Discrete optimization methods to fit piecewise affine models to data points

E. Amaldi[a], S. Coniglio[b,1], L. Taccari[a]

[a] *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy*

[b] *Department of Mathematical Sciences, University of Southampton, University Road, Southampton, SO17 1BJ, UK*

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 Piecewise Linear Separability problem* (k-PAMF-PLS) where, given a set of m points $\{\mathbf{a}_1, \ldots, \mathbf{a}_m\} \subset \mathbb{R}^n$ and the corresponding observations $\{b_1, \ldots, b_m\} \subset \mathbb{R}$, we have to partition the domain $\mathbb{R}^n$ into k piecewise linearly (or affinely) 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 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 multi-category linear classification. Differently from traditional approaches proposed in the literature for similar fitting problems, in both our exact and heuristic 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

Email addresses: edoardo.amaldi@polimi.it (E. Amaldi), s.coniglio@soton.ac.uk (S. Coniglio), leonardo.taccari@polimi.it (L. Taccari)

[1] 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 and with Lehrstuhl II für Mathematik, RWTH Aachen University, supported. While with the latter, he was supported by the German Federal Ministry of Education and Research (BMBF), grant 05M13PAA, and Federal Ministry for Economic Affairs and Energy (BMWi), grant 03ET7528B.

is quite substantial when compared to standard variants not including them. We conclude with an application to the identification of dynamical piecewise affine systems for which we obtain promising results of comparable quality with those achieved with state-of-the-art methods from the literature on benchmark data sets.

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 statistics (see, e.g., the recent work in [BM14]), to data mining (see, e.g., [AM02, BS07]), and to system identification (see, for instance, [FTMLM03, BGPV05, 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 $\mathcal{A} = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m\} \subset \mathbb{R}^n$, with index set $I = \{1, \ldots, m\}$, with the corresponding observations $\{b_1, \ldots, b_m\} \subset \mathbb{R}$ and a positive integer k, the general problem of fitting a piecewise affine model to the data points $\{(\boldsymbol{a}_1, b_1), \ldots, (\boldsymbol{a}_m, b_m)\}$ consists in partitioning the domain $\mathbb{R}^n$ into k continuous *subdomains* $D_1, \ldots, D_k$, with index set $J = \{1, \ldots, k\}$, and in determining, for each subdomain D_j, an *affine submodel* (an affine function) $f_j : D_j \to \mathbb{R}$, so as to minimize a measure of the total fitting error. Adopting the notation[2] $f_j(\boldsymbol{x}) = \boldsymbol{w}^j \boldsymbol{x} - w_0^j$ with coefficients $(\boldsymbol{w}^j, w_0^j) \in \mathbb{R}^{n+1}$, the j-th affine submodel corresponds to the hyperplane $H_j = \{(\boldsymbol{x}, f_j(\boldsymbol{x})) \in \mathbb{R}^{n+1} : f_j(\boldsymbol{x}) = \boldsymbol{w}^j \boldsymbol{x} - w_0^j\}$ where $\boldsymbol{x} \in D_j$. 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 $f_{j(i)}(\boldsymbol{a}_i)$ 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 $\boldsymbol{a}_i$.

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.

[2] For greater readability, transposition symbols will be omitted throughout the paper. If one wanted to specify the row and column nature of the vectors that we use, $\boldsymbol{a}_i$ would be a column vector, while $\boldsymbol{w}^j$ and, in the following, $\boldsymbol{y}^j$, would be row vectors.

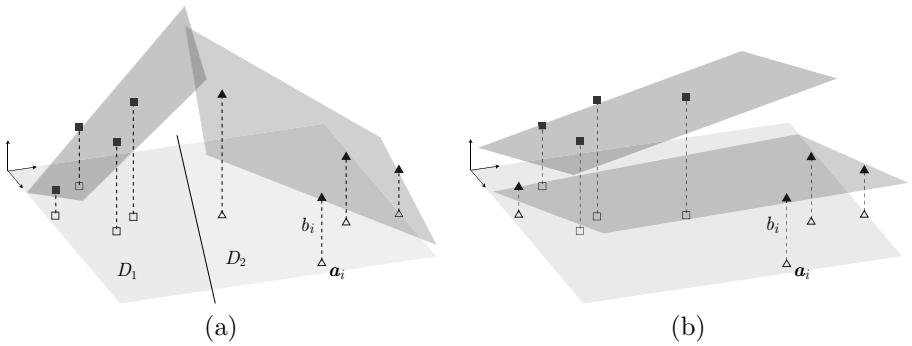


Figure 1: (a) A piecewise affine model with $k = 2$, fitting the eight data points $\mathcal{A} = \{\boldsymbol{a}_i\}_{i \in I}$ and their observations $\{b_i\}_{i \in I}$ with two submodels (in dark gray). The points $(\boldsymbol{a}_i, b_i)$ assigned to each submodel are indicated by ■ and ▲. The model adopts a linearly separable partition of the domain $\mathbb{R}^2$ (represented in light gray). (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 $\boldsymbol{a}_i$ assigned to the first submodel (indicated by □) cannot be linearly separated from those assigned to the second submodel (indicated by △). 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.

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 piecewise linearly[3] separable subdomains. We refer to it as to the *k-Piecewise Affine Model Fitting with Piecewise 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 $\mathbb{R}^{n+1}$ which minimize the sum, over all the data points $\{(\boldsymbol{a}_1, b_1), \ldots, (\boldsymbol{a}_m, b_m)\}$, of the ℓ_2 distance from $(\boldsymbol{a}_i, b_i)$ to the hyperplane the point is assigned to. See [BM00, AC13, Con11, Con15] for some recent work on the problem and [ADC13] for the problem variant aiming at minimizing the number of hyperplanes needed to fit all the points within a prescribed tolerance $\varepsilon > 0$.

It is nevertheless crucial to note that, differently from many of the approaches in the 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. A naive application of algorithms designed for k-HC to tackle k-PAMF-PLS can indeed lead to solutions with a large fitting error, as a consequence of the domain partitioning

[3] Although this kind of separation employs piecewise affine functions, it is usually referred to, in the literature, as *piecewise linear*. We will do so also here for uniformity with previous work.

aspect being entirely neglected. 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 two are consistent. For this reason, most of the algorithms in the literature incorporate techniques to enforce the piecewise linear separability of the domain, typically after first solving k-HC (or one of its variants). In this work, we propose exact methods for k-PAMF-PLS based on mixed-integer linear programming as well as heuristic algorithms which consider both aspects of the problem simultaneously, rather than deferring the domain partitioning aspect to a later stage of the solution process.

The paper is organized as follows. After summarizing previous and related works 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 using it for 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. In Section 7, we consider the application of k-PAMF-PLS to problems in the area of dynamical system identification and compare the obtained results with those provided by state-of-the-art methods. Section 8 contains some concluding remarks. Portions of this work appeared, in a preliminary stage, in [ACT11, ACT12].

2. Previous and related work

Recently, there has been a growing interest in mixed-integer programming and discrete optimization approaches to a wide range of problems in the areas of data mining and statistics, see, e.g., [IR03, CSK06, BS07, CBR12, BM14, MT15]. As to the problem of fitting a piecewise affine model to data points, many variants have been considered in the literature. 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 $D_j = \{\boldsymbol{x} \in \mathbb{R}^n \ : \ f_j(\boldsymbol{x}) \geq f_{j'}(\boldsymbol{x}) \ \forall j' \in J\}$ (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. In the case of hinging hyperplane models, see [Bre93] or [RBL04] and the references therein, the domain partition does not need to be explicitly derived due to the special structure of this type of piecewise affine models.

In more general versions of the problem, 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, most of the available methods are *two-phase* in nature, in that they 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 $\{(\boldsymbol{a}_1, b_1), \ldots, (\boldsymbol{a}_m, b_m)\} \subset \mathbb{R}^{n+1}$, whereas the classification problem considers the original points $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m\} \subset \mathbb{R}^n$ 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. Although such two-phase approaches turn out to perform satisfactorily in several of the control applications that are cited in this section, in the general case they may lead to solutions with a large fitting error (due to deferring the domain partition to the end), as also illustrated in Section 6 with our computational experiments.

Among the two-phase methods available in the literature, we first mention [BGPV05], which includes a postprocessing phase aiming at enforcing piecewise linear separability. In its 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], resorting to SVM for the classification phase. 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 [Vap96] and [BM94], respectively. The method proposed in [FTMLM03] associates, to each data point $\boldsymbol{a}_i$, a hyperplane computed by solving a least square problem over the c closest points, and then clusters them into k groups via a weighted version of the k-means algorithm[4] [Mac67]. To avoid grouping together hyperplanes corresponding to far away points and, thus, partially accounting for the domain partitioning aspect, clustering is carried out in a $2n + 1$-dimensional feature space which includes the hyperplane parameters and the centroid of the c points corresponding to the neighbourhood. Finally, SVM is applied to derive the domain partition *a posteriori*.

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

Note that, in several of the cited articles, the authors also propose heuristics to select a suitable number k of submodels which, in our work, we assume to be given. In Subsection 6.6, we will give an idea on how to estimate a suitable value for k using a cross-validation technique.

As already mentioned in the previous section, these two-phase 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 $\boldsymbol{a}_i$ 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.

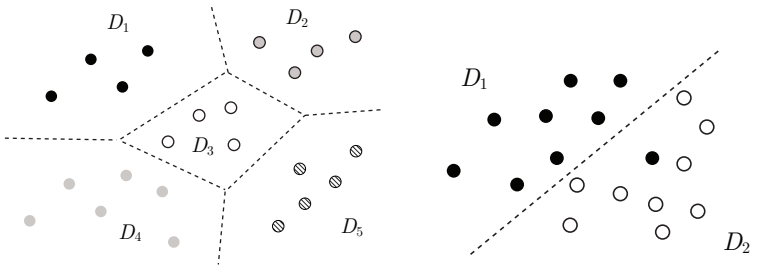
A large body of work on problems related to piecewise affine model fitting has been carried out in the context of dynamical system identification for the case of so-called *hybrid systems*. Many approaches typically encompass two-phase heuristic methods such as those in [BGPV05, TPSM06, FTMLM03], which we have already mentioned. A few mixed-integer programming formulations have also been proposed for similar fitting problems. In the case of switched system identification, such as in [PJFTV07, MK05], the focus is on finding an optimal assignment of the points to the submodels without determining the domain partition. In the case of hinging hyperplane models [Bre93, RBL04], no explicit domain partition needs to be derived. Another line of research is that pursued in, among others, [OL13, OSLC12], where the authors consider sparse optimization approaches aiming at the identification of piecewise affine models with a trade-off between number of pieces and fitting error. For further details on similar applications and methodologies, we refer the reader to the surveys [PJFTV07, GPV12], as well as to Section 7.

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 *piecewise linear separability*, which forms the basis of *multicategory linear classification* [BM94]. In the following, we briefly recall some key features of this well-known classification paradigm, which we will then leverage in the remainder of the paper.

3.1. Piecewise 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 $D_1, \dots, D_k$ which, as shown in [DF66] (also see [BM94]) can be defined by introducing, for each group of points A_j with $j \in J$, a vector of parameters $(\boldsymbol{y}^j, y_0^j) \in \mathbb{R}^{n+1}$ such that a point $\boldsymbol{a}_i \in A_j$ belongs to the subdomain D_j if and only if, for every $j' \neq j$, we have $(\boldsymbol{y}^j - \boldsymbol{y}^{j'})\boldsymbol{a}_i - (y_0^j - y_0^{j'}) > 0$. This equivalently amounts to imposing that, 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 $H_{j_1 j_2} = \{\boldsymbol{x} \in \mathbb{R}^n : (\boldsymbol{y}^{j_1} - \boldsymbol{y}^{j_2})\boldsymbol{x} = y_0^{j_1} - y_0^{j_2}\}$ with coefficients $(\boldsymbol{y}^{j_1} - \boldsymbol{y}^{j_2}, y_0^{j_1} - y_0^{j_2}) \in \mathbb{R}^{n+1}$, see



(a) (b)

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

Figure 2 (a) for an illustration. It follows that, for any $j \in J$, the domain D_j is defined as:

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

If the group of points $A_1, \ldots, A_k$ are not piecewise linearly separable, there exists at least a point $\boldsymbol{a}_i$ and a pair j_1, j_2 for which the inequality $(\boldsymbol{y}^{j_1} - \boldsymbol{y}^{j_2})\boldsymbol{a}_i - (y_0^{j_1} - y_0^{j_2}) > 0$ is violated for any choice of the vectors of parameters $(\boldsymbol{y}^j, y_0^j)$ with $j \in J$. 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 $\boldsymbol{a}_i \in A_{j(i)}$, where $j(i)$ is the index of the group the point belongs to, the misclassification error is defined as:

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

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

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

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

3.2. k-Piecewise affine model fitting problem with piecewise linear separability

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

> k-PAMF-PLS: Given a set of m points $\mathcal{A} = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m\} \subset \mathbb{R}^n$ with the corresponding observations $\{b_1, \ldots, b_m\} \subset \mathbb{R}$ and a positive integer k:
>
> i) partition $\mathcal{A}$ into k subsets $A_1, \ldots, A_k$ which are *piecewise linearly separable* via a domain partition $D_1, \ldots, D_k$ of $\mathbb{R}^n$ induced, according to Equation (1), by a set of vectors $(\boldsymbol{y}^j, y_0^j) \in \mathbb{R}^{n+1}$, for $j \in J$,
>
> ii) determine, for each subdomain D_j, an affine function $f_j : D_j \to \mathbb{R}$ where $f_j(\boldsymbol{x}) = \boldsymbol{w}^j \boldsymbol{x} - w_0^j$ with parameters $(\boldsymbol{w}^j, w_0^j) \in \mathbb{R}^{n+1}$,
>
> so as to minimize the linear error function $\sum_{i=1}^{m} \left| b_i - \left(\boldsymbol{w}^{j(i)} \boldsymbol{a}_i - w_0^{j(i)} \right) \right|$, where $j(i) \in J$ is the index for which $\boldsymbol{a}_i \in A_{j(i)} \subset D_{j(i)}$.

It is worth emphasizing that, in this problem, both the domain partition and the fitting hyperplanes have to be determined jointly.

4. Strengthened mixed-integer linear programming formulation

In this section, we propose an MILP formulation to solve k-PAMF-PLS to optimality via a branch-and-cut method, 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, as we will explain in the following.

Our MILP is derived by combining an adapted hyperplane clustering formulation with a multicategory linear classification one. The former allows us to partition the data points into k subsets $A_1, \ldots, A_k$ and to determine an affine submodel for each of them (see, e.g., [AC13] or [MK05, PJFTV07] for closely related formulations for the identification of switched systems). The latter guarantees a piecewise linearly separable domain partition $D_1, \ldots D_k$, consistent with the k subsets $A_1, \ldots, A_k$ (see the previous section and [BM94]). Consistently with the definition of the problem, in the resulting formulation, the parameters of the fitting affine submodels and the piecewise linear domain partition are determined *simultaneously*.

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 $\boldsymbol{a}_i$ is contained in the subset A_j and 0 otherwise. Let z_i be the fitting error of point $\boldsymbol{a}_i \in \mathcal{A}$ for each $i \in I$, $(\boldsymbol{w}^j, w_0^j) \in \mathbb{R}^{n+1}$ be the parameters of the submodel of index $j \in J$, and $(\boldsymbol{y}^j, y_0^j) \in \mathbb{R}^{n+1}$, with $j \in J$, be the parameters used to enforce piecewise linear separability. Let also M_1 and

M_2 be large enough constants (whose value is discussed below). The formulation is as follows:

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

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

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

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

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

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

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

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

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

Constraints (5) guarantee that each point $\boldsymbol{a}_i \in \mathcal{A}$ be assigned to exactly one submodel. Constraints (6) and (7) impose that $z_i = |b_i - \boldsymbol{w}^{j(i)}\boldsymbol{a}_i + w_0^{j(i)}|$. This is because, together, they imply that $z_i \geq |b_i - \boldsymbol{w}^j \boldsymbol{a}_i + w_0^j| - M_1(1 - x_{ij})$. When $x_{ij} = 1$, this amounts to imposing $z_i \geq |b_i - \boldsymbol{w}^j \boldsymbol{a}_i + w_0^j|$ (which will be tight in any optimal solution due to the objective function direction), whereas it becomes 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 $(\boldsymbol{y}^{j_1} - \boldsymbol{y}^{j_2})\boldsymbol{a}_i - (y_0^{j_1} - y_0^{j_2}) \geq 1$, 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 piecewise 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 $(\boldsymbol{a}, b) = (\boldsymbol{e}, -1) \in \mathbb{R}^{n+1}$, where $\boldsymbol{e}$ is the all-one vector, and the affine function $f = \boldsymbol{w}\boldsymbol{a} - 1$ is equal to $\|\boldsymbol{w}\|_1$ (the ℓ_1 norm of $\boldsymbol{w}$) and, thus, it is unbounded and arbitrarily large for an arbitrarily large $\|\boldsymbol{w}\|_1$. Let $j(i) \in J$ such that $x_{ij(i)} = 1$. The introduction of a finite M_1 corresponds to letting:

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

rather than $z_i = |b_i - \boldsymbol{w}^{j(i)}\boldsymbol{a}_i + w_0^{j(i)}|$. Therefore, a finite M_1 introduces a

penalization term into the objective function equal to:

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

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 with Constraints (8) due to the presence of the parameter M_2. Indeed, for any finite M_2 and for $x_{ij_1} = 0$, each of such constraints implies $(\boldsymbol{y}^{j_1} - \boldsymbol{y}^{j_2})\boldsymbol{a}_i - (y_0^{j_1} - y_0^{j_2}) \geq 1 - M_2$. Hence, Constraints (8) impose that the *linear distance*[5] $-(\boldsymbol{y}^{j_1} - \boldsymbol{y}^{j_2})\boldsymbol{a}_i + (y_0^{j_1} - y_0^{j_2})$ (notice that the signs are consistent with Equation (3)) between each point $\boldsymbol{a}_i$ and the hyperplane separating *any* pair of subdomains D_{j_1}, D_{j_2} be smaller than $M_2 - 1$ even if $\boldsymbol{a}_i$ is not contained in either of the subdomains, i.e., even if $\boldsymbol{a}_i \notin A_{j_1}$ and $\boldsymbol{a}_i \notin A_{j_2}$.

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.

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 $1, \ldots, k$ 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 $M = M_1 = M_2 \geq \frac{k}{k-1} \max\{1, |b_1|, |b_2|, \ldots, |b_m|\}$, 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 $i \in I$,

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

$(\boldsymbol{w}^j, w_0^j) = (\mathbf{0}, 0)$ and $(\boldsymbol{y}^j, y_0^j) = (\mathbf{0}, 0)$ 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 nonincreasing order. According to [KP08], we call the convex hull of such lexicographically sorted matrices $X \in \{0,1\}^{m \times k}$ *orbitope*.

4.3. Symmetry breaking constraints from the partitioning orbitope

Since, in our case, X is a *partitioning matrix* (a matrix $X \in \{0,1\}^{m \times k}$ 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 $B_{(i,j)} = \{(i,j), (i,j+1), \ldots, (i, \min\{i,k\})\}$, and $\mathrm{col}_{(i,j)}$ a *column*, defined as $\mathrm{col}_{(i,j)} = \{(j,j), (j+1,j), ..., (i,j)\}$. Intuitively, a *shifted column* $S_{(i,j)}$ is a subset of the indices of X obtained by *shifting* some of the indices in $\mathrm{col}_{(i,j)}$ diagonally towards the upper-left portion of X. More formally, for a pair of indices (i,j), a shifted column $S_{(i,j)}$ is a collection of indices in $I \times J$ such that $S_{(i,j)} = \{(c_1, c_1), (c_2+1, c_2), \ldots, (c_\eta + \eta - 1, c_\eta)\}$ with $\eta = i - j + 1$ and $1 \leq c_1 \leq c_2 \cdots \leq c_\eta \leq j$. For $c_1 = \cdots = c_\eta = j$, we obtain the original (nonshifted) column $col_{(i,j)}$. For an illustration, see Figure 3. For two 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:

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

$$\sum_{(i',j') \in B_{(i,j)}} x_{i'j'} - \sum_{(i',j') \in S_{(i-1,j-1)}} x_{i'j'} \leq 0 \quad \forall B_{(i,j)}, S_{(i-1,j-1)} : i,j \in J, i,j \geq 2 \quad (16)$$

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

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

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].

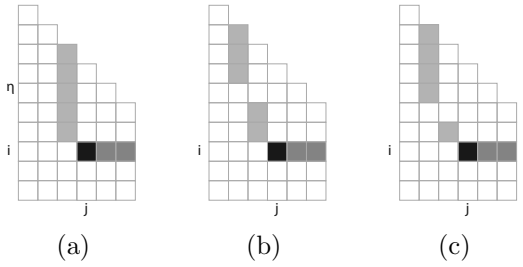


Figure 3: (a) The bar $B_{(i,j)}$ (in black and dark gray) and the column $\mathrm{col}_{(i-1,j-1)}$ (in light gray). (b) and (c) Two shifted columns (in light gray) obtained by shifting $\mathrm{col}_{(i-1,j-1)}$. Note how the shifting operation introduces empty rows.

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 for the solution of 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, at each iteration, all the three aspects of the problem, namely, affine submodel fitting, point partition, and domain partition. At each iteration, the heuristic alternately and coordinately carries out a sequence of four steps, the last two of which guarantee that the current solution always admits a piecewise linear domain partition. Before describing the details of the four steps, it is worth pointing out that the structure of our algorithm differs from other heuristics (such as [BGPV05, FTMLM03]) where the domain partitioning aspect is considered directly just once, in the very last iteration. Moreover, our method does not amount to just running an off-the-shelf clustering algorithm (tackling the first and second aspects of the problem) until convergence to a local minimum, deriving a domain partition, and then feeding a refined solution again to the clustering method as a warm start.

We start from a feasible solution composed of a point partition $A_1, \ldots, A_k$, a domain partition $D_1, \ldots, D_k$ (induced by the parameters $(\boldsymbol{y}^j, y_0^j)$ for $j \in J$), and a set of affine submodels of parameters $(\boldsymbol{w}^j, w_0^j)$, for $j \in J$. At each iteration, 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 $A_1, \ldots, A_k$ of $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$, determine, for each $j \in J$, an affine submodel with param-

eters $(\boldsymbol{w}^j, w_0^j)$ which minimizes the linear fitting error over all the data points $\{(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_m, b_m)\} \subset \mathbb{R}^{n+1}$. As we shall see, this is carried out by solving a single linear program.

ii) `Point Partition`: Given the current set of affine submodels $f_j : D_j \to \mathbb{R}$ with $f_j(\boldsymbol{x}) = \boldsymbol{w}^j \boldsymbol{x} - w_0^j$ 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 $A_1, \ldots, A_k$ of $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$, a multicategory linear classification problem is solved via linear programming to either find a piecewise linearly separable domain partition $D_1, \ldots, D_k$ 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 $D_1, \ldots, D_k$.

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 $\boldsymbol{a}_i$ (if any) belonging to A_j (i.e., for any $\boldsymbol{a}_i \in \mathcal{A}$ where $\boldsymbol{a}_i \in A_j$ and $\boldsymbol{a}_i \in D_{j'}$, for some $j, j' \in J$ such that $j \neq j'$), $\boldsymbol{a}_i$ is reassigned to the subset $A_{j'}$ associated with $D_{j'}$.

We now describe the four steps in greater detail.

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

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

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

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

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

$$d_i \geq 0 \qquad \forall i \in I, \tag{23}$$

where $j(i)$ denotes the submodel to which the point $\boldsymbol{a}_i$ is currently assigned. Note that this linear program, which is standard for solving regression problems in ℓ_1-norm, 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 $\boldsymbol{a}_i$ to a subset A_j whose corresponding submodel yields the smallest fitting error. As shown in Figure 4 (a), and as can be confirmed computationally,

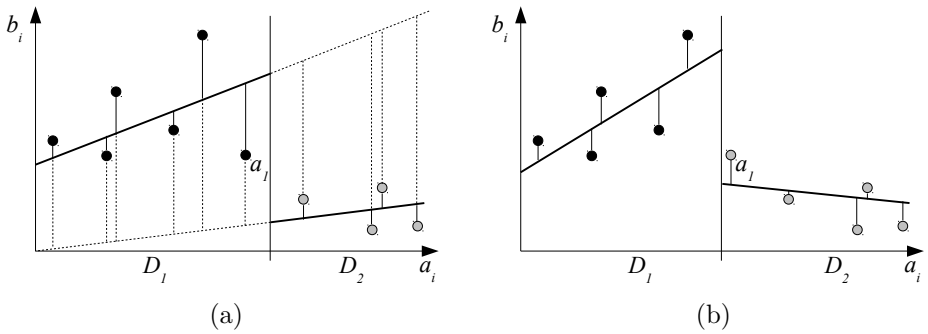


Figure 4: (a) A solution corresponding to a local minimum for an algorithm where each point is reassigned to the submodel yielding the smallest fitting error. (b) An improved solution that can be obtained by reassigning the point $\boldsymbol{a}_1$ in (a) which, according to our criterion, achieves the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

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 one employed in the Distance-Based Point Reassignment heuristic (DBPR) proposed in [AC13] for the k-hyperplane clustering problem.

The idea of the criterion 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 can 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 $\boldsymbol{a}_i \in A_j$ 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 $\boldsymbol{a}_i$ but which is also different from $j(i)$. Formally, for each $j \in J$, the criterion ranks each point $\boldsymbol{a}_i \in A_j$ with respect to the quantity:

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

The `Point Partition` step relies on a *control parameter* $\alpha \in [0, 1)$. Given a current solution characterized by a point partition $A_1, \ldots, A_k$ 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).

Since the reassignment of critical points, as identified by our criterion, in-

troduces a high variability in the sequence of solutions that are generated, the search process is stabilized by decreasing α to 0 over the iterations, thus progressively shifting from introducing (for a large α) substantial changes in the current solution to performing fine polishing operations (for a smaller α).

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 and 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 piecewise linear separation of the sets $A_1, \ldots, A_k$ which minimizes the total misclassification error. This is achieved by solving a Multicategory Linear Classification problem via the linear program proposed in [BM94]:

$$\min \sum_{i=1}^{m} e_i \tag{25}$$

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

$$e_i \geq 0 \qquad \forall i \in I \tag{27}$$

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

where $j(i)$ denotes the submodel to which the point $\boldsymbol{a}_i$ is currently assigned, and e_i represents the misclassification error of point $\boldsymbol{a}_i$, for all $i \in I$. If this subproblem admits an optimal solution with total misclassification error equal to 0, then the k subsets are piecewise linearly separable. Otherwise, the current solution contains at least a misclassified point $\boldsymbol{a}_i \in A_{j(i)}$ with $\boldsymbol{a}_i \in D_j$ for $j \in J$ with $j \neq j(i)$. 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 $(\boldsymbol{a}_i, b_i)$ to a submodel yielding the smallest fitting error (ties are broken arbitrarily). The four steps are then 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, thus avoiding infeasible solutions which are good from a submodel fitting point of view but do not admit a piecewise

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 and some variants on larger instances, and assess the impact of the main components of 4S-CR on the overall quality of the solutions it finds.

6.1. Experimental setup

Our 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)[6]. They are constructed by randomly sam-

[6] 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.

pling the data points $\boldsymbol{a}_i$ 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 $D_1, \ldots, D_k$ by solving a multicategory linear classification problem on k randomly chosen representative points in $\mathbb{R}^n$. Then, we randomly choose the submodel parameters $(\boldsymbol{w}^j, w_0^j)$ for all $j \in J$ and sample, uniformly at random, the m points $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m\} \in \mathbb{R}^n$. For each sampled point $\boldsymbol{a}_i$, 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 $\boldsymbol{a}_i$, i.e., $\boldsymbol{w}^{j(i)}\boldsymbol{a}_i - w_0^{j(i)}$. 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 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 its deterministic mode on a single thread with default settings. In all the cases, we set $M = M_1 = M_2 = 1000$.

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

The results are reported in Table 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.

It has recently been argued that, in a number of data mining problems, feeding a heuristically generated solution to the MILP solver as a warm-start might give a significant performance boost [BM14]. Unfortunately, when experimenting with this approach on k-PAMF-PLS, we only observed a negligible impact. This is arguably due to the fact that, for hard instances, most of the difficulty in proving optimality lies in improving the dual bound, and running a heuristic (to obtain a primal bound) beforehand only incurs a overhead. For this reason, we have not used this approach in our experiments.

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 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 tested 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 $m = 100,\ n = 5$ (on which we register 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 that 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. 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 and some variants

We now present and discuss the results obtained with 4S-CR and some variants 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`

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.

		$k=2$			$k=3$			$k=4$			$k=5$		
			Objective			Objective			Objective			Objective	
n	m	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**

`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 could 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 $(\boldsymbol{a}_i, b_i)$ is (re)assigned to a submodel yielding the smallest fitting error. This is in line with the reassignment step of many popular clustering heuristics, as reported in Sections 2 and 5. 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), as well as to compare 4S-CR to classical algorithms, 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, thus tackling the problem in two distinct, successive phases: a clustering phase and a classification phase. In the algorithm, which we consider in two versions, we iteratively alternate between the `Submodel Fitting` and `Point Partition` steps. In the `Point Partition` 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 piecewise linearly separable domain partition with minimum misclassification error is derived by solving only once a multicategory 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, we consider it as the baseline method and compare the other methods (4S-CR, 4S-CL, and STD-CR) to it.

For the sake of comparison, we also consider an extension of this standard method, STD-CL, obtained after introducing an extra third phase, based on the refinement point-reassignment criterion proposed in [BGPV05], taking place between the clustering and classification phases. We refer to this algorithm as STD-CL-B. This refinement criterion is based on the identification of so-called *undecidable* points, namely, data points which lie within a maximum given error $\delta \geq 0$ w.r.t., at least, two affine submodels. According to this criterion, each

undecidable data point $\boldsymbol{a}_i$ is iteratively reassigned to the submodel to which the majority of the c points which are closer, in ℓ_2-norm, to $\boldsymbol{a}_i$ are currently assigned. We remark that [BGPV05] addresses a piecewise-affine model fitting problem where k is minimized, subject to a maximum fitting error of δ. To adapt the criterion to k-PAMF-PLS, we set, in our experiments, δ to the average fitting error between each data point and its current submodel. As suggested in [BGPV05], we select $c = 10$ and carry out the reassignment of undecidable points 5 times, each time followed by a `Submodel Fitting` step[7].

The results for the medium, large, and UCI datasets, obtained within 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% on the medium instances, of 12%, 17%, 17%, and 11% on 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 our criterion based on critical points which 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 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

[7]Experiments with δ equal to the average fitting error scaled by a factor of $\{0.1, 0.25, 0.50, 0.75, 1\}$ revealed that the best results are obtained with a factor of 1. We also experimented with $c = 5, 20, 50$ and carrying out 1, 10, and 20 passes of the criterion. On average, on average, we obtained comparable results.

Table 4: Results obtained with 4S-CR, when compared to STD-CL, STD-CR, 4S-CL, and STD-CL-B 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.

			$k=2$					$k=3$					$k=4$					$k=5$				
	n	m	STD-CL	STD-CR	4S-CL	4S-CR	STD-CL-B	STD-CL	STD-CR	4S-CL	4S-CR	STD-CL-B	STD-CL	STD-CR	4S-CL	4S-CR	STD-CL-B	STD-CL	STD-CR	4S-CL	4S-CR	STD-CL-B
medium	500	2	658.6	863.4	659.7	**592.7**	658.7	660.2	734.7	592.7	**472.9**	660.2	638.3	790.6	485.8	**345.7**	610.8	638.3	784.9	493.1	**472.9**	663.4
	500	2	**406.2**	**406.2**	**406.2**	**406.2**	**406.2**	400.1	400.1	400.1	**274.1**	400.1	416.8	416.8	387.8	**254.0**	416.8	416.8	400.1	383.6	**254.0**	416.8
	500	2	408.5	408.5	**408.4**	**408.4**	408.5	365.6	292.7	394.3	**267.4**	408.6	288.7	**265.4**	325.6	266.7	281.0	330.5	424.4	325.6	**226.3**	329.1
	500	3	392.1	386.7	378.7	**358.2**	391.8	334.0	346.1	310.7	**279.2**	334.6	318.6	319.9	309.9	**286.6**	329.8	345.5	351.0	310.5	**281.3**	288.4
	500	3	518.3	484.1	472.9	**448.5**	506.8	500.3	371.6	300.0	**262.7**	501.3	496.2	511.8	288.2	**257.4**	494.9	491.3	533.3	292.4	**256.8**	408.1
	500	3	699.7	698.6	580.4	**574.6**	750.0	612.2	627.9	508.8	**447.1**	522.9	502.1	548.1	**447.9**	449.3	459.0	532.7	564.3	**411.1**	427.6	503.0
	500	4	421.1	426.6	412.2	**403.7**	424.6	378.0	392.0	334.1	**325.3**	360.3	308.4	325.2	268.8	**260.2**	295.8	328.1	339.1	206.2	**193.0**	234.0
	500	4	669.0	670.0	643.0	**623.8**	668.5	536.6	521.1	505.5	**474.3**	546.2	435.5	492.3	395.7	**375.2**	394.8	423.4	429.2	377.2	**318.8**	339.4
	500	4	642.2	625.2	587.0	**569.3**	641.6	553.9	516.2	528.5	**513.2**	597.5	550.9	580.9	**474.9**	493.1	539.9	566.9	566.9	479.5	**472.4**	534.2
	500	5	531.7	511.6	520.5	**503.9**	531.7	381.8	403.2	370.4	**360.8**	368.5	280.4	348.2	264.0	**242.8**	288.9	277.6	278.7	**230.3**	264.7	288.7
	500	5	373.9	399.0	371.4	**344.9**	384.4	257.3	388.7	261.0	**252.3**	257.3	246.1	273.9	240.7	**230.5**	241.8	232.0	298.9	237.7	**218.8**	233.8
	500	5	635.1	616.0	603.4	**593.7**	626.1	610.4	592.0	536.4	**509.8**	605.9	449.3	616.0	472.5	**421.7**	456.6	441.4	620.6	464.5	489.0	**426.0**
large	1000	2	**654.5**	**654.5**	**654.5**	**654.5**	**654.5**	654.1	654.1	**572.5**	**572.5**	654.1	654.6	670.0	572.5	**557.6**	654.6	565.7	644.5	572.5	**557.8**	565.7
	1000	2	1685.4	1618.5	1452.5	**1241.9**	1685.4	1685.4	916.0	931.0	**790.5**	1685.4	1715.6	1994.0	931.0	**581.6**	1918.1	1208.2	1903.3	931.0	**647.8**	1504.8
	1000	2	1423.0	1042.1	1019.9	**953.2**	1423.0	1102.5	970.6	849.1	**634.3**	1124.7	1190.9	1405.6	809.0	**501.1**	1195.7	953.2	1289.1	752.0	**490.1**	1233.1
	1000	3	1371.2	1127.6	1095.8	**976.8**	1397.5	1182.8	1053.2	743.2	**729.7**	1196.6	780.3	1089.1	732.0	**621.9**	1041.5	1003.7	1132.4	714.2	**605.7**	1075.6
	1000	3	1099.5	1072.9	1023.5	**935.8**	1099.5	969.5	1010.8	953.6	**829.5**	974.8	997.3	1014.7	934.8	**793.6**	1022.0	957.3	1047.9	943.4	**654.5**	1033.6
	1000	3	1545.1	1408.3	1107.6	**1084.2**	1545.1	1217.2	1171.3	853.9	**805.0**	1172.7	1152.5	1209.7	777.9	**536.4**	1145.5	919.5	1258.6	777.9	**616.5**	993.0
	1000	4	520.1	516.2	519.4	**503.9**	520.1	520.1	547.7	481.1	**422.3**	520.1	520.1	557.3	**438.2**	455.0	537.7	503.7	514.3	483.1	**443.1**	536.1
	1000	4	672.8	672.6	666.0	**655.0**	672.8	536.9	544.7	502.9	**480.8**	518.6	500.7	513.8	481.9	**450.9**	484.4	511.2	521.9	478.5	**463.6**	484.3
	1000	4	1154.7	1076.1	951.0	**924.1**	1154.7	963.3	979.4	854.6	**797.7**	984.5	869.5	895.1	705.8	**625.1**	765.3	928.0	950.6	758.5	**710.4**	794.2
	1000	5	1174.6	1165.9	1110.6	**1092.8**	1185.5	1079.3	1061.9	972.2	**941.1**	1073.8	910.1	883.1	861.4	**846.0**	868.6	915.8	1049.4	880.2	**856.1**	793.9
	1000	5	985.3	936.9	878.6	**859.3**	994.5	760.3	912.1	704.9	**685.8**	785.1	671.9	689.2	641.4	**614.7**	654.5	671.2	717.1	656.9	646.8	**631.8**
	1000	5	570.3	568.1	573.2	**565.2**	579.2	487.7	531.5	491.3	**471.5**	488.2	474.0	521.2	469.9	**464.5**	481.6	481.4	486.5	476.2	**451.3**	481.2
UCI	**auto**	3	397	23.0	23.0	22.6	**22.1**	23.8	21.2	23.9	21.0	**20.8**	21.5	21.2	23.9	**19.5**	21.9	21.0	23.3	18.2	**19.1**	21.6
	breast	5	698	40.0	**35.7**	40.0	**35.7**	40.0	35.7	**33.6**	35.7	**33.6**	36.1	35.7	33.6	35.7	**31.5**	35.7	33.6	33.6	**31.5**	35.7
	cpu	5	209	28.0	28.2	27.6	**27.0**	27.9	25.2	27.5	23.4	**22.2**	25.8	24.4	29.2	21.6	**20.7**	26.2	25.9	29.2	21.0	**20.1**	25.5
	house	8	506	142.6	145.9	135.5	**134.9**	142.1	132.8	146.6	114.6	**112.4**	134.1	131.1	150.1	110.0	**107.0**	135.5	135.4	134.5	**103.8**	106.7	135.4

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.

The comparison between 4S-CR and STD-CL-B yields results that are comparable to those between 4S-CR and STD-CL. Indeed, when focusing on STD-CL-B, we register a slight improvement w.r.t. STD-CL only for $k = 5$, equal to 2% on average. For $k = 3, 4$, the results of STD-CL-B and STD-CL are, on average, identical. For $k = 2$, those for STD-CL-B are slightly inferior than those for STD-CL, with the former method providing a fitting error 1% larger. Overall, when considering the whole dataset and all the values of k, STD-CL-B provides an improvement over STD-CL of less that 0.5% on average.

6.6. Generalization for different values of k

While a thorough evaluation of the accuracy and generalization capabilities of the models obtained with k-PAMF-PLS lies outside of the scope of this work, we carry out a cross-validation to estimate the so-called *generalization error*, i.e., the fitting error of the models corresponding to the solutions found with our techniques on unseen data.

We compare the results obtained with k-PAMF-PLS on the UCI instances for increasing values of k. As baseline for the comparisons, we also report the results obtained when fitting a single affine model ($k = 1$) by minimizing the ℓ_1-norm error (this linear program is solved with CPLEX). Since these k-PAMF-PLS instances are larger than those we tackled with our exact formulation (see Tables 1,2 and Subsection 6.3) and solving them with our MILP formulation would require too much computing time, for $k \geq 2$ we take the best solutions found via our heuristic 4S-CR in 900 seconds.

Each data set is randomly split into two parts, with 75% of the data used for training and the remaining 25% retained for validation. The training set is used to compute an optimal k-PAMF-PLS model for each method, whereas the remaining points in the validation set are only used *a posteriori* as unseen data to evaluate the generalization error.

We construct 5 training/validation splits by sampling the data points at random and solve k-PAMF-PLS for each split. In Table 5, we report, for both the training set and the validation set, the ℓ_1-norm fitting error divided by the number of data points (the so-called *mean absolute error*), averaged over the 5 training/validation splits. The best result obtained in the validation phase is highlighted in boldface.

Table 5: Mean absolute error of k-PAMF-PLS solutions on the of UCI dataset for $k = 1, 2, 3, 4, 5$. Five data splits are considered to generate training and validation sets containing 75% and 25% of the data points. The results are reported on average for the different values of k, with the best error on the training set highlighted in boldface for each instance.

k	1		2		3		4		5	
	Train	Valid	Train	Valid	Train	Valid	Train	Valid	Train	Valid
auto	0.073	0.076	0.054	**0.060**	0.051	0.062	0.048	0.066	0.046	0.073
breast	0.263	0.285	0.052	**0.070**	0.047	0.074	0.046	0.077	0.044	0.089
cpus	0.186	0.243	0.117	0.219	0.096	**0.210**	0.086	0.213	0.081	0.244
hous	0.345	0.378	0.257	0.324	0.219	**0.292**	0.208	0.360	0.205	0.434

We observe that, as expected, the fitting error over the training sets decreases monotonically for larger values of k. As to the generalization error, computed, by definition, on the validation set, it achieves the smallest error for low values of k, between 2 and 3, while, contrary to the training case, it increases for larger values of $k = 4, 5$. This suggests an overfitting phenomenon where models with a small k allow for a better generalization error on the unseen data due to their low complexity (the so-called *Occam's razor* principle). Indeed, a closer look at the results reveals that, by employing too many affine submodels, we may obtain solutions where some of the hyperplanes are used to fit only a small subset of points with a close to zero fitting error (which is always possible if the number of assigned points is not larger than the dimension of the feature space). Such solutions, as it is intuitive to see, could generalize quite badly.

7. Application to the identification of piecewise affine dynamical systems

Let us consider an application of k-PAMF-PLS to the identification of hybrid dynamical systems. A dynamical system is said to be *hybrid* if it exhibits a combination of a continuous and discrete behavior. Such models are of use in a number of practical cases to approximate the behavior of continuous, but nonlinear, systems. Adopting the standard notation, a piecewise affine system can be formally described by an equation of the form:

$$y_t = g(\boldsymbol{\varphi}_t) + e_t,$$

where y_t is the output of the system at time t, $\boldsymbol{\varphi}_t$ is the regression vector, e_t is white noise, and g is a piecewise affine function. In *piecewise affine autoregressive exogenous* models, the vector $\boldsymbol{\varphi}_t$ consists of a collection of previous inputs and outputs, namely:

$$\boldsymbol{\varphi}_t = [y_{t-1} \dots y_{t-n_y} u_{t-1} \dots u_{t-n_u}],$$

where n_y and n_u define the orders of the model, and u_t is the input at time t.

Given an input vector $\boldsymbol{u}$ and an output vector $\boldsymbol{y}$, the problem of identifying a piecewise affine model that best reproduces the dynamics of the underlying

process has been extensively studied in the system identification literature. We refer the reader to the surveys [PJFTV07, GPV12] and the references therein, as well as to the works we cited in Section 2.

Since the identification of this type of systems can be seen as a natural testbed for k-PAMF-PLS, in this section we tackle this task with our exact and heuristic algorithms. In particular, we compare our exact formulation to the MILP formulation proposed in [RBL04] for the subclass of piecewise affine models based on *hinging hyperplanes* and compare the results of our heuristic 4S-CR to a state-of-the-art approach [OL13] on a real-world benchmark dataset that has been often used in the literature.

7.1. Comparison with a MILP formulation for the identification of hinging hyperplane models

Hinging Hyperplane (HH) models, originally introduced in [Bre93], are piecewise affine models defined as a sum of so-called *hinge functions* of the form $g(\boldsymbol{x}) = \pm \max\left\{\boldsymbol{w}^+\boldsymbol{x} - w_0^+, \boldsymbol{w}^-\boldsymbol{x} - w_0^-\right\}$. The graph of a hinge function is thus the union of two intersecting half-hyperplanes with normal vectors $(\boldsymbol{w}^+, w_0^+)$ and $(\boldsymbol{w}^-, w_0^-)$ which, depending on the sign, can be either convex or concave. See Figure 5 for an illustration.

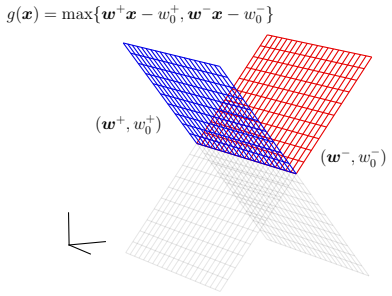


Figure 5: A convex hinge function.

HH models provide a convenient way to represent a large subclass of piecewise affine models, and can be effectively used in the identification of hybrid systems, as proposed in [RBL04]. After a reparameterization, an HH model composed of h hinge functions can be written as:

$$f(\boldsymbol{x}) = \boldsymbol{w}^0\boldsymbol{x} - w_0^0 + \sum_{j=1}^{h} s_j \max\left\{\boldsymbol{w}^j\boldsymbol{x} - w_0^j, 0\right\},$$

where $s_j \in \{-1, +1\}$ represents the sign of the j-th hinge function. The problem of identifying an HH model (here referred to as Hinging Hyperplane Model

Fitting, or HHMF) can be formulated [RBL04] as the following MILP:

$$\min \sum_{i=1}^{m} z_i \tag{29}$$

$$z_i \geq b_i - \boldsymbol{w}^0 \boldsymbol{a}_i + w_0^0 - \sum_{j=1}^{h} s_j \beta_{ij} \qquad \forall i \in I \tag{30}$$

$$z_i \geq -b_i + \boldsymbol{w}^0 \boldsymbol{a}_i - w_0^0 + \sum_{j=1}^{h} s_j \beta_{ij} \qquad \forall i \in I \tag{31}$$

$$0 \leq \beta_{ij} \leq \overline{M}_{ij} \delta_{ij} \qquad \forall i \in I, j \in J \tag{32}$$

$$\boldsymbol{w}^j \boldsymbol{a}_i - w_0^j \leq \beta_{ij} \qquad \forall i \in I, j \in J \tag{33}$$

$$\beta_{ij} + \underline{M}_{ij}(1 - \delta_{ij}) \leq \boldsymbol{w}^j \boldsymbol{a}_i - w_0^j \qquad \forall i \in I, j \in J \tag{34}$$

$$(\boldsymbol{w}^1 \ w_0^1)\boldsymbol{v} \geq (\boldsymbol{w}^2 \ w_0^2)\boldsymbol{v} \geq \cdots \geq (\boldsymbol{w}^h \ w_0^h)\boldsymbol{v} \tag{35}$$

$$\delta_{ij} \in \{0, 1\} \qquad \forall i \in I, j \in J \tag{36}$$

$$z_i \geq 0 \qquad \forall i \in I \tag{37}$$

$$(\boldsymbol{w}^j, w_0^j) \in \mathbb{R}^{n+1} \qquad \forall j \in J. \tag{38}$$

Consistently with the notation used throughout this paper, the data points are denoted by $(\boldsymbol{a}_i, b_i) \in \mathbb{R}^{n+1}$, while $(\boldsymbol{w}^j, w_0^j)$ are the variables describing the fitting hyperplanes (in this case, hinging hyperplanes), and the nonnegative variables z_i represent the absolute error for each point $i \in I$. The auxiliary variables β_{ij} are introduced to linearize the term $\max\left\{\boldsymbol{w}^j \boldsymbol{x} - w_0^j, 0\right\}$ via Constraints (32)–(34), where $\underline{M}_{ij}$ and $\overline{M}_{ij}$ are lower and upper bounds on $\boldsymbol{w}^j \boldsymbol{a}_i - w_0^j$. We remark that, in principle, $\boldsymbol{w}^j \boldsymbol{a}_i - w_0^j$ is unbounded. Since the bounds $\underline{M}_{ij}$ and $\overline{M}_{ij}$ are necessary for the linearization though, they should be chosen carefully, lest the optimality of the solution is invalidated. Constraint (35) is included to break some of the symmetries of the model, where $\boldsymbol{v} \in \mathbb{R}^{n+1}$ is an arbitrary vector.

Due to their structure, hinging hyperplane models are clearly a restriction of the more general class of piecewise affine models. Quite conveniently, the subdomains D_j are implicitly induced by the hinge functions and, therefore, they can be reconstructed analytically *a posteriori*, if necessary. Note also that, differently from the models found via k-PAMF-PLS, the actual number of affine submodels in an optimal solution of HHMF cannot be fixed *a priori* to an arbitrary number, as it depends on the number of regions into which h hyperplanes can partition the domain D. When $D = \mathbb{R}^2$, such number is $\frac{h(h+1)}{2} + 1$, i.e., the maximum number of regions into which h lines divide a plane[8].

When comparing the formulation for HHMF to that for k-PAMF-PLS, we see that Formulation (29)–(38) contains $O(n|J| + |I||J|)$ variables and $O(|I||J|)$ con-

[8] Usually referred to as the *lazy caterer's sequence*.

straints, while Formulation (4)–(12) has $O(n|J|+|I||J|)$ variables and $O(|I||J|^2)$ constraints. The HHMF formulation has a few intrinsic limitations. First of all, the sign determining the convexity or concavity of each hinging function has to be selected in advance, and may clearly affect the quality of the model. Secondly, the formulation does not allow for discontinuities in the fitted model. To overcome these limitations, extra variables and constraints must be added, as also mentioned in [RBL04]. Since the more general formulation thus obtained is even more challenging to solve, we will restrict ourselves to Formulation (29)–(38), which is used in all the experiments in [RBL04].

To compare k-PAMF-PLS to HHMF, we consider the two main instances adopted in [RBL04], neglecting the toy examples. By imposing, for both of them, $n_y = 1, n_u = 1$, each data point $(\boldsymbol{a}_t, b_t)$ in the corresponding MILP formulation is defined in terms of the system inputs and outputs as:

$$\boldsymbol{a}_t := [y_{t-1}\ u_{t-1}],\ b_t := y_t.$$

7.1.1. Instance 1 (Example 5 in [RBL04])

Let us consider 100 data points obtained from a system whose output at time t is computed according to the bivariate HH function:

$$y_t = -0.3 + 1.2y_{t-1} - u_{t-1} + \max\{1.2 + 2u_{t-1}, 0\} - \max\{-0.2y_{t-1}, 0\} + e_t, \quad (39)$$

where e_t is a Gaussian noise with a variance of 0.01. The data points and the original HH function from which they are sampled are displayed in Figure 6.

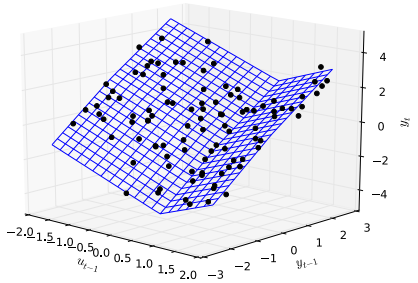


Figure 6: The HH function in Equation (39) with 100 sampled data points.

Our MILP formulation for k-PAMF-PLS is solved in 4.6 seconds for $k = 2$, yielding a total absolute error of 8.67. Consider now HHMF with a single hinge function (thus, two affine submodels). We assume that the appropriate choice $s_1 = +1$ is known, since the data comes from a mostly convex function (see Figure 6). Formulation (29)–(38) is solved in 5.8 seconds, and the corresponding ℓ_1-norm error in the optimal solution is 8.805. It is worth noting that the k-PAMF-PLS solution achieves a better fit by allowing for a discontinuity that cannot be reproduced with the considered HHMF formulation.

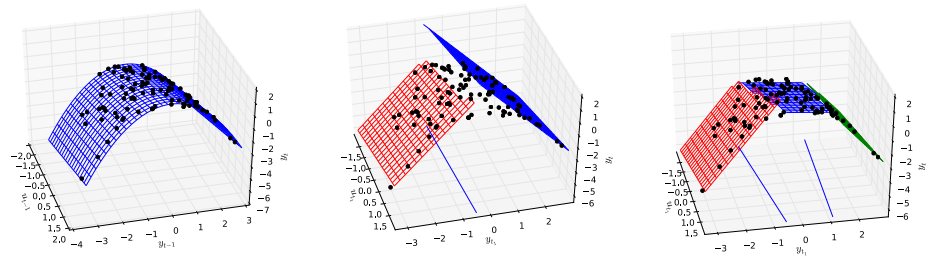
On this instance, HHMF is not solved to optimality for any $h \geq 2$ (50.6% gap at the time limit with $h = 2$), suggesting that the identification of hinging hyperplane models is a difficult task even when the sign of the hinge functions is fixed *a priori*. Similarly, k-PAMF-PLS becomes very challenging already for $k = 3$, for which CPLEX obtains a solution with 19.8% gap in 3600 seconds.

7.1.2. Instance 2 (Example 6 in [RBL04])

Let us consider an instance with 100 data points, obtained from a system whose output at time t is defined according to the quadratic function:

$$y_t = -0.5y_{t-1}^2 + 0.7u_{t-1} + e_t, \tag{40}$$

where e_t is Gaussian noise with a variance of 0.01. The data points and the original quadratic function from which they are sampled are displayed in Figure 7(a).



(a) (b) (c)

Figure 7: (a) The quadratic function in Equation (40) with 100 sampled data points. (b) The model estimated by solving k-PAMF-PLS for $k = 2$. (c) The model estimated by solving k-PAMF-PLS for $k = 3$.

For $k = 2$, k-PAMF-PLS is solved to optimality in 5.2 seconds. With a single concave hinge function (we set $s_1 = -1$ due to the nonlinear component of the function in Equation (40) being concave), HHMF is solved to optimality within 2.4 seconds. Interestingly, the two solutions obtained with k-PAMF-PLS and HHMF have the same fitting error of 26.77, and very similar submodel equations. The solution obtained with k-PAMF-PLS is represented in Figure 7(b).

Both k-PAMF-PLS and HHMF become significantly harder when more than two affine submodels are considered. For $k = 3$, k-PAMF-PLS is solved to optimality in 1211 seconds, yielding an error of 15.16; the optimal solution is shown in Figure 7(c). For $k = 4$, optimality appears out of reach within the time limit. As far as HHMF is concerned, CPLEX does not reach optimality within one hour for any value of h larger than 1 (for $h = 2$, we register a gap of 36.4% at the time limit).

These experiments indicate that the MILP formulations for k-PAMF-PLS and HHMF are of comparable difficulty. This may be due to the fact that the bounds from the linear programming relaxations at each node of the branch-and-bound tree are, typically, rather weak.

7.2. System identification for a pick-and-place machine model

We now consider a real-world benchmark case study: the pick-and-place machine example adopted in [BGPV05, JHFT$^+$05, OL13]. The dataset is sampled from a *pick-and-place machine* used to place electronic components on a circuit board. It consists of 15 seconds of input (the input voltage of the machine, $\boldsymbol{u}$) and output (the vertical position of the mounting head, $\boldsymbol{y}$) sampled at 4000Hz. See Figure 8 for a depiction. The physical system has two main modes: the *free mode*, active when the head moves freely, and the *impact mode*, active when the head is in contact with the circuit board.

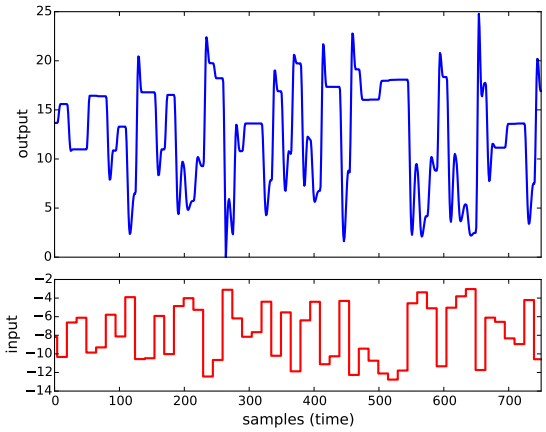


Figure 8: Input and output vectors $\boldsymbol{u}$ and $\boldsymbol{y}$ sampled from the pick-and-place machine.

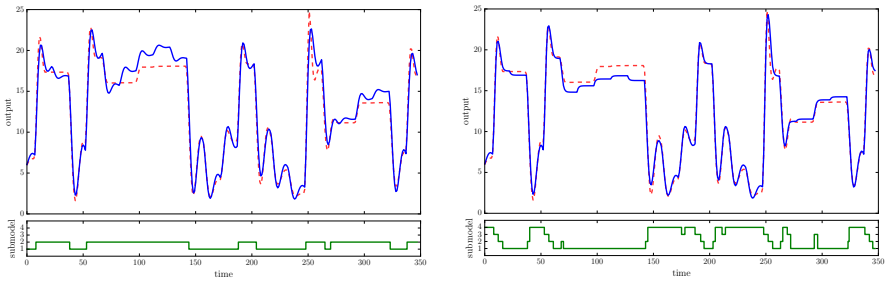
In our experiments, we adopt the settings of [OL13] which, to the best of our knowledge, is one of the latest references using this dataset. We will compare the results achieved by k-PAMF-PLS to those in [OL13], obtained with a method which aims at balancing the fitting error and the number of submodels via a *sum-of-norms regularization term*. The data are downsampled to 50 Hz, so as to obtain a total of 750 samples. We consider a model of orders $n_y = 2,\ n_u = 2$. The data points $(\boldsymbol{a}_t, b_t)$ are thus constructed as follows:

$$\boldsymbol{a}_t := [y_{t-1}\ y_{t-2}\ u_{t-1}\ u_{t-2}],\ b_t := y_t.$$

To test the effectiveness of the identification approach and assess its generalization capabilities, we split the data set in a training and a validation set, using the first 8 seconds (400 points) for training and the remaining 7 seconds (350 points) for validation. The validation phase is carried out by running a complete simulation of the system with the estimated model and the original input vector $\boldsymbol{u}$. Note that the task is significantly more challenging than predicting the output y_t at time t given perfect information on the past, i.e., assuming perfect knowledge of y_{t-1} and y_{t-2}. Indeed, the simulated output y_t is a function (of the

original input and) of the simulated outputs at time $t-1$ and $t-2$, which were also obtained by applying the identified piecewise affine model. Identification errors may clearly propagate over time.

For $k = 2$, our 4S-CR heuristic finds in 900 seconds a solution with a total error of 54.3 on the training set. When simulating the system with the inputs contained in the validation set, we obtain the output reported in Figure 9(a). Observe that this output is very similar to the solution with two affine submodels reported in [OL13]. The *fit percentage*[9] of our solution is 81.11%, slightly better than that of the solution reported in [OL13], which amounts to 78.6%.



(a) (b)

Figure 9: Simulation results on the pick-and-place machine example with (a) $k = 2$ and (b) $k = 4$, with real and simulated outputs reported as, respectively, a dashed and a solid line. The lower panels report the submodel that is active at each point in time.

Considering k-PAMF-PLS with $k = 4$ submodels, 4S-CR provides in 900 seconds a solution with an objective function of 29.4 on the training set and a 86.35% fit percentage in simulation, see the output in Figure 9(b). Even though a smaller fitting error on the training set can be obtained by further increasing the value of k, the performance in simulation (on the validation data) does not improve for larger values of k, most likely due to overfitting. Note that the k-PAMF-PLS solution well reproduces the behavior of the system over most of the time horizon, although the range [75, 150] appears to be particularly challenging, as also observed in [OL13].

Overall, the experiments suggest that the techniques that we proposed to solve k-PAMF-PLS could provide a valid alternative to state-of-the-art methods for piecewise affine system identification.

8. Concluding remarks

We have addressed the k-PAMF-PLS problem of fitting a piecewise affine model with a piecewise linearly separable domain partition to a set of data

[9] The *fit percentage*, computed as $100\left(1 - \frac{\|\boldsymbol{y}_{sim} - \boldsymbol{y}_{true}\|}{\|\boldsymbol{y}_{true} - \bar{y}\|}\right)$, is a measure of the fitting error, where $\boldsymbol{y}_{sim}$ is the vector of simulated outputs, $\boldsymbol{y}_{true}$ is the vector of the original, sampled outputs, and $\bar{y}$ is the mean of the original, sampled output. See [BGPV05] for more details.

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 4S-CR, a four-step heuristic algorithm which simultaneously deals with the various aspects of the problem.

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. Our four-step heuristic provides close-to-optimal solutions for small-size instances, while also allowing us 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, suggesting to address them in a joint way to achieve high quality solutions to k-PAMF-PLS. On the instances from the UCI repository, the results obtained with cross-validation confirm that low-complexity models (with a small value of k) provide solutions with good generalization properties.

Finally, the performance of our k-PAMF-PLS exact and heuristic algorithms is confirmed on problems arising in the field of dynamical system identification, for which, on benchmark instances from the literature, the quality of the solutions provided by our approaches well compares with state-of-the-art methods.

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.

[BGPV05] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A bounded-error approach to piecewise affine system identification. *Automatic Control, IEEE Transactions on*, 50(10):1567–1580, 2005.

[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.

[BM14] D. Bertsimas and R. Mazumder. Least quantile regression via modern optimization. *The Annals of Statistics*, 42(6):2494–2525, 2014.

[Bre93] L. Breiman. Hinging hyperplanes for regression, classification, and function approximation. *Information Theory, IEEE Transactions on*, 39(3):999–1013, 1993.

[BS07] D. Bertsimas and R. Shioda. Classification and regression via integer optimization. *Operations Research*, 55:252–271, 2007.

[CBR12] A. Chang, D. Bertsimas, and C. Rudin. An integer optimization approach to associative classification. In F. Pereira, C. Burges, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 269–277, 2012.

[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.

[CSK06] O. Chapelle, V. Sindhwani, and S. Keerthi. Branch and bound for semi-supervised support vector machines. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 217–224, 2006.

[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.

[GPV12] A. Garulli, S. Paoletti, and A. Vicino. A survey on switched and piecewise affine system identification. In *System Identification*, volume 16, pages 344–355, 2012.

[IR03] F. Iannarilli and P. Rubin. Feature selection for multiclass discrimination via mixed-integer linear programming. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):779–783, 2003.

[JHFT+05] A. Juloski, W. Heemels, G. Ferrari-Trecate, R. Vidal, S. Paoletti, and J. Niessen. Comparison of four procedures for the identification of hybrid systems. In *Hybrid systems: computation and control*, pages 354–369. Springer, 2005.

[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 Statististics 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.

[MK05] E. Münz and V. Krebs. Continuous optimization approaches to the identification of piecewise affine systems. In *Proceedings of the 16th IFAC World Congress*, 2005.

[MRT05] O. Mangasarian, J. Rosen, and M. Thompson. Global minimization via piecewise-linear underestimation. *Journal of Global Optimization*, 32(1):1–9, 2005.

[MT15] R. Miyashiro and Y. Takano. Mixed integer second-order cone programming formulations for variable selection in linear regression. *European Journal of Operational Research*, 247(3):721 – 731, 2015.

[OL13] H. Ohlsson and L. Ljung. Identification of switched linear regression models using sum-of-norms regularization. *Automatica*, 49(4):1045–1050, 2013.

[OSLC12] N. Ozay, M. Sznaier, C. Lagoa, and O. Camps. A sparsification approach to set membership identification of switched affine systems. *Automatic Control, IEEE Transactions on*, 57(3):634–648, 2012.

[PJFTV07] S. Paoletti, A. Juloski, G. Ferrari-Trecate, and R. Vidal. Identification of hybrid systems: a tutorial. *European Journal of Control*, 13(2):242–260, 2007.

[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.