

Discrete optimization methods to fit piecewise-affine models to data points

E. Amaldi⁺, S. Coniglio^{**}, L. Taccari⁺

⁺Dipartimento di Elettronica, Informazione e Bioingegneria,
Politecnico di Milano

Piazza Leonardo da Vinci 32, 20133 Milano, Italy

^{*}Lehrstuhl II für Mathematik

RWTH Aachen University

Pontdriesch 14-16, 52062 Aachen, Germany

edoardo.amaldi@polimi.it, coniglio@math2.rwth-aachen.de,
leonardo.taccari@polimi.it

Abstract. Fitting piecewise affine models to data points is a pervasive task in many scientific disciplines. In this work, we address the *k-Piecewise Affine Model Fitting with Pairwise Linear Separability problem* (*k*-PAMF-PLS) where, given a set of m points $\{\mathbf{a}_1, \dots, \mathbf{a}_m\} \subset \mathbb{R}^n$ and the corresponding observations $\{b_1, \dots, b_m\} \subset \mathbb{R}$, we have to partition the domain \mathbb{R}^n into k pairwise linearly separable subdomains and to determine an affine submodel (function) for each of them so as to minimize the total linear fitting error w.r.t. the observations b_i .

To solve *k*-PAMF-PLS to optimality, we propose a mixed-integer linear programming (MILP) formulation where symmetries are broken by separating the so-called shifted column inequalities. For medium-to-large scale instances, we develop a four-step heuristic involving, among others, a point reassignment step based on the identification of critical points and a domain partition step based on multicategory linear classification. Differently from traditional approaches proposed in the literature for similar fitting problems, in our methods the domain partitioning and submodel fitting aspects are taken into account simultaneously.

Computational experiments on real-world and structured randomly generated instances show that, with our MILP formulation with symmetry breaking constraints, we can solve to proven optimality many small-size instances. Our four-step heuristic turns out to provide close-to-optimal solutions for small-size instances, while allowing to tackle instances of much larger size. The experiments also show that the combined impact of the main features of our heuristic is quite substantial when compared to standard variants not including them.

* The work of S. Coniglio was carried out, for a large part, while he was with Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano. The part carried out while with Lehrstuhl II für Mathematik, RWTH Aachen University, is supported by the German Federal Ministry of Education and Research (BMBF), grant 05M13PAA, and Federal Ministry for Economic Affairs and Energy (BMWi), grant 03ET7528B.

1 Introduction

Fitting a set of data points in \mathbb{R}^n with a combination of low complexity models is a pervasive problem in, essentially, any area of science and engineering. It naturally arises, for instance, in prediction and forecasting when determining a model to approximate the value of an unknown function, or whenever one wishes to approximate a highly complex nonlinear function with a simpler one. Applications range from optimization (see, e.g., [TV12] and the references therein) to data mining (see, e.g., [AM02]) and system identification (see, for instance, [BGPV03,FTMLM03,TPSM06]), only to cite a few.

Among the different options, *piecewise affine models* have a number of advantages with respect to other model fitting approaches. Indeed, they are compact and simple to evaluate, visualize, and interpret, in contrast to models obtained with other techniques such as, e.g., neural networks, while allowing to approximate even highly nonlinear functions.

Given a set of m points $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\} \subset \mathbb{R}^n$, where $I = \{1, \dots, m\}$, with the corresponding observations $\{b_1, \dots, b_m\} \subset \mathbb{R}$ and a positive integer k , the general problem of fitting a piecewise affine model to the data points $\{(\mathbf{a}_1, b_1), \dots, (\mathbf{a}_m, b_m)\}$ consists in partitioning the sub-domain \mathbb{R}^n into k continuous *subdomains* D_1, \dots, D_k , where $J = \{1, \dots, k\}$, and in determining, for each subdomain D_j , an *affine submodel* (an affine function) $f_j : D_j \rightarrow \mathbb{R}$, so as to minimize a measure of the total fitting error. Adopting the notation $f_j(\mathbf{x}) = \mathbf{w}^j \mathbf{x} - w_0^j$ with coefficients $(\mathbf{w}^j, w_0^j) \in \mathbb{R}^{n+1}$, the j -th affine submodel corresponds to the hyperplane $H_j = \{(\mathbf{x}, f_j(\mathbf{x})) \in \mathbb{R}^{n+1} : f_j(\mathbf{x}) = \mathbf{w}^j \mathbf{x} - w_0^j\}$ where $\mathbf{x} \in D_j$. The total fitting error is defined as the sum, over all $i \in I$, of a function of the difference between b_i and the value $f_{j(i)}(\mathbf{a}_i)$ provided by the piecewise affine model, where $j(i)$ is the index of the affine submodel corresponding to the subdomain $D_{j(i)}$ which contains the point \mathbf{a}_i .

In the literature, different error functions (e.g., linear or quadratic) as well as different types of domain partition (with linearly or nonlinearly separable subdomains) have been considered. See Figure 1 (a) for an illustration of the case with $k = 2$ and a domain partition with linearly separable subdomains.

In this work, the focus is on the version of the general piecewise affine model fitting problem with a linear error function (L_1 norm) and a domain partition with pairwise linearly separable subdomains. We refer to it as to the *k-Piecewise Affine Model Fitting with Pairwise Linear Separability problem (k-PAMF-PLS)*. A more formal definition of the problem will be provided in Section 3.

k-PAMF-PLS shares a connection with the so-called *k-Hyperplane Clustering problem (k-HC)*, an extension of a classical clustering problem which calls for k hyperplanes in \mathbb{R}^{n+1} that minimize the sum, over all the data points $\{(\mathbf{a}_1, b_1), \dots, (\mathbf{a}_m, b_m)\}$, of the L_2 distance from (\mathbf{a}_i, b_i) to the hyperplane it is assigned to. See [BM00,AC13,Con11,Con15] for some recent work on the problem and [ADC13] for the problem variant where we minimize the number of hyperplanes needed to fit the points within a prescribed tolerance ε . It is nevertheless crucial to note that, differently from many of the approaches in the

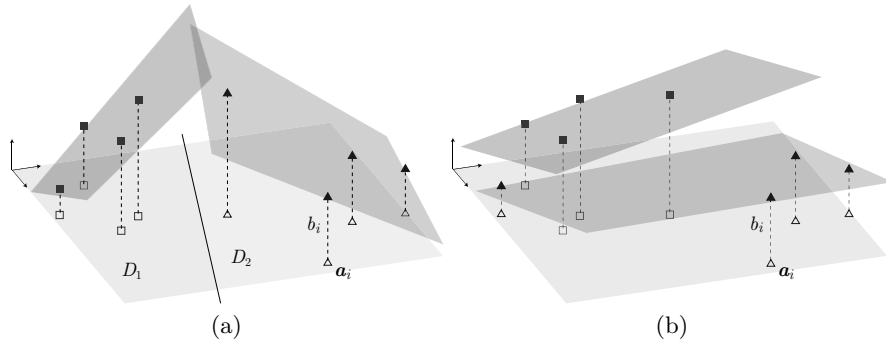


Fig. 1. (a) A piecewise affine model with $k = 2$, fitting the eight data points $\mathcal{A} = \{\mathbf{a}_i\}_{i \in I}$ and their observations $\{b_i\}_{i \in I}$ with two submodels (in dark grey). The points (\mathbf{a}_i, b_i) assigned to each submodel are indicated by \blacksquare and \blacktriangle . The model adopts a linearly separable partition of the domain \mathbb{R}^2 (represented in light grey). (b) An infeasible solution obtained by solving a k -hyperplane clustering problem in \mathbb{R}^3 with $k = 2$. Although yielding a smaller fitting error than that in (a), this solution induces a partition A_1, A_2 of \mathcal{A} where the points \mathbf{a}_i assigned to the first submodel (indicated by \square) cannot be linearly separated from those assigned to the second submodel (indicated by \triangle). In other words, the solution does not allow for a domain partition D_1, D_2 of \mathbb{R}^2 with linearly separable subdomains that is consistent with the point partition A_1, A_2 .

literature (which we briefly summarize in Section 2) and depending on the type of the domain partition that is adopted, a piecewise affine function cannot be determined by just solving an instance of k -HC. As illustrated in Figure 1 (b), the two aspects of k -PAMF-PLS, namely, *submodel fitting* and *domain partitioning*, should be taken into account at once to obtain a solution where the submodels and the domain partition are consistent. In this work, we propose exact and heuristic algorithms for k -PAMF-PLS which simultaneously consider both aspects.

The paper is organized as follows. After summarizing previous and related work in Section 2, we formally define the problem under consideration in Section 3. In Section 4, we provide a mixed-integer linear programming (MILP) formulation for k -PAMF-PLS. We then strengthen the formulation, when solving the problem in a branch-and-cut setting, by generating symmetry-breaking constraints. In Section 5, we propose a four-step heuristic to tackle larger-size instances. Computational results are reported and discussed in Section 6. Section 7 contains some concluding remarks. Portions of this work appeared, in a preliminary stage, in [ACT11,ACT12].

2 Previous and related work

In the literature, many variants of the general problem of fitting a piecewise affine model to data points have been considered. We briefly mention some of the most relevant ones in this section.

In some works, the domain is partitioned *a priori*, exploiting the domain-specific information about the dataset at hand. This approach has a typically limited applicability, as it requires knowledge of the underlying structure of the data, which may often not be available. For some examples, the reader is referred to [TV12] (which admits the use of a predetermined domain partition as a special case of a more general approach) and to the references therein.

In other works, a domain partition is easily derived when the attention is restricted to convex or concave piecewise affine models. Indeed, if the model is convex, each subdomain D_j is uniquely defined as $D_j = \{\mathbf{x} \in \mathbb{R}^n : f_j(\mathbf{x}) \geq f_{j'}(\mathbf{x}) \forall j' \in J\}$ (similarly, for concave models, with \leq instead of \geq). This is, for instance, the case of [MB09] and [MRT05], where the fitting function is the pointwise maximum (or minimum) of a set of k affine functions. A similar case is that of [RBL04], where the authors address the identification of a special type of dynamic systems which, due to the properties of the models they consider, do not require an explicit definition of the subdomains.

In the general version of the problem (that we address in this paper), a partition of the domain has to be explicitly derived together with the fitting submodels in order to obtain a piecewise affine function from \mathbb{R}^n to \mathbb{R} . To the best of our knowledge, the available methods split the problem into two subproblems that are solved sequentially: i) a *clustering problem* aiming at partitioning the data points and simultaneously fitting each subset with an affine submodel, and ii) a *classification problem* asking for a domain partition consistent with the previously determined submodels and the corresponding point partition. Note that the clustering problem considers the data points $\{(\mathbf{a}_1, b_1), \dots, (\mathbf{a}_m, b_m)\} \subset \mathbb{R}^{n+1}$, whereas the classification problem considers the original points $\{\mathbf{a}_1, \dots, \mathbf{a}_m\} \subset \mathbb{R}^n$ but not the observations b_i . The clustering phase is typically carried out by either choosing a given number k of hyperplanes which minimize the fitting error, or by finding a minimum number of hyperplanes yielding a fitting error of, at most, a given ε . We remark that these *two-phase* approaches, due to deferring the domain partition to the end of the method, may lead to poor quality solutions.

Such an approach is adopted, for instance, in [BGPV03]. In the clustering phase, as proposed in [AM02], the problem of fitting the data points in \mathbb{R}^{n+1} with a minimum number of linear submodels within a given error tolerance $\varepsilon > 0$ is formulated and solved as a Min-PFS problem, which amounts to partitioning a given infeasible linear system into a minimum number of feasible subsystems. Then, in the classification phase, the domain is partitioned via a *Support Vector Machine* (SVM). In [TPSM06], the authors solve a k -hyperplane clustering problem via the heuristic proposed in [BM00] for the clustering phase, resorting to SVM for the classification phase. The authors of [FTMLM03] adopt a variant of k -means¹ [Mac67] for the clustering phase, but fit each affine submodel *a posteriori* by solving a linear regression problem where the weighted least square

¹ k -means is a well-known heuristic to partition m points $\{\mathbf{a}_1, \dots, \mathbf{a}_m\}$ into k groups (clusters) so as to minimize the total distance between each point and the centroid (mean) of the corresponding group.

error is minimized and, then, partition the domain via SVM. A similar approach is also adopted in [BS07], where, in the first phase, a k -hyperplane clustering problem is solved as a mixed-integer linear program and, in the second phase, the domain partition is derived via *Multicategory Linear Classification* (MLC). For references to SVM and MLC, see [BM94] and [Vap96].

As already mentioned in the previous section, this kind of approaches may produce, in the first phase, affine submodels inducing a partition A_1, \dots, A_k of the points of \mathcal{A} which does not allow for a consistent domain partition D_1, \dots, D_k , i.e., for a partition where all the points \mathbf{a}_i in a subset A_j are contained into one and only one subdomain $D_{j(i)}$. Refer again to Figure 1 (b) for an illustration.

3 Problem definition

In this work, we require that the domain partition D_1, \dots, D_k of \mathbb{R}^n satisfy the property of *pairwise linear separability*, which is the basis of the so-called multicategory linear classification problem.

3.1 Pairwise linear separability and multicategory linear classification

Given k groups of points $A_1, \dots, A_k \subset \mathbb{R}^n$, the multicategory linear classification problem calls for a partition of the domain \mathbb{R}^n into k subdomains D_1, \dots, D_k where: i) for every $j \in J$, each group A_j is completely contained into the subdomain D_j , and ii) for any pair of indices $j_1, j_2 \in J$ with $j_1 \neq j_2$, the subdomains D_{j_1} and D_{j_2} of \mathbb{R}^n can be linearly separated by a hyperplane.

As shown in [DF66], such a partition can be conveniently defined by introducing, for each group of points A_j with $j \in J$, a vector of parameters $(\mathbf{y}^j, y_0^j) \in \mathbb{R}^{n+1}$ such that a point $\mathbf{a}_i \in A_j$ belongs to the subdomain D_j if and only if, for every $j' \neq j$, we have $(\mathbf{y}^j - \mathbf{y}^{j'})\mathbf{a}_i - (y_0^j - y_0^{j'}) > 0$. Note that, this way, for any pair of indices $j_1, j_2 \in J$ with $j_1 \neq j_2$, the sets of points A_{j_1} and A_{j_2} are separated by the hyperplane $H_{j_1 j_2} = \{\mathbf{x} \in \mathbb{R}^n : (\mathbf{y}^{j_1} - \mathbf{y}^{j_2})\mathbf{x} = y_0^{j_1} - y_0^{j_2}\}$ with coefficients $(\mathbf{y}^{j_1} - \mathbf{y}^{j_2}, y_0^{j_1} - y_0^{j_2}) \in \mathbb{R}^{n+1}$. See Figure 2 (a) for an illustration. It follows that, for any $j \in J$, the domain D_j is defined as:

$$D_j = \left\{ \mathbf{x} \in \mathbb{R}^n : (\mathbf{y}^j - \mathbf{y}^{j'})\mathbf{x} - (y_0^j - y_0^{j'}) > 0 \quad \forall j' \in J \setminus \{j\} \right\}. \quad (1)$$

If the group of points A_1, \dots, A_k are not linearly separable, for any choice of the vectors of parameters (\mathbf{y}^j, y_0^j) with $j \in J$, there exists at least a pair j_1, j_2 for which the inequality $(\mathbf{y}^{j_1} - \mathbf{y}^{j_2})\mathbf{a}_i - (y_0^{j_1} - y_0^{j_2}) > 0$ is violated. In this case, the typical approach is to look for a solution which minimizes the sum, over all the data points, of the so-called *misclassification error*. For a point $\mathbf{a}_i \in A_{j(i)}$, where $j(i)$ is the index of the group it belongs to, the latter is defined as:

$$\max \left\{ 0, \max_{j \in J \setminus \{j(i)\}} \left\{ -(\mathbf{y}^{j(i)} - \mathbf{y}^j)\mathbf{a}_i + (y_0^{j(i)} - y_0^j) \right\} \right\}, \quad (2)$$

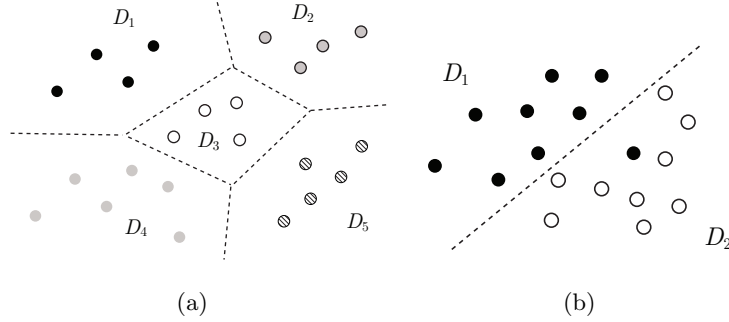


Fig. 2. (a) Pairwise linear separation of five linearly separable groups of points. (b) Classification with minimum misclassification error of two linearly inseparable groups of points (note the misclassified black point).

thus corresponding to the largest violation among the inequalities $(\mathbf{y}^{j(i)} - \mathbf{y}^j)\mathbf{a}_i - (y_0^{j(i)} - y_0^j) > 0$. For an illustration, see Figure 2 (b).

Since the set of vectors (\mathbf{y}^j, y_0^j) , for $j \in J$, satisfying constraint $(\mathbf{y}^{j(i)} - \mathbf{y}^j)\mathbf{a}_i - (y_0^{j(i)} - y_0^j) > 0$ is an open subset of \mathbb{R}^{n+1} , it is common practice to replace it by the inhomogeneous constraint $(\mathbf{y}^{j(i)} - \mathbf{y}^j)\mathbf{a}_i - (y_0^{j(i)} - y_0^j) \geq 1$, which induces a closed feasible set. This can be done without loss of generality if we assume that the norm of the vectors $(\mathbf{y}^{j(i)}, y_0^{j(i)})$, (\mathbf{y}^j, y_0^j) can be arbitrarily large, for all $j \in J$. Indeed, if $(\mathbf{y}^{j(i)} - \mathbf{y}^j)\mathbf{a}_i - (y_0^{j(i)} - y_0^j) > 0$ but $(\mathbf{y}^{j(i)} - \mathbf{y}^j)\mathbf{a}_i - (y_0^{j(i)} - y_0^j) < 1$ for some $\mathbf{a}_i \in \mathcal{A}$, then a feasible solution which satisfies the inhomogeneous constraint can be obtained by just scaling $(\mathbf{y}^{j(i)}, y_0^{j(i)})$ and (\mathbf{y}^j, y_0^j) by a constant $\lambda \geq \frac{1}{(\mathbf{y}^{j(i)} - \mathbf{y}^j)\mathbf{a}_i - (y_0^{j(i)} - y_0^j)}$. In the inhomogeneous version, the misclassification error becomes:

$$\max \left\{ 0, \max_{j \in J \setminus \{j(i)\}} \left\{ 1 - (\mathbf{y}^{j(i)} - \mathbf{y}^j)\mathbf{a}_i + (y_0^{j(i)} - y_0^j) \right\} \right\}. \quad (3)$$

3.2 k -Piecewise affine model fitting problem with pairwise linear separability

We can now provide a formal definition of k -Piecewise Affine Model Fitting with Pairwise Linear Separability.

k -PAMF-PLS: Given a set of m points $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\} \subset \mathbb{R}^n$ with the corresponding observations $\{b_1, \dots, b_m\} \subset \mathbb{R}$ and a positive integer k :

- i) partition \mathcal{A} into k subsets A_1, \dots, A_k which are *pairwise linearly separable* via a domain partition D_1, \dots, D_k of \mathbb{R}^n induced, according to Equation (1), by a set of vectors $(\mathbf{y}^j, y_0^j) \in \mathbb{R}^{n+1}$, for $j \in J$,
- ii) determine, for each subdomain D_j , an affine function $f_j : D_j \rightarrow \mathbb{R}$ where $f_j(\mathbf{x}) = \mathbf{w}^j \mathbf{x} - w_0^j$ with parameters $(\mathbf{w}^j, w_0^j) \in \mathbb{R}^{n+1}$,

so as to minimize the linear error function $\sum_{i=1}^m |b_i - (\mathbf{w}^{j(i)} \mathbf{a}_i - w_0^{j(i)})|$, where $j(i) \in J$ is the index for which $\mathbf{a}_i \in A_{j(i)} \subset D_{j(i)}$.

4 Strengthened mixed-integer linear programming formulation

In this section, we propose an MILP formulation to solve k -PAMF-PLS to optimality via branch-and-cut, as implemented in state-of-the-art MILP solvers. To enhance the efficiency of the solution algorithm, we break the symmetries that naturally arise in the formulation by generating symmetry-breaking constraints.

Our MILP formulation is derived by combining a hyperplane clustering formulation (to partition the data points into k subsets A_1, \dots, A_k and to determine an affine submodel for each of them) with multicategory linear classification constraints (to guarantee a pairwise linearly separable domain partition D_1, \dots, D_k , consistent with the k subsets A_1, \dots, A_k).

4.1 MILP formulation

For each $i \in I$ and $j \in J$, we introduce a binary variable x_{ij} which takes value 1 if the point \mathbf{a}_i is contained in the subset A_j and 0 otherwise. Let z_i be the fitting error of point $\mathbf{a}_i \in \mathcal{A}$ for each $i \in I$, $(\mathbf{w}^j, w_0^j) \in \mathbb{R}^{n+1}$ the parameters of the submodel of index $j \in J$, and $(\mathbf{y}^j, y_0^j) \in \mathbb{R}^{n+1}$, with $j \in J$, the parameters used to enforce pairwise linear separability. Let also M_1 and M_2 be large enough constants (whose value is discussed below). The formulation is as follows:

$$\min \sum_{i=1}^m z_i \tag{4}$$

$$\text{s.t. } \sum_{j=1}^k x_{ij} = 1 \quad \forall i \in I \tag{5}$$

$$z_i \geq b_i - \mathbf{w}^j \mathbf{a}_i + w_0^j - M_1(1 - x_{ij}) \quad \forall i \in I, j \in J \tag{6}$$

$$z_i \geq -b_i + \mathbf{w}^j \mathbf{a}_i - w_0^j - M_1(1 - x_{ij}) \quad \forall i \in I, j \in J \tag{7}$$

$$(\mathbf{y}^{j_1} - \mathbf{y}^{j_2}) \mathbf{a}_i - (y_0^{j_1} - y_0^{j_2}) \geq 1 - M_2(1 - x_{ij_1}) \quad \forall i \in I, j_1, j_2 \in J : j_1 \neq j_2 \tag{8}$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, j \in J \tag{9}$$

$$z_i \geq 0 \quad \forall i \in I \tag{10}$$

$$(\mathbf{w}^j, w_0^j) \in \mathbb{R}^{n+1} \quad \forall j \in J \tag{11}$$

$$(\mathbf{y}^j, y_0^j) \in \mathbb{R}^{n+1} \quad \forall j \in J. \tag{12}$$

Constraints (5) guarantee that each point $\mathbf{a}_i \in \mathcal{A}$ be assigned to exactly one submodel. Constraints (6) and (7) impose that $z_i = |b_i - \mathbf{w}^{j(i)} \mathbf{a}_i + w_0^{j(i)}|$. Indeed, together they imply that $z_i \geq |b_i - \mathbf{w}^j \mathbf{a}_i + w_0^j| - M_1(1 - x_{ij})$. When $x_{ij} = 1$, this amounts to imposing $z_i \geq |b_i - \mathbf{w}^j \mathbf{a}_i + w_0^j|$ (which will be tight in any optimal solution due to the objective function direction), while being redundant (since

$z_i \geq 0$) when $x_{ij} = 0$ and M_1 is large enough. For each $j_1 \in J$, Constraints (8) impose that all the points assigned to the subset A_{j_1} (for which the term $-M_2(1 - x_{ij})$ vanishes) belong to the intersection of all the halfspaces defined by $(\mathbf{y}^{j_1} - \mathbf{y}^{j_2})\mathbf{a}_i - (y_0^{j_1} - y_0^{j_2}) \geq 1$, whereas they are deactivated when $x_{ij_1} = 0$ and M_2 is sufficiently large. This way, we impose a zero misclassification error for each data point, thus guaranteeing pairwise linear separability among the points assigned to the different submodels. Note that, if Constraints (8) are dropped, we obtain a relaxation corresponding to a k -hyperplane clustering problem where the objective function is measured according to (4), (6) and (7).

It is important to observe that, in principle, there exists no (large enough) finite value for the parameter M_1 in Constraints (6) and (7). As an example, the fitting error between a point $(\mathbf{a}, b) = (\mathbf{e}, -1) \in \mathbb{R}^{n+1}$, where \mathbf{e} is the all-one vector, and the affine function $f = \mathbf{w}\mathbf{a} - 1$ is equal to $\|\mathbf{w}\|_1$ (the L_1 norm of \mathbf{w}) and, thus, it is unbounded and arbitrarily large for an arbitrary large $\|\mathbf{w}\|_1$. Let $j(i) \in J$ such that $x_{ij(i)} = 1$. The introduction of a finite M_1 corresponds to letting:

$$z_i = \max \left\{ \overbrace{|b_i - \mathbf{w}^{j(i)}\mathbf{a}_i + w_0^{j(i)}|}^{j=j(i) \text{ and } x_{ij(i)}=1}, \overbrace{\max_{j \in J \setminus \{j(i)\}} \{|b_i - \mathbf{w}^j\mathbf{a}_i + w_0^j| - M_1\}}^{j \neq j(i) \text{ and } x_{ij}=0} \right\}, \quad (13)$$

rather than $z_i = |b_i - \mathbf{w}^{j(i)}\mathbf{a}_i + w_0^{j(i)}|$. Therefore, a finite M_1 introduces a penalization term into the objective function, equal to:

$$\sum_{i=1}^m \max \left\{ 0, \max_{j \in J \setminus \{j(i)\}} \{|b_i - \mathbf{w}^j\mathbf{a}_i + w_0^j| - M_1\} - |b_i - \mathbf{w}^{j(i)}\mathbf{a}_i + w_0^{j(i)}| \right\}. \quad (14)$$

The effect is of penalizing solutions where the fitting error between *any* point and *any* submodel is too large, regardless of the submodels to which each point is assigned.

We face a similar issue for Constraints (8) due to the presence of the parameter M_2 . Indeed, for any finite M_2 and for $x_{ij_1} = 0$, the constraint implies $(\mathbf{y}^{j_1} - \mathbf{y}^{j_2})\mathbf{a}_i - (y_0^{j_1} - y_0^{j_2}) \geq 1 - M_2$. Hence, Constraints (8) impose that the “linear distance”² between each point \mathbf{a}_i and the hyperplane separating *any* pair of subdomains D_{j_1}, D_{j_2} be smaller than $M_2 - 1$ even if \mathbf{a}_i is not contained in either of the subdomains, i.e., even if $\mathbf{a}_i \notin A_{j_1}$ and $\mathbf{a}_i \notin A_{j_2}$.

In spite of the lack of theoretically finite values for M_1 and M_2 , setting them to a value a few orders of magnitude larger than the size of the box encapsulating the data points in \mathbb{R}^{n+1} typically suffices to produce good quality (if not optimal) solutions. We will mention an occurrence where this is not the case in Section 6.

² Given a point \mathbf{a} and a hyperplane of equation $\mathbf{w}\mathbf{x} - w_0 = 0$, the distance from \mathbf{a} to the closest point belonging to the hyperplane amounts to $\frac{|\mathbf{w}\mathbf{a} - w_0|}{\|\mathbf{w}\|_2}$. Then, the “linear distance” mentioned in the text corresponds to the point-to-hyperplane distance multiplied by $\|\mathbf{w}\|_2$.

4.2 Symmetries

Let $X \in \{0, 1\}^{m \times k}$ be the binary matrix with entries $\{X\}_{ij} = x_{ij}$ for $i \in I$ and $j \in J$. We observe that Formulation (4)–(12) admits symmetric solutions as a consequence of the existence of a symmetry group acting on the columns of X . This is because, for any X representing a feasible solution, an equivalent solution can be obtained by permuting the columns of X , an operation which corresponds to permuting the labels $1, \dots, k$ by which the submodels and subdomains are indexed.

From a computational point of view, the solvability of our MILP formulation for k -PAMF-PLS is hindered by the existence of symmetries. On the one hand, this is because, when adopting methods based on branch-and-bound, symmetries typically lead to an unnecessarily large search tree where equivalent (symmetric) solutions are discovered again and again at different nodes. On the other hand, the presence of symmetries usually leads to weaker Linear Programming (LP) relaxations, for which the barycenter of each set of symmetric solutions, which often yields very poor LP bounds, is always feasible [KP08]. This is the case of our formulation where, for a sufficiently large $M = M_1 = M_2 \geq \frac{k}{k-1} \max\{1, |b_1|, |b_2|, \dots, |b_m|\}$, the LP relaxation of Formulation (4)–(12) admits a solution of value 0. To see this, let $x_{ij} = \frac{1}{k}$ for all $i \in I, j \in J$. Constraints (5) are clearly satisfied. Let then $z_i = 0$ for all $i \in I$, $(\mathbf{w}^j, w_0^j) = (\mathbf{0}, 0)$ and $(\mathbf{y}^j, y_0^j) = (\mathbf{0}, 0)$ for all $j \in J$. Constraints (6), (7), and (8) are then satisfied whenever we have, respectively, $M_1 \frac{k-1}{k} \geq b_i$, $M_1 \frac{k-1}{k} \geq -b_i$, and $M_2 \frac{k-1}{k} \geq 1$.

A way to deal with this issue is to partition the set of feasible solutions into equivalence classes (or *orbits*) under the symmetry group, selecting a single representative per class. Different options are possible. We refer the reader to [Mar10] for an extensive survey on symmetry in mathematical programming. A possibility, originally introduced in [MDZ01, MDZ06], is of selecting as a representative the (unique) feasible solution of each orbit where the columns of X are lexicographically sorted in non-increasing order. According to [KP08], we call the convex hull of such lexicographically sorted matrices $X \in \{0, 1\}^{n \times k}$ *orbitope*.

4.3 Symmetry breaking constraints from the partitioning orbitope

Since, in our case, X is a *partitioning matrix* (a matrix $X \in \{0, 1\}^{n \times k}$ with exactly a 1 per row), we are interested in the so-called *partitioning orbitope*, whose complete linear description is given in [KP08].

Neglecting the trivial constraints, the partitioning orbitope is defined by the set of so-called *Shifted Column Inequalities* (SCIs). Call $B_{(i,j)}$ a *bar*, defined as $B_{(i,j)} = \{(i, j), (i, j+1), \dots, (i, \min\{i, k\})\}$, and $\text{col}_{(i,j)}$ a *column*, defined as $\text{col}_{(i,j)} = \{(j, j), (j+1, j), \dots, (i, j)\}$. A *shifted column* $S_{(i,j)}$ is a subset of the indices of X obtained by *shifting* some of the indices in $\text{col}_{(i,j)}$ diagonally towards the upper-left portion of X . For an illustration, see Figure 3. For two

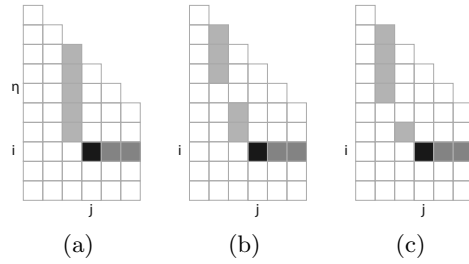


Fig. 3. (a) The bar $B_{(i,j)}$ (in black and dark gray) and the column $\text{col}_{(i-1,j-1)}$ (in light gray). (b) and (c) Two shifted columns (in light gray) obtained by shifting $\text{col}_{(i-1,j-1)}$. Note how the shifting operation introduces empty rows.

subsets of indices $B_{(i,j)}, S_{(i-1,j-1)} \subset I \times J$ thus defined, an SCI reads:

$$\sum_{(i',j') \in B_{(i,j)}} x_{i'j'} - \sum_{(i',j') \in S_{(i-1,j-1)}} x_{i'j'} \leq 0.$$

As shown in [KP08], the linear description of the partitioning orbitope is:

$$\sum_{j=1}^k x_{ij} = 1 \quad \forall i \in I \quad (15)$$

$$\sum_{(i',j') \in B_{(i,j)}} x_{i'j'} - \sum_{(i',j') \in S_{(i-1,j-1)}} x_{i'j'} \leq 0 \quad \forall B_{(i,j)}, S_{(i-1,j-1)} \quad (16)$$

$$x_{ij} = 0 \quad \forall i \in I, j \in J : j \geq i + 1 \quad (17)$$

$$x_{ij} \geq 0 \quad \forall i \in I, j \in J, \quad (18)$$

where Constraints (16) are SCIs, while Constraints (17) restrict the problem to the only elements of X that are either on the main diagonal or below it.

Although there are exponentially many SCIs, the corresponding separation problem can be solved in linear time by dynamic programming, as shown in [KP08].

When solving the MILP formulation (4)–(12) with a branch-and-cut algorithm, we generate maximally violated SCIs both at each node of the enumeration tree (by separating the corresponding fractional solution) and every time a new integer solution is found (thus separating the integer incumbent solution).

5 Four-step heuristic algorithm

As we will see in Section 6, the introduction of SCIs has a remarkable impact on the solution times. Nevertheless, even with them the MILP formulation only allows to solve small to medium size instances in a reasonable amount of computing time.

To tackle instances of larger size, we propose an efficient heuristic that takes into account all the aspects of the problem at each iteration by alternately and

coordinately solving a sequence of subproblems, namely, affine submodel fitting, point partition, and domain partition, as explained in detail in the following. Differently from other heuristic approaches in the literature (see Section 2), the domain partitioning aspect is considered at each iteration, rather than deferred to a final stage.

We start from a feasible solution composed of a point partition A_1, \dots, A_k , a domain partition D_1, \dots, D_k (induced by the parameters (\mathbf{y}^j, y_0^j) for $j \in J$), and a set of affine submodels of parameters (\mathbf{w}^j, w_0^j) , for $j \in J$. Iteratively, the algorithm tries to improve the current solution by applying the following four steps (until convergence or until a time limit is met):

- i) **Submodel Fitting:** Given the current point partition A_1, \dots, A_k of $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, determine for each $j \in J$ an affine submodel with parameters (\mathbf{w}^j, w_0^j) that minimize the linear fitting error over all the data points $\{(\mathbf{a}_1, b_1), \dots, (\mathbf{a}_m, b_m)\} \subset \mathbb{R}^{n+1}$. As we shall see, this is carried out by solving a single linear program.
- ii) **Point Partition:** Given the current set of affine submodels $f_j : D_j \rightarrow \mathbb{R}$ with $f_j(\mathbf{x}) = \mathbf{w}^j \mathbf{x} - w_0^j$ and $j \in J$, identify a set of *critical* data points $(\mathbf{a}_i, b_i) \in \mathbb{R}^{n+1}$ and (re)assign them to other submodels in an attempt to improve (decrease) the total linear fitting error over all the dataset. As described below, the identification and reassignment of such points is based on an *ad hoc* criterion and on a related control parameter.
- iii) **Domain Partition:** Given the current point partition A_1, \dots, A_k of $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, a multicategory linear classification problem is solved via Linear Programming to either find a pairwise linearly separable domain partition D_1, \dots, D_k of \mathbb{R}^n , consistent with the current point partition, or, if none exists, to construct a domain partition which minimizes the total misclassification error. In the latter case, i.e., when there is at least an index $j \in J$ for which $A_j \not\subset D_j$, we say that the previously constructed point partition is not *consistent* with the resulting domain partition D_1, \dots, D_k .
- iv) **Partition Consistency:** If the current point partition and domain partition are inconsistent, the former is modified to make it consistent with the latter. For every index $j \in J$ and every misclassified point \mathbf{a}_i (if any) belonging to A_j (i.e., for any $\mathbf{a}_i \in \mathcal{A}$ where $\mathbf{a}_i \in A_j$ and $\mathbf{a}_i \in D_{j'}$, for some $j, j' \in J$ such that $j \neq j'$), \mathbf{a}_i is reassigned to the subset $A_{j'}$ associated with $D_{j'}$.

In the following, we describe the four steps in greater detail.

In the **Submodel Fitting** step, we determine, for each $j \in J$, the submodel parameters (\mathbf{w}^j, w_0^j) yielding the smallest fitting error by solving the following

linear program:

$$\min \sum_{i=1}^m d_i \quad (19)$$

$$\text{s.t. } d_i \geq b_i - \mathbf{w}^{j(i)} \mathbf{a}_i + w_0^{j(i)} \quad \forall i \in I \quad (20)$$

$$d_i \geq -b_i + \mathbf{w}^{j(i)} \mathbf{a}_i + w_0^{j(i)} \quad \forall i \in I \quad (21)$$

$$(\mathbf{w}^j, w_0^j) \in \mathbb{R}^{n+1} \quad \forall j \in J \quad (22)$$

$$d_i \geq 0 \quad \forall i \in I, \quad (23)$$

where $j(i)$ denotes the submodel to which the point \mathbf{a}_i is currently assigned. Note that this linear program decomposes into k independent linear programs, one per submodel.

Let us now consider the **Point Partition** step. Many clustering heuristics (see, e.g., [Mac67,BM00]) are based on the iterative reassignment of each point \mathbf{a}_i to a subset A_j whose corresponding submodel yields the smallest fitting error. As shown in Figure 4 (a), and as it can be confirmed computationally, this choice is likely to lead to poor quality local minima. In our method, we adopt a criterion to help identify a set of *critical points* which might jeopardize the overall quality of the current solution. The criterion is an adaptation of the corresponding one employed in the Distance-Based Point Reassignment heuristic (DBPR) proposed in [AC13] for the k -hyperplane clustering problem.

The idea is to identify as critical those points which, not only give a large contribution to the total fitting error for their current submodel, but also have another submodel to which they could be reassigned without increasing too much the overall fitting error. The set of such points is determined by ranking, for each subset A_j , each point $\mathbf{a}_i \in A_j$ in nonincreasing order with respect to the ratio between its fitting error w.r.t. the *current* submodel of index $j(i)$ and the fitting error w.r.t. a *candidate* submodel. The latter is defined as the submodel that best fits \mathbf{a}_i but which is different from $j(i)$. Formally, for each $j \in J$, the criterion ranks each point $\mathbf{a}_i \in A_j$ with respect to the quantity:

$$\frac{|b_i - \mathbf{w}^{j(i)} \mathbf{a}_i + w_0^{j(i)}|}{\min_{j \in J \setminus \{j(i)\}} \{|b_i - \mathbf{w}^j \mathbf{a}_i + w_0^j|\}}. \quad (24)$$

The **Point Partition** step also relies on a *control parameter* $\alpha \in [0, 1)$. Given a current solution characterized by a point partition A_1, \dots, A_k and the k associated affine submodels, let $m(j)$ denote, for all $j \in J$, the cardinality of A_j . At each iteration and for each submodel of index $j \in J$, $\lceil \alpha m(j) \rceil$ of the points with highest rank are reassigned to the corresponding candidate submodel, even if this leads to a worse objective function value. The remaining points are simply reassigned to the closest submodel (if they are not already assigned to it). Then, α is decreased exponentially, by updating it as $\alpha := 0.99\rho^t$, for some parameter $\rho \in (0, 1)$, where t is the index of the current iteration. For an illustration, see Figure 4 (b).

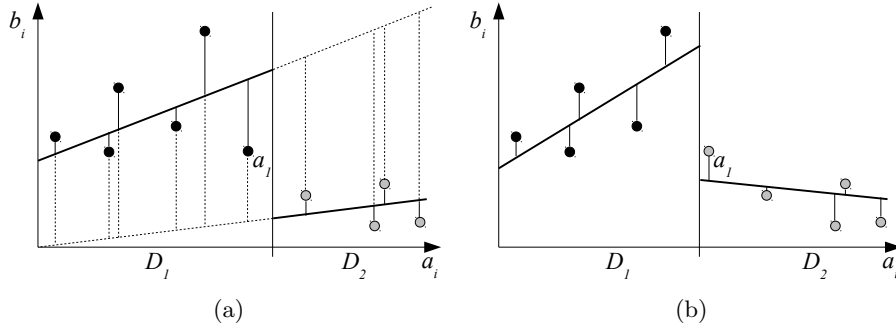


Fig. 4. (a) A solution corresponding to a local minimum for an algorithm where each point is reassigned to the submodel yielding the smallest fitting error. (b) An improved solution that can be obtained by reassigning the point \mathbf{a}_1 in (a) which, according to our criterion, achieves the highest ranking, to the rightmost submodel, and by updating the affine submodel fitting as well as the domain partition.

Since the reassignment of critical points, as identified by our criterion, introduces a high variability in the sequence of solutions that are generated, the search process is stabilized by decreasing α to 0 over the iterations, which progressively leads from substantial changes of the current solution to fine polishing operations.

To avoid cycling effects, whenever a worse solution is found we add the whole set of the point-to-submodel assignments that we carried out involving critical points to a tabu list of short memory. We also consider an aspiration criterion, that is, a criterion that allows to override the tabu status of a move if, by performing it, a solution with an objective function value that is better than that of the best solution found so far can be achieved. Since it is computationally too demanding to compute the exact objective function value of a solution obtained after the reassignment of a model from a submodel to another one (as it would require to carry out the **Submodel Fitting**, **Domain Partition**, and **Partition Consistency** steps for each point, at each iteration), we consider a partial aspiration criterion in which we override the tabu status of a point-to-submodel reassignment only if the corresponding fitting error is strictly smaller than the value that was registered when the reassignment move was added to the tabu list.

In the **Domain Partition** step, we derive a domain partition by constructing a pairwise linear separation of the sets A_1, \dots, A_k which minimizes the total

misclassification error. This is achieved by solving the following linear program:

$$\min \sum_{i=1}^m e_i \quad (25)$$

$$\text{s.t. } e_i \geq -(\mathbf{y}^{j(i)} - \mathbf{y}^j)\mathbf{a}_i + (y_0^{j(i)} - y_0^j) + 1 \quad \forall i \in I, j \in J \setminus \{j(i)\} \quad (26)$$

$$e_i \geq 0 \quad \forall i \in I \quad (27)$$

$$(\mathbf{y}^j, y_0^j) \in \mathbb{R}^{n+1} \quad \forall j \in J, \quad (28)$$

where $j(i)$ denotes the submodel to which the point \mathbf{a}_i is currently assigned, and e_i represents the misclassification error of point \mathbf{a}_i , for all $i \in I$. If this subproblem admits an optimal solution with total misclassification error equal to 0, then the k subsets are linearly separable, otherwise the current solution contains at least a misclassified point $\mathbf{a}_i \in A_{j(i)}$ where $\mathbf{a}_i \in D_j$, for some $j \in J \setminus \{j(i)\}$. Each such point is then reassigned to A_j in the **Partition Consistency** step.

The overall four-step algorithm, which we refer to with the shorthand 4S-CR (4 Steps-CRiterion), starts from a point assignment obtained by randomly generating the coefficients of k affine submodels and assigning each point (\mathbf{a}_i, b_i) to a submodel yielding the smallest fitting error (ties are broken arbitrarily). The above four steps are repeated until $\alpha = 0$, while storing the best solution found so far. The method is then restarted until the time limit is reached.

Note that the **Domain Partition** step drives the search towards solutions that induce a suitable domain partition, avoiding infeasible solutions which are good from a submodel fitting point of view but do not admit a pairwise linearly separable domain partition, i.e., where $A_j \subset D_j$ does not hold for all $j \in J$. Then the **Partition Consistency** step makes sure that the point partition and domain partition are consistent at the end of each iteration.

6 Computational results

In this section, we report and discuss on a set of computational results obtained when solving k -PAMF-PLS either to optimality with branch-and-cut and symmetry breaking constraints or with our four-step heuristic 4S-CR. First, we investigate the impact of symmetry breaking constraints when solving the problem to global optimality. On a subset of instances for which the exact approach is viable, we compare the best solutions obtained with the exact algorithm (within a time limit) to those produced by our heuristic method. Then we experiment with 4S-CR on larger instances and also assess the impact of its main components on the overall quality of the solutions found.

6.1 Experimental setup

The exact formulation is solved with CPLEX 12.5, interfaced with the Concert library in C++. The separation algorithm for SCIs and the heuristic methods

are implemented in C++ and compiled with GNU-g++-4.3. SCIs are added to the default branch-and-cut algorithm implemented in CPLEX via both a *lazy constraint* callback and a *user cut* callback, thus separating SCIs for both integer and fractional solutions. This way, with lazy constraints we guarantee the lexicographic maximality of the columns of the partitioning matrix X for any feasible solution found by the method. With user cuts, we also allow for the introduction of SCIs at the different nodes of the branch-and-cut tree, thus tightening the LP relaxations. In 4S-CR (and its variants, as introduced in the following), the `Submodel Fitting` and `Domain Partition` steps are carried out by solving the corresponding LPs, namely (19)–(23) and (25)–(28), with CPLEX.

The experiments are conducted on a Dell PowerEdge Quad Core Xeon 2.0 Ghz, with 16 GB of RAM. In the heuristics, we set $\rho = 0.5$ and adopt a tabu list with a short memory of two iterations.

6.2 Test instances

We consider both a set of structured, randomly generated instances, as well as some real-world ones taken from the UCI repository [FA13].

We classify the random instances into four groups: *small* ($m = 20, 30, 40, 50, 60, 75, 100$ and $n = 2, 3, 4, 5$), *medium* ($m = 500$ and $n = 2, 3, 4, 5$), and *large* ($m = 1000$ and $n = 2, 3, 4, 5$)³. They are constructed by randomly sampling the data points \mathbf{a}_i and the corresponding observations b_i from a randomly generated (discontinuous) piecewise affine model with $k = 5$ pieces and an additional Gaussian noise. First, we generate k subdomains D_1, \dots, D_k by solving a multiway linear classification problem on k randomly chosen representative points in \mathbb{R}^n . Then, we randomly choose the submodel parameters (\mathbf{w}^j, w_0^j) for all $j \in J$ and sample, uniformly at random, the m points $\{\mathbf{a}_1, \dots, \mathbf{a}_m\} \in \mathbb{R}^n$. For each sampled point \mathbf{a}_i , we keep track of the subdomain $D_{j(i)}$ which contains it and set b_i to the value that the affine submodel of index $j(i)$ takes in \mathbf{a}_i , i.e., $\mathbf{w}^{j(i)}\mathbf{a}_i - w_0^{j(i)}$. Then, we add to b_i an additive Gaussian noise with 0 mean and a variance which is chosen, for each submodel, by sampling uniformly at random within $[\frac{7}{10} \cdot \frac{3}{1000}, \frac{3}{1000}]$. For convenience, but w.l.o.g., after an instance has been constructed, we rescale all its data points (and their observations) so that they belong to $[0, 10]^{n+1}$.

As to the real-world instances, we consider four datasets from the UCI repository: *Auto MPG* (`auto`), *Breast Cancer Wisconsin Original* (`breast`), *Computer Hardware* (`cpu`), and *Housing* (`house`). We remove data points with missing features, convert each categorical attribute (if any) to a numerical value, and normalize the data so that each point belongs to the interval $[0, 10]^{n+1}$. We then perform feature extraction via Principal Component Analysis (PCA), using the

³ We do not consider instances with $n = 1$ since k -PAMF-PLS is pseudopolynomially solvable in this case. Indeed, if the domain coincides with \mathbb{R} , then the number of linear domain partitions is, at most, $O(m^k)$. An optimal solution to k -PAMF-PLS can thus be found by constructing all such partitions and then solving, for each of them, an affine model fitting problem in polynomial time by Linear Programming.

Matlab toolbox PRTools, calling the function `PCAM(A,0.9)`, where `A` is the Matlab data structure where the data points are stored. After preprocessing, the instances are of the following size: $m = 397, n = 3$ (`auto`), $m = 698, n = 5$ (`breast`), $m = 209, n = 5$ (`cpu`), and $m = 506, n = 8$ (`house`).

All the instances are solved with different values of k , namely, $k = 2, 3, 4, 5$. This way, the experiments are in line with a real-world scenario where the complexity of the underlying model is unknown.

Throughout the section, speedup factors and average improvements will be reported as ratios of geometric means.

6.3 Exact solutions via the MILP formulations

We test our MILP formulation with and without SCIs on the small dataset, considering four figures:

- total computing time (in seconds) needed to solve the problem, including the generation of SCIs as symmetry breaking constraints (**Time**);
- total number of branch-and-bound nodes that have been generated, divided by 1000 (**Nodes[k]**);
- percent gap at the end of the computations (**Gap**), defined as $100 \frac{|LB-UB|}{10^{-4}+|LB|}$, where LB and UB are the tightest lower and upper bounds that have been found; if $LB = 0$, a “-” is reported;
- total number of generated symmetry breaking constraints (**Cuts**).

The instances are solved for $k = 2, 3, 4, 5$, within a time limit of 3600 seconds. We run CPLEX in deterministic mode on a single thread with default settings. In all the cases, we set $M = M_1 = M_2 = 1000$.

The results are reported in Tables 1 for $k = 2, 3$ and in Table 2 for $k = 4, 5$. In the second table, we omit the results for the instances with $n = 4, 5$ as, both with or without SCIs, no solutions with a finite gap are found within the time limit (i.e., the lower bound LB is always 0). Note that, for $k = 2$, symmetry is broken by just fixing the top left element of the matrix X to 1, i.e., by letting, w.l.o.g., $x_{11} = 1$. Hence, we do not resort to the generation of SCIs in this case.

Let us neglect the case of $k = 2$ and focus on the full set of 56 k -PAMF-PLS problems that are considered for this dataset (28 instances for $k = 3$ and 14 for $k = 4, 5$). Without SCI inequalities, we achieve an optimal solution in 24 cases out of 56 (43%). The introduction of SCIs has a very positive impact. They allow to solve to optimality 10 more instances, for a total of 34 (60.1%). SCIs also yield a substantial reduction in both computing time and number of nodes. When focusing on the 24 instances solved by both variants of the algorithm, the overall results show that the introduction of SCIs yields a speedup, on (geometric) average, of almost 3 times, corresponding to a reduction of 66% of the computing times. The number of nodes is reduced by the same factor of 66%. Interestingly, this improvement is obtained by adding a rather small number of cuts which, in practice, prove to be highly effective. See, e.g., the instance with $m = 40, n = 3$ which, when solved for $k = 3$ with SCIs, presents

Table 1. Results obtained on the small dataset when solving the MILP formulation for $k = 2$ (without SCIs) and for $k = 3$ (with and without SCIs). For $k = 3$ and for each instance, if both variants achieve an optimal solution, the smallest number of nodes and computing time are highlighted in boldface. If at least a variant does not achieve an optimal solution, the smallest gap is highlighted.

		$k=2$			$k=3$						
		without SCIs			without SCIs			with SCIs			
n	m	Time	Nodes[k]	Gap	Time	Nodes[k]	Gap	Time	Nodes[k]	Gap	Cuts
2	20	0.1	0.2	0.0	2.3	2.0	0.0	2.1	1.1	0.0	12
2	30	0.7	0.5	0.0	4.8	6.5	0.0	4.3	5.0	0.0	10
2	40	0.7	0.7	0.0	7.9	11.3	0.0	4.1	4.0	0.0	18
2	50	0.8	0.6	0.0	9.3	9.8	0.0	4.0	5.2	0.0	17
2	60	1.2	1.0	0.0	14.7	17.9	0.0	8.7	8.3	0.0	8
2	75	2.4	1.2	0.0	20.2	20.7	0.0	8.8	7.7	0.0	8
2	100	3.8	1.7	0.0	43.6	28.9	0.0	43.1	28.4	0.0	13
3	20	0.4	0.9	0.0	13.9	29.0	0.0	5.6	9.3	0.0	7
3	30	0.7	1.2	0.0	58.3	124.4	0.0	35.6	55.1	0.0	20
3	40	1.8	2.2	0.0	226.9	291.7	0.0	49.1	64.6	0.0	19
3	50	1.5	3.1	0.0	603.6	467.0	0.0	117.7	122.2	0.0	15
3	60	4.7	5.1	0.0	615.9	440.3	0.0	171.5	147.8	0.0	22
3	75	5.4	7.4	0.0	3600.0	802.4	76.7	512.7	324.8	0.0	29
3	100	9.9	19.6	0.0	3600.0	757.1	89.4	3196.2	1272.8	0.0	42
4	20	1.2	2.3	0.0	131.0	270.1	0.0	35.2	84.0	0.0	10
4	30	2.1	3.8	0.0	633.1	802.8	0.0	167.8	224.8	0.0	11
4	40	5.6	11.1	0.0	3600.0	1519.4	79.3	3345.2	1867.2	0.0	21
4	50	8.6	20.8	0.0	3600.0	1096.8	-	3600.0	1206.0	85.3	19
4	60	15.2	39.0	0.0	3600.0	970.8	-	3600.0	1238.7	89.7	22
4	75	50.3	87.1	0.0	3600.0	851.1	-	3600.0	1014.9	86.6	17
4	100	98.9	192.3	0.0	3600.0	529.5	-	3600.0	508.1	-	26
5	20	1.9	5.6	0.0	679.6	974.1	0.0	170.0	311.7	0.0	11
5	30	6.3	16.2	0.0	3600.0	1775.3	-	3600.0	2054.0	73.3	18
5	40	27.7	61.4	0.0	3600.0	1422.8	-	3600.0	1363.3	-	13
5	50	54.4	125.6	0.0	3600.0	1118.3	-	3600.0	1132.1	-	24
5	60	491.2	830.6	0.0	3600.0	963.5	-	3600.0	1009.1	-	17
5	75	1751.1	1841.4	0.0	3600.0	788.0	-	3600.0	815.1	-	17
5	100	3600.0	3649.2	9.3	3600.0	769.3	-	3600.0	623.1	-	32

Table 2. Results obtained on the small dataset when solving the MILP formulation for $k = 4, 5$ with and without SCIs. For each instance, if both variants achieve an optimal solution, the smallest number of nodes and computing time are highlighted in boldface. If at least a variant does not achieve an optimal solution, the smallest gap is highlighted.

		$k=4$						$k=5$							
		without SCIs			with SCIs			without SCI			with SCI				
n	n	Time	Nodes[k]	Gap	Time	Nodes[k]	Gap	Cuts	Time	Nodes[k]	Gap	Time	Nodes[k]	Gap	Cuts
2	20	7.6	12.9	0.0	4.5	4.0	0.0	27	112.2	152.4	0.0	31.1	35.3	0.0	268
2	30	31.6	41.9	0.0	7.2	9.0	0.0	39	554.8	441.3	0.0	59.6	48.2	0.0	397
2	40	105.8	115.3	0.0	19.0	18.6	0.0	86	3600.0	1308.2	28.7	98.2	75.1	0.0	472
2	50	156.0	145.3	0.0	35.8	39.6	0.0	174	3600.0	1347.0	20.3	347.4	153.2	0.0	893
2	60	411.9	275.1	0.0	244.1	143.0	0.0	86	3600.0	865.1	90.2	1062.9	453.1	0.0	783
2	75	520.9	328.3	0.0	158.5	118.0	0.0	155	3600.0	577.0	97.6	2385.9	739.7	0.0	1034
2	100	3600.0	673.3	15.7	367.0	169.6	0.0	233	3600.0	356.0	98.6	3600.0	319.2	98.2	1105
3	20	1067.9	1246.5	0.0	118.8	176.2	0.0	67	3600.0	2306.5	-	2013.0	1401.2	0.0	428
3	30	3600.0	1590.5	-	2336.6	1480.7	0.0	125	3600.0	1491.6	-	3600.0	1236.9	-	825
3	40	3600.0	1188.1	-	3600.0	1344.4	34.2	91	3600.0	1077.4	-	3600.0	871.3	-	385
3	50	3600.0	896.5	-	3600.0	847.6	94.4	174	3600.0	738.2	-	3600.0	704.0	-	496
3	60	3600.0	754.1	-	3600.0	697.5	-	120	3600.0	725.4	-	3600.0	597.2	-	580
3	75	3600.0	626.7	-	3600.0	458.8	-	187	3600.0	444.9	-	3600.0	331.0	-	843
3	100	3600.0	440.5	-	3600.0	370.5	-	267	3600.0	324.4	-	3600.0	221.4	-	1691

a speedup in the computing time, when compared to the case without SCIs, of 4.6 times (corresponding to a reduction of 78%) with the sole introduction of 19 symmetry breaking constraints.

In our preliminary experiments, we observed the generation of a higher number of cuts when employing older versions of CPLEX, such as 12.1 and 12.2 whereas, with CPLEX 12.5, their number is significantly smaller. This is, most likely, a consequence of the introduction of more aggressive techniques for symmetry detection and symmetry breaking in the latest versions of CPLEX. We nevertheless remark that the improvement in computing time provided by the introduction of SCIs appears to be comparable for all the versions of CPLEX 12, regardless of the number of cuts that are generated.

Although the introduction of SCIs clearly increases the number of instances which can be solved to optimality, the results in Tables 1 and 2 show that the exact approach via mixed-integer linear programming might require large computing times even for fairly small instances with $n \geq 3$ and $m \geq 40$ for $k \geq 4$. For $k = 2$, all the instances are solved to optimality, with the sole exception of the instance with $m = 100$, $n = 5$ (which reports a gap of 9.3%). For $k = 3$, Table 1 shows that, already for $n = 4$ and $m \geq 50$, the gap after one hour is still larger than 80%. According to Table 2, the exact approach becomes impractical for $n = 3$ and $m \geq 40$ for $k = 4$, and for $n = 3$ and $m \geq 30$ for $k = 5$.

6.4 Comparison between the four-step heuristic 4S-CR and the MILP formulation

Before assessing the effectiveness of 4S-CR on larger instances, we compare the solutions it provides with the best ones found via mixed-integer linear programming on the small dataset (within the time limit). The results are reported in Table 3. For a fair comparison, 4S-CR is run, for the instances that are solved to optimality by the exact method, for the same time taken by the latter. For the instances for which an optimal solution has not been found, 4S-CR is run up to the time limit of 3600 seconds.

When comparing the quality of the solutions found by 4S-CR with those found by the MILP formulation with SCIs, we register, for $k = 2$, very close to optimal solutions with, on (geometric) average, a 4% larger fitting error. This number decreases to 1% for $k = 3$. For larger values of k , namely $k = 4$ and $k = 5$, for which the number of instances that are unsolved when adopting the MILP formulation is much larger, 4S-CR yields solutions that are much better than those found via mixed-integer linear programming. When neglecting the instances with an optimal solution of value 0 (which would skew the geometric mean), the solutions provided by 4S-CR are, on geometric average, better than those obtained via the exact method by 14% for $k = 4$ and by 20% for $k = 5$.

For $k = 2, 3, 4, 5$, 4S-CR finds equivalent or better solutions than those obtained via mixed-integer linear programming in, respectively, 11, 15, 18, and 19 cases, with strictly better solutions in, respectively, 1, 5, 16, and 15 cases.

Table 3. Comparison between the best results obtained for $k = 2, 3, 4, 5$ on the small instances when solving the MILP formulation (with symmetry breaking constraints and within a time limit of 3600 seconds) and those obtained via 4S-CR. The latter is run for as much time as that required to solve the MILP formulation (within the time limit). For each instance, the value of the best solution found is highlighted in boldface.

		$k = 2$			$k = 3$			$k = 4$			$k = 5$		
n	m	Objective			Objective			Objective			Objective		
		Time	MILP	4S-CR	Time	MILP	4S-CR	Time	MILP	4S-CR	Time	MILP	4S-CR
2	20	0.1	18.3	22.4	2.1	9.6	9.6	4.5	6.1	7.1	31.1	4.5	6.1
2	30	0.1	30.5	30.5	4.3	19.3	19.3	7.2	10.0	10.0	59.6	7.7	8.8
2	40	0.5	61.3	61.3	4.1	37.8	45.4	19.0	24.7	35.5	98.2	14.8	21.2
2	50	0.3	56.0	56.0	4.0	39.1	40.9	35.8	26.9	29.5	347.4	22.1	22.1
2	60	1.2	86.6	91.7	8.7	53.1	61.8	244.1	43.2	53.1	1062.9	35.7	43.5
2	75	1.7	40.5	45.4	8.8	31.3	31.6	158.5	28.6	30.5	2385.9	27.3	28.4
2	100	1.6	114.9	114.9	43.1	62.9	62.9	367.0	48.4	48.5	3600.0	187.4	48.4
3	20	0.3	11.3	11.3	5.6	4.3	4.6	118.8	2.3	2.4	2013.0	0.4	0.9
3	30	0.7	14.1	13.8	35.6	9.4	9.0	2336.6	6.3	6.3	3600.0	4.5	4.7
3	40	2.9	29.3	32.3	49.1	19.6	21.2	3600.0	14.7	15.5	3600.0	11.7	11.7
3	50	2.0	48.6	55.5	117.7	26.3	26.3	3600.0	20.2	17.9	3600.0	21.2	15.1
3	60	4.9	40.0	40.0	171.5	22.7	25.0	3600.0	22.7	17.5	3600.0	17.3	12.9
3	75	5.9	72.5	82.4	512.7	43.9	47.9	3600.0	35.2	31.6	3600.0	54.0	31.5
3	100	10.3	86.5	88.0	3196.2	51.3	51.3	3600.0	77.2	33.6	3600.0	69.2	33.2
4	20	1.2	7.5	7.5	35.2	2.3	2.3	3600.0	0.2	0.3	2.1	0.0	0.4
4	30	1.2	20.0	20.3	167.8	6.8	6.8	3600.0	3.5	3.4	3600.0	1.3	1.4
4	40	4.1	34.4	34.4	3345.2	16.5	16.5	3600.0	11.0	10.3	3600.0	7.4	6.8
4	50	6.2	38.0	38.0	3600.0	20.5	20.0	3600.0	19.0	13.4	3600.0	17.7	6.3
4	60	12.1	40.1	41.0	3600.0	28.8	27.3	3600.0	24.8	17.9	3600.0	21.1	15.5
4	75	23.0	85.0	88.5	3600.0	49.4	53.9	3600.0	50.4	37.8	3600.0	46.1	32.4
4	100	57.0	110.3	118.0	3600.0	113.9	74.0	3600.0	139.3	49.7	3600.0	117.6	43.3
5	20	2.6	8.9	8.9	170.0	0.5	0.5	0.3	0.0	0.5	0.6	0.0	0.0
5	30	4.0	17.1	19.2	3600.0	6.7	6.7	3600.0	1.9	1.7	3600.0	0.3	0.6
5	40	12.1	37.9	41.7	3600.0	18.7	19.5	3600.0	13.3	8.9	3600.0	6.7	5.1
5	50	32.0	28.9	29.4	3600.0	21.7	17.9	3600.0	15.8	12.4	3600.0	9.7	8.4
5	60	457.1	58.4	58.6	3600.0	43.9	44.0	3600.0	37.5	31.9	3600.0	33.6	21.2
5	75	820.9	62.6	65.0	3600.0	26.7	29.4	3600.0	50.9	20.7	3600.0	53.0	18.4
5	100	3600.0	79.3	84.1	3600.0	56.0	60.8	3600.0	60.5	49.3	3600.0	61.6	42.9

Overall, when considering the instances jointly, 4S-CR performs as good or better than mixed-integer linear programming in 63 cases out of 112 (28 instances, each solved 4 times, once per value of k), strictly improving over the latter in 37 cases. This indicates that the quality of the solutions found via 4S-CR can be quite high even for small-size instances and that the difference w.r.t. the exact method, at least on the instances for which a comparison is viable, seems to be increasing with the number of points m , the number of dimensions n , and the number of submodels k .

Note that, for the instance with $m = 30, n = 3$ and for both $k = 2$ and $k = 3$, the MILP formulation yields, in strictly less than the time limit, a solution which is worse than the corresponding one found by 4S-CR. As discussed in Section 4, this is most likely due to the selection of too small values for the parameters M_1 and M_2 . Experimentally, we observed that the issue can be avoided by choosing $M = M_1 = M_2 = 10000$, although at the cost of a substantially larger computing time (due to the need for a higher numerical precision to handle the larger differences between the magnitudes of the coefficients in the formulation).

6.5 Experiments with 4S-CR on larger instances and impact of the main 4S-CR features

We now present the results obtained with 4S-CR on larger instances and assess the impact of the main features of 4S-CR (i.e., the criterion for identifying and reassigning critical points in the `Point Partition` step, the `Domain Partition` step, and the `Partition Consistency` step, applied at each iteration) on the quality of the solutions found.

As already mentioned, we set $\rho = 0.5$ in all the experiments involving our criterion for identifying critical points and we consider a tabu list with a memory of two iterations. When tuning the parameters, we observed improved results on the smaller instances when increasing ρ , as opposed to worse ones on the larger instances. This is, most likely, a consequence of the number of iterations carried out within the time limit, which becomes much smaller for a larger value of ρ , thus forcing the method to halt with a solution which is too close to the starting one. As to the tabu list, we observed that a short memory of two iterations suffices to prevent loops. Indeed, due to the nature of the problem as well as due to the many aspects of k -PAMF-PLS that our method considers, the values of the parameters of the piecewise-affine submodels change often dramatically within very few iterations. This way, few iterations taking place after a worsening move typically suffice to prevent that a point-to-submodel reassignment take place twice, thus making the occurrence of loops extremely unlikely.

To assess the impact of the `Point Partition` step based on the criterion for critical points (and on the corresponding control parameter), we introduce a variant of 4S-CR where, in the former step, *every* point (\mathbf{a}_i, b_i) is (re)assigned to a submodel yielding the smallest fitting error. This is in line with many popular clustering heuristics, as reported in Section 2. We refer to this method as 4S-CL (where “CL” stands for “closest”).

To evaluate the relevance of considering, in 4S-CR, the domain partition aspect directly at each iteration (via the `Domain Partition` and `Partition Consistency` steps), we also consider a standard (STD) two-phase method which, first, addresses the clustering aspect of k -PAMF-PLS and, only at the end, before halting, takes the domain partition aspect into account. In the algorithm, which we consider in two versions, we iteratively alternate between the `Submodel Fitting` and `Point Partition` steps. In the latter step, we either reassign every point to the “closest” submodel (STD-CL) or to that indicated by our criterion (STD-CR). After a local minimum has been reached, a pairwise linearly separable domain partition with minimum misclassification error is derived by solving a multiway linear classification problem via linear programming, as in Problem (25)–(28).

Since STD-CL is similar to most of the standard techniques proposed in the literature (see Section 2), we consider it as the baseline method and compare the other methods (4S-CR, 4S-CL, and STD-CR) to it. The results for the medium, large, and UCI datasets, obtained with a time limit of, respectively, 900, 1800, and 900 seconds, are reported in Table 4.

The comparison shows that 4S-CR outperforms the baseline method STD-CL in almost all the cases. When considering the medium instances, 4S-CR yields an improvement in objective function value of, on geometric average, 8%, 21%, 21%, and 24% for, respectively, $k = 2, 3, 4,$ and 5 . On the large instances, the improvement is of 16%, 24%, 29%, and 24%. For the four UCI instances, the improvement is of 6%, 9%, 13%, and 16%. When considering the three datasets jointly, the improvement is of 8%, 21%, 21%, 24%. On geometric average, for all the values of k , 4S-CR improves on the fitting error of STD-CL by 20%. On a total of 112 instances (for the different values of k) of k -PAMF-PLS, 4S-CR achieves the best solution in 103 cases (92%).

When comparing 4S-CL to STD-CL, we register, on geometric mean and for the different values of k , an improvement of 4%, 9%, 10%, and 16% for the medium instances, of 12%, 17%, 17%, and 11% for the large ones, and of 4%, 9%, 10%, and 16% on the UCI datasets. When considering all the datasets and all the values of k jointly, the improvement is of 12%. Although still substantial, this value is not as large as that for 4S-CR, thus highlighting the relevance of the criterion based on critical points that is adopted in the `Point Partition` step. At the same time, it also shows that, even without the criterion, the central idea of 4S-CR (i.e., considering the domain partition aspect of the problem directly at each iteration, rather than deferring it to a final phase) has a large positive impact on the solution quality.

Most interestingly, the results for STD-CR are quite poor. When considering all the 112 instances (for the different values of k), the method yields, on geometric average, a 4% larger fitting error w.r.t. STD-CL. This is not surprising as, by constructing a domain partition only at the very end of the algorithm, the solutions that are obtained before its derivation typically contain a large number of misclassified points, which yield a large negative contribution to the final fitting error. Indeed, when comparing the value of the solutions that are

Table 4. Results obtained with 4S-CR, when compared to STD-CL, STD-CR, and 4S-CL on the medium, large, and UCI instances, within a time limit of, respectively, 900, 1800, and 900 seconds, for $k = 2, 3, 4, 5$. For each instance, the value of the best solution found is highlighted in boldface.

	n	m	$k = 2$		$k = 3$		$k = 4$		$k = 5$										
			STD-CL	STD-CR	4S-CL	4S-CR	STD-CL	STD-CR	4S-CL	4S-CR	STD-CL	STD-CR							
medium	2	500	658.6	863.4	659.7	592.7	734.7	592.7	472.9	638.3	790.6	485.8	345.7	638.3	784.9	493.1	472.9		
	2	500	406.2	406.2	406.2	406.2	400.1	400.1	274.1	416.8	416.8	387.8	254.0	416.8	400.1	383.6	254.0		
	2	500	408.5	408.5	408.4	408.4	365.6	292.7	394.3	267.4	288.7	265.4	325.6	266.7	330.5	424.4	325.6	226.3	
	3	500	392.1	386.7	378.7	358.2	334.0	346.1	310.7	279.2	318.6	319.9	309.9	286.6	345.5	351.0	310.5	281.3	
	3	500	518.3	484.1	472.9	448.5	500.3	371.6	300.0	262.7	496.2	511.8	288.2	257.4	491.3	533.3	292.4	256.8	
	3	500	699.7	698.6	580.4	574.6	612.2	627.9	508.8	447.1	502.1	548.1	447.9	449.3	532.7	564.3	411.1	427.6	
	4	500	421.1	426.6	412.2	403.7	378.0	392.0	334.1	325.3	308.4	325.2	268.8	260.2	328.1	339.1	206.2	193.0	
	4	500	669.0	670.0	643.0	623.8	536.6	521.1	505.5	474.3	435.5	492.3	395.7	375.2	423.4	429.2	377.2	318.8	
	4	500	642.2	625.2	587.0	569.3	553.9	516.2	528.5	513.2	550.9	580.9	474.9	493.1	566.9	566.9	479.5	472.4	
	5	500	531.7	511.6	520.5	503.9	381.8	403.2	370.4	360.8	280.4	348.2	264.0	242.8	277.6	278.7	230.3	264.7	
	5	500	373.9	399.0	371.4	344.9	257.3	388.7	261.0	252.3	246.1	273.9	240.7	230.5	232.0	298.9	237.7	218.8	
	5	500	635.1	616.0	603.4	593.7	610.4	592.0	536.4	509.8	449.3	616.0	472.5	421.7	441.4	620.6	464.5	489.0	
	large	2	1000	654.5	654.5	654.5	654.5	654.1	654.1	572.5	572.5	654.6	670.0	572.5	557.6	565.7	644.5	572.5	557.8
		2	1000	1685.4	1618.5	1452.5	1241.9	1685.4	916.0	931.0	790.5	1715.6	1994.0	931.0	581.6	1208.2	1903.3	931.0	647.8
		2	1000	1423.0	1042.1	1019.9	953.2	1102.5	970.6	849.1	634.3	1190.9	1405.6	809.0	501.1	953.2	1289.1	752.0	490.1
3		1000	1371.2	1127.6	1095.8	976.8	1182.8	1053.2	743.2	729.7	780.3	1089.1	732.0	621.9	1003.7	1132.4	714.2	605.7	
3		1000	1099.5	1072.9	1023.5	935.8	969.5	1010.8	953.6	829.5	997.3	1014.7	934.8	793.6	957.3	1047.9	943.4	654.5	
3		1000	1545.1	1408.3	1107.6	1084.2	1217.2	1171.3	853.9	805.0	1152.5	1209.7	777.9	536.4	919.5	1258.6	777.9	616.5	
4		1000	520.1	516.2	519.4	503.9	520.1	547.7	481.1	422.3	520.1	557.3	438.2	455.0	503.7	514.3	483.1	443.1	
4		1000	672.8	672.6	666.0	655.0	536.9	544.7	502.9	480.8	500.7	513.8	481.9	450.9	511.2	521.9	478.5	463.6	
4		1000	1154.7	1076.1	951.0	924.1	963.3	979.4	854.6	797.7	869.5	895.1	705.8	625.1	928.0	950.6	758.5	710.4	
5		1000	1174.6	1165.9	1110.6	1092.8	1079.3	1061.9	972.2	941.1	910.1	883.1	861.4	846.0	915.8	1049.4	880.2	856.1	
5		1000	985.3	936.9	878.6	859.3	760.3	912.1	704.9	685.8	671.9	689.2	641.4	614.7	671.2	717.1	656.9	646.8	
5		1000	570.3	568.1	573.2	565.2	487.7	531.5	491.3	471.5	474.0	521.2	469.9	464.5	481.4	486.5	476.2	451.3	
auto		3	397	23.0	23.0	22.6	22.1	21.2	23.9	21.0	20.8	21.2	23.9	19.5	19.7	21.0	23.3	18.2	19.1
UCI		breast	5	698	40.0	35.7	40.0	35.7	35.7	33.6	35.7	33.6	35.7	33.6	31.5	35.7	33.6	33.6	31.5
		cpu	5	209	28.0	28.2	27.6	27.0	25.2	27.5	23.4	22.2	24.4	29.2	21.6	20.7	25.9	29.2	21.0
	house	8	506	142.6	145.9	135.5	134.9	132.8	146.6	114.6	112.4	131.1	150.1	110.0	107.0	135.4	134.5	103.8	106.7

found before and after carrying out the domain partition phase at the end of the method, we register an increase in fitting error of up to 4 times for both STD-CR and STD-CL. This suggests the lack of a strong correlation, in both algorithms, between the quality of the solutions found before and after constructing the final domain partition. Also note that, with the adoption of our criterion for the identification of critical points, the `Point Partition` step becomes more time consuming. Indeed, our experiments show that the average number of iterations carried out in the time limit by STD-CR with respect to those for STD-CL can be up to 40% smaller (as observed for the large instances with $k = 5$). Therefore, investing more computing time in a more refined criterion for the `Point Partition` step turns out to be not effective for a method (such as STD-CR) which only considers the domain partition aspect in a second phase.

7 Concluding remarks

We have addressed the k -PAMF-PLS problem of fitting a piecewise affine model with a pairwise linearly separable domain partition to a set of data points. We have proposed an MILP formulation to solve the problem to optimality, strengthened via symmetry breaking constraints. To solve larger instances, we have developed a four-step heuristic algorithm which simultaneously deals with the various aspects of the problem. It is based on two key ideas: a criterion for the identification of a set of critical points to be reassigned and the introduction of a domain partitioning step at each iteration of the method.

Computational experiments on a set of structured randomly generated and real-world instances show that with our MILP formulation with symmetry breaking constraints we can solve to optimality small-size instances, while our four-step heuristic provides close-to-optimal solutions for small-size instances and allows to tackle instances of much larger size. The results not only indicate the high quality of the solutions found by 4S-CR when compared to those obtained with either an exact method or a standard two-phase heuristic algorithm, but they also highlight the relevance of the different features of 4S-CR, which must be adopted in a joint way to yield higher quality solutions to k -PAMF-PLS.

References

- [AC13] E. Amaldi and S. Coniglio. A distance-based point-reassignment heuristic for the k -hyperplane clustering problem. *European Journal of Operational Research*, 227(1):22–29, 2013.
- [ACT11] E. Amaldi, S. Coniglio, and L. Taccari. Formulations and heuristics for the k -piecewise affine model fitting problem. In *Proc. of 10th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW), Frascati (Rome), Italy*, pages 48–51, 2011.
- [ACT12] E. Amaldi, S. Coniglio, and L. Taccari. k -Piecewise Affine Model Fitting: heuristics based on multiway linear classification. In *Proc. of 11th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW)*,

- Munich, Germany, pages 16–19. Universität der Bundeswehr München, 2012.
- [ADC13] E. Amaldi, K. Dhyani, and A. Ceselli. Column generation for the minimum hyperplanes clustering problem. *INFORMS Journal on Computing*, 25(3):446–460, 2013.
- [AM02] E. Amaldi and M. Mattavelli. The MIN PFS problem and piecewise linear model estimation. *Discrete Applied Mathematics*, 118(1-2):115–143, 2002.
- [BGPV03] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A greedy approach to identification of piecewise affine models. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 97–112. Springer Berlin Heidelberg, 2003.
- [BM94] K. Bennet and O. Mangasarian. Multicategory discrimination via linear programming. *Optimization Methods and Software*, 3:27–39, 1994.
- [BM00] P. Bradely and O. Mangasarian. k -plane clustering. *Journal of Global Optimization*, 16:23–32, 2000.
- [BS07] D. Bertsimas and R. Shioda. Classification and regression via integer optimization. *Operations Research*, 55:252–271, 2007.
- [Con11] S. Coniglio. The impact of the norm on the k -Hyperplane Clustering problem: relaxations, restrictions, approximation factors, and exact formulations. In *Proc. of 10th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW), Frascati (Rome), Italy*, pages 118–121, 2011.
- [Con15] S. Coniglio. On the optimization of vector norms and the k -hyperplane clustering problem: tightened exact and approximated formulations within an approximation factor. Technical report, Lehrstuhl II für Mathematik, RWTH Aachen University, 2015.
- [DF66] R. Duda and H. Fossum. Pattern classification by iteratively determined linear and piecewise linear discriminant functions. *IEEE Transactions on Electronic Computers*, 15:220–232, 1966.
- [FA13] A. Frank and A. Asuncion. UCI machine learning repository, 2013. <http://archive.ics.uci.edu/ml>.
- [FTMLM03] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, 2003.
- [KP08] V. Kaibel and M. E. Pfetsch. Packing and partitioning orbitopes. *Mathematical Programming, Series A*, 114:1–36, 2008.
- [Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability, Los Angeles, California*, volume 1, pages 281–297. California University Press, 1967.
- [Mar10] F. Margot. Symmetry in integer linear programming. In M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 647–686. Springer Berlin Heidelberg, 2010.
- [MB09] A. Magnani and S. Boyd. Convex piecewise-linear fitting. *Optimization and Engineering*, 10:1–17, 2009.
- [MDZ01] I. Méndez-Díaz and P. Zabala. A polyhedral approach for graph coloring. *Electronic Notes in Discrete Mathematics*, 7:178–181, 2001.
- [MDZ06] I. Méndez-Díaz and P. Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154(5):826–847, 2006.

- [MRT05] O. Mangasarian, J. Rosen, and M. Thompson. Global minimization via piecewise-linear underestimation. *Journal of Global Optimization*, 32(1):1–9, 2005.
- [RBL04] J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40:37–50, 2004.
- [TPSM06] M. Tabatabaei-Pour, K. Salahshoor, and B. Moshiri. A modified k-plane clustering algorithm for identification of hybrid systems. In *Proc. of 6th World Congress on Intelligent Control and Automation (WCICA)*, volume 1, pages 1333–1337. IEEE, 2006.
- [TV12] A. Toriello and J.P. Vielma. Fitting piecewise linear continuous functions. *European Journal of Operational Research*, 219(1):86–95, 2012.
- [Vap96] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1996.