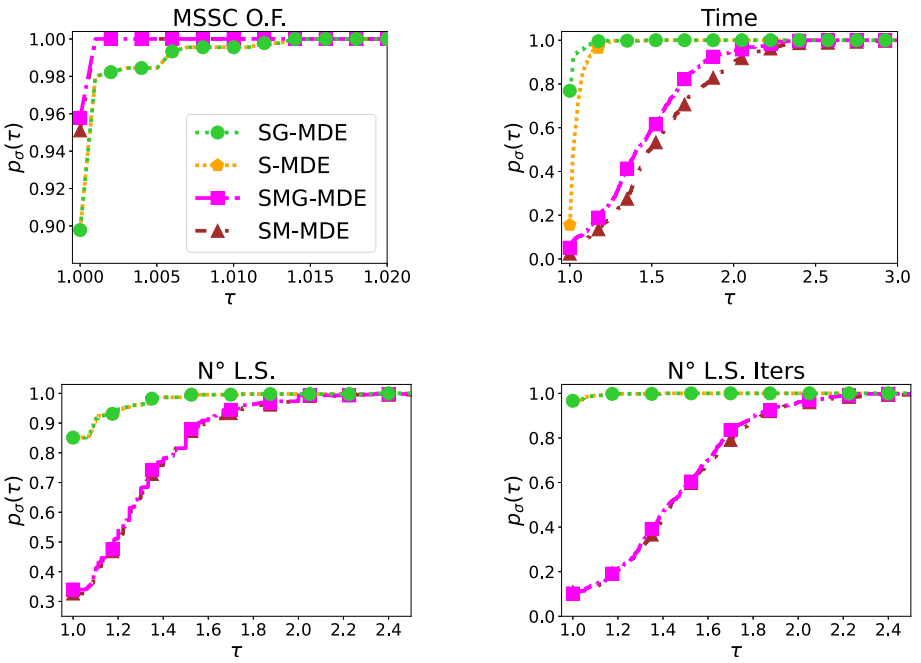


Fig. 3 Performance profiles for SG-MDE, S-MDE, SMG-MDE, and SM-MDE in terms of MSSC O.F., time, N° L.S. and N° L.S. Iters on a subset of the datasets listed in Table 1 (Iris, Accent, ECG5000). Note that the intervals of the axes were set for a better visualization of the numerical results. For further details on the interpretation of the axes, we refer the reader to Section 4.1.2

4.3 Comparison with Well-Known Heuristic Approaches

Since, as far as we know, no other memetic approach designed to return strictly feasible solutions has been proposed in the literature yet, we tested the goodness of S-MDEClust comparing it with well-known local heuristics for semi-supervised clustering run in a multi-start fashion: COP-K-means (Wagstaff et al., 2011), introduced in Section 1 and referred as COP-KM in the following for the sake of brevity; BLP-KM (Baumann, 2020), the Baumann approach also employed as local search in our framework (see Section 3.4). The implementation code of COP-KM can be found at github.com/ggalmagro/TFG/tree/master/Software/COPKmeans, while the one of BLP-KM can be found at github.com/phil85/BLPKM-CC.

As for the parameter setting, we run the two competitors with the values recommended in the respective papers and codes. SG-MDE, SMG-MDE, COP-KM, and BLP-KM were executed in order to process 100 solutions: each execution of COP-KM and BLP-KM consisted of 100 runs of the two heuristics with 100 initial random solutions; as for SG-MDE and SMG-MDE, they were executed with size of the population $P = 10$ and a maximum number of iterations equal to 10. For all the tested algorithms, the maximum number of iterations for the local search optimization was set to 25. Finally, in order to be less sensitive to the random operations, all the approaches were executed three times with different seeds for the pseudo-random number generator; the metric results were the averages of the values obtained in the three executions.

We start analyzing the performance of SG-MDE, BLP-KM, and COP-KM overall on the synthetic datasets of Table 2. In Table 4, we report the following metrics for each dataset: the

Table 4 Numerical results obtained by SG-MDE, BLP-KM, and COP-KM in terms of MSSC O.F., N° L.S. Iters and N_{unfeas} on the synthetic datasets of Table 2

N	K	%S				$\%N_{\text{unfeas}}$
		MSSC O.F.			N° L.S. Iters	
		SG-MDE	BLP-KM	COP-KM	SG-MDE	COP-KM
500	2	**100**	**100**	57.14	**100**	42.86
500	20	**100**	14.29	0	**100**	28.57
500	100	**100**	0	0	**100**	0
1000	2	**100**	**100**	42.86	**100**	42.86
1000	20	**100**	0	0	**100**	14.29
1000	100	**100**	0	0	**100**	0
2000	2	**100**	85.71	28.57	**100**	57.14
2000	20	**100**	0	0	**100**	0
2000	100	**100**	0	0	**100**	0
5000	2	**100**	57.14	0	**100**	57.14
5000	20	**100**	0	0	**100**	0
5000	100	**100**	0	0	**100**	0
10,000	2	**100**	71.43	0	**100**	71.43
10,000	20	**100**	0	0	**100**	0
10,000	100	**100**	0	0	**100**	0
20,000	2	**100**	42.86	0	**100**	71.43
20,000	20	**100**	0	0	**100**	0
20,000	100	**100**	0	0	**100**	0

The numbers in bold indicate the best values/performance w.r.t. the considered metric and problem

percentage (%S) of constraints configurations where SG-MDE/BLP-KM/COP-KM reached the best objective function (MSSC O.F.) mean value w.r.t. the others; the percentage of times where SG-MDE performed, on average, the minimum number of local search iterations (N° L.S. Iters); the rate $\%N_{\text{unfeas}}$ indicating the times where COP-KM did not manage to find any feasible solution in the three executions (we indeed remind that this approach is not guaranteed to generate a feasible clustering partition).

We easily observe that SG-MDE was by far the most effective and efficient approach in the tested benchmark. As for N° L.S. Iters, the lack of columns for BLP-KM and COP-KM is justified since the two competitors did not manage to perform less local search iterations than our approach. We can thus conclude that the stopping condition based on the population convergence turned out to be effective in the SG-MDE mechanisms to save computational resources without spoiling the solution quality. Regarding the MSSC O.F. metric, we notice that the performance difference between SG-MDE and the competitors is sharper as the values for N and K grow: SG-MDE was undoubtedly the best choice for problems characterized by numerous samples and partitions.

In the following, we then focus on the hardest synthetic datasets of Table 2, i.e., the ones with $N = 20,000$; in Table 5, we report, for each tested value of K and for each constraints configuration, the MSSC O.F. and N° L.S. Iters metrics average values attained by the three approaches.

Table 5 Numerical results obtained by SG-MDE, BLP-KM, and COP-KM in terms of MSSC O.F. and N° L.S. Iters on the synthetic datasets of Table 2 with $N = 20,000$

K	$\lvert\mathcal{M}_{\mathcal{L}} \cup \mathcal{C}_{\mathcal{L}}\rvert$	MSSC O.F.			N° L.S. Iters		
		SG-MDE	BLP-KM	COP-KM	SG-MDE	BLP-KM	COP-KM
2	100	**26766**	26,766	26,818	**725**	2490.67	2008.67
	200	**26,838.26**	26,838.35	26,932.75	**724.67**	2462.67	1991.67
	500	**27,039.26**	**27,039.26**	N/A	**723.33**	2475.67	N/A
	1000	**27,361.06**	27,361.06	N/A	**692**	2480.33	N/A
	2000	**27,984.05**	27,984.05	N/A	**525**	2401	N/A
	5000	**29,800.8**	**29,800.8**	N/A	**426.33**	2381.33	N/A
	10,000	**32,244.7**	**32,244.7**	N/A	**223**	1842	N/A
20	100	**3465.17**	3477.7	3486.42	**2040**	2500	2497
	200	**3477.67**	3490.82	3514.07	**1960.67**	2500	2497.33
	500	**3487.97**	3500.65	3529.72	**2122.33**	2500	2499
	1000	**3547.5**	3561.12	3635.31	**1971.33**	2500	2497.67
	2000	**3626.01**	3638.91	3786.6	**2119.67**	2500	2497
	5000	**3898.78**	3908.63	4297.12	**1921.33**	2500	2496.67
	10,000	**4332.8**	4335.76	5089.48	**2068.33**	2500	2495.67
100	100	**728.5**	754	754.03	**2403**	2500	2500
	200	**732.27**	756.75	759.45	**2429**	2500	2500
	500	**735.92**	759.8	761.91	**2471**	2500	2500
	1000	**735.23**	763.07	768.45	**2422**	2500	2500
	2000	**769.6**	795.83	837.09	**2404**	2500	2500
	5000	**817.14**	840.88	928.88	**2454**	2500	2500
	10,000	**880.54**	904.8	1062.33	**2464**	2500	2500

The numbers in bold indicate the best values/performance w.r.t. the considered metric and problem

The "N/A" sign reported in the columns of COP-KM indicates the scenarios where COP-KM was not able to generate a feasible solution for the problem at hand; we notice that the sign is reported only on problem instances with $K = 2$, leading us to the (reasonable) conclusion that higher values for K increase the chances of finding feasible solutions. Table 5 allows us to analyze the performance of the approaches for each specific value of K. When $K = 2$, SG-MDE was highly competitive in effectiveness with BLP-KM: the differences in terms of MSSC O.F. are indeed negligible. Still, as the number for K grows, we observe that such differences become more and more noticeable, especially in problems with a high number of constraints. In all these scenarios, COP-KM struggled to reach the same effectiveness as its competitors. In terms of N° L.S. Iters, SG-MDE was the clear winner; in this context, unlike the competitors, our approach even managed to generate high-quality solutions in the hardest scenarios without reaching the maximum possible number of local search iterations (2500).

We conclude the section showing in Fig. 4 the performance profiles obtained by SG-MDE, SMG-MDE, BLP-KM, and COP-KM on the large-size datasets of Table 3. In terms of MSSC O.F., the results meet our expectations: our proposals were the most robust algorithms, with BLP-KM and COP-KM being the third and the forth best algorithm, respectively. Regarding the two S-MDEClust variants, SMG-MDE leveraged the mutation operator effectiveness

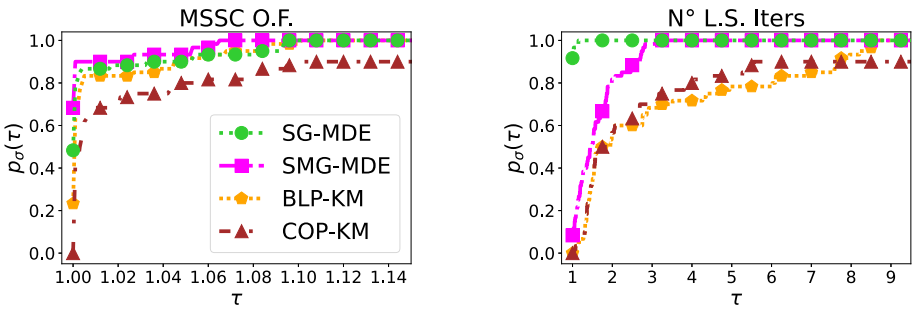


Fig. 4 Performance profiles for SG-MDE, SMG-MDE, BLP-KM, and COP-KM in terms of MSSC O.F. and N° L.S. Iters on the large-size datasets of Table 3. Note that the intervals of the x-axes were set for a better visualization of the numerical results. For further details on the interpretation of the axes, we refer the reader to Section 4.1.2

to outperform SG-MDE. As for N° L.S. Iters, the difference between BLP-KM and COP-KM is more negligible; in terms of efficiency, SG-MDE turned out to be the best choice, unsurprisingly outperforming the second-best, SMG-MDE, as well.

4.4 Comparison with Global State-of-the-Art Methodologies

We finally decided to test our framework with some recent, albeit different types, state-of-the-art global optimization approaches.

1. PC-SOS-SDP: a branch-and-cut algorithm for semi-supervised MSSC problems proposed by Piccialli et al. (2022); for the lower bound procedure, a semi-definite programming (SDP) relaxation of the MSSC discrete optimization model and a cutting-plane procedure for strengthening the bound are employed; for the upper bound, the authors use BLP-KM (Baumann, 2020) with an initialization technique based on the primal solution of the SDP relaxation; the algorithm was proved to be effective with problems whose dimensionalities are about four times larger than the ones tested with other exact algorithms from the literature. The implementation code of PC-SOS-SDP can be found at github.com/antoniosudoso/pc-sos-sdp.
2. S-HG-means: a probabilistic-based adaptation for the semi-supervised scenario of the HG-means algorithm (Gribel et al., 2022), which was proven to be highly effective in unconstrained MSSC problems (Gribel & Vidal, 2019; Gribel et al., 2022) propose a generative model that assumes Gaussian-distributed data samples along with must-link and cannot-link constraints generated by stochastic block models; adopting a maximum-likelihood approach, they employ the HG-means algorithm equipped with modified and new operators to take into account the new generative model; the resulting procedure also has a parameter $p \in [0, 1]$ indicating the accuracy of the provided pairwise constraints, so as to handle possible inaccuracies in their definitions. The implementation code of S-HG-means can be found at github.com/danielgribel/SSC-IPA.

As for PC-SOS-SDP, we considered all the parameter settings recommended by Piccialli et al. (2022) and in the published implementation code; the algorithm was tested with a pool of 12 threads for the branch-and-cut procedure, 12 threads for the computation at the root node, and 1 thread for each lower bound computation. Similarly, the parameter values for S-HG-means were set according to Gribel et al. (2022); Gribel and Vidal (2019) and the published

code. Moreover, since in our setting the constraints were considered accurate, we set $p = 1$. Similar to the S-MDEClust variants, S-HG-means involves random-based operations, and thus, it was run five times with different seeds for the pseudo-random number generator. The results were again obtained by calculating the averages of the metric values achieved in the five tests.

4.4.1 The Exact Brand-and-Cut Algorithm

In Tables 6 and 7, we compare the performance of SG-MDE and SMG-MDE w.r.t. PC-SOS-SDP on the datasets of Table 1. For each dataset, we report the following:

1. %S: It indicates the percentage of the tested constraint configurations (30 for each dataset; see Section 4.1.1 for more information) where we had a success, i.e., SG-MDE/SMG-MDE obtained a better or equal performance compared to PC-SOS-SDP; in the time context, in order to have a success, we also required that in the considered instance our method had obtained a success in terms of MSSC O.F.; we indeed considered a computational time saving w.r.t. a competitor useless if it was at the expense of the solution quality.
2. Δ_{Mi}, Δ_{Me}, Δ_{Ma}: The minimum, median, and maximum of all the relative percentage differences obtained by our method on the tested constraint configurations; in the time context, we considered only the differences achieved in instances where our methodology obtained a success in terms of MSSC O.F..

Since PC-SOS-SDP spends additional time to prove the solution optimality, in order to have a fairer comparison, we decided to consider as time metric for the exact approach the elapsed time to find the best solution.

As for MSSC O.F., SG-MDE and SMG-MDE proved to be competitive with PC-SOS-SDP, reaching most of the time a similar performance. We further observe that the use of the mutation operator in SMG-MDE improved most of the statistics in terms of MSSC O.F.. The results w.r.t. time are justified by the different nature of the algorithms: the exact approach is composed of more time-consuming operations than our memetic framework, which, as a consequence, achieved the best performance overall. The time metric is thus inconsistent

Table 6 Numerical results obtained by the SG-MDE algorithm w.r.t. PC-SOS-SDP in terms of MSSC O.F. and time on the datasets of Table 1

Dataset	MSSC O.F.				Time			
	%S	Δ_{Mi}	Δ_{Me}	Δ_{Ma}	%S	Δ_{Mi}	Δ_{Me}	Δ_{Ma}
Iris	**100.0**	**0.0**	**0.0**	**0.0**	**93.33**	**−95.02**	**−70.09**	35.73
Wine	**100.0**	**0.0**	**0.0**	**0.0**	**96.67**	**98.2**	**−94.01**	59.79
Connectionist	**93.33**	**−0.001**	**0.0**	0.001	**93.33**	**−98.2**	**−94.58**	**−83.2**
Seeds	**100.0**	**0.0**	**0.0**	**0.0**	**100.0**	**−97.54**	**−89.46**	**−38.69**
Glass	23.33	**0.0**	0.014	0.425	23.33	**−93.01**	**−88.64**	**−82.49**
Accent	46.67	**−0.004**	0.0	0.409	46.67	**−97.3**	**−91.4**	**−81.8**
Ecoli	36.67	**0.0**	0.003	0.479	36.67	**−94.75**	**−88.0**	**−54.15**
ECG5000	46.67	**0.0**	0.0	0.03	46.67	**−99.37**	**−93.29**	**−72.48**
Computers	**76.67**	**0.0**	**0.0**	0.001	**76.67**	**−99.27**	**−98.17**	**−88.56**
Gene	**100.0**	**0.0**	**0.0**	**0.0**	**100.0**	**−92.12**	**−78.99**	**−57.45**

The numbers in bold indicate the best values/performance w.r.t. the considered metric and problem

Table 7 Numerical results obtained by the SMG-MDE algorithm w.r.t. PC-SOS-SDP in terms of MSSC O.F. and time on the datasets of Table 1

Dataset	MSSC O.F.				Time			
	%S	Δ_{Mi}	Δ_{Me}	Δ_{Ma}	%S	Δ_{Mi}	Δ_{Me}	Δ_{Ma}
Iris	**100.0**	**0.0**	**0.0**	**0.0**	**86.67**	**−94.86**	**−62.62**	89.94
Wine	**96.67**	**0.0**	**0.0**	0.009	**93.33**	**−98.2**	**−90.7**	101.45
Connectionist	**93.33**	**−0.001**	**0.0**	0.002	**93.33**	**−97.04**	**−89.96**	**−62.33**
Seeds	**100.0**	**0.0**	**0.0**	**0.0**	**96.67**	**−97.38**	**−85.41**	3.83
Glass	56.67	**0.0**	**0.0**	0.357	56.67	**−89.21**	**−82.49**	**−60.85**
Accent	**80.0**	**−0.004**	**0.0**	0.007	**80.0**	**−96.73**	**−88.0**	**−56.13**
Ecoli	53.33	**0.0**	**0.0**	0.077	53.33	**−93.55**	**−83.6**	**−36.29**
ECG5000	70.0	**0.0**	**0.0**	0.012	70.0	**−98.97**	**−93.25**	**−64.85**
Computers	**83.33**	**0.0**	**0.0**	0.001	**83.33**	**−98.61**	**−96.54**	**−72.28**
Gene	**100.0**	**0.0**	**0.0**	**0.0**	**100.0**	**−87.39**	**−61.02**	**−20.72**

The numbers in bold indicate the best values/performance w.r.t. the considered metric and problem

alone, but the analysis of it in conjunction with MSSC O.F. leads to an interesting, not obvious a priori, conclusion: SG-MDE and SMG-MDE, with only fractions of the CPU times needed by PC-SOS-SDP, managed to reach an effectiveness level which is highly competitive with that of an exact algorithm.

At first glance, cases where SG-MDE and/or SMG-MDE outperformed PC-SOS-SDP in terms of MSSC O.F., corresponding to values of $\Delta_{\text{Mi}} < 0$, may seem surprising. After all, PC-SOS-SDP is an exact global optimization method capable of returning a provably optimal solution through systematic updates of upper and lower bounds. However, as already noted, this level of exactness comes at a high computational cost, which makes the approach practically prohibitive when a very low branch-and-bound tolerance, governing the certification level of the optimal solution, is required. Based on this limitation, and following the parameter settings recommended by the PC-SOS-SDP authors (see Piccialli et al., 2022, and the published code), the method was run with a branch-and-bound tolerance of 10^{-4}. In most cases, this setting was sufficient to obtain a provably optimal solution; in others, it at least provided a bound on the maximum error. The cases where SG-MDE and/or SMG-MDE outperformed PC-SOS-SDP fall into this latter category: our approaches, being significantly more lightweight in terms of computational cost, were able to reach within reasonable computational time solutions that, on average, slightly outperform the upper error bounds returned by PC-SOS-SDP. Therefore, these results do not fundamentally indicate any anomaly or unexpected behavior of the three considered approaches, but rather reflect the practical infeasibility of employing a computationally intensive exact global optimization method to attain a high level of solution certification.

Finally, we compare our approaches, SG-MDE and SMG-MDE, with PC-SOS-SDP under two different scenarios: when dealing with sparse datasets, where distances between points may become less significant for clustering, and non-sparse datasets. As representatives, we selected the Connectionist dataset for the sparse case and the Seeds dataset for the non-sparse case. For both datasets, we used a constraint configuration composed of 50 must-link and 50 cannot-link constraints, and we varied the number of clusters $K \in \{3, 4, 6, 8, 10\}$. The constraints were originally defined based on $K = 2$ for Connectionist and $K = 3$ for Seeds; thus, for higher values of K, the feasibility of the solution is preserved. Tables 8 and 9 report the results in terms of MSSC O.F. and time, including the relative percentage difference in elapsed time Δ_T with respect to the $K = 3$ case.

Table 8 Numerical results obtained by SG-MDE and SMG-MDE in terms of MSSC O.F. and time (given in seconds) on the datasets Connectionist and Seeds of Table 1 for different numbers of K

Dataset	K	SG-MDE			SMG-MDE		
		MSSC O.F	Time	Δ_T	MSSC O.F	Time	Δ_T
Connectionist	3	**279.82**	**5.45**		**279.82**	10.62	
	4	258.36	**9.88**	81.17	258.36	18.0	**69.41**
	6	**233.03**	**24.64**	351.88	**233.03**	34.29	**222.76**
	8	216.97	**35.72**	554.99	216.9	51.75	**387.06**
	10	204.64	**47.17**	765.02	204.15	72.22	**579.72**
Seeds	3	**634.58**	**5.16**		**634.58**	8.2	
	4	**524.89**	**8.14**	57.87	**524.89**	12.46	51.93
	6	**409.4**	**18.53**	259.35	**409.4**	23.29	183.95
	8	**336.21**	**32.64**	533.03	336.26	47.25	475.98
	10	298.27	**48.74**	845.17	298.01	66.44	709.93

The symbol Δ_T denotes the relative percentage difference in elapsed time with respect to the case $K = 3$. Each bolded metric value indicates the best performance for that metric on the given problem. To determine the best values, those obtained by PC-SOS-SDP in Table 9 were also taken into account
The numbers in bold indicate the best values/performance w.r.t. the considered metric and problem

From the tables, we observe that both SG-MDE and SMG-MDE require more time as K increases, which is expected since many operations scale with K. Notably, the increase in time relative to the $K = 3$ baseline is less pronounced for SMG-MDE than for SG-MDE, while the latter generally achieves the lowest absolute times. Both approaches are faster on the non-sparse Seeds dataset, but the overall trend in terms of time and Δ_T remains consistent across both datasets, suggesting that the performance of our methods is robust regardless of the sparsity characteristic of the dataset. On the other hand, PC-SOS-SDP shows a different

Table 9 Numerical results obtained by PC-SOS-SDP in terms of MSSC O.F. and time (given in seconds) on the datasets Connectionist and Seeds of Table 1 for different numbers of K

Dataset	K	PC-SOS-SDP		
		MSSC O.F	Time	Δ_T
Connectionist	3	**279.82**	223.11	
	4	**258.34**	490.97	120.06
	6	233.24	1307.85	486.18
	8	**216.83**	3679.22	1549.04
	10	**204.13**	3217.8	1342.23
Seeds	3	**634.58**	126.05	
	4	**524.89**	64.41	**−48.9**
	6	**409.4**	90.24	**−28.4**
	8	**336.21**	103.37	**−17.99**
	10	**297.98**	170.8	**35.51**

The symbol Δ_T denotes the relative percentage difference in elapsed time with respect to the case $K = 3$. Each bolded metric value indicates the best performance for that metric on the given problem. To determine the best values, those obtained by SG-MDE and SMG-MDE in Table 8 were also taken into account
The numbers in bold indicate the best values/performance w.r.t. the considered metric and problem

Table 10 Numerical results obtained by the SG-MDE algorithm w.r.t. S-HG-means in terms of MSSC O.F. and time on the dataset instances of Table 1 where no constraint is violated by S-HG-means; the numbers of such instances are indicated in the parentheses

Dataset	MSSC O.F.				Time			
	%S	Δ_{Mi}	Δ_{Me}	Δ_{Ma}	%S	Δ_{Mi}	Δ_{Me}	Δ_{Ma}
Iris (30)	**100.0**	**−1.28**	**−0.001**	**0.0**	40.0	**−52.83**	18.59	160.91
Seeds (30)	**100.0**	**−1.134**	**−0.044**	**0.0**	26.67	**−60.81**	24.69	211.93
Glass (3)	**100.0**	**−29.552**	**−28.591**	**−26.053**	0.0	3.11	7.23	94.26
Accent (10)	**100.0**	**−2.72**	**−1.73**	**−0.985**	0.0	77.43	143.9	191.1
Ecoli (16)	**100.0**	**−22.062**	**−16.94**	**−4.068**	0.0	69.79	119.63	334.3

The numbers in bold indicate the best values/performance w.r.t. the considered metric and problem

behavior, with its efficiency being highly sensitive to such dataset characteristics, as illustrated in Table 9. For the non-sparse Seeds dataset, the best absolute execution times of PC-SOS-SDP correspond to cases where $K > 3$, while on the sparse Connectionist dataset, the increase in time becomes progressively more significant as the value of K increases. In particular, these increases are substantially larger than those observed with our proposed methods. This experimentation further confirms the efficiency of our approaches: in all cases, they achieved better time values than the competitor. Furthermore, as already evidenced in Tables 6 and 7, the time savings in many cases did not come at the cost of solution quality.

4.4.2 Performance Evaluation of the Global Memetic Methods

In Tables 10, 11, 12, and 13, we report a similar comparison with S-HG-means w.r.t. MSSC O.F., time, and N° L.S. on the dataset instances where no constraint is violated by S-HG-means.

Although our approach better than the competitor in terms of N° L.S. (Tables 12 and 13), they took more time than S-HG-means in most of the instances, as can be seen from Tables 10 and 11; these results are a direct consequence of the fact that S-HG-means employs a less computationally demanding local search, which, however, does not have any feasibility guarantee. Such efficiency did not allow S-HG-means to reach feasibility in most of the sce-

Table 11 Numerical results obtained by the SMG-MDE algorithm w.r.t. S-HG-means in terms of MSSC O.F. and time on the dataset instances of Table 1 where no constraint is violated by S-HG-means; the numbers of such instances are indicated in the parentheses

Dataset	MSSC O.F.				Time			
	%S	Δ_{Mi}	Δ_{Me}	Δ_{Ma}	%S	Δ_{Mi}	Δ_{Me}	Δ_{Ma}
Iris (30)	**100.0**	**−1.28**	**−0.001**	**0.0**	26.67	**−52.56**	71.54	436.99
Seeds (30)	**100.0**	**−1.134**	**−0.044**	**0.0**	16.67	**−61.5**	85.75	482.71
Glass (3)	**100.0**	**−29.552**	**−28.597**	**−26.053**	0.0	59.82	85.28	150.33
Accent (10)	**100.0**	**−2.72**	**−1.731**	**−0.985**	0.0	171.49	244.17	337.0
Ecoli (16)	**100.0**	**−22.062**	**−16.94**	**−4.068**	0.0	121.88	195.29	529.58

The numbers in bold indicate the best values/performance w.r.t. the considered metric and problem

Table 12 Numerical results obtained by the SG-MDE algorithm w.r.t. S-HG-means in terms of N° L.S. on the dataset instances of Table 1 where no constraint is violated by S-HG-means; the numbers of such instances are indicated in the parentheses

Dataset	N° L.S.			
	%S	Δ_{Mi}	Δ_{Me}	Δ_{Ma}
Iris (30)	**86.67**	**−94.45**	**−79.08**	33.33
Seeds (30)	**90.0**	**−96.08**	**−67.31**	40.0
Glass (3)	**100.0**	**−75.16**	**−74.51**	**−64.71**
Accent (10)	**100.0**	**−66.27**	**−52.78**	**−43.8**
Ecoli (16)	**100.0**	**−58.43**	**−46.05**	**−1.75**

The numbers in bold indicate the best values/performance w.r.t. the considered metric and problem

narios anyway; moreover, in the instances where it did, it was outperformed w.r.t. MSSC O.F. (Tables 10 and 11) (with remarkable results in some cases) by SG-MDE and SMG-MDE.

5 Conclusions

In this paper, we considered the semi-supervised MSSC problem, where instance-level constraints, each of them indicating the membership of two points to the same or to a different cluster, are given a priori. The presence of such must-link and cannot-link constraints makes the semi-supervised MSSC problem at least as hard as the unsupervised one (NP-hard).

After reviewing the (relatively new) related literature, we proposed a novel methodology for the considered problem where the given constraints must be strictly satisfied at each iteration. In particular, the proposal originates from a differential evolution-based memetic framework, proposed by Mansueto and Schoen (2021) for unsupervised clustering and proved to be highly competitive w.r.t. the state-of-the-art. Being a memetic approach, the new framework, called S-MDEClust, combines random-based genetic operations with an effective local search procedure, placed at the end of each iteration to refine every new solution. S-MDEClust presents innovative extensions, such as exact and greedy strategies for the points assignment and the possibility of managing temporarily infeasible solutions. To the best of our knowledge, it is the first attempt in the semi-supervised clustering literature to propose a memetic algorithm designed to return good quality solutions satisfying all the constraints.

Finally, we compared our framework with state-of-the-art local and global methodologies through a vast set of computational experiments on well-known and synthetic datasets: S-MDEClust turned out to be effective and efficient w.r.t. the competitors in any scenario and, especially, on the larger instances. In particular, (1) it was superior w.r.t. well-known local

Table 13 Numerical results obtained by the SMG-MDE algorithm w.r.t. S-HG-means in terms of N° L.S. on the dataset instances of Table 1 where no constraint is violated by S-HG-means; the numbers of such instances are indicated in the parentheses

Dataset	N° L.S.			
	%S	Δ_{Mi}	Δ_{Me}	Δ_{Ma}
Iris (30)	**83.33**	**−94.45**	**−75.27**	113.33
Seeds (30)	**90.0**	**−96.08**	**−66.67**	106.67
Glass (3)	**100.0**	**−68.63**	**−62.35**	**−58.82**
Accent (10)	**100.0**	**−60.0**	**−44.12**	**−33.82**
Ecoli (16)	**87.5**	**−50.59**	**−37.14**	26.32

The numbers in bold indicate the best values/performance w.r.t. the considered metric and problem

methodologies in terms of both solution quality and computational demanding resources; (2) it was proved to have an effectiveness in finding good solutions similar to the one typical of exact approaches, with only fractions of CPU times required by the latter; although S-MDEClust is a metaheuristic framework and thus offers no guarantee of global optimality, the comparison shows that our proposed approach was indeed frequently able to find a global optimum; (3) it performed better than soft-constrained memetic approaches, having guarantee of always finding feasible solutions which, in practice, turned out to be of higher quality.

Possible future research directions include the introduction of mechanisms in the greedy assignment methodology so as to maximize the chances of final assignments that do not violate the constraints, an extension of the presented framework to handle other clustering models, and the management of cluster constraints other than the instance-level ones.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00357-025-09528-z.

Funding Open access funding provided by Università degli Studi di Firenze within the CRUI-CARE Agreement.

Data Availability The synthetic datasets, the constraint configurations for the synthetic and large-size datasets (Table 2), and all the computational experiments results can be found in the supplementary material.

Code Availability The S-MDEClust source code can be found in the supplementary material.

Declarations

Conflict of Interest The authors declare no competing interests.

References

Aloise, D., Deshpande, A., Hansen, P., & Popat, P. (2009). NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning, 75*(2), 245–248. https://doi.org/10.1007/s10994-009-5103-0

Aloise, D., Hansen, P., & Liberti, L. (2012a). An improved column generation algorithm for minimum sum-of-squares clustering. *Mathematical Programming, 131*(1), 195–220. https://doi.org/10.1007/s10107-010-0349-7

Aloise, D., Hansen, P., & Rocha, C. (2012b). A column generation algorithm for semisupervised minimum sum-of-squares clustering. In *Global optimization workshop 2012* (pp. 19–22).

Al-Sultan, K. (1995). A tabu search approach to the clustering problem. *Pattern Recognition, 28*, 1443–1451. https://doi.org/10.1016/0031-3203(95)00022-R

Arthur, D., & Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms* (pp. 1027–1035). USA: Society for Industrial and Applied Mathematics.

Bagirov, A., Taheri, S., & Ugon, J. (2016). Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems. *Pattern Recognition, 53*, 12–24. https://doi.org/10.1016/j.patcog.2015.11.011

Basu, S., Banerjee, A., Mooney, R. J. (2004). Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM international conference on data mining (SDM)* (pp. 333–344).

Baumann, P. (2020). A binary linear programming-based k-means algorithm for clustering with must-link and cannot-link constraints. In *2020 IEEE international conference on industrial engineering and engineering management (IEEM)* (pp. 324–328).

Baumann, P., & Hochbaum, D. S. (2024). An algorithm for clustering with condence-based must-link and cannot-link constraints. *INFORMS Journal on Computing*. https://doi.org/10.1287/ijoc.2023.0419.

Dao, T.-B.-H., Duong, K.-C., & Vrain, C. (2017). Constrained clustering by constraint programming. *Articial Intelligence, 244*, 70–94. https://doi.org/10.1016/j.artint.2015.05.006. (Combining Constraint Solving with Mining and Learning)

Dau, H. A., Keogh, E., Kamgar, K., Yeh, C. -C. M., Zhu, Y., Gharghabi, S., Hexagon-ML et al. (2018). *The UCR time series classication archive*. Retrieved from https://www.cs.ucr.edu/eamonn/time_series_data_2018

Davidson, I., & Ravi, S. S. (2005). Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 SIAM international conference on data mining (SDM)* (pp. 138–149).

Davidson, I., & Ravi, S. S. (2007). Intractability and clustering with constraints. In *Proceedings of the 24th international conference on machine learning* (pp. 201–208). New York, NY, USA: Association for Computing Machinery.

Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming, 91*(2), 201–213. https://doi.org/10.1007/s101070100263

Ganji, M., Bailey, J., Stuckey, P. J. (2016). Lagrangian constrained clustering. In *Proceedings of the 2016 SIAM international conference on data mining (SDM)* (pp. 288–296).

González-Almagro, G., Luengo, J., Cano, J.-R., & García, S. (2020). DILS: Constrained clustering through dual iterative local search. *Computers & Operations Research, 121*, 104979. https://doi.org/10.1016/j.cor.2020.104979

Gribel, D., Gendreau, M., & Vidal, T. (2022). Semi-supervised clustering with inaccurate pairwise annotations. *Information Sciences, 607*, 441–457. https://doi.org/10.1016/j.ins.2022.05.035

Gribel, D., & Vidal, T. (2019). HG-means: A scalable hybrid genetic algorithm for minimum sum-of-squares clustering. *Pattern Recognition, 88*, 569–583. https://doi.org/10.1016/j.patcog.2018.12.022

Guns, T., Dao, T.-B.-H., Vrain, C., & Duong, K.-C. (2016). Repetitive branch-andbound using constraint programming for constrained minimum sum-of-squares clustering. In *Proceedings of the twenty-second European conference on articial intelligence* (pp. 462–470). NLD: IOS Press.

Gurobi Optimization, LLC (2023). *Gurobi optimizer reference manual*. Retrieved from https://www.gurobi.com.

Han, J., Pei, J., & Tong, H. (2022). *Data mining: Concepts and techniques*. Morgan Kaufmann.

Hansen, P., & Jaumard, B. (1997). Cluster analysis and mathematical programming. *Mathematical Programming, 79*(1), 191–215. https://doi.org/10.1007/BF02614317

Hartigan, J., & Wong, M. (1979). Algorithm AS 136: A k-means clustering algorithm. *Applied Statistics, 28*, 100–108. https://doi.org/10.2307/2346830

Hu, G., Zhou, S., Guan, J., & Hu, X. (2008). Towards effective document clustering: A constrained k-means based approach. *Information Processing & Management, 44*(4), 1397–1409. https://doi.org/10.1016/j.ipm.2008.03.001

Huang, Y., & Mitchell, T.M. (2006). Text clustering with extended user feedback. In *Proceedings of the 29th annual international ACM SIGIR conference on researchand development in information retrieval* (pp. 413–420). New York, NY, USA: Association for Computing Machinery.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters, 31*(8), 651–666. https://doi.org/10.1016/j.patrec.2009.09.011. (Award winning papers from the 19th International Conference on Pattern Recognition (ICPR))

Kelly, M., Longjohn, R., & Nottingham, K. (2024). The UCI machine learning repository. Retrieved from https://archive.ics.uci.edu

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly, 2*(1–2), 83–97. https://doi.org/10.1002/nav.3800020109

Liberti, L., & Manca, B. (2022). Side-constrained minimum sum-of-squares clustering: mathematical programming and random projections. *Journal of Global Optimization, 83*(1), 83–118. https://doi.org/10.1007/s10898-021-01047-6

Likas, A., Vlassis, N., Verbeek, J., & J. (2003). The global k-means clustering algorithm. *Pattern Recognition, 36*(2), 451–461. https://doi.org/10.1016/S0031-3203(02)00060-2 (Biometrics)

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory, 28*(2), 129–137. https://doi.org/10.1109/TIT.1982.1056489

MacQueen, J. (1965). Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley symposium on math., stat., and prob* (p. 281).

Mansueto, P., & Schoen, F. (2021). Memetic differential evolution methods for clustering problems. *Pattern Recognition, 114*, 107849. https://doi.org/10.1016/j.patcog.2021.107849

Maulik, U., & Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique. *Pattern Recognition, 33*, 1455–1465. https://doi.org/10.1016/S0031-3203(99)00137-5

Nghiem, N.-V.-D., Vrain, C., Dao, T.-B.-H., & Davidson, I. (2020). Constrained clustering via post-processing. In A. Appice, G. Tsoumakas, Y. Manolopoulos, & S. Matwin (Eds.), *Discovery science* (pp. 53–67). Cham: Springer International Publishing.

Ordin, B., & Bagirov, A. M. (2015). A heuristic algorithm for solving the minimum sum-of-squares clustering problems. *Journal of Global Optimization, 61*(2), 341–361. https://doi.org/10.1007/s10898-014-0171-5

Öncan, T., Zhang, R., & Punnen, A. P. (2013). The minimum cost perfect matching problem with conflict pair constraints. *Computers & Operations Research, 40*(4), 920–930. https://doi.org/10.1016/j.cor.2012.10.022

Pensa, R. G., & Boulicaut, J. -F. (2008). Constrained co-clustering of gene expression data. In *Proceedings of the 2008 SIAM international conference on data mining (SDM)* (pp. 25–36).

Piccialli, V., Russo Russo, A., & Sudoso, A. M. (2022). An exact algorithm for semi-supervised minimum sum-of-squares clustering. *Computers & Operations Research, 147*, 105958. https://doi.org/10.1016/j.cor.2022.105958

Rutayisire, T., Yang, Y., Lin, C., & Zhang, J. (2011). A modified Cop-kmeans algorithm based on sequenced cannot-link set. J. Yao, S. Ramanna, G. Wang, & Z. Suraj (Eds.), Rough sets and knowledge technology (pp. 217–225). Berlin, Heidelberg: Springer Berlin Heidelberg

Sarkar, M., Yegnanarayana, B., & Khemani, D. (1997). A clustering algorithm using an evolutionary programming-based approach. *Pattern Recognition Letters, 18*, 975–986. https://doi.org/10.1016/S0167-8655(97)00122-0

Selim, S., & Alsultan, K. (1991). A simulated annealing algorithm for the clustering problem. *Pattern Recognition, 24*, 1003–1008. https://doi.org/10.1016/0031-3203(91)90097-O

Storn, R., & Price, K. (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization, 11*(4), 341. https://doi.org/10.1023/A:1008202821328

Tan, W., Yang, Y., & Li, T. (2010). An improved COP-KMeans algorithm for solving constraint violation. In *Computational intelligence* (Vol. 4, pp. 690– 696). World Scientic.

Tran, D. H., Babaki, B., Van Daele, D., Leyman, P., & De Causmaecker, P. (2021). Local search for constrained graph clustering in biological networks. *Computers & Operations Research, 132*, 105299. https://doi.org/10.1016/j.cor.2021.105299

Wagstaff, K., Cardie, C., Rogers, S., & Schrödl, S. (2001). Constrained k-means clustering with background knowledge. In *ICML* (Vol. 1, pp. 577–584).

Xia, Y. (2009). A global optimization method for semi-supervised clustering. *Data Mining and Knowledge Discovery, 18*(2), 214–256. https://doi.org/10.1007/s10618-008-0104-3