Noname manuscript No.
(will be inserted by the editor)

Side-constrained minimum sum-of-squares clustering: Mathematical Programming and random projections

Leo Liberti · Benedetto Manca

Received: date / Accepted: date

Abstract This paper investigates a mathematical programming based methodology for solving the minimum sum-of-squares clustering problem, also known as the "k-means problem", in the presence of side constraints. We propose several exact and approximate mixed-integer linear and nonlinear formulations. The approximations are based on norm inequalities and random projections, the approximation guarantees of which are based on an additive version of the Johnson-Lindenstrauss lemma. We perform computational testing (with fixed CPU time) on a range of randomly generated and real data instances of medium size, but with high dimensionality. We show that when side constraints make k-means inapplicable, our proposed methodology — which is easy and fast to implement and deploy — can obtain good solutions in limited amounts of time.

Keywords MINLP, k-means, random projections, side constraints.

1 Introduction

Given a set P of n entities and some pairwise similarity function $P \times P \to \mathbb{R}$, cluster analysis aims at finding a set of k subsets $C_1, \ldots, C_k \subseteq P$ such that each cluster contains as many similar entities, and as few dissimilar entities, as possible. Cluster analysis — as a field — grew out of statistics in the course of the second half of the 20th century, encouraged by the advances in computing power. But some forms of

The first author has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement n. 764759 "MINOA". The second author was supported by KASBA, funded by *Regione Autonoma della Sardegna*.

Leo Liberti
CNRS LIX Ecole Polytechnique 91128 Palaiseau, France
E-mail: liberti@lix.polytechnique.fr

Benedetto Manca
Dip. Matematica e Informatica, Università degli Studi di Cagliari, Via Ospedale 72, 09124 Cagliari, Italy
E-mail: bmanca@unica.it

cluster analysis may also be attributed to earlier scientists (e.g. Aristotle, Buffon, Cuvier, Linné [36]).

We look at one of the most studied cluster analysis problems in Euclidean spaces:

Minimum Sum-of-Squares Clustering (MSSC). Given an integer $k > 0$ and a set $P \subset \mathbb{R}^m$ of n vectors, find a set $\mathcal{C} = \{C_1, \ldots, C_k\}$ of subsets of P such that the function

$$f(\mathcal{C}) = \sum_{j \leq k} \sum_{p \in C_j} \|p - \mathsf{centroid}(C_j)\|_2^2 \tag{1}$$

is minimum, where

$$\mathsf{centroid}(C_j) = \frac{1}{|C_j|} \sum_{p \in C_j} p. \tag{2}$$

The MSSC is the problem which the celebrated k-means algorithm [46] aims at solving. The k-means algorithm improves a given initial clustering $\mathcal{C}$ by means of the two following operations:

1. compute centroids $c_j = \mathsf{centroid}(C_j)$ for each $j \leq k$;
2. for any pair of clusters $C_h, C_j \in \mathcal{C}$ and any point $p \in C_h$, if p is closest to c_j than to c_h, move p from C_h to C_j.

These two operations are repeated until $\mathcal{C}$ no longer changes. Since the only decision operation (i.e. operation 2) carries out a change only if it decreases $f(\mathcal{C})$, it follows that k-means is a local descent algorithm. We note that it offers no guarantee on the approximation of the objective function with respect to the global optimum. It turns out that the MSSC problem can also be seen as a discrete analogue of the problem of partitioning a body into smaller bodies having minimum sum of moments of inertia [55].

In this paper we look at the MSSC from the point of view of Mathematical Programming (MP), and focus on a general class of MSSC variants obtained by adding more or less arbitrary side constraints. Achieving an almost complete flexibility in the nature of these constraints is possible because our solution methodology is MP-based. On the other hand, generality comes at an efficiency cost. We attempt to moderate this cost by reducing the dimensionality of the input data without losing too much accuracy.

Methodologically, we construct approximate MP formulations of the MSSC problems with reduced dimensionality. These formulations have desirable properties (their continuous relaxations are convex) and they are small enough that good feasible solutions can be obtained by off-the-shelf MP solvers with a time limit. Any side constraint can be applied to these formulations as long as the solver can handle them.

The main original theoretical contributions of this paper are:

– a new formulation for the MSSC which has a convex continuous relaxation (Sect. 3), and the corresponding norm approximations (Sect. 4.1);
– the application of random projections to the MSSC formulation and their approximations, as well as its theoretical analysis (Sect. 4.2).

The rest of this paper is organized as follows. In Sect. 2 we review the literature about the main topics we touch on. In Sect. 3 we derive a MP formulation of the MSSC the continuous relaxation of which is a convex program. In Sect. 4 we discuss some approximations of the formulation of the previous section. In Sect. 5 we present some computational experiments.

2 Preliminary notions

In this section we introduce some basic notions about the three main topics treated in this paper. We first discuss the MP formulation class of interest, namely Mixed-Integer Nonlinear Programming (MINLP). Then we discuss the dimensionality reduction technique we use on the formulations, namely Random Projection (RP), and their previous applications to MINLP. Then, we briefly survey the MSSC, with and without side constraints, and discuss existing applications of RPs to the MSSC.

2.1 MINLP

MP is a formal language for describing optimization problems in terms of parameters (the input), decision variables (the solution), an objective function to optimize, and some constraints to be satisfied: formal expressions for objective and constraints can be built up recursively from parameters, decision variables, numerical constants, and a finite number of mathematical functions and operators [34, 44, 65]. A valid MP sentence is called a *formulation*.

As with any formal language, formulations are interpreted by a program (called *solver*) which implements an algorithm for reading the input (i.e. assigning values to parameter symbols) and computing the output (i.e. searching for values to assign to decision variables that optimize the objective and satisfy the constraints). MP solvers usually aim at solving fairly large subclasses of MP, and try to strike a good compromise between generality and solution speed (two desirable characteristics which are most often in a trade-off balance). For an MP formulation $\mathcal{F}$, we denote $\mathsf{val}(\mathcal{F})$ the optimal objective function value of $\mathcal{F}$.

The basic taxonomy for MP formulations depends on the absence/presence of nonlinear terms in the objective function and constraints, on the absence/presence of integer variables, and on the provable convexity of the objective function and the feasible set. Linear forms in continuous variables yield Linear Programs (LP). Nonlinear terms in continuous variables yield Nonlinear Programs (NLP); if a proof of convexity of the objective and feasible set is available, we speak of convex NLP (cNLP). Variants of LP, NLP, cNLP with some integer variables are called MILP, MINLP, cMINLP. A note on cMINLP: obviously, any MP formulation with integer variables cannot have a convex feasible set. By "cMINLP" we mean a MINLP having a cNLP as a continuous relaxation (i.e. after relaxing all of the integrality constraints). The MINLP class obviously contains all of the other MP classes we mentioned. In general, MINLPs are undecidable [42]. If the feasible set can be proved bounded (e.g. if there are range constraints on the decision variables), then MINLPs become decidable, but most of the practically interesting

classes are **NP**-hard (SAT can easily be formulated as a MILP, which is a subset of MINLP).

When working with floating point numbers and nonlinear functions, using the Turing machine computation model is unnatural. We therefore often resort to the *real RAM* computation model [10], which assumes that elementary operations on reals can be carried out exactly in unit time, and to a variety of numerical analysis techniques to keep the algorithms as numerically stable as possible.

The main solution algorithm for general MINLP is spatial Branch-and-Bound (sBB) [54, 57, 8, 32], which identifies ε-approximate globally optimal objective function values and its corresponding optima. The sBB algorithm is nonetheless termed "exact" since we often think of it in the real RAM model of computation. It can be used heuristically by terminating its execution prematurely based on CPU time. More efficient solution methods can be used for cMINLP [25, 29, 30, 11]. Most cNLPs can be solved to global optimality by using any local descent method [28, 63].

sBB works by partitioning the search space recursively into regions and subregions. Guaranteed lower and and upper bounds for the objective function value are computed at each sub-region. A region is fathomed and then discarded if: (i) upper and lower bounds for the current node are within ε (in which case the solution yielding the upper bound on the region is taken as the ε-approximate global optimum for the region); (ii) if the lower bound for the current region is worse than the incumbent (in which case the current region has no chance of containing the global optimum of the problem). Thus, finding lower bounds for MINLP is important. The main techniques in this area are derived from relaxations of the original MINLP formulations. A *relaxation* is a reformulation the solution of which yields a lower bound to the optimal objective function value of the original formulation. Typically, relaxations are obtained by enlarging the feasible region of the original formulation: the globally optimal objective function value of the relaxation is then guaranteed to be a lower bound. In general, relaxing integer to continuous variables in MINLPs yields a nonconvex NLP relaxation, which is just as hard to solve as the original MINLP.

While MP solvers work best based on the mathematical properties of their input formulations, for any given optimization problem there exist infinitely many MP formulations that describe it. There arises an interest in modifying the mathematical properties of formulations while making sure that they still describe the problem either correctly or at least approximately. This modification process is called *reformulation* [43]. Reformulations that describe the problem correctly are called *exact*. Reformulations whose optimal objective function value is a guaranteed bound (in the optimization direction) of the optimal objective function value of the original formulation are called *relaxations*. Families of reformulations depending on a parameter, which "tend to" a correct description of the problem as the parameter tends to some value, are called *approximations* [41]; members of these families are called *approximate* reformulations.

In this paper we propose a new set of cMINLP reformulations of the MSSC, both exact and approximate.

2.2 Random projections

RPs are $d \times m$ matrices T which pre-multiply $m \times n$ data matrices P in view of obtaining a lower dimensional version TP of P with some approximation guarantee. The type of approximation we consider in this context concerns the Euclidean distances between pairs of columns of P. More precisely, the Johnson-Lindenstrauss Lemma (JLL) [39] states that, for any $P \in \mathbb{R}^{m \times n}$ and $\epsilon > 0$, there exists a $T \in \mathbb{R}^{d \times m}$ with $d = O(\frac{1}{\epsilon^2} \ln n)$ such that

$$\forall i, \ell \leq n \quad (1-\epsilon)\|P_i - P_\ell\|_2 \leq \|TP_i - TP_\ell\|_2 \leq (1+\epsilon)\|P_i - P_\ell\|_2, \tag{3}$$

where P_i, P_ℓ denote the i-th and ℓ-th columns of P.

It turns out that, by choosing T so that each component is sampled from a normal distribution $\mathsf{N}(0, \frac{1}{\sqrt{d}})$ (where $\frac{1}{\sqrt{d}}$ is the standard deviation), after some repeated samplings of T Eq. (3) is satisfied with probability exceeding $1 - \theta e^{u(d)}$, where θ is a universal constant and $u(\cdot)$ a linear function [21]. Many variants of this RP construction have been proposed in the literature, with respect to sparsity [1,58,3,2,60] and other properties [59].

We remark that the application of RP to MP problems is not new [66,50,51, 61,62,49,60,19] but recent. In particular, RPs applied to linear integer feasibility problems were partly discussed in [62], and RPs applied to quadratic programs were presented in [60,19]. But, as far as we know, this is the first application of RPs to a MINLP, albeit with a specific structure imposed by the underlying application.

2.3 The MSSC problem

The MSSC is a fundamental problem that has attracted a lot of attention from the data science, optimization, and algorithmic communities over the years. Thus, a considerable amount of literature is available, most of it under the keyword "k-means". Although most papers about this problem are actually about the k-means algorithm, almost every paper also comments on the problem formulation. An almost universal consideration is that the MSSC is based on two arrays of decision variables: $x_{ij} \in \{0,1\} = 1$ if point $p_i \in P$ (for $i \leq n$) is assigned to cluster $j \leq k$ and zero otherwise, and $y_j \in \mathbb{R}^m$ denoting $\mathsf{centroid}(C_j)$, the centroid of cluster j. This immediately gives a formulation of Eq. (1) in terms of x, y:

$$f(x,y) = \sum_{j \leq k} \sum_{\substack{i \leq n \\ x_{ij}=1}} \|p_i - y_j\|_2^2. \tag{4}$$

The function $f(x,y)$ cannot be employed "as is" as an objective function to be minimized, since the decision variable x appears in a sum quantifier, which makes $\min f(x,y)$ an invalid sentence in the MP language. We shall see in Sect. 3 that there is an easy reformulation of $f(x,y)$ so that it becomes valid.

The data science oriented survey [56] presents interesting matrix formulations of the MSSC. In particular, [56, Eq. (10)] shows that the MSSC is equivalent to

finding the projection matrix $x(x^\top x)^{-1}x^\top$ (where $x \in \{0,1\}^{n\times k}$ is the point-cluster assignment matrix, as in Eq. (4)) minimizing

$$f(x) = \mathsf{tr}(P^\top(I - x(x^\top x)^{-1}x^\top)P). \tag{5}$$

We note that [56, Eq. (33)] shows how Eq. (5) changes when a pre-conditioning matrix A is applied to the data matrix P. While these considerations do not bear a direct impact on the present work, they clarify the role that projections play in the MSSC, and, more specifically, the distinction between a dimensional reduction operator used as a preconditioner on the data (the matrix A), and a projection matrix derived by the binary decision variable matrix x. We note, as a curiosity, that: (i) [56, Eq. (10)] is ascribed to a referee of [33]; (ii) at the publication time of [56], Eq. (5) had not yielded any major progress on MSSC research; (iii) to the best of our knowledge, the latter is still the case.

The survey [9] is almost completely algorithmic, but reports an **NP**-hardness proof and mentions some inapproximability results. Notably, there exists an $\epsilon > 0$ such that it is **NP**-hard to approximate Eq. (1) within a factor of $(1 + \epsilon)$ for arbitrary n, k.

The introductions of [47,4] provide brief surveys to the MSSC from the point of view of MP. In particular, it was shown in [4] (and we shall see in Sect. 3) that the MSSC can be naturally formulated as a Mixed-Integer Nonlinear Program (MINLP), which is one of the largest subclasses of MP. We are going to reformulate this nonconvex MINLP formulation of the MSSC to a cMINLP. We are then going to apply a RP technique to the data points, and show that we obtain an *approximating reformulation* [41] of the MSSC having much smaller size.

2.3.1 Side constraints

The title of this paper refers to "side constraints". In Sect. 1 we stated that the object of our attention was a "general class of MSSC variants obtained by adding more or less arbitrary side constraints". We now give a more precise description of the classes of constraints that can be added to the formulations, as well as a literature review of the MSSC with side constraints.

Adjoining new constraints to an existing MP formulation may make the resulting problem harder or easier (both empirically and in terms of worst-case analysis), depending on the underlying problem and the constraints. In view of keeping solution difficulty from changing too much w.r.t. the underlying MSSC problem, we pose two limitations to the side constraints we allow.

(i) The new constraint set must consist of a polynomial number of constraints in function of a polynomial number of variables in the input size of the MSSC instance.
(ii) The most appropriate MP solver used for solving the underlying MSSC formulation must also be able to solve the formulation once the new constraint set is added.

Item (i) is designed to keep worst-case complexity of reading the formulation polynomially bounded. Item (ii) is somewhat informal, as the "most appropriate MP solver" for a given formulation is not well defined. Insofar as MP prescribes certain solution algorithm classes for certain formulation classes (e.g. sBB for nonconvex

MINLPs or NLPs, outer approximation or simpler types of Branch-and-Bound (BB) algorithms for cMINLPs, various types of local descent algorithms for convex NLPs, etc.), Eq. (ii) is a practical limitation. Obviously the new constraint set must be a valid sentence in the MP language.

Side constraints for clustering problems are usually categorized into "cluster-level" (e.g. imposing minimum or maximum cardinality on the clusters) and "instance-level" (e.g. requiring that certain given subsets of the points should be in the same or different clusters).

The literature about solving MSSC variants with side constraints goes under the name of "constrained clustering", and, perhaps unsurprisingly, mostly comes from the Constraint Programming (CP) community. There are two main methodological approaches: (a) trying to adapt k-means to the new constraints, and (b) using typical CP algorithms such as domain filtering and backtracking. Approach (a) trades efficiency off for generality, while (b) does the reverse (since it devises specific algorithms based on the constraint structure).

Here are some typical examples of papers in the group (a) above. Instance-level constraints concerning given pairs of points, called "must-link" (the two points must be in the same cluster) and "cannot-link" (the reverse), were first proposed in [64]. In [40], instance-level constraints are interpreted in a Euclidean space context, which allows the definition of implied constraints on the point positions: for example, nearby points to points involved in "must-link" constraints should also be "must-link" too; methodologically, a CP filtering step is added to the main clustering algorithm. In [22], more side constraints concerning sets of points are introduced; the problem determining the feasibility of general subsets of side constraints is shown to be **NP**-complete; and a k-means algorithm variant capable of integrating these side constraints is presented.

An MP approach to solving the MSSC with general side constraints is presented in [5], which builds on the column generation algorithms of [47,4], but integrates various types of side constraints by adapting the pricing subproblem and its solver. An approach based on CP-based modelling and solving is given in [35]. Another approach based on CP is given in [20], which proposes a global optimization constraint to represent Eq. (1), and uses CP filtering to handle the side constraints; the solution algorithm is based on dynamic programming. In [24], a general approach to solving bi-criteria constrained MSSCs is presented, based on an existing general-purpose CP solver. The tested instances, however, have very few features w.r.t. the those we consider in order to apply the dimensional reduction approach. Moreover, only one instance is solved using the classic MSSC objective function. This prevents any meaningful computational comparison between [24] and the methodology presented herein. Both approaches, however, provide for allowing general classes of side-constraints, each according to the specificities of the underlying languages (CP in [24], MP in this paper).

Compared to the approaches in the literature, we do not focus on the side constraint structure, which we leave as general as possible (similarly to [24]); neither do we focus on the solution algorithm, for which we exploit an off-the-shelf MP solver. Instead, we focus on the formulation, which we reduce to a cMINLP and then approximate using RPs.

2.3.2 Application of RPs to the MSSC

Based on Eq. (3), RPs are ideal candidates to decrease the dimensionality of the MSSC. In fact, RPs have been already applied to the k-means algorithm [13], yielding an approximation ratio of $2+\epsilon$. We offer a critique of the analysis of [13] in Sect. 4.2.1.

One of the most common approaches when applying RPs to the MSSC consists in considering a relaxation of the MSSC to a low rank approximation problem. To see this, consider the following formulation of the MSSC [13]:

$$\left.\begin{array}{rl} \min\limits_{\bar{x}\in\mathcal{X},Y} & \|P^\top - Y\|_F^2 \\ & \bar{x}\bar{x}^\top P^\top = Y, \end{array}\right\} \tag{6}$$

where $\mathcal{X}$ is the set of all $n\times k$ *cluster indicator matrices*, meaning that for any $\bar{x}\in\mathcal{X}$, $i\leq n$ and $j\leq k$ the i-th point belongs to the j-th cluster iff $\bar{x}_{ij}=\frac{1}{\sqrt{s_j}}$, where s_j is the cardinality of cluster j. We note that these $\bar{x}$ matrices have the same sparsity structure, but not the same entries, as the x matrices appearing in Eq. (4)-(5).

Since $\mathsf{rank}(\bar{x})\leq k$ for any $\bar{x}\in\mathcal{X}$ (by definition), we also have $\mathsf{rank}(Y)\leq k$ in Eq. (6). We can therefore relax the constraints $\bar{x}\bar{x}^\top P^\top = Y$ to $\mathsf{rank}(Y)\leq k$. By [26], the solution to this relaxation is given by the truncated singular value decomposition of the matrix P. We note that this relaxation is also equivalent to finding the rank-k projection operator $\bar{X}$ minimizing $\|P^\top - \bar{X}\bar{X}^\top P^\top\|_F^2$. Based on this observation, a $(1\pm\varepsilon)$ approximation for the optimal rank-k projection operator was achieved in [53] using RPs mapping m-dimensional vectors to either $O(k\log k + k/\varepsilon)$ or $O(k\log k\varepsilon^{-2})$ dimensions, depending on the type of RP used. These results were improved to $O(k\varepsilon^{-2})$ dimensions in [17].

These results on the aforementioned relaxation were adapted to the MSSC in [13], which achieves a $2+\varepsilon$ approximation error using RPs mapping to $O(k\varepsilon^{-2})$ dimensions with high probability (whp). This result was improved in [18] to a $(1\pm\varepsilon)$ approximation error; the same paper also presents a $9+\varepsilon$ approximation error with $O(\varepsilon^{-2}\log k)$ dimensions. It was shown in [6] that there is a RP mapping to $O(\frac{\log k+\log\log n}{\varepsilon^6}\log(\frac{1}{\varepsilon}))$ dimensions that preserves the cost of any k-clustering up to a $(1\pm\varepsilon)$ approximation error. It was also shown that it is possible to further reduce the projected dimension to $O(\frac{\log k+\log(1-\delta)}{\varepsilon^4}\log(\frac{1}{\varepsilon}))$ and obtain a $(1\pm\varepsilon)$ approximation error whp.

In another line of research following [53,17], a merge-and-reduce approach based on [14] which finds a (k,ε)-coreset for the MSSC objective function is proposed in [16].

3 A cMINLP formulation for the MSSC

The MSSC with side constraints can be naturally formulated as follows.

$$\left.\begin{array}{lrl} \min\limits_{x,y,s,\gamma} & \sum\limits_{i\leq n}\sum\limits_{j\leq k} \|p_i - y_j\|_2^2\, x_{ij} & \\ \forall j \leq k & \frac{1}{s_j}\sum\limits_{i\leq n} p_i x_{ij} = y_j & \\ \forall j \leq k & \sum\limits_{i\leq n} x_{ij} = s_j & \\ \forall i \leq n & \sum\limits_{j\leq k} x_{ij} = 1 & \\ \forall j \leq k & y_j \in \mathbb{R}^m & \\ & G(x,y,\gamma) \leq 0 & \\ & x \in \{0,1\}^{nk} & \\ & s \in \mathbb{N}^k. & \end{array}\right\} \quad \text{(MSSC)} \tag{7}$$

The parameters of formulation Eq. (7) are the given set $P = \{p_1, \ldots, p_n\}$ of vectors in $\mathbb{R}^m$, and the number of clusters k. The decision variables are:

- for each $i \leq n$ and $j \leq k$, the binary assignment variables x_{ij}, set to 1 iff vector i is assigned to cluster j and to 0 otherwise;
- for each $j \leq k$, the integer variables s_j, equal to the cardinality of cluster j;
- for each $j \leq k$, the vector $y_j \in \mathbb{R}^m$, containing the centroid of cluster j.

The formula $G(x, y, \gamma) \leq 0$ encodes the side constraints (possibly with additional decision variables γ), which may be arbitrary as long as they satisfy the limitations (i)-(ii) given in Sect. 2.3.1 above. Insofar as we focus on a cMINLP reformulation, we require that the continuous relaxation of the set $\{(x, y, \gamma) \mid G(x, y, \gamma) \leq 0\}$ should be convex, in line with limitation (ii). Since our reformulations are not going to exploit the structure of the side constraints, we do not list them explicitly in the reformulations below.

We note that Eq. (7) imposes a restriction on the MSSC solution, namely that each cluster must be non-empty, since the cardinality s_j appears in a denominator. This restriction can be relaxed by multiplying both sides of the equation constraint by s_j.

Solving Eq. (7) directly is unadvisable for several reasons. It has a mixture of continuous, binary and general integer variables; some decision variables appear in the denominator of a fraction; while the objective function consists of sums of products of convex terms, the products makes it (generally) nonconvex. We shall address all of these issues by using elementary reformulation steps, and construct a cMINLP reformulation of the MSSC.

3.1 Removing centroid constraints

The first reformulation of MSSC consists in eliminating the centroid constraints. We shall see that this is an *exact reformulation* (i.e. preserving global optima).

$$\left.\begin{array}{lll} \min\limits_{x,y} \sum\limits_{i\leq n}\sum\limits_{j\leq k} \|p_i - y_j\|_2^2\, x_{ij} & & \\ \forall j \leq k & \sum\limits_{i\leq n} x_{ij} \geq 1 & \\ \forall i \leq n & \sum\limits_{j\leq k} x_{ij} = 1 & \\ \forall j \leq k & y_j \in \mathbb{R}^m & \\ & x \in \{0,1\}^{nk}. & \end{array}\right\} \tag{8}$$

Lemma 3.1
For any $j \leq k$, $v = \mathsf{centroid}(C_j)$ iff $\sum\limits_{p\in C_j} \|p - v\|_2^2$ is minimum over all $v \in \mathbb{R}^m$.

Proof By [4, p. 199]. □

Proposition 3.2
$\mathcal{C}^*$ is a global optimum of Eq. (8) *iff it is also a global optimum of* MSSC.

Proof Note that any clustering $\mathcal{C}$ is completely defined by the binary assignment variables x, since once the values of x are known, one can easily compute centroids y and their cardinalities s. Moreover, because of the constraint $\sum_i x_{ij} \geq 1$ in Eq. (8), no cluster in $\mathcal{C}^*$ may be empty; and, since the term $1/s_j$ in MSSC forces $s_j \geq 1$ for all $j \leq k$, the same must also hold for an optimum of MSSC. Let x^* be the solution for the binary variables x determined by $\mathcal{C}^*$: since the objective functions of the two formulations (MSSC and Eq. (8)) are identical, when x is fixed at x^*, their optimal values w.r.t. y must match, whence x^* yields the same optimal objective function values in both. If we denote $f^1(x)$ the objective function of MSSC and $f^2(x)$ that of Eq. (8), we have

$$f^1(x^*) = f^2(x^*). \qquad (\dagger)$$

We also observe that Eq. (8) was obtained by MSSC by removing the constraints $\frac{1}{s_j}\sum_i p_i x_{ij} = y_j$ and $\sum_i x_{ij} = s_j$, and adjoining the constraint $\sum_i x_{ij} \geq 1$. But the latter is always satisfied in MSSC as observed above, which implies that Eq. (8) is a relaxation of MSSC, i.e.

$$\min_x f^1(x) \geq \min_x f^2(x). \qquad (\ddagger)$$

($\Rightarrow$) Eq. ($\dagger$) implies $\min_x f^1(x) \leq \min_x f^2(x)$ since x^* is a global optimum of Eq. (8). By Eq. ($\ddagger$), we have $\min_x f^1(x) = \min_x f^2(x)$, showing that x^* is a global optimum of MSSC.

($\Leftarrow$) Let $\mathcal{C}^*$ be a global optimum for MSSC corresponding to x^* and suppose x^* is not a global optimum for Eq. (8). By Eq. ($\dagger$)-($\ddagger$), x^* is feasible in Eq. (8) and has the same objective function value, so the only way it can fail to be a global optimum is that $\mathcal{C}^*$ is not an optimal clustering for Eq. (8). So let $\mathcal{C}'$ be an optimal clustering for Eq. (8) with corresponding variables (x', y', s'): the only way it can be better than $\mathcal{C}^*$ is that it should be infeasible for MSSC. By Lemma 3.1, y' defines the centroids of the clusters of $\mathcal{C}'$. Moreover, $s'_j = \sum_i x'_{ij} \geq 1$ makes $1/s'_j$ well defined. Hence (x', y', s') must be feasible in MSSC, against the assumption. □

The advantage of Eq. (8) w.r.t. MSSC is that the former has fewer nonlinear terms as well as fewer constraints.

3.2 Linearization of products

In this section we reformulate products of terms involving decision variables in an exact way. While this is not generally possible, when at least one of the terms in the product takes a finite set of values and the other can be constrained to lie in a set of variable ranges, it becomes possible [43, §3.3]. In the case of the formulation in Eq. (8), we aim at linearizing all products $\alpha_{ij}(x,y) = \|p_i - y_j\|_2^2 x_{ij}$ over all $i \leq n$ and $j \leq k$.

We first remark that, even though the centroid variables y are unconstrained in Eq. (8), no centroid may ever lie outside the hyper-rectangle

$$[y^L, y^U] = [\min_{i \leq n} p_i, \max_{i \leq n} p_i], \tag{9}$$

where the min and max operators are applied componentwise to the vectors. This means that $\|p_i - y_j\|_2^2$ lies in $[0, P^U]$ for any $i \leq n$, $j \leq k$, where

$$P^U = \max_{i<h\leq n} \|p_i - p_h\|_2^2. \tag{10}$$

Next, we replace the products denoted with $\alpha_{ij}(x,y)$ by additional variables $\chi_{ij} \in [0, P^U]$ in the objective function, and adjoin the defining constraints $\chi_{ij} = \alpha(x,y)$ for each i, j. Since $x_{ij} \in \{0,1\}$, we have $\chi_{ij} \in [0, P^U]$ for all i, j. Again because the x variables are binary,

$$\chi_{ij} = \begin{cases} \|p_i - y_j\|_2^2 & \text{if } x_{ij} = 1 \\ 0 & \text{if } x_{ij} = 0. \end{cases}$$

For each $i \leq n$, $j \leq k$, let $D = \{0,1\} \times [y^L, y^U]$, and

$$A_{ij} = \{(\chi_{ij}, x_{ij}, y_j) \mid \chi_{ij} = \|p_i - y_j\|_2^2 x_{ij} \wedge (x_{ij}, y_j) \in D\}$$
$$B_{ij} = \{(\chi_{ij}, x_{ij}, y_j) \mid 0 \leq \chi_{ij} \leq P^U x_{ij} \wedge \|p_i - y_j\|_2^2 \leq \chi_{ij} + P^U(1 - x_{ij}) \wedge (x_{ij}, y_j) \in D\}.$$

It is easy to verify by inspection that $A_{ij} \subseteq B_{ij}$ by checking the two cases $x_{ij} = 0$ and $x_{ij} = 1$. Now let

$$\bar{B}_{ij} = \arg\min_{\chi_{ij}} B_{ij}.$$

We claim that $A_{ij} = \bar{B}_{ij}$. Let $\beta' = (\chi'_{ij}, x'_{ij}, y'_j) \in \bar{B}_{ij}$, then χ'_{ij} must be minimal in B_{ij}. If $x'_{ij} = 0$ then $\chi'_{ij} = 0$ which yields $\beta' \in A_{ij}$. If $x'_{ij} = 1$ then $\chi'_{ij} = \|p_i - y'_j\|_2^2$, which again implies that $\beta' \in A_{ij}$. Therefore $A_{ij} \supseteq \bar{B}_{ij}$. Conversely, if $\beta' \in A_{ij}$ then $\beta' \in B_{ij}$ as shown above. In order to obtain $\chi'_{ij} = 0$ if $x'_{ij} = 0$ and $\chi'_{ij} = \|p_i - y_j\|_2^2$ if $x'_{ij} = 1$ then χ'_{ij} must be minimal in B_{ij}, i.e. $\beta' \in \bar{B}_{ij}$ as claimed.

Thus, we can reformulate Eq. (8) as follows:

$$\left.\begin{array}{rr} \min\limits_{\chi,x,y} \sum\limits_{i\leq n}\sum\limits_{j\leq k} \chi_{ij} & \\ \forall i \leq n, j \leq k & \chi_{ij} \leq P^U x_{ij} \\ \forall i \leq n, j \leq k & \|p_i - y_j\|_2^2 \leq \chi_{ij} + P^U(1-x_{ij}) \\ \forall j \leq k & \sum\limits_{i\leq n} x_{ij} \geq 1 \\ \forall i \leq n & \sum\limits_{j\leq k} x_{ij} = 1 \\ \forall i \leq n, j \leq k & \chi_{ij} \geq 0 \\ \forall j \leq k & y_j \in [y^L, y^U] \\ & x \in \{0,1\}^{nk}. \end{array}\right\} \tag{11}$$

By the above discussion, Eq. (11) is an exact reformulation of Eq. (8) and hence, by transitivity, also of MSSC. The advantage of Eq. (11) w.r.t. Eq. (8) is that the former is a cMINLP instead of a (nonconvex) MINLP.

4 Approximating reformulations

In this section we introduce two types of approximating reformulations: one based on replacing the ℓ_2 norm with ℓ_1 or ℓ_∞ (the so-called "linearizable norms"), and the other obtained by applying a RP to the point set P.

4.1 Linearizable norms

In this section we exploit the well-known inequalities

$$\|\zeta\|_\infty^2 \leq \|\zeta\|_2^2 \leq \|\zeta\|_1^2 \tag{12}$$

$$\frac{1}{m}\|\zeta\|_1^2 \leq \|\zeta\|_2^2 \leq m\|\zeta\|_\infty^2 \tag{13}$$

which hold for every $\zeta \in \mathbb{R}^m$. We replace the ℓ_2 norm in MSSC with the ℓ_1 and ℓ_∞ norms, for which we provide approximate MILP reformulations of Eq. (11). Since MILP solvers (such as e.g. [37]) are technologically more advanced than cMINLP solvers (such as e.g. [12]), we can hope to solve larger instances of MSSC with the MILP formulations: these will derive bounds on the optimal objective values by means of Eq. (12)-(13).

4.1.1 The ℓ_∞ norm

We are going to replace $\|p_i - y_j\|_2^2$ in MSSC by $\|p_i - y_j\|_\infty$, and compute P^U with the ℓ_∞ norm. The same reformulations follow through, and we get to a variant of Eq. (11) where the (convex) nonlinear constraints are:

$$\|p_i - y_j\|_\infty \leq \chi_{ij} + P^U(1-x_{ij}) \tag{14}$$

for all $i \leq n, j \leq k$. This is equivalent to

$$\max_{\ell\leq m} |p_{i\ell} - y_{j\ell}| \leq \chi_{ij} + P^U(1-x_{ij})$$

which can be reformulated to

$$\forall \ell \leq m \quad |p_{i\ell} - y_{j\ell}| \leq \chi_{ij} + P^U(1 - x_{ij}),$$

whence

$$\begin{aligned} \forall \ell \leq m \quad p_{i\ell} - y_{j\ell} &\leq \chi_{ij} + P^U(1 - x_{ij}) \\ \forall \ell \leq m \quad y_{j\ell} - p_{i\ell} &\leq \chi_{ij} + P^U(1 - x_{ij}). \end{aligned}$$

Note that the replacement of a square ℓ_2 norm with a non-square ℓ_∞ norm makes χ_{ij} take a linear, rather than squared, value at the optimum (as long as $x_{ij} = 1$). Thus the terms χ_{ij} on the objective function should be squared.

This yields the following approximating reformulation:

$$\left.\begin{array}{rl} \min\limits_{\chi,x,y} \sum\limits_{i\leq n}\sum\limits_{j\leq k} \chi_{ij}^2 & \\ \forall i \leq n, j \leq k \quad \chi_{ij} & \leq P^U x_{ij} \\ \forall i \leq n, j \leq k, \ell \leq m \quad p_{i\ell} - y_{j\ell} & \leq \chi_{ij} + P^U(1 - x_{ij}) \\ \forall i \leq n, j \leq k, \ell \leq m \quad y_{j\ell} - p_{i\ell} & \leq \chi_{ij} + P^U(1 - x_{ij}) \\ \forall j \leq k \quad \sum\limits_{i\leq n} x_{ij} & \geq 1 \\ \forall i \leq n \quad \sum\limits_{j\leq k} x_{ij} & = 1 \\ \forall i \leq n, j \leq k \quad \chi_{ij} & \geq 0 \\ \forall j \leq k \quad y_j & \in [y^L, y^U] \\ x & \in \{0,1\}^{nk}, \end{array}\right\} \tag{15}$$

which is again a cMINLP, where the only nonlinearities are in the objective function (the constraints are wholly linear).

Lastly, we propose to optimize the following linear objective function:

$$\min \sum_{i\leq n}\sum_{j\leq k} \chi_{ij} \tag{16}$$

instead of the quadratic form in Eq. (15), so as to be able to use a MILP solver. We remark that this last step is wholly heuristic. In the worst case, it may find clusterings which are arbitrarily different from the optima of the MSSC problem.

4.1.2 The ℓ_1 norm

Similarly to Sect. 4.1.1, we are going to replace $\|p_i - y_j\|_2^2$ in MSSC by $\|p_i - y_j\|_1$ for each $i \leq n$, $j \leq k$, and compute P^U with the ℓ_1 norm. Again, we get to a variant of Eq. (11) where the (convex) nonlinear constraints are:

$$\|p_i - y_j\|_1 \leq \chi_{ij} + P^U(1 - x_{ij}) \tag{17}$$

for all $i \leq n, j \leq k$. This is equivalent to

$$\sum_{\ell\leq m} |p_{i\ell} - y_{j\ell}| \leq \chi_{ij} + P^U(1 - x_{ij})$$

which, by means of some additional variables $\eta \in \mathbb{R}^{nkm}$ can be reformulated to

$$\begin{aligned} \forall \ell \leq m \quad -\eta_{ij\ell} \leq p_{i\ell} - y_{j\ell} \leq \eta_{ij\ell} \\ \sum_{\ell \leq m} \eta_{ij\ell} \leq \chi_{ij} + P^U(1 - x_{ij}). \end{aligned}$$

As in Sect. 4.1.1, the terms χ_{ij} on the objective function should be squared.

This yields the following approximating reformulation:

$$\left.\begin{array}{rl} \min\limits_{\chi,x,y} \sum\limits_{i \leq n} \sum\limits_{j \leq k} \chi_{ij}^2 & \\ \forall i \leq n, j \leq k & \chi_{ij} \leq P^U x_{ij} \\ \forall i \leq n, j \leq k, \ell \leq m & p_{i\ell} - y_{j\ell} \leq \eta_{ij\ell} \\ \forall i \leq n, j \leq k, \ell \leq m & p_{i\ell} - y_{j\ell} \geq -\eta_{ij\ell} \\ \forall i \leq n, j \leq k & \sum\limits_{\ell \leq m} \eta_{ij\ell} \leq \chi_{ij} + P^U(1 - x_{ij}) \\ \forall j \leq k & \sum\limits_{i \leq n} x_{ij} \geq 1 \\ \forall i \leq n & \sum\limits_{j \leq k} x_{ij} = 1 \\ \forall i \leq n, j \leq k & \chi_{ij} \geq 0 \\ \forall j \leq k & y_j \in [y^L, y^U] \\ & x \in \{0,1\}^{nk}. \end{array}\right\} \quad (18)$$

We again obtain a cMINLP where the only nonlinearities are in the objective function. The same comment about heuristically optimizing Eq. (16) with a MILP solver holds.

4.1.3 Approximation guarantees

Proposition 4.1
If χ^*, x^*, y^* is a global optimum of Eq. (15) (resp. Eq. (18)), then the globally optimal objective function value $\sum_{i,j}(\chi^*_{ij})^2$ is equal to the globally minimal value of $f(\mathcal{C})$ (see Eq. (1)) with the ℓ_2 norm replaced by the ℓ_∞ norm (resp. the ℓ_1 norm).

Proof If p_i is not assigned to the cluster C_j in Eq. (1), then $x_{ij} = 0$ and $\chi^*_{ij} = 0$ by the optimization direction and Eq. (14) (resp. Eq. (17)). For the same reasons, if p_i is assigned to C_j then $x_{ij} = 1$ and $\chi^*_{ij} = \|p_i - y_j\|_\infty$ (resp. $\chi^*_{ij} = \|p_i - y_j\|_1$). □

By Eq. (12)-(13) and Prop. 4.1, we have:

$$\begin{aligned} \mathsf{val}(15) \leq \mathsf{val}(\mathsf{MSSC}) \leq \mathsf{val}(18) \\ \frac{1}{m}\mathsf{val}(18) \leq \mathsf{val}(\mathsf{MSSC}) \leq m\,\mathsf{val}(15), \end{aligned} \quad (19)$$

whence

$$\max\left(\mathsf{val}(15), \frac{1}{m}\mathsf{val}(18)\right) \leq \mathsf{val}(\mathsf{MSSC}) \leq \min\left(\mathsf{val}(18), m\,\mathsf{val}(15)\right) \quad (20)$$

and

$$\frac{1}{m}\mathsf{val}(\mathsf{MSSC}) \leq \mathsf{val}(15) \leq \mathsf{val}(18) \leq m\,\mathsf{val}(\mathsf{MSSC}). \quad (21)$$

Eq. (20)-(21) is most useful for cases where m is fixed and small (e.g. clustering in the plane).

4.2 Randomly projected formulations

We now address the computational issues arising from assuming m large by pre-multiplying the vector set P (seen as an $m \times n$ matrix) by a $d \times m$ RP matrix $T = (T_{h\ell})$, where $d = O(\frac{1}{\epsilon^2} \ln n)$ and $\epsilon > 0$ appears in the approximation guarantee in Eq. (3). We obtain the same formulations and reformulations as above, with p_i replaces by Tp_i, as well as y_j replaced by Ty_j. While P is given, so TP can be computed, Ty_j depends on a decision variable. More precisely,

$$\forall j \leq k \quad Ty_j = \left(\sum_{\ell \leq m} T_{h\ell} y_{j\ell} \;\middle|\; h \leq d \right)^\top.$$

We employ an additional variable matrix $z \in \mathbb{R}^{kd}$, replace Ty_j by z_j, and relax the defining constraints $z_j = Ty_j$, yielding projected versions of MSSC, Eq. (11), Eq. (15), Eq. (18) where $p_i - y_j \in \mathbb{R}^m$ is replaced by $Tp_i - z_j \in \mathbb{R}^d$. If $m \gg 1$ and $d = O(\ln n)$, these projected reformulations have considerably fewer variables than their original counterparts.

In the rest of this section we only consider the MSSC formulation (i.e. Eq. (7)), since the others follow by exact or approximate reformulation operations, whether we use P or TP as data. We denote the randomly projected MSSC formulation by TMSSC.

4.2.1 Applicability of Boutsidis' approach [13]

In [13], the authors provide an analysis of a k-means algorithm which solves the TMSSC instead of the MSSC. The formulation they consider is a slight modification of Eq. (6), where Y is replaced with its definition:

$$\min_{X \in \mathcal{X}} \|P^\top - XX^\top P^\top\|_F^2. \tag{22}$$

We recall that $\mathcal{X}$ is the set of clustered indicator matrices (see Sect. 2.3.2).

The result provided in [13] is as follows. Suppose the $m \times n$ data matrix P (containing the n points in $\mathbb{R}^m$ as columns) is replaced by TP, where T is a $d \times m$ RP matrix with $d \geq \tilde{c}k/\varepsilon^2$, such that $\tilde{c}$ is an instance-independent constant, k is the number of clusters, and $\varepsilon \in (0, \frac{1}{3})$. Then, any exact algorithm for the MSSC turns into an approximation algorithm with error bounded above by $2+\varepsilon$, whp. As stated, the given result clearly applies to our setting. This should let us conclude that our approximate formulations have a $2 + \varepsilon$ error bound at worst whp.

Intuitively, we find this result surprising. It is well known that the JLL essentially says that the reduced dimension d (given as $O(\ln n/\varepsilon^2)$ in [39]) is independent of the original dimension [39,21]. On the other hand, the result proposed in [13] appears to take this independence one (sizable) step further: it claims that $d = O(k/\varepsilon^2)$, which makes the reduced dimension independent of the the number of points too. The proof of [13, Thm. 1] rests heavily on [53], and proposes a result [13, Lemma 2] which summarizes other results from [53, Lemma 6, Lemma 8, Cor. 11]. In particular, transposed in the notation of the present paper, the first part of [13, Lemma 2] states that

$$\forall j \leq k \quad |1 - \sigma_j(TU_k)| \leq \varepsilon \tag{23}$$

whp, where $\sigma_j(\cdot)$ is the j-th singular value of the argument matrix TU_k, and U_k is the $m \times k$ matrix of left singular vectors of P corresponding to the diagonal matrix Σ_k of top k singular values of P in non-increasing order. Among the results in [53], the only one that proves Eq. (23) is [53, Cor. 11], which, however, states that, in this case, the reduced dimension d should be $d = O(f(\delta)k \ln(k/\varepsilon)/\varepsilon^2)$, where $f(\delta)$ controls the probability $1-\delta$ of success, as explained in [53, Defn. 1]. On the other hand, [53, Cor. 11] is a corollary of [53, Lemma 10], which is introduced as follows:

> The JLL states that k vectors from $\mathbb{R}^m$ can be embedded into $O(\ln(k)/\varepsilon^2)$ dimensions [...]. As a consequence we prove that given a k-dimensional subspace [...], embedding it into $O(k \ln(k/\varepsilon)/\varepsilon^2)$ dimensions preserves the length of all vectors from V. (†)

Therefore, the "k" symbol appearing in [53, Cor. 11] corresponds to the number n of points in this paper (we denote by (‡) this symbolic correspondence between k in [53] and n). So, what [53, Cor. 11] implies is that, for Eq. (23) to hold, we need $d = O(f(\delta)n \ln(n/\varepsilon)/\varepsilon^2)$.

We now consider the value of d. For the two estimates

- $d = O(k/\varepsilon^2)$, given in [13, Step 1, Alg. 1]
- $d = O(f(\delta)n \ln(n/\varepsilon)/\varepsilon^2)$, given in [53]

to match, we should either have

$$f(\delta) = \frac{k}{n \ln(n/\varepsilon)} \tag{24}$$

or $k = n$ and $f(\delta) = \frac{1}{\ln(k/\varepsilon)}$. This latter case cannot hold since the number of clusters k and the number of points n must differ for the MSSC to be nontrivial.

In our opinion, the former case of Eq. (24) also cannot hold: the assumption made in (†), considering the change of symbol (‡), is that [53, Lemma 10] is a statement about embedding an n-dimensional subspace of $\mathbb{R}^m$ in d dimensions. The paper [13], however, uses the number of clusters k rather than the number of points n.

We were not able to find a proof of the first part of [53, Lemma 2] that circumvents the difficulties just discussed.

4.2.2 Applicability of the JLL [39]

An alternative to invoking [13] would be to simply rely on the approximation of Euclidean distances given by the JLL [39], which would directly apply to Eq. (4). We pursue a different critique for this alternative.

The symbols y_j appearing in Eq. (4) are decision variables ranging in continuous space. As such, they represent a potentially uncountable infinity of vectors. The JLL, however, only applies to finite subsets of vectors.

Observe, however, that there are only as many centroids as there are possible different partitions of n entities in k clusters — so the number of vectors assigned to the y variables may not be known in advance, but it is not infinite. In the worst case, there are B_n partitions of n elements, where B_n is the n-th Bell number [7]. It turns out that B_n grows like a product of two exponentials in n [45, §1.14, Problem 9], and that $\ln B_n$ behaves asymptotically like $O(n \ln n)$ (plus smaller terms) [15,

p. 108]. Thus, the JLL would yield $d = O(\ln B_n/\varepsilon^2) = O(n \ln n/\varepsilon^2)$. This would only be useful in cases where $n \ll m$, which are a minority.

We also note that the k-means algorithm is not expected to run into the worst case from complete enumeration: since k-means is a heuristic, it is expected to reach termination reasonably quickly, after having examined only a few indicator matrices. This also holds for heuristic utilizations of exact algorithms, such as e.g. BB algorithms with a time or iteration limit. The JLL therefore provides a simple and valid analysis for such cases, which are the cases we actually test in practice in this paper.

On the other hand, we believe that a formulation-based analysis (i.e. independent of the algorithm used) would be better. We address this issue next.

4.2.3 The additive JLL for infinite sets

In this section we prove a result that is similar to [13, Thm. 1], but which avoids the issues described in Section 4.2.1. In view of Sect. 4.2.2 we do not employ the JLL, but a similar result that also applies to infinite sets, namely the *additive JLL* (see Thm. 4.2 below).

We first introduce two quantities, the sub-Gaussian norm of a sub-Gaussian random variable, and the Gaussian width of a set. A random variable X is *sub-Gaussian* iff there exists a constant K such that:

$$\forall t \geq 0 \quad \mathsf{Prob}(|X| \geq t) \leq 2e^{-(t/K)^2}.$$

We denote the *sub-Gaussian norm* of X by

$$\|X\|_{\psi_2} = \inf\{t > 0 \mid \mathbb{E}(e^{(X/t)^2}) \leq 2\}.$$

Given a set $S \subseteq \mathbb{R}^m$, the *Gaussian width* of S is

$$w(T) = \mathbb{E}\{\sup_{x \in S} \langle g, x \rangle \mid g \sim \mathsf{N}(0, I_m)\},$$

where the expectation is computed over all multivariate normal samples g.

Theorem 4.2
Let $S \subset \mathbb{R}^m$, and consider a $d \times m$ matrix T' having independent isotropic sub-Gaussian random vectors T'_i for rows, $K = \max_{h \leq d} \|T'_i\|_{\psi_2}$, and $T = \frac{1}{\sqrt{d}} T'$. Then, with high probability, there exists a universal constant κ such that:

$$\forall x, y \in S \quad \|x - y\|_2 - \delta \leq \|Tx - Ty\|_2 \leq \|x - y\|_2 + \delta, \tag{25}$$

where $\delta = \frac{\kappa K^2 w(S)}{\sqrt{d}}$.

Proof See [59, Prop. 9.3.2]. □

Our treatment follows the same logical order as the results in [53,13]. Each of the results below corresponds to a result in [53,13]. We adapted the proofs to employ the additive RP T satisfying Theorem 4.2 instead of the standard JLL. We first show that T preserves inner products approximately.

Corollary 4.3 *For every pair of elements $x, y \in \mathbb{R}^m$, the following holds with high probability*

$$|\langle Tx, Ty\rangle - \langle x, y\rangle| \leq \frac{\delta}{2}\|x\|_2\|y\|_2.$$

Proof We observe that the parallelogram rule and Eq. (25) imply that, whp, for any $x, y \in \mathbb{R}^m$

$$\begin{aligned} 4\langle Tx, Ty\rangle &= \|Tx + Ty\|_2^2 - \|Tx - Ty\|_2^2 \\ &\geq \|x + y\|_2^2 - \delta - \delta - \|x - y\|_2^2 \\ &= 4\langle x, y\rangle - 2\delta. \end{aligned}$$

Therefore, $\langle Tx, Ty\rangle - \langle x, y\rangle \geq -\frac{\delta}{2}$. Analogously, we obtain $\langle Tx, Ty\rangle - \langle x, y\rangle \leq \frac{\delta}{2}$. Thus,

$$|\langle Tx, Ty\rangle - \langle x, y\rangle| \leq \frac{\delta}{2}. \tag{26}$$

Since T is linear, we have

$$\langle Tx, Ty\rangle = \|x\|_2\|y\|_2 \, \langle Tx/\|x\|_2, Ty/\|y\|_2\rangle. \tag{27}$$

The result follows by replacement of Eq. (27) in Eq. (26). □

Lemma 4.4 *For every $x, y \in \mathbb{R}^m$, the following hold:*

(i) $\mathbb{E}(\langle Tx, Ty\rangle) = \langle x, y\rangle$
(ii) $\mathsf{Var}(\langle Tx, Ty\rangle) \leq \frac{\delta^2+4\delta}{4}\|x\|_2^2\|y\|_2^2$.

Proof We recall that $T = \frac{1}{\sqrt{d}}T'$, thus

$$\langle Tx, Ty\rangle = \frac{1}{d}\langle T'x, T'y\rangle. \tag{28}$$

Moreover,

$$\left.\begin{aligned} T'x &= \big(\langle T'_1, x\rangle, \ldots, \langle T'_d, x\rangle\big) \\ T'y &= \big(\langle T'_1, y\rangle, \ldots, \langle T'_d, y\rangle\big). \end{aligned}\right\}$$

Thus, we can write

$$\langle T'x, T'y\rangle = \langle T'_1, x\rangle\langle T'_1, y\rangle + \cdots + \langle T'_d, x\rangle\langle T'_d, y\rangle. \tag{29}$$

We first prove (i) for $x = y$. In this case Eq. (29) becomes

$$\langle T'x, T'x\rangle = \langle T'_1, x\rangle^2 + \cdots + \langle T'_d, x\rangle^2.$$

Therefore,

$$\begin{aligned} \mathbb{E}(\langle T'x, T'x\rangle) &= \mathbb{E}(\langle T'_1, x\rangle^2) + \cdots + \mathbb{E}(\langle T'_d, x\rangle^2) \\ &= \|x\|_2^2 + \cdots + \|x\|_2^2 \\ &= d\|x\|_2^2, \end{aligned}$$

where we have used the isotropy of the random vectors T'_i. From Eq. (28) we obtain

$$\mathbb{E}(\langle Tx, Tx\rangle) = \langle x, x\rangle. \tag{30}$$

In order to prove (i) for generic $x, y \in \mathbb{R}^m$, it is enough to apply Eq. (30) to the vector $x - y$ and use the linearity of the expected value. For the proof of (ii) we first

set $X = \langle Tx, Ty\rangle - \mathbb{E}(\langle Tx, Ty\rangle) = \langle Tx, Ty\rangle - \langle x, y\rangle$. The properties of the variance and (i) give

$$\mathbb{E}(X^2) = \mathsf{Var}(X) + \mathbb{E}(X)^2 = \mathsf{Var}(\langle Tx, Ty\rangle).$$

Moreover,

$$\begin{aligned}\mathbb{E}(X^2) &= \mathbb{E}(\langle Tx, Ty\rangle^2 + \langle x, y\rangle^2 - 2\langle Tx, Ty\rangle\langle x, y\rangle)\\ &= \mathbb{E}(\langle Tx, Ty\rangle^2) + \langle x, y\rangle^2 - 2\langle x, y\rangle^2\\ &= \mathbb{E}(\langle Tx, Ty\rangle^2) - \langle x, y\rangle^2.\end{aligned}$$

From Corollary 4.3 we have

$$\langle Tx, Ty\rangle^2 \leq \langle x, y\rangle^2 + \frac{\delta}{4}\|x\|_2^2\|y\|_2^2 + \delta\langle x, y\rangle\|x\|_2\|y\|_2. \tag{31}$$

Exploiting Eq. (31) and $\langle x, y\rangle \leq \|x\|_2\|y\|_2$ we obtain

$$\begin{aligned}\mathsf{Var}(\langle Tx, Ty\rangle) &= \mathbb{E}(X^2) = \mathbb{E}(\langle Tx, Ty\rangle^2) - \langle x, y\rangle^2\\ &\leq \mathbb{E}(\langle x, y\rangle^2 + \tfrac{\delta}{4}\|x\|_2^2\|y\|_2^2 + \delta\langle x, y\rangle\|x\|_2\|y\|_2) - \langle x, y\rangle\\ &= \langle x, y\rangle^2 + \tfrac{\delta}{4}\|x\|_2^2\|y\|_2^2 + \delta\langle x, y\rangle\|x\|_2\|y\|_2 - \langle x, y\rangle\\ &\leq \tfrac{\delta^2}{4}\|x\|_2^2\|y\|_2^2 + \delta\|x\|_2^2\|y\|_2^2\\ &= \tfrac{\delta^2+4\delta}{4}\|x\|_2^2\|y\|_2^2,\end{aligned}$$

which concludes the proof. □

Following the work of Sarlos [53], it is possible to use Corollary 4.3 and Lemma 4.4 to prove that the RP T can also be used to approximate the product of two matrices.

Lemma 4.5 *Let $A \in \mathbb{R}^{r\times m}$ and $B \in \mathbb{R}^{m\times s}$ be two matrices. Then the following hold:*

(a) $\|AB - AT^\top TB\|_F \leq \frac{\delta}{2}\|A\|_F\|B\|_F$ *whp;*
(b) $\mathbb{E}(AT^\top TB) = AB$;
(c) $\mathbb{E}(\|AB - AT^\top TB\|_F^2) \leq \frac{\delta^2+4\delta}{4}\|A\|_F^2\|B\|_F^2$.

Proof Set $a_i = A_i$ and $b_j = B^j$. Then,

$$(AT^\top)_i = Ta_i, \quad (TB)^j = TB_j.$$

Let $Y_{ij} = (AB)_{ij} - (AT^\top TB)_{ij} = \langle a_i, b_j\rangle - \langle Ta_i, Tb_j\rangle$, then from Corollary 4.3 we have that, whp,

$$|Y_{ij}| \leq \frac{\delta}{2}\|a_i\|_2\|b_j\|_2.$$

Hence,

$$\|AB - AT^\top TB\|_F^2 = \sum_{i,j} Y_{ij}^2 \leq \sum_{ij}\frac{\delta^2}{4}\|a_i\|_2^2\|b_j\|_2^2 = \frac{\delta^2}{4}\|A\|_F^2\|B\|_F^2,$$

which proves (a). Applying Lemma 4.4 to the random variable Y_{ij} we obtain $\mathbb{E}(Y_{ij}) = 0$ for all i, j and thus (b). Moreover, $\mathbb{E}(Y_{ij}^2) = \mathsf{Var}(\langle Ta_i, Tb_j\rangle)$ and, from Lemma 4.4 we obtain

$$\mathbb{E}(\|AB - AT^\top TB\|_F^2) = \sum_{i,j}\mathbb{E}(Y_{ij}^2) \leq \frac{\delta^2+4\delta}{4}\|A\|_F^2\|B\|_F^2,$$

which concludes the proof. □

Moreover, the RP T almost preserves the Frobenius norm of any matrix, in the sense of the next result.

Lemma 4.6 *Let $C \in \mathbb{R}^{m\times n}$ be a matrix and $X = \|TC\|_F^2$. Then we have: (i) $\mathbb{E}(X) = \|C\|_F^2$, and (ii) $\mathsf{Var}(X) \leq \frac{2}{d}\|C\|_F^4$.*

Proof We prove (i) first. We recall that $T = \frac{1}{\sqrt{d}}T'$, where the rows of T' are isotropic sub-Gaussian random vectors. Let $Y_i = \|T_i'C\|_2^2$, then

$$X = \sum_{i=1}^{d} \frac{1}{d} Y_i.$$

Using the isotropy of T_i', we obtain

$$\begin{aligned}\mathbb{E}(Y_i) = \mathbb{E}(\|T_i'C\|_2^2) &= \mathbb{E}(\textstyle\sum_j \langle T_i', C^j\rangle^2)\\ &= \textstyle\sum_j \mathbb{E}(\langle T_i', C^j\rangle^2)\\ &= \textstyle\sum_j \|C^j\|_2^2\\ &= \|C\|_F^2.\end{aligned}$$

Thus,

$$\mathbb{E}(X) = \mathbb{E}(\sum_{i=1}^{d} \frac{1}{d} Y_i) = \sum_{i=1}^{d} \frac{1}{d}\mathbb{E}(Y_i) = \|C\|_F^2.$$

As for (ii), from Lemma 4.4 and $\mathsf{Var}(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$ we have

$$\mathbb{E}((\langle T_i', x\rangle\langle T_i', y\rangle)^2) \leq 3\|x\|_2^2\|y\|_2^2 \quad \forall x, y \in \mathbb{R}^m.$$

Thus,

$$\begin{aligned}\mathbb{E}(Y_i^2) = \mathbb{E}((\textstyle\sum_j \langle T_i', C^j\rangle^2)^2) &= \textstyle\sum_{j,k} \mathbb{E}((\langle T_i', C^j\rangle\langle T_i', C^k\rangle)^2)\\ &\leq 3\textstyle\sum_{j,k} \|C^j\|_2^2\|C^k\|_2^2 = 3\|C\|_F^4\end{aligned}$$

Thus $\mathsf{Var}(Y_i) = \mathbb{E}(Y_i^2) - \mathbb{E}(Y_i)^2 \leq 2\|C\|_F^4$ and we obtain

$$\mathsf{Var}(X) = \mathsf{Var}(\sum_{i=1}^{d} \frac{1}{d} Y_i) = \sum_{i=1}^{d} \frac{1}{d^2}\mathsf{Var}(Y_i) \leq \frac{2}{d}\|C\|_F^4,$$

which concludes the proof. □

The next result corresponds to [53, Cor. 11] discussed in Sect. 4.2.1, where T, however, is the RP in Thm. 4.2, which can be applied to possibly infinite sets $S \subset \mathbb{R}^m$.

Corollary 4.7 *Let $U \in \mathbb{R}^{m\times k}$ be a unitary matrix. Then, the following holds whp:*

$$\forall i \leq k \quad |1 - \sigma_i(TU)| \leq \delta. \tag{32}$$

Proof The singular values $\sigma_i(TU)$ are the square roots of the eigenvalues λ_i of the matrix $U^\top T^\top TU$. Let x be the unitary eigenvector corresponding to λ_i for a generic $i \in \{1, \ldots, k\}$. Then,

$$\begin{aligned}\|TUx\|_2 &= \sqrt{\langle TUx, TUx\rangle} = \sqrt{(TUx)^\top (TUx)} \\ &= \sqrt{x^\top U^\top T^\top TUx} = \sqrt{x^\top \lambda_i x} \\ &= \sqrt{\lambda_i}\|x\|_2 = \sqrt{\lambda_i}.\end{aligned}$$

Therefore, Eq. (32) is equivalent to

$$|1 - \|TUx\|_2| \leq \delta. \tag{33}$$

In order to prove Eq. (33), since $\|x\|_2 = 1$ and U is a unitary matrix, we have $\|Ux\|_2 = 1$. Thus, if the set S in Theorem 4.2 contains the points

$$\{Ux \mid x \text{ unitary eigenvector of } U^\top T^\top TU\},$$

we have, whp,

$$|1 - \|TUx\|_2| = |\|Ux\|_2 - \|TUx\|_2| \leq \delta,$$

which concludes the proof. □

With all the preliminary results we proved so far, we can now prove the equivalent of Lemmata 3-5 in [13] for the RP T satisfying Eq. (25).

Lemma 4.8 *Let $P \in \mathbb{R}^{m\times n}$ be the matrix representing the points we want to cluster and $P_k = U_k \Sigma_k V_k{}^\top$ its SVD of rank k. Then, whp,*

$$\|(TU_k)^+ - (TU_k)^\top\|_2 \leq \frac{2-\delta^2}{1-\delta},$$

where M^+ is the Moore-Penrose left pseudoinverse of M.

Proof Let $\Phi = TU_k$ and $\Phi = U_\Phi \Sigma_\Phi V_\Phi{}^\top$ be its SVD. If we consider the SVD of Φ^+ and $\Phi^\top$ and the fact that the spectral norm of a matrix is invariant with respect to unitary matrices, we obtain

$$\begin{aligned}\|(TU_k)^+ - (TU_k)^\top\|_2 &= \|V_\Phi \Sigma_\Phi^{-1} U_\Phi{}^\top - V_\Phi \Sigma_\Phi U_\Phi{}^\top\|_2 \\ &= \|V_\Phi(\Sigma_\Phi^{-1} - \Sigma_\Phi)U_\Phi{}^\top\|_2 \\ &= \|\Sigma_\Phi^{-1} - \Sigma_\Phi\|_2.\end{aligned}$$

Now, let $\Psi = \Sigma_\Phi^{-1} - \Sigma_\Phi$, σ_i be the i-th singular value of Φ and τ_i the i-th entry of Ψ. Then, a simple computation shows that

$$\tau_i = \frac{1 - \sigma_i \sigma_{k+1-i}}{\sigma_{k+1-i}}.$$

From Corollary 4.7 we know that $1 - \delta \leq \sigma_i \leq 1 + \delta$, for $i = 1, \ldots, k$. Therefore,

$$\tau_1 = \frac{1}{\sigma_{k+1-i}} - \sigma_i \leq \frac{1}{1-\delta} + 1 + \delta = \frac{2-\delta^2}{1-\delta}.$$

Since Ψ is diagonal we have $\|\psi\|_2 = \max_i \tau_i$, which concludes the proof. □

Lemma 4.9 *Let $C \in \mathbb{R}^{m\times n}$, then, whp,*

$$\|TC\|_F \leq \sqrt{1+\delta}\|C\|_F.$$

Proof Let $Z = \|TC\|_F^2$, from Lemma 4.6 we have $\mathbb{E}(Z) = \|C\|_F^2$ and $\mathsf{Var}(Z) \leq \frac{2}{d}\|C\|_F^4$. Applying the Chebyshev inequality to the random variable Z we obtain

$$\mathsf{Prob}(|Z - \mathbb{E}(Z)| \geq \delta\|C\|_F^2) \leq \frac{\mathsf{Var}(Z)}{\delta^2\|C\|_F^4} \leq \frac{2}{d\delta}.$$

Hence,

$$\mathsf{Prob}(Z \geq (1+\delta)\|C\|_F^2) \leq \frac{2}{d\delta} \Leftrightarrow \mathsf{Prob}(\sqrt{Z} \geq \sqrt{1+\delta}\|C\|_F) \leq \frac{2}{d\delta},$$

which concludes the proof. □

Lemma 4.10 *The following holds whp:*

$$P_k = U_k(TU_k)^+TP + E,$$

where $E \in \mathbb{R}^{m\times n}$ satisfies $\|E\|_F \leq f_1(\delta)\|P - P_k\|$, $f_1(\delta) = \frac{\delta}{2} + \sqrt{1+\delta}\frac{2-\delta^2}{1-\delta}$.

Proof Define $E = P_k - U_k(TU_k)^+TP$. We want to prove that

$$\|E\|_F \leq f_1(\delta)\|P - P_k\|.$$

Let $\rho = \mathsf{rank}(P)$, then we can write $P = P_k + P_{\rho-k}$, where $P_{\rho-k} = P - P_k$. Replacing P with $P_k + P_{\rho-k}$ in the definition of E we have

$$\begin{aligned} E &= P_k - U_k(TU_k)^+T(P_k + P_{\rho-k}) \\ &= (P_k - U_k(TU_k)^+TP_k) + (U_k(TU_k)^+TP_{\rho-k}). \end{aligned}$$

If we consider the Frobenius norm and we use the triangular inequality we get

$$\|E\|_F \leq \|P_k - U_k(TU_k)^+TP_k\|_F + \|U_k(TU_k)^+TP_{\rho-k}\|_F. \tag{34}$$

In the first term appearing in the second member of Eq. (34) we can replace $P_k = U_k\Sigma_k{V_k}^\top$ and exploit the fact that $(TU_k)^+(TU_k) = I$, to obtain

$$\begin{aligned} \|P_k - U_k(TU_k)^+TP_k\|_F &= \|U_k\Sigma_k{V_k}^\top - U_k(TU_k)^+TU_k\Sigma_k{V_k}^\top\| \\ &= \|U_k\Sigma_k{V_k}^\top - U_k\Sigma_k{V_k}^\top\| = 0. \end{aligned}$$

It remains to bound the second term in the second member of Eq. (34). We add and subtract $U_k(TU_k)^\top TP_{\rho-k}$, we use the fact that the Frobenius norm is invariant with respect to unitary matrices, as well as the strong sub-multiplicativity of every pair of matrices X, Y of appropriate dimensions, namely that $\|XY\|_F \leq \|X\|_F\|Y\|_2$. We then obtain

$$\left.\begin{aligned} &\|U_k(TU_k)^+TP_{\rho-k} - U_k(TU_k)^\top TP_{\rho-k} + U_k(TU_k)^\top TP_{\rho-k}\|_F \\ &\leq \|U_k(TU_k)^\top TP_{\rho-k}\|_F + \|U_k((TU_k)^+ - (TU_k)^\top)TP_{\rho-k}\|_F \\ &\leq \|(TU_k)^\top TP_{\rho-k}\|_F + \|TP_{\rho-k}\|_F\|(TU_k)^+ - (TU_k)^\top\|_2 \\ &= \|U_k{}^\top T^\top TP_{\rho-k}\|_F + \|TP_{\rho-k}\|_F\|(TU_k)^+ - (TU_k)^\top\|_2. \end{aligned}\right\} \tag{35}$$

We bound the last three terms in Eq. (35) separately.

1. For $\|{U_k}^\top T^\top T P_{\rho-k}\|_F$, we observe that

$$ {U_k}^\top P_{\rho-k} = {U_k}^\top U_{\rho-k} \Sigma_{\rho-k} {V_{\rho-k}}^\top = 0 \, \Sigma_{\rho-k} {V_{\rho-k}}^\top = 0, $$

since U_k and $U_{\rho-k}$ have orthogonal columns by definition. Therefore, we can apply Lemma 4.5 (a) and obtain, whp,

$$ \begin{aligned} \|{U_k}^\top T^\top T P_{\rho-k}\|_F &= \|{U_k}^\top P_{\rho-k} - U^\top T^\top T P_{\rho-k}\|_F \\ &\leq \frac{\delta}{2} \|{U_k}^\top\|_F \|P_{\rho-k}\|_F = \frac{\delta}{2} \|P_{\rho-k}\|_F. \end{aligned} $$

2. In order to bound $\|TP_{\rho-k}\|_F$, it is enough to apply Lemma 4.9 with $C = P_{\rho-k}$. This yields

$$ \|TP_{\rho-k}\|_F \leq \sqrt{1+\delta} \|P_{\rho-k}\|_F. $$

3. As concerns $\|(TU_k)^+ - (TU_k)^\top\|_2$, we can apply Lemma 4.8 and obtain, whp,

$$ \|(TU_k)^+ - (TU_k)^\top\|_2 \leq \frac{2-\delta^2}{1-\delta}. $$

Putting these bounds together we obtain:

$$ \begin{aligned} \|E\|_F &\leq \|{U_k}^\top T^\top T P_{\rho-k}\|_F + \|TP_{\rho-k}\|_F \|(TU_k)^+ - (TU_k)^\top\|_2 \\ &\leq \tfrac{\delta}{2} \|P_{\rho-k}\|_F + \sqrt{1+\delta} \tfrac{2-\delta^2}{1-\delta} \|P_{\rho-k}\|_F \\ &= \left(\tfrac{\delta}{2} + \sqrt{1+\delta} \tfrac{2-\delta^2}{1-\delta}\right) \|P_{\rho-k}\|_F \\ &= \left(\tfrac{\delta}{2} + \sqrt{1+\delta} \tfrac{2-\delta^2}{1-\delta}\right) \|P - P_k\|_F, \end{aligned} $$

which concludes the proof. □

Remark 4.11 Let $X_{\mathsf{opt}} = \underset{X \in \mathcal{X}}{\arg\min} \|P^\top - XX^\top P^\top\|_2^2$, where $\mathcal{X}$ is the set of all cluster indicator matrices. Since $X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top P^\top$ is a matrix with rank at most k, the Theorem of Eckart-Young implies

$$ \|{P_{\rho-k}}^\top\|_F^2 = \|P^\top - P_k^\top\|_F^2 \leq \|P^\top - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top P^\top\|_F^2. $$

We are finally ready to state and prove the equivalent of [13, Theorem 1] for the RP T satisfying Eq. (25).

Theorem 4.12 *Let $P \in \mathbb{R}^{m \times n}$ be the matrix representing the points we want to cluster, k the number of clusters, and $T \in \mathbb{R}^{d \times m}$ the RP satisfying Theorem 4.2. Assume that we have access to an algorithm which takes as input P, k, T and returns a cluster indicator matrix X_γ which satisfies, with high probability*

$$ \|(TP)^\top - X_\gamma {X_\gamma}^\top (TP)^\top\|_F^2 \leq \gamma \min_{X \in \mathcal{X}} \|(TP)^\top - XX^\top (TP)^\top\|_F^2, $$

where $\mathcal{X}$ is the set of all cluster indicator matrices and $\gamma \geq 1$. Then, with high probability we have

$$ \|P^\top - X_\gamma {X_\gamma}^\top P^\top\|_F^2 \leq \phi(\delta, \gamma) \|P^\top - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top P^\top\|_F^2, \tag{36} $$

where $X_{\mathsf{opt}} = \underset{X \in \mathcal{X}}{\arg\min} \|P^\top - XX^\top P^\top\|_F^2$ and

$$ \phi(\delta, \gamma) = \left(1 + \frac{(2-\delta^2)(\sqrt{\gamma}+1)\sqrt{1+\delta}}{1-\delta} + \frac{\delta}{2}\right)^2. $$

Proof In Eq. (36) we replace $P = P_k + P_{\rho-k}$ and we use the fact that ${P_k}^\top - X_\gamma {X_\gamma}^\top {P_k}^\top$ and ${P_{\rho-k}}^\top - X_\gamma {X_\gamma}^\top {P_{\rho-k}}^\top$ generate orthogonal subspaces to obtain

$$\left.\begin{array}{l} \|P^\top - X_\gamma {X_\gamma}^\top P^\top\|_F^2 \\ = \|{P_k}^\top - X_\gamma {X_\gamma}^\top {P_k}^\top\|_F^2 + \|{P_{\rho-k}}^\top - X_\gamma {X_\gamma}^\top {P_{\rho-k}}^\top\|_F^2 \\ = \|(I - X_\gamma {X_\gamma}^\top){P_k}^\top\|_F^2 + \|(I - X_\gamma {X_\gamma}^\top){P_{\rho-k}}^\top\|_F^2. \end{array}\right\} \tag{37}$$

We can bound the second term in the second member of Eq. (37) using the fact that $I - X_\gamma {X_\gamma}^\top$ is a projection matrix and the Frobenius norm does not increase if we drop a projection matrix

$$\|(I - X_\gamma {X_\gamma}^\top){P_{\rho-k}}^\top\|_F^2 \leq \|{P_{\rho-k}}^\top\|_F^2 \leq \|P^\top - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top P^\top\|_F^2, \tag{38}$$

where we used Remark 4.11 in the last inequality. We now bound the term $\|(I - X_\gamma {X_\gamma}^\top){P_k}^\top\|_F^2$. Using Lemma 4.10, the triangular inequality and the fact that $I - X_\gamma {X_\gamma}^\top$ is a projection matrix we get

$$\left.\begin{array}{rl} \|(I - X_\gamma {X_\gamma}^\top){P_k}^\top\|_F \leq & \|(I - X_\gamma {X_\gamma}^\top)(U_k(TU_k)^+TP)^\top\|_F + \|E^\top\|_F \\ = & \|(I - X_\gamma {X_\gamma}^\top)(TP)^\top((TU_k)^+)^\top\|_F + \|E\|_F, \end{array}\right\}$$

where the last member is obtained from the fact that the Frobenius norm is invariant with respect to unitary matrices and that $\|A^\top\|_F = \|A\|_F$ holds for every matrix A. Now we can apply strong sub-multiplicativity:

$$\left.\begin{array}{l} \|(I - X_\gamma {X_\gamma}^\top)(TP)^\top((TU_k)^+)^\top\|_F + \|E\|_F \\ \leq \|(I - X_\gamma {X_\gamma}^\top)(TP)^\top\|_F \|(TU_k)^+\|_2 + \|E\|_F. \end{array}\right\}$$

From the construction of X_γ we have that

$$\|(I - X_\gamma {X_\gamma}^\top)(TP)^\top\|_F \leq \sqrt{\gamma}\|(I - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top)(TP)^\top\|_F.$$

Thus, applying Lemma 4.8, Lemma 4.10 and Remark 4.11 we obtain

$$\left.\begin{array}{l} \|(I - X_\gamma {X_\gamma}^\top)(TP)^\top\|_F \|(TU_k)^+\|_2 + \|E\|_F \\ \leq \sqrt{\gamma}\|(I - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top)(TP)^\top\|_F \frac{2-\delta^2}{1-\delta} \\ + \left(\frac{\delta}{2} + \sqrt{1+\delta}\frac{2-\delta^2}{1-\delta}\right)\|(I - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top)P^\top\|_F. \end{array}\right\} \tag{39}$$

Now we observe that

$$\begin{array}{l} \|(I - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top)(TP)^\top\|_F \\ = \|\big((I - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top)P^\top T^\top\big)^\top\|_F \\ = \|T\big((I - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top)P^\top\big)^\top\|_F. \end{array}$$

Therefore, we can apply Lemma 4.9 with $C = \big((I - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top)P^\top\big)^\top$ and obtain

$$\|(I - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top)(TP)^\top\|_F \leq \sqrt{1+\delta}\|(I - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top)P^\top\|_F. \tag{40}$$

Replacing Eq. (40) in Eq. (39) we get

$$\left.\begin{array}{l} \|(I - X_\gamma {X_\gamma}^\top){P_k}^\top\|_F \\ \leq \sqrt{\gamma}\sqrt{1+\delta}\frac{2-\delta^2}{1-\delta}\|(I - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top)P^\top\|_F \\ + \left(\frac{\delta}{2} + \sqrt{1+\delta}\frac{2-\delta^2}{1-\delta}\right)\|(I - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top)P^\top\|_F \\ = \left(\sqrt{1+\delta}\frac{2-\delta^2}{1-\delta}\right)\left(\sqrt{\gamma}+1\right) + \frac{\delta}{2}\|(I - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top)P^\top\|_F. \end{array}\right\} \tag{41}$$

Finally, putting Eq. (38) and Eq. (41) together we obtain:

$$\left.\begin{aligned} &\|P^\top - X_\gamma {X_\gamma}^\top P^\top\|_F^2 \\ &\leq \left(1 + \left(\sqrt{1+\delta}\,\tfrac{2-\delta^2}{1-\delta}\right)\left(\sqrt{\gamma}+1\right) + \tfrac{\delta}{2}\right)^2 \|P^\top - X_{\mathsf{opt}} {X_{\mathsf{opt}}}^\top P^\top\|_F^2, \end{aligned}\right\}$$

which concludes the proof. □

5 Computational experiments

In this section we present results from experiments with the exact and approximate reformulations presented in Sect. 3-4. All results were obtained on a Linux machine with 8 quad-core Intel Xeon E5-2620 CPUs at 2.1GHz and 64GB RAM.

The formulations we test are:

- Eq. (11) (labeled `f2` with original data, `Tf2` with RP);
- Eq. (18) (labeled `f1` with original data, `Tf1` with RP);
- Eq. (18) with Eq. (16) as objective function (labeled `f1m` with original data, `Tf1m` with RP);
- Eq. (15) (labeled `finf` with original data, `Tfinf` with RP);
- Eq. (15) with Eq. (16) as objective function (labeled `finfm` with original data, `Tfinfm` with RP).

Given that our approximation guarantees apply to the optimum, whereas we solve our formulations with a given time limit, we structure our tests in two phases. In the first phase, which we run on instances without side constraints, we simply want to make sure that the relative difference in MSSC cost between the solutions of original and projected formulations is reasonably low. This will justify the application of projected formulations with a time limit. In the second phase, we add side constraints that the k-means algorithm cannot deal with, and show that our methods can.

For any MSSC instance, our code tests each of the 10 formulations above, and runs k-means on both P and TP whenever no side constraints are present. So every instance is solved by 10 or 12 different methods, depending on the presence of the side constraints. We label k-means on the original data by `fkm`, and on the projected data by `Tfkm`. For each of these methods, we compute the optimal objective function value found and the CPU time. The code, written in a mixture of Python 3 [52] and `bash` shell scripts, uses the `amplpy` interface to call AMPL [31] and solves the corresponding instance with the specified method using CPLEX 12.10 [38]. The k-means implementation is that of the `sklearn` [48] python library with default configuration (see the user manual for more details).

Our code was deployed on two instance sets: `random` and `UCI`. The `random` set contains randomly generated instances of the MSSC without side constraints. Again, the only aim of the experiments on the `random` set is to verify that RPs yield good approximations on the tested formulations. The `UCI` set contains some instances from the UCI data archive [23] having relatively few points in many dimensions, to which we adjoined some side constraints, as detailed in Sect. 5.3 below.

5.1 Improvement of the P^U bound

The formulations presented above all use P^U as a "big M" parameter used in order to model a variable product linearly. It can be shown that formulation tightness (i.e. the extent of the gap in optimality/feasibility between an integer formulation and its continuous relaxation) can be improved by making such parameters as small as possible, without losing exactness or relaxation guarantees. In the case of P^U, which is defined as the maximum possible distance (in various norms) between two points in P, an obvious tightening consists in defining a different bound depending on $i \leq n$:

$$\forall i \leq n \quad P_i^U = \max_{h \neq i} \|p_i - p_h\|, \tag{42}$$

where the norm can be $\ell_1, \ell_2, \ell_\infty$. This applies to formulations in Eq. (11), (15), (18).

5.2 Tests on random instances

Our algorithm for creating random MSSC instances with given n, m, k is as follows:

1. initialize scalars $B > 0$ (domain, set to 10 in our experiments), $\lambda \in [0, 1]$ (cluster overlap factor), $N \in [0, \frac{1}{2}]$ (cluster cardinality difference factor);
2. randomly generate k candidate centroid vectors $\bar{y}_1, \ldots, \bar{y}_k \in [-B, B]^m$;
3. randomly generate k cluster cardinality integers $n_1, \ldots, n_k$ such that: (i) $\sum_{j \leq k} n_j = n$, (ii) $\forall j \leq k \; n_j \in \lfloor (1 \pm N)n/k \rceil$;
4. for each cluster $j \leq k$ let $\rho_j = \frac{1+\lambda}{k} \sum_{\ell \neq j} \frac{\|\bar{y}_j - \bar{y}_\ell\|_2}{1 + n_\ell / n_j}$ be the radius of a sphere around $\bar{y}_j$ such that these spheres overlap by a factor of their volume controlled by λ;
5. for each cluster $j \leq k$ uniformly generate n_j points in the ball $B(\bar{y}_j, \rho_j)$, and label them with j.

This algorithm creates some centroids and cluster cardinalities randomly, then defines appropriately sized balls around the centroids and samples points uniformly in those balls. The parameters N and λ control how much the cluster cardinalities differ from n/k, and, respectively, the overlap of the balls. The algorithm also saves the obtained clusterings: while these may not be optimal w.r.t. Eq. (4), in general they provide a good approximation by construction.

We generated three subsets of random instances: `small`, `medium` and `large`. The purpose of this partition based on size is that we would like to test formulations on both original and projected data; and given the relative size differences, fair testing requires instances with different sizes. The first subset, `small`, is intended to allow formulations on the original data to be solved at optimality. The second, `medium`, allows feasible solutions to be found by both original and projected formulations. The third, `large` is such that feasible solutions are mostly found with randomly projected formulations. All these instances were tested using a CPU time limit of 900s. Those instances reporting a CPU time exceeding 900s may not have been solved to global optimality. We remark that the value of Eq. (4) considered in the results table is evaluated on the best clustering found by the corresponding

formulation. Thus, certain clusterings found by approximating reformulations yield values of Eq. (4) that are worse than the optima found by k-means.

The details and results of the `small` set are reported in Table 1. The results validate the formulation approach: when k-means does not achieve a global optimum, MP formulations can. In this table we only report formulations on the original data, since none of the instances has sufficiently high dimensionality to undergo effective dimensional reduction using RP. We note that certain formulations solved to global optimality on certain instances improve the quality of the solution found by k-means. Obviously, the exact cMINLP reformulation Eq. (11) is always among them; but there is no clear rule about the approximations. The CPU time of k-means is always a fraction of that of CPLEX solving the MP formulations: this is a fixture of the proposed methodology.

The details and results of the `medium` set are reported in Table 2. For every formulation considered in the paper, we computed the relative difference between the value of Equation (4) for the original formulation and its projected version. These results show that RPs yield lower dimensional instances that are very accurate in practice (very few pairs of original vs. projected formulations exhibit a noticeable difference). Projected formulations were obtained with a RP with $\varepsilon = 0.3$ and density 0.9. We note that no feasible solutions were found by the `f2`, `Tf2`, and `finf` formulations in the 900s of allotted computation time: we therefore do not report the columns corresponding to `f2 - Tf2` and `finf - Tfinf`. Moreover, every solution process was terminated after 900s of computation time, so we do not report CPU time. We recall that d only depends on n, not on m.

The details and results of the `large` set are reported in Table 3. For every formulation considered in the paper, we computed the relative difference between the value of Equation (4) for the original formulation and its projected version. For most of the instances, formulations `f1`, `f1m` and `finfm` were not able to find a solution, but in all the other cases we observe that the projected formulations were able to obtain a smaller or equal value of Equation (4). Projected formulations were obtained with a RP with $\varepsilon = 0.3$ and density 0.9. Again, no feasible solutions were found by the `f2`, `Tf2`, and `finf` formulations in the 900s of allotted computation time: we therefore do not report the columns corresponding to `f2 - Tf2` and `finf - Tfinf`. And, again, every solution process was terminated after 900s of computation time, so we do not report CPU time. We recall that d only depends on n, not on m.

The fact that formulations do not achieve the same quality as k-means is due to the limit on CPU time. To show this, we took the most promising formulation (`Tfinf`) and solved it with larger time limit (4h), yielding 3 instances (one from `medium` and two from `large`) where `Tfinf` found a clustering yielding the same value of Eq. (4) as k-means.

Instance						*Value of Eq.* (4)	
n	m	d	k	λ	N	`Tfinf`	`fkm`
500	500	155	2	0.1	0.1	5136288.6	5136288.6
1000	1000	173	2	0.1	0.1	199329869.3	199329869.3
1000	10000	173	2	0.1	0.1	20175599.2	20175599.2

The projected formulation `Tfinf` used in the above table was obtained with a RP with $\varepsilon = 0.2$ and density 0.5. Raising the CPU time limit to 8h yielded a few more instances having the same value of Eq. (4) as k-means.

Instance					*Value of Eq.* (4)						*CPU time*					
n	m	k	λ	N	`f2`	`f1`	`f1m`	`finf`	`finfm`	`fkm`	`f2`	`f1`	`f1m`	`finf`	`finfm`	`fkm`
10	2	2	0.1	0.1	111.089	111.089	111.089	111.089	111.089	111.089	2.77	0.96	1.45	0.48	1.38	0.02
10	2	2	0.1	0.3	142.937	142.937	142.937	142.937	154.461	142.937	2.03	1.20	1.20	0.84	1.69	0.01
10	2	2	0.3	0.1	95.511	95.511	95.511	107.359	95.511	95.511	2.28	0.82	2.80	2.56	1.24	0.01
10	2	2	0.3	0.3	77.582	77.582	77.582	77.582	90.016	77.582	1.83	0.71	2.14	0.86	1.96	0.01
10	2	3	0.1	0.1	58.414	58.414	58.414	58.414	58.414	58.414	2.09	1.49	1.62	1.76	2.11	0.02
10	2	3	0.1	0.3	130.465	133.579	133.579	130.465	130.465	130.465	2.09	1.30	1.44	1.39	3.73	0.01
10	2	3	0.3	0.1	191.885	191.885	191.885	196.659	191.885	191.885	2.09	1.42	1.49	1.44	1.88	0.01
10	2	3	0.3	0.3	**207.349**	220.556	220.556	**207.349**	241.860	224.134	1.95	1.52	1.35	1.25	2.95	0.01
10	5	2	0.1	0.1	1449.714	1449.714	1449.714	2130.912	2147.024	1449.714	2.30	1.03	0.82	0.61	0.56	0.01
10	5	2	0.1	0.3	527.950	527.950	527.950	527.950	527.950	527.950	2.80	0.93	0.72	0.59	0.89	0.01
10	5	2	0.3	0.1	883.103	883.103	883.103	883.103	883.103	883.103	2.04	2.55	0.75	0.70	0.8	0.01
10	5	2	0.3	0.3	455.986	455.986	455.986	519.779	519.779	455.986	2.37	0.77	1.45	0.64	0.77	0.01
10	5	3	0.1	0.1	648.298	665.388	665.388	648.298	665.388	648.298	2.14	1.64	1.57	1.49	1.83	0.03
10	5	3	0.1	0.3	479.700	586.810	586.810	479.700	613.309	479.700	2.50	1.42	1.49	1.45	1.52	0.01
10	5	3	0.3	0.1	520.428	533.656	533.656	520.428	520.428	520.428	2.63	1.70	1.59	1.35	1.32	0.01
10	5	3	0.3	0.3	**170.775**	**175.940**	**175.940**	**177.624**	**176.149**	185.192	2.39	1.67	3.73	1.46	1.46	0.01
20	2	2	0.1	0.1	44.770	44.770	44.770	44.770	44.770	44.770	3.11	0.86	1.28	0.64	1.12	0.01
20	2	2	0.1	0.3	402.803	402.803	402.803	402.803	421.095	402.803	2.22	0.65	1.42	0.73	0.71	0.01
20	2	2	0.3	0.1	645.169	645.169	645.169	645.169	645.169	645.169	1.86	0.70	1.40	0.70	0.65	0.01
20	2	2	0.3	0.3	581.553	588.950	588.950	583.957	583.957	581.553	2.60	0.84	1.69	1.24	1.83	0.01
20	2	3	0.1	0.1	81.574	81.574	81.574	81.574	81.574	81.574	2.23	1.47	5.21	1.70	1.65	0.02
20	2	3	0.1	0.3	277.817	287.993	287.993	277.817	277.817	277.817	2.17	1.53	6.63	1.63	2.03	0.02
20	2	3	0.3	0.1	**162.896**	**162.896**	**162.896**	163.158	163.158	163.158	3.15	4.23	13.99	1.91	3.71	0.02
20	2	3	0.3	0.3	440.198	461.291	461.291	440.198	440.198	440.198	4.07	5.58	25.52	1.68	4.47	0.03
20	5	2	0.1	0.1	2084.071	2084.071	2084.071	2084.071	2084.071	2084.071	2.36	1.59	2.92	1.34	0.97	0.01
20	5	2	0.1	0.3	543.065	544.044	544.044	543.065	544.044	543.065	2.05	1.70	3.27	1.37	4.2	0.01
20	5	2	0.3	0.1	2175.156	2175.156	2175.156	2244.987	2244.987	2175.156	2.74	1.49	5.17	1.24	4.41	0.01
20	5	2	0.3	0.3	1451.049	1463.957	1463.957	1451.049	1451.049	1451.049	2.03	1.63	4.17	1.37	4.43	0.01
20	5	3	0.1	0.1	867.195	893.865	893.865	867.195	867.195	867.195	11.37	15.87	255.14	12.63	65.83	0.02
20	5	3	0.1	0.3	1434.383	1439.256	1439.256	1439.256	1556.837	1434.383	10.30	11.97	112.99	12.78	31.81	0.02
20	5	3	0.3	0.1	1867.578	1867.578	1867.578	2247.412	2247.412	1867.578	19.27	15.42	335.3	23.52	104.44	0.02
20	5	3	0.3	0.3	2427.466	2506.625	2506.625	2959.899	3184.599	2427.466	54.23	26.19	381.61	104.31	405.88	0.02

Table 1 Details and results of `small` random set. Optima better than those found by k-means are reported in bold.

Instance						*Value of Eq.* (4)		
n	m	d	k	λ	N	`f1 - Tf1`	`f1m - Tf1m`	`finfm - Tfinfm`
500	300	69	2	0.1	0.1	0.0	0.3483	0.1986
500	300	69	2	0.1	0.3	0.0	0.3445	0.1661
500	300	69	2	0.3	0.1	0.0	0.0912	0.3014
500	300	69	2	0.3	0.3	0.0	0.0034	0.1589
500	300	69	5	0.1	0.1	0.0	0.0	0.0007
500	300	69	5	0.1	0.3	0.0	0.0	0.0220
500	300	69	5	0.3	0.1	0.0	0.0	0.0
500	300	69	5	0.3	0.3	0.0	0.0	0.0041
500	300	69	10	0.1	0.1	0.0	0.0	0.0010
500	300	69	10	0.1	0.3	0.0	0.0	-0.0001
500	300	69	10	0.3	0.1	0.0	0.0	0.0005
500	300	69	10	0.3	0.3	0.0	0.0	0.0012
500	500	69	2	0.1	0.1	0.0	0.4254	0.1546
500	500	69	2	0.1	0.3	0.0	0.2603	0.0561
500	500	69	2	0.3	0.1	0.0	0.1448	0.0431
500	500	69	2	0.3	0.3	0.0	0.1022	0.1802
500	500	69	5	0.1	0.1	0.0	0.0	0.0041
500	500	69	5	0.1	0.3	0.0	0.0	0.0
500	500	69	5	0.3	0.1	0.0	0.0	0.0112
500	500	69	5	0.3	0.3	0.0	0.0	0.0
500	500	69	10	0.1	0.1	0.0	0.0	-0.0017
500	500	69	10	0.1	0.3	0.0	0.0	0.0012
500	500	69	10	0.3	0.1	0.0	0.0	0.0018
500	500	69	10	0.3	0.3	0.0	0.0	0.0003
1000	300	77	2	0.1	0.1	N.a.	N.a.	0.0553
1000	300	77	2	0.1	0.3	N.a.	N.a.	0.0619
1000	300	77	2	0.3	0.1	0.0	0.0	0.0515
1000	300	77	2	0.3	0.3	0.0	0.0	0.0214
1000	300	77	5	0.1	0.1	0.0	0.0	0.0
1000	300	77	5	0.1	0.3	N.a.	0.0	0.0001
1000	300	77	5	0.3	0.1	0.0	0.0	0.0
1000	300	77	5	0.3	0.3	N.a.	0.0	0.0003
1000	300	77	10	0.1	0.1	N.a.	N.a.	-0.0002
1000	300	77	10	0.1	0.3	N.a.	N.a.	N.a.
1000	300	77	10	0.3	0.1	N.a.	N.a.	-0.0002
1000	300	77	10	0.3	0.3	N.a.	N.a.	-0.0001
1000	500	77	2	0.1	0.1	0.0	N.a.	0.0544
1000	500	77	2	0.1	0.3	0.0	N.a.	0.0352
1000	500	77	2	0.3	0.1	0.0	0.0	0.1143
1000	500	77	2	0.3	0.3	0.0	0.0	0.0233
1000	500	77	5	0.1	0.1	N.a.	0.0	0.0
1000	500	77	5	0.1	0.3	N.a.	0.0	0.0002
1000	500	77	5	0.3	0.1	N.a.	0.0	0.0
1000	500	77	5	0.3	0.3	N.a.	0.0	0.0
1000	500	77	10	0.1	0.1	N.a.	N.a.	-0.0004
1000	500	77	10	0.1	0.3	N.a.	N.a.	-0.0019
1000	500	77	10	0.3	0.1	N.a.	N.a.	-0.0005
1000	500	77	10	0.3	0.3	N.a.	N.a.	N.a.

Table 2 Details and results of `medium` random set. We computed the value of Equation (4) for the formulations `f1`, `f1m`, `finfm` and their projected versions. In the table it is shown the relative difference `(f-Tf)/`max{`|f|,|Tf|`} between the value of Equation (4) of the initial formulation and the corresponding projected formulation. A positive (resp. negative) difference means that the projected formulation reached a smaller (resp. bigger) value. For every instance for which the original formulation or its projection was not able to obtain a solution we indicated N.a. (Not applicable). No feasible solution were found by the `f2`, `Tf2`, and `finf` formulations in the 900s of allotted computation time: we therefore do not report the corresponding columns.

The fact that we were not able to improve strictly on the k-means results may simply be a consequence of the fact that k-means found global optima on `medium` and `large`. An indication to this effect is the fact that the value of Eq. (4) of the (known) clustering used to generate the instances is always equal to the value of Eq. (4) of the clustering found by k-means, with one exception. This exception is the single instance ($n = 1000, m = 500, k = 10, \lambda = 0.3, N = 0.3$), where the values of Eq. (4) are 14435118.41 for the generating clustering, and 15532981.13 for k-means. Unfortunately, the solver ran out of memory on the `Tfinf` formulation applied to this instance before finding any feasible solution, even when run on a more powerful computer with 256GB RAM.

Instance						*Value of Eq.* (4)		
n	m	d	k	λ	N	`f1-Tf1`	`f1m - Tf1m`	`finfm - Tfinfm`
500	1000	69	2	0.1	0.1	0.0	0.2887	0.3139
500	1000	69	2	0.1	0.3	0.0	0.0021	0.3523
500	1000	69	2	0.3	0.1	0.0	0.0299	0.2640
500	1000	69	2	0.3	0.3	0.0	0.1072	0.0561
500	1000	69	5	0.1	0.1	0.0	0.0	0.0
500	1000	69	5	0.1	0.3	0.0	0.0	0.0362
500	1000	69	5	0.3	0.1	0.0	0.0	0.0
500	1000	69	5	0.3	0.3	0.0	0.0	0.0
500	1000	69	10	0.1	0.1	N.a.	0.0	0.0004
500	1000	69	10	0.1	0.3	N.a.	0.0	0.0017
500	1000	69	10	0.3	0.1	N.a.	0.0	0.0002
500	1000	69	10	0.3	0.3	N.a.	N.a.	0.0
500	10000	69	2	0.1	0.1	N.a.	N.a.	N.a.
500	10000	69	2	0.1	0.3	N.a.	N.a.	N.a.
500	10000	69	2	0.3	0.1	N.a.	N.a.	N.a.
500	10000	69	2	0.3	0.3	N.a.	N.a.	N.a.
500	10000	69	5	0.1	0.1	N.a.	N.a.	N.a.
500	10000	69	5	0.1	0.3	N.a.	N.a.	N.a.
500	10000	69	5	0.3	0.1	N.a.	N.a.	N.a.
500	10000	69	5	0.3	0.3	N.a.	N.a.	N.a.
500	10000	69	10	0.1	0.1	N.a.	N.a.	N.a.
500	10000	69	10	0.1	0.3	N.a.	N.a.	N.a.
500	10000	69	10	0.3	0.1	N.a.	N.a.	N.a.
500	10000	69	10	0.3	0.3	N.a.	N.a.	N.a.
1000	1000	77	2	0.1	0.1	0.0	N.a.	0.2549
1000	1000	77	2	0.1	0.3	0.0	N.a.	0.1275
1000	1000	77	2	0.3	0.1	0.0	N.a.	0.1629
1000	1000	77	2	0.3	0.3	0.0	N.a.	0.0195
1000	1000	77	5	0.1	0.1	N.a.	N.a.	0.0
1000	1000	77	5	0.1	0.3	N.a.	N.a.	0.0
1000	1000	77	5	0.3	0.1	N.a.	N.a.	0.0006
1000	1000	77	5	0.3	0.3	N.a.	N.a.	0.0007
1000	1000	77	10	0.1	0.1	N.a.	N.a.	N.a.
1000	1000	77	10	0.1	0.3	N.a.	N.a.	N.a.
1000	1000	77	10	0.3	0.1	N.a.	N.a.	N.a.
1000	1000	77	10	0.3	0.3	N.a.	N.a.	N.a.
1000	10000	77	2	0.1	0.1	N.a.	N.a.	N.a.
1000	10000	77	2	0.1	0.3	N.a.	N.a.	N.a.
1000	10000	77	2	0.3	0.1	N.a.	N.a.	N.a.
1000	10000	77	2	0.3	0.3	N.a.	N.a.	N.a.
1000	10000	77	5	0.1	0.1	N.a.	N.a.	N.a.
1000	10000	77	5	0.1	0.3	N.a.	N.a.	N.a.
1000	10000	77	5	0.3	0.1	N.a.	N.a.	N.a.
1000	10000	77	5	0.3	0.3	N.a.	N.a.	N.a.

Table 3 Details and results of `large` random set. We computed the value of Equation (4) for the formulations `f1`, `f1m`, `finfm` and their projected version. In the table it is shown the difference `(f-Tf)/`max{`|f|,|Tf|`} between the value of Equation (4) of the initial formulation and the corresponding projected formulation. A positive (resp. negative) difference means that the projected formulation reached a smaller (resp. bigger) value. For every instance for which the original formulation or its projection was not able to obtain a solution we indicated N.a. (Not applicable). No feasible solution were found by the `f2`, `Tf2`, and `finf` formulations in the 900s of allotted computation time: we therefore do not report the corresponding columns.

5.3 Side constraints

Having ascertained that the proposed formulation-based methods are sound and behave according to expectations, we now exploit the flexibility and generality of MP-based approaches w.r.t. purely algorithmic ones, and adjoin some side constraints to the formulation. In particular, we consider classic "must-link" and "cannot-link" constraints. We then also add more exotic constraints, i.e. that centroids must be inside given balls (to the best of our knowledge, no k-means variant targets such constraints).

- *Must-link*: for a given set $L \subset \{\{i,h\} \mid i < h \leq n\}$ and for every $j \leq k$ we require $x_{ij} = x_{hj}$.
- *Cannot-link*: for a given set $L \subset \{\{i,h\} \mid i < h \leq n\}$ and for every $j \leq k$ we require $x_{ij} \leq 1 - x_{hj}$.
- *Centroids in balls*: given a set $\mathcal{B} = \{(\hat{y}_j, \hat{r}_j) \mid j \leq k\}$ of ℓ_2-balls centered at $\hat{y}_j$ with radii $\hat{r}_j$, we require that $\|y_j - \hat{y}_j\|_2^2 \leq \hat{r}_j^2$ for all $j \leq k$.

We remark that the centroid constraints turn every MILP formulation into a more challenging cMINLP one.

In computational experiments, we generate a certain number of side constraints randomly from the above families, and adjoin them to instances downloaded from the UCI archive [23]. Specifically:

- we generate "must-link" constraints by picking a random sample $\hat{S}$ of $0.1\frac{n}{k}$ random points in P, and considering the pairs p_i, p_j (for $i, j \in \hat{S}$) whose Euclidean distance is within a threshold $\frac{1}{2}(\texttt{avg}D + \min D)$, where D is the Euclidean distance matrix of the points in $\hat{S}$;
- "cannot-link" constraints are generated similarly: we consider random sample of points generated like $\hat{S}$, and take all the point pairs from the sample, excluding those that were used in "must-link" constraints (if any);
- the centers $\hat{y}$ of the balls in $\mathcal{B}$ defining the centroid constraints are chosen randomly in a box containing all the points in P; the radii are chosen to be all equal to $\frac{1}{k}\min_{\ell \leq m}(y^U_\ell - y^L_\ell)$ (see Eq. (9)).

5.3.1 Easing feasibility by local branching constraints

The first tests on these constrained instances witnessed the extreme difficulty to find a feasible solution. We therefore add one more class of *local branching* constraints, the only purpose of which was to encode a nearly-feasible solution in the formulation. For each instance, we consider the solution $\hat{x}$ given by k-means, and then adjoin the following constraints [27, Eq. (7)]:

$$\sum_{\substack{i \leq n, j \leq k \\ \hat{x}_{ij}=0}} x_{ij} + \sum_{\substack{i \leq n, j \leq k \\ \hat{x}_{ij}=1}} (1 - x_{ij}) \leq \kappa. \tag{43}$$

Such constraints allow the encoding of a local search neighbourhood (centered at $\hat{x}$ and having Hamming radius κ) within the formulation itself. They essentially states that at most κ binary variables can switch their value from $\hat{x}$. We note that letting $\kappa = 0$ makes $\hat{x}$ the only possible solution, and that reasonable pre-solving codes within commercial solvers would immediately fix $x = \hat{x}$ based on Eq. (43) with $\kappa = 0$. In thise sense, local branching constraints can endow a formulation with some knowledge of a good solution.

In the present case, $\hat{x}$ is feasible for the MSSC problem, but not w.r.t. the side constraints. Moreover, we still want the search to be global rather than local. We therefore let $\kappa = nk$, which means that the constraint is satisfied by every binary vector x. Even though Eq. (43) with $\kappa = nk$ are redundant constraints, our computational experience shows that they allow solvers to find a feasible solution much more easily.

5.4 Tests on UCI instances

We considered datasets from `archive.ics.uci.edu/ml/datasets/` with relatively few points in a large dimensional spaces. Notably, we consider the following datasets:

- *SCADI* (labelled `scadi`);

– *DBWorld e-mails* (labelled `dbworld`);
– *Gastrointestinal lesions in regular colonoscopy* (labelled `gastroent`);
– *A study of Asian religious and biblical texts* (labelled `asianrel`);
– *Gas+sensor array under flow modulation* (labelled `pulmon`);
– *DrivFace* (labelled `drivface`).

For those instances containing categorical values, we mapped each categorical string to ASCII codes and concatenated them. The properties of these instances are given in Table 4.

Name	n	m	d	k
`dbworld`	64	4702	46	2
`scadi`	70	206	47	2
`asianrel`	590	8266	71	2
`drivface`	606	6400	71	2
`gastroent`	700	152	73	2
`pulmon`	928	7509	76	2

Table 4 Properties of the UCI instances.

Our first validation experiment consists in solving these instances with $k = 2$, without side constraints, allowing each formulation 900s of CPU time. Detailed instance data and results concerning the value of Eq. (4) at the best optima found by each formulation are given in Table 5 (above). CPU times taken by k-means are negligible (in the range 0.22s-3.36s). In one case (`dbworld`) the best optimum was found by a formulation method rather than k-means. The large number of failures found in `pulmon` might be due to ill scaling of the instance data, which is present both in the original numerical data (five orders of magnitude), and also partly derived by our own category-to-number mapping.

Finally, the results of our experiment with side constraints are reported in Table 5 (below). For these tests, we allowed a maximum CPU time of 8h per formulation. We note that most randomly projected formulations can be solved to feasibility.

6 Conclusion

"Minimum Sum-of-Squares Clustering" is the name of the problem that the famous k-means heuristic aims at solving: given a set of points in some Euclidean space, find clusters and their centroids so that the sum of squared Euclidean distances between points and the corresponding centroid is minimum. It is one of the foundational problems related to clustering using Euclidean distances. Its natural Mathematical Programming formulation is of the Mixed-Integer Nonlinear Programming class. We considered variants that add arbitrary constraints to the classic formulation. We proposed reformulations yielding convex continuous relaxations. We presented approximations based on other norms than Euclidean and on the application of random projection techniques to decrease dimensionality of the data points.

We emphasize the extreme ease and speed of implementation and deployment of our proposed methodology. The algorithmic part rests almost completely on

Instance	*Value of Eq.* (4)											
	Without side constraints											
Name	`f2`	`Tf2`	`f1`	`Tf1`	`f1m`	`Tf1m`	`finf`	`Tfinf`	`finfm`	`Tfinfm`	`fkm`	`Tfkm`
`dbworld`	-1	10738.15	10810.24	10519.35	**10293.89**	10603.04	10767.33	10744.57	10855.97	10746.00	10487.82	10701.22
`scadi`	1957.76	2401.46	2050.57	1823.73	2050.57	1823.73	1811.48	1871.63	1811.48	1849.49	**1811.26**	**1811.26**
`asianrel`	-1	-1	165387.4	165387.4	149017.6	165387.4	-1	155280.1	163279.7	158913.6	**141465.4**	141858.8
`drivface`	-1	-1	110471.94	110428.39	10293.89	102307.86	-1	96953.231	110506.472	105406.01	**79509.72**	79561.14
`gastroent`	-1	-1	1.3×10^{11}	1.3×10^{11}	1.3×10^{11}	1.3×10^{11}	-1	1.3×10^{11}	3.4×10^{10}	3.9×10^{10}	$\mathbf{3.2 \times 10^{10}}$	$\mathbf{3.2 \times 10^{10}}$
`pulmon`	-1	-1	-1	-1	-1	1.6×10^{36}	-1	-1	-1	-1	$\mathbf{7.9 \times 10^{31}}$	$\mathbf{7.9 \times 10^{31}}$
	With side constraints											
`dbworld`	-1	10602.6	-1	10561.1	-1	10607.5	-1	10727.6	-1	-1	-	-
`scadi`	-1	2074.98	-1	1822.71	-1	1833.18	2211.36	1836.19	2266.87	-1	-	-
`asianrel`	-1	159328	-1	158449	-1	148695	-1	155012	-1	157576	-	-
`drivface`	-1	106411	-1	86258.1	-1	86201.5	-1	86124.2	-1	107421	-	-
`gastroent`	-1	-1	-1	-1	1.23×10^{11}	1.26×10^{11}	-1	1.31×10^{11}	-1	1.21×10^{11}	-	-
`pulmon`	-1	-1	-1	-1	-1	1.62×10^{36}	-1	-1	-1	1.61×10^{36}	-	-

Table 5 UCI dataset experiments (above: without side constraints; below: with side constraints). We note that k-means cannot solve the instances with side constraints).

off-the-shelf solver components. Random projections, though their analyses are certainly not simple, are based on well-known sampling techniques and a single matrix multiplication. Computational results were carried out on both random and real data instances. While k-means is decidedly the better option for solving the original problem as stated, the variant with side constraints limits the applicability of this successful heuristic. We think that, in such cases, our methodology based on randomly projected formulations is a viable alternative, in that it provides good quality solutions in limited amounts of time.

Data availability statement

The datasets generated and analysed in this paper are available upon request from the corresponding author.

References

1. Achlioptas, D.: Database-friendly random projections: Johnson-Lindenstrauss with binary coins. Journal of Computer and System Sciences **66**, 671–687 (2003)
2. Ailon, N., Chazelle, B.: Approximate nearest neighbors and fast Johnson-Lindenstrauss lemma. In: Proceedings of the Symposium on the Theory Of Computing, *STOC*, vol. '06. ACM, Seattle (2006)
3. Allen-Zhu, Z., Gelashvili, R., Micali, S., Shavit, N.: Sparse sign-consistent Johnson-Lindenstrauss matrices: Compression with neuroscience-based constraints. Proceedings of the National Academy of Sciences **111**(47), 16,872–16,876 (2014)
4. Aloise, D., Hansen, P., Liberti, L.: An improved column generation algorithm for minimum sum-of-squares clustering. Mathematical Programming A **131**, 195–220 (2012)
5. Babaki, B., Guns, T., Nijssen, S.: Constrained clustering using column generation. In: H. Simonis (ed.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR), *LNCS*, vol. 8451. Springer, Heidelberg (2014)
6. Becchetti, L., Bury, M., Cohen-Addad, V., Grandoni, F., Schwiegelshohn, C.: Oblivious dimension reduction for *k*-means: Beyond subspaces and the Johnson-Lindenstrauss lemma. In: Proceedings of the 51st Annual ACM Symposium on the Theory of Computing, STOC, pp. 1039–1050. ACM, New York (2019)
7. Bell, E.: The iterated exponential integers. Annals of Mathematics **39**, 539–557 (1938)
8. Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bounds tightening techniques for non-convex MINLP. Optimization Methods and Software **24**(4), 597–634 (2009)
9. Blömer, J., Lammersen, C., Schmidt, M., Sohler, C.: Theoretical analysis of the k-means algorithm: A survey. In: L. Kliemann, P. Sanders (eds.) Algorithm Engineering, *LNCS*, vol. 9220, pp. 81–116. Springer, Cham (2016)
10. Blum, L., Shub, M., Smale, S.: On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions, and universal machines. Bulletin of the AMS **21**(1), 1–46 (1989)
11. Bonami, P., Biegler, L., Conn, A., Cornuéjols, G., Grossmann, I., Laird, C., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wächter, A.: An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optimization **5**, 186–204 (2008)
12. Bonami, P., Lee, J.: `BONMIN` User's Manual. Tech. rep., IBM Corporation (2007)
13. Boutsidis, C., Zouzias, A., Drineas, P.: Random projections for *k*-means clustering. In: Advances in Neural Information Processing Systems, NIPS, pp. 298–306. NIPS Foundation, La Jolla (2010)
14. Braverman, V., Meyerson, A., Ostrovsky, R., Roytman, A., Shindler, M., Tagiku, B.: Streaming k-means on well-clusterable data. In: Proceedings of the 22nd annual ACM Symposium on Discrete Algorithms, *SODA*, vol. 22, pp. 26–40. ACM, Philadelphia (2011)
15. de Bruijn, N.: Asymptotic methods in analysis. Dover, New York (1981)

16. Bury, M., Schwiegelshohn, C.: Random projection for k-means: Maintaining coresets beyond merge & reduce. Tech. Rep. 1504.01584v3, arXiv (2015)
17. Clarkson, K., Woodruff, D.: Numerical linear algebra in the streaming model. In: Proceedings of the 41st Annual ACM Symposium on the Theory of Computing, STOC, pp. 205–241. ACM, New York (2009)
18. Cohen, M., Elder, S., Musco, C., Musco, C., Persu, M.: Dimensionality reduction for k-means clustering and low-rank approximation. In: Proceedings of the 47th Annual ACM Symposium on the Theory of Computing, STOC, pp. 163–172. ACM, New York (2015)
19. D'Ambrosio, C., Liberti, L., Poirion, P.L., Vu, K.: Random projections for quadratic programs. Mathematical Programming B (accepted)
20. Dao, T.B.H., Duong, K.C., Vrain, C.: Constrained minimum sum of squares clustering by constraint programming. In: G. Pesant (ed.) Principles and Practice of Constraint Programming, *LNCS*, vol. 9255, pp. 557–573. Springer, Heidelberg (2015)
21. Dasgupta, S., Gupta, A.: An elementary proof of a theorem by Johnson and Lindenstrauss. Random Structures and Algorithms **22**, 60–65 (2002)
22. Davidson, I., Ravi, S.: Clustering with constraints: Feasibility issues and the k-means algorithm. In: Proceedings of the SIAM International Conference on Data Mining, ICDM, pp. 138–149. SIAM, Philadelphia (2005)
23. Dua, D., Graff, C.: UCI machine learning repository (2017). URL `http://archive.ics.uci.edu/ml`
24. Duong, K.-C., Vrain, C.: Constrained clustering by constraint programming. Artificial Intelligence, **244**, 70–94 (2017)
25. Duran, M., Grossmann, I.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Mathematical Programming **36**, 307–339 (1986)
26. Eckart, C., Young, G.: The approximation of one matrix by another of lower rank. Psychometrika **1**, 211–218 (1936)
27. Fischetti, M., Lodi, A.: Local branching. Mathematical Programming **98**, 23–37 (2005)
28. Fletcher, R.: Practical Methods of Optimization, second edn. Wiley, Chichester (1991)
29. Fletcher, R., Leyffer, S.: Solving mixed integer nonlinear programs by outer approximation. Mathematical Programming **66**, 327–349 (1994)
30. Fletcher, R., Leyffer, S.: Numerical experience with lower bounds for MIQP branch-and-bound. SIAM Journal of Optimization **8**(2), 604–616 (1998)
31. Fourer, R., Gay, D.: The AMPL Book. Duxbury Press, Pacific Grove (2002)
32. Gleixner, A., Bastubbe, M., Eifler, L., Gally, T., Gamrath, G., Gottwald, R.L., Hendel, G., Hojny, C., Koch, T., Lübbecke, M.E., Maher, S.J., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schlösser, F., Schubert, C., Serrano, F., Shinano, Y., Viernickel, J.M., Walter, M., Wegscheider, F., Witt, J.T., Witzig, J.: The SCIP Optimization Suite 6.0. Technical report, Optimization Online (2018). URL `http://www.optimization-online.org/DB_HTML/2018/07/6692.html`
33. Gordon, A., Henderson, J.: An algorithm for Euclidean sum of squares classification. Biometrics **33**(2), 355–362 (1977)
34. Goubault, E., Roux, S.L., Leconte, J., Liberti, L., Marinelli, F.: Static analysis by abstract interpretation: a mathematical programming approach. In: A. Miné, E. Rodriguez-Carbonell (eds.) Proceeding of the Second International Workshop on Numerical and Symbolic Abstract Domains, *Electronic Notes in Theoretical Computer Science*, vol. 267(1), pp. 73–87. Elsevier (2010)
35. Grossi, V., Monreale, A., Nanni, M., Pedreschi, D., Turini, F.: Clustering formulation using constraint optimization. In: D.B. et al. (ed.) SEFM Workshops, *LNCS*, vol. 9509, pp. 93–107. Srpinger, Heidelberg (2015)
36. Hansen, P., Jaumard, B.: Cluster analysis and mathematical programming. Mathematical Programming **79**, 191–215 (1997)
37. IBM: ILOG CPLEX 12.8 User's Manual. IBM (2017)
38. IBM: ILOG CPLEX 12.10 User's Manual. IBM (2020)
39. Johnson, W., Lindenstrauss, J.: Extensions of Lipschitz mappings into a Hilbert space. In: G. Hedlund (ed.) Conference in Modern Analysis and Probability, *Contemporary Mathematics*, vol. 26, pp. 189–206. AMS, Providence, RI (1984)
40. Klein, D., Kamvar, S., Manning, C.: From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In: Proceedings of the 19th International Conference on Machine Learning, ICML, p. 307314. Morgan Kaufmann, San Francisco (2002)

41. Liberti, L.: Reformulations in mathematical programming: Definitions and systematics. RAIRO-RO **43**(1), 55–86 (2009)
42. Liberti, L.: Undecidability and hardness in mixed-integer nonlinear programming. RAIRO-Operations Research **53**, 81–109 (2019)
43. Liberti, L., Cafieri, S., Tarissan, F.: Reformulations in mathematical programming: A computational approach. In: A. Abraham, A.E. Hassanien, P. Siarry, A. Engelbrecht (eds.) Foundations of Computational Intelligence Vol. 3, no. 203 in Studies in Computational Intelligence, pp. 153–234. Springer, Berlin (2009)
44. Liberti, L., Marinelli, F.: Mathematical programming: Turing completeness and applications to software analysis. Journal of Combinatorial Optimization **28**(1), 82–104 (2014)
45. Lovasz, L.: Combinatorial problems and exercises. North-Holland, Amsterdam (1993)
46. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley symposium on mathematical statistics and probability, vol. 1, pp. 281–297. University of California Press (1967)
47. du Merle, O., Hansen, P., Jaumard, B., Mladenović, N.: An interior point algorithm for minimum sum-of-squares clustering. SIAM Journal Scientific Computing **21**(4), 1485–1505 (2000)
48. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
49. Pham, N.T.: Quantile regression in large energy datasets. Master's thesis, LIX, Ecole Poltyechnique (2018)
50. Pilanci, M., Wainwright, M.: Randomized sketches of convex programs with sharp guarantees. In: International Symposium on Information Theory (ISIT), pp. 921–925. IEEE, Piscataway (2014)
51. Pilanci, M., Wainwright, M.: Newton sketch: A linear time optimization algorithm with linear-quadratic convergence. SIAM Journal on Optimization **27**(1), 205–245 (2017)
52. van Rossum, G., *et al.*: Python Language Reference, version 3. Python Software Foundation (2019)
53. Sarlós, T.: Improved approximation algorithms for large matrices via random projections. In: Proceedings of the Annual IEEE Symposium on Foundations of Computer Science, *FOCS*, vol. 47, pp. 143–152. IEEE, Washington (2006)
54. Smith, E., Pantelides, C.: A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. Computers & Chemical Engineering **23**, 457–478 (1999)
55. Steinhaus, H.: Sur la division des corps matériels en parties. Bulletin de l'Académie Polonaise des Sciences Cl. III **4**(12), 801–804 (1956)
56. Steinley, D.: K-means clustering: A half-century synthesis. British Journal of Mathematical and Statistical Psychology **59**, 1–34 (2006)
57. Tawarmalani, M., Sahinidis, N.: Global optimization of mixed integer nonlinear programs: A theoretical and computational study. Mathematical Programming **99**, 563–591 (2004)
58. Vempala, S.: The Random Projection Method. No. 65 in DIMACS Series in Discrete Mathematics and Theoretical Computer Science. AMS, Providence, RI (2004)
59. Vershynin, R.: High-dimensional probability. CUP, Cambridge (2018)
60. Vu, K., Poirion, P.L., D'Ambrosio, C., Liberti, L.: Random projections for quadratic programs over a Euclidean ball. In: A. Lodi, *et al.* (eds.) Integer Programming and Combinatorial Optimization (IPCO), *LNCS*, vol. 11480, pp. 442–452. Springer, New York (2019)
61. Vu, K., Poirion, P.L., Liberti, L.: Random projections for linear programming. Mathematics of Operations Research **43**(4), 1051–1071 (2018)
62. Vu, K., Poirion, P.L., Liberti, L.: Gaussian random projections for Euclidean membership problems. Discrete Applied Mathematics **253**, 93–102 (2019)
63. Wächter, A., Biegler, L.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming **106**(1), 25–57 (2006)
64. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: Proceedings of the 17th International Conference on Machine Learning, ICML, p. 11031110. Morgan Kaufmann, San Francisco (2000)
65. Wang, O., de Sainte Marie, C., Ke, C., Liberti, L.: Business Rules are universal and unlearnable. Computational Intelligence (accepted)
66. Yang, J., Meng, X., Mahoney, M.: Quantile regression for large-scale applications. SIAM Journal of Scientific Computing **36**(5), S78–S110 (2014)