Discrete Applied Mathematics 336 (2023) 15–36

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

A unified framework for bivariate clustering and regression problems via mixed-integer linear programming

John Alasdair Warwicker *, Steffen Rebennack

Institute of Operations Research (IOR), Karlsruhe Institute of Technology, 76185 Karlsruhe, Baden-Württemberg, Germany

ARTICLE INFO

Article history:
Received 8 June 2022
Received in revised form 27 February 2023
Accepted 8 March 2023
Available online 1 April 2023

Keywords:
Clustering
Regression
Mixed-integer linear programming
Outlier detection
Decomposition
Function fitting

ABSTRACT

Clustering and regression are two of the most important problems in data analysis and machine learning. Recently, mixed-integer linear programs (MILPs) have been presented in the literature to solve these problems. By modelling the problems as MILPs, they are able to be solved very quickly by commercial solvers. In particular, MILPs for bivariate clusterwise linear regression (CLR) and (continuous) piecewise linear regression (PWLR) have recently appeared. These MILP models make use of binary variables and logical implications modelled through big-$\mathcal{M}$ constraints. In this paper, we present these models in the context of a unifying MILP framework for bivariate clustering and regression problems. We then present two new formulations within this framework, the first for ordered CLR, and the second for clusterwise piecewise linear regression (CPWLR). The CPWLR problem concerns simultaneously clustering discrete data, while modelling each cluster with a continuous PWL function. Extending upon the framework, we discuss how outlier detection can be implemented within the models, and how specific decomposition methods can be used to find speedups in the runtime. Experimental results show when each model is the most effective.

1. Introduction

Two of the most common problems in the fields of data analysis and statistics are *data fitting* and *classification*. Such problems allow the interpolation and extrapolation from discrete data by means of predicting future trends and classifying data points based on their inherent qualities. Regression analysis is the most common form of data fitting, and is used to estimate the relationship between the dependent variables and the independent variables. Linear regression attempts to model the data with a linear function.

Classification is usually achieved by clustering the data. Clustering is the process of partitioning a set of objects (data) into clusters, such that objects in the same cluster share some similarity, and are dissimilar to objects in different clusters. That is, the goal of clustering problems is to maximise the homogeneity within each cluster, while also maximising the heterogeneity between different clusters. From a machine learning perspective, clustering is a form of unsupervised learning, since inferences must be drawn from input data without labelled responses. In general, clustering is considered one of the most difficult and challenging problems in the field of machine learning. Applications of clustering are found in the fields of biology, medicine, business and computer science [54]. Perhaps the most famous approach to clustering is the k-Means approach, where data points are assigned to the nearest clusters, whose centres are iteratively updated to the mean position of the inherent data points [29]. Recent approaches have used clustering approaches

* Corresponding author.
E-mail addresses: john.warwicker@kit.edu (J.A. Warwicker), steffen.rebennack@kit.edu (S. Rebennack).

https://doi.org/10.1016/j.dam.2023.03.010

within multivariate data modelling algorithms [49], and utilised branch-and-bound approaches alongside semi-definite programming relaxations to solve very large instances [41].

Clustering problems can be divided into two classes [22]:

1. Each cluster is considered as an area of high density of the original data set distribution;
2. Each cluster is considered as a subset of the whole set of data points, and each cluster can be separately modelled with a given function. In these instances, one is interested in the classification and estimation within each cluster.

Problems within the first category group the data into different clusters. However, this is usually an exploratory first step in the data analysis, since no further information or characteristics are immediately given from the data within each cluster. While smaller data sets run the risk of being overfitted, post cluster analysis can be performed on larger data sets.

Clusterwise linear regression (CLR) problems fall into the second category. The aim of CLR is to simultaneously cluster objects while performing linear regression on each of the clusters [44,45]. Early attempts to solve the CLR problem used heuristic methods, such as maximum likelihood [12] and simulated annealing [13]. Recently, Bertsimas and Shioda [6] approached the problem from an integer optimisation perspective by constructing a software package called CRIO (classification and regression via integer optimisation), which gives promising results compared to heuristic approaches. Park et al. [40] presented various heuristic and exact methods to solve the generalised CLR problem (which allows each data point to have multiple observations), including mathematical programming methods and adapted k-Means approaches. Angün and Altınoy [1] presented a nested algorithm to solve the CLR problem which decomposes to a mixed-integer linear programming (MILP) model when the number of clusters is fixed. Recently, Joki et al. [24] utilised support vector machines for regression within a difference of convex approach for CLR, resulting in a constrained nonsmooth optimisation problem, while da Silva and de Carvalho [11] introduced a weighted CLR method which is robust against overfitting. In general, typical approaches to CLR in the literature consider high-dimensional data points with many features.

As opposed to standard linear regression models, fitting data with a continuous piecewise linear (PWL) function is slightly more complex. PWL regression (PWLR), also known as segmented regression or linear spline regression, is the problem of fitting a (continuous) PWL function to discrete data. A continuous PWL function consists of a given number of linear segments which intersect at breakpoints. PWLR allows us to predict data trends and make observations based on modelled data which does not necessarily adhere to a strict singular linear relationship.

Early attempts at PWLR used information provided by the convex hull of the data to find PWL functions in polynomial time [19,20,23]. Another typical approach has been the use of dynamic programming (DP) to solve the PWLR problem, including the early approach by Bellman and Roth [2] whereby breakpoints are selected from a grid of uniformly distributed candidate points. Recent DP approaches have implemented an adaptive refinement approach on the candidate breakpoints [18], as well as robust approaches to deal with noise in the data set [7].

The PWLR problem has recently been modelled as a MILP problem, with binary variables assigning data points to their associated linear segments and big-$\mathcal{M}$ constraints working to model logical implications [27,42]. Recently, Warwicker and Rebennack [52] showed that the MILP approach presented by Rebennack and Krasko [42] is preferable on most instances to the similar approach presented by Kong and Maravelias [27] over three different distance metrics and a number of data sets with differing landscapes. Typical PWL modelling works consider bivariate data, since modelling PWL functions in higher dimensions is prohibitive and only heuristic or approximate non-linear methods exist to date (see e.g., [14,15]).

The MILP models for bivariate CLR and PWLR fit into the same monolithic framework, where binary variables are used to assign data points to their linear functions. In this paper, we extend upon this framework by introducing the clusterwise piecewise linear regression (CPWLR) problem. CPWLR is another problem in the second category of clustering problems, where each cluster is modelled with a continuous PWL function. This allows for a greater level of precision, since it is not necessarily the case that each cluster adheres to a strict linear relationship. Since CPWLR can be seen as an advancement of both the CLR and PWLR problems, all applications from these two areas benefit from models for CPWLR, including supermarket forecasting [40], pavement condition prediction [25,30], utilities networks [17,33], classification [6,32,55], and combinatorial optimisation [38,39]. DeSarbo and Cron [12] also suggest possible applications of CLR in marketing, psychological analysis and political science, for which CPWLR would present more accurate models.

For CPWLR, the measure of similarity within each cluster can be given by the objective of the PWLR problem. We aim to model the bivariate CPWLR problem within our overarching MILP framework, such that standard solvers (such as ILOG-Cplex) can be employed to solve the problem quickly and efficiently. The MILP simultaneously clusters the data and fits each cluster with a PWL function. The initial model we present in this paper relies on three assumptions: firstly, that the clusters are ordered. Formally, we seek a partition of the data set $\{x_1, \dots, x_I\}$ into $K \geq 1$ clusters, where $C_1 = \{x_1, \dots, x_{C_1}\}$, $C_2 = \{x_{C_1+1}, \dots, x_{C_2}\}$, $\dots$, $C_K = \{x_{C_{K-1}+1}, \dots, x_{C_K}\}$, and $x_{C_1} < x_{C_2} < \dots < x_{C_K} = x_I$ holds. Secondly, we assume that each data point has only one observation, and thirdly, that the number of clusters (and the total number of linear segments) is known in advance. That is, we present a MILP for ordered, non-generalised, non-hierarchical CPWLR.

Throughout this paper, we present MILP models for the (bivariate) CLR problem [40], the ordered CLR problem, the PWLR problem [42], and the CPWLR problem. Each MILP formulation builds on the previous models by introducing constraints to bypass the difficulty in modelling the extra requirements. Furthermore, we show how each model can be adapted to remove outliers during the solution process, and how combinatorial Benders decomposition [8] can be used to take advantage of the special structure of the models in order to find optimal solutions quicker.

This paper has the following main contributions:

- We present a unified MILP framework for bivariate clustering and regression problems, which generalises existing models from the literature using a consistent structure;
- We introduce two new programs within this framework, one for an ordered variant of CLR, and an innovative formulation for CPWLR. The formulation for CPWLR simultaneously clusters the data and performs piecewise linear regression on each of the clusters;
- We discuss how outlier detection and decomposition methods can be implemented to bring improvements in robustness and efficiency to the models presented within the framework;
- We present a comparison of the models on a series of real-world data sets with differing landscapes, drawing conclusions on situations in which each of the models is advantageous.

The rest of the paper is structured as follows. In Section 2, we introduce the existing MILP models in the literature. We present the formulations for CLR, the newly introduced ordered CLR (oCLR), and the existing model for PWLR, discussing how the formulations are related. In Section 3, we introduce the first MILP model for the CPWLR problem. This model takes inspiration from the previous models and allows the simultaneous clustering and piecewise linear regression of discrete data. In Section 4, we discuss how to implement outlier detection into the MILP models, and how combinatorial Benders decomposition can be used to bring speedups to the runtime. We present experimental results in Section 5, and we finish with some conclusions and ideas for future work.

2. Existing mixed-integer linear models

In this section, we discuss the existing mixed-integer linear models for clusterwise linear regression (including an ordered variant) and piecewise linear regression from the literature.

Throughout this paper we use the following notation: $[n]$ to denote the set $\{1, \dots, n\}$. For each problem, we are given a set of I data tuples, of the form $(X_i, Y_i) \in \mathbb{R}^2$, $i \in [I]$, and we are looking to model this data using functions defined over the interval $[\underline{X}, \overline{X}]$. We can assume without loss of generality that:

$$-\infty < \underline{X} = X_1 \leq \cdots \leq X_i \leq X_{i+1} \leq \cdots \leq X_I = \overline{X} < \infty.$$

The standard model for linear regression can be represented as a MILP in the following way (see e.g., [46]).

$$\begin{aligned}
\underline{\text{LR}}: \quad \min \quad & \sum_{i=1}^{I} \xi_i \\
\text{s.t.} \quad & Y_i - (cX_i + d) \leq \xi_i && \forall\, i \in [I] \\
& (cX_i + d) - Y_i \leq \xi_i && \forall\, i \in [I] \\
& \xi_i \geq 0 && \forall\, i \in [I] \\
& c \in [\underline{C}, \overline{C}],\ d \in [\underline{D}, \overline{D}]
\end{aligned}$$

This model fits a linear function of the form $y = cx + d$ (with slope c and intercept d) such that the sum of absolute errors between the regression function and the data points (i.e., the *residuals*, each calculated by ξ_i) is minimised. The extreme values for the slope (i.e., $\underline{C}$ and $\overline{C}$) can, for example, be calculated by interpolating between every pair of data points. The corresponding values for the intercept (i.e., $\underline{D}$ and $\overline{D}$) can then be calculated by using the extreme values of the slopes to calculate the maximal and minimal possible values for the x-intercept of segments (see Section A of the Appendix, or, e.g., [42] for detailed derivations for these values).

2.1. Clusterwise linear regression

We first present a MILP model for the clusterwise linear regression (CLR) problem, in which data is simultaneously clustered and each cluster is modelled with a linear regression function. This model forms the basis of our universal framework, and is based on the model presented by Park et al. [40] (yet assumes only one observation from each data point, i.e., it is non-generalised). The model returns B optimal CLR functions for the data (i.e., one linear segment for each of the B clusters), subject to the given distance metric. We introduce the following variables, which are used throughout each of the models in our framework presented in this section.

Continuous variables

- c_b, the slope of segment $b \in [B]$;
- d_b, the intercept of segment $b \in [B]$;
- ξ_i, absolute error at data point $i \in [I]$.

Binary variables

- $\delta_{i,b}$, set to 1 if data point X_i ($i \in [I]$) is associated with segment $b \in [B]$;

That is, the model returns B affine functions, $y = c_b x + d_b$ (for $b \in [B]$), defined by their slopes and intercepts. We present the model in formulation (1).

$$
\begin{aligned}
(1)\ \underline{\text{CLR}}: \quad \min \quad & \sum_{i=1}^{I} \xi_i && && && (1a)\\
\text{s.t.} \quad & Y_i - (c_b X_i + d_b) \leq \xi_i + M_i^1(1-\delta_{i,b}) && && \forall\, i \in [I];\ b \in [B] && (1b)\\
& (c_b X_i + d_b) - Y_i \leq \xi_i + M_i^1(1-\delta_{i,b}) && && \forall\, i \in [I];\ b \in [B] && (1c)\\
& \sum_{b=1}^{B} \delta_{i,b} = 1 && && \forall\, i \in [I] && (1d)\\
& \xi_i \geq 0 && && \forall\, i \in [I] && (1e)\\
& c_b \in [\underline{C}_b, \overline{C}_b] && && \forall\, b \in [B] && (1f)\\
& d_b \in [\underline{D}_b, \overline{D}_b] && && \forall\, b \in [B] && (1g)\\
& \delta_{i,b} \in \{0,1\} && && \forall\, i \in [I];\ b \in [B] && (1h)
\end{aligned}
$$

The objective function of the CLR problem is given by constraint (1a). For the given example, we present the sum of absolute differences metric. The contribution towards the value of the objective function by each data point is evaluated in constraints (1b)–(1c). Constraint (1d) ensures that each data point is only associated to one linear segment. Finally, constraints (1e)–(1h) give the domains of the variables. Note that the formulation itself does not ensure the ordering of the data points, and hence clusters may consist of disjoint sets (i.e., non-continuous sets in x-space).

The model also includes the big-$\mathcal{M}$ constants M_i^1, for each $i \in [I]$, which should be set as tightly as possible for efficient performance. In order for constraints (1b)–(1c) to hold, they should be greater than or equal to the largest possible difference between Y_i and any linear segment (i.e., $M_i^1 \geq \max_{b\in[B]} |Y_i - (c_b X_i + d_b)|$). Hence, we can set:

$$M_i^1 = \max\{|Y_i - \overline{C} X_i - \overline{D}|, |Y_i - \overline{C} X_i - \underline{D}|, |Y_i - \underline{C} X_i - \overline{D}|, |Y_i - \underline{C} X_i - \underline{D}|\} \qquad \forall\, i \in [I].$$

The MILP formulation introduced by Park et al. [40] also includes a constraint which requires each cluster to have a minimum number ($n \geq 1$) of data points associated with it:

$$\sum_{i=1}^{I} \delta_{i,b} \geq n \qquad \forall\, b \in [B].$$

A different mixed-integer model for CLR has also been presented by Bertsimas and Shioda [6], which first classifies the data into a pre-defined number of clusters, before assigning each cluster with regression co-efficients. However, due to the non-linearity of this approach, we consider the MILP presented in formulation (1) (inspired by Park et al. [40]) as the canonical MILP for CLR within the framework we present.

2.2. Distance metric

Formulation (1) provides optimal CLR functions for the given data set, subject to the chosen objective function. There are many objective functions in the literature to measure the *closeness* of clusters and the fit of the PWL function. For most applications of CLR and PWLR problems, the absolute vertical distance between the data points and their associated linear segments (i.e., the values ξ_i, $i \in [I]$) contributes to the objective function.

Constraint (1a) presents the objective function for the *sum of absolute differences* metric (also referred to as the L_1 norm in the literature). If the term being summed is replaced by ξ_i^2, then this results in the *sum of squared differences* metric (L_2 norm), which is commonly used in regression problems. However, we note that this results in formulation (1) becoming a (quadratically constrained convex) mixed-integer non-linear program (MINLP). These two metrics (which take each data point into account when calculating the objective function) are less prone to outliers, and are generally preferred in realistic settings due to the quality of their approximations. However, due to their complexity, the time required to find an optimal fit (with regard to the given metric) is significantly increased in comparison with 'simpler' metrics, even more so for the sum of squared differences due to the non-linearity in the objective function.

For the use of the *maximum absolute difference* metric (L_∞ norm), we adapt the constraints (1a)–(1c) and the variable domain in constraint (1e) as such:

$$
\begin{aligned}
\min \quad & \xi\\
\text{s.t.} \quad & Y_i - (c_b X_i + d_b) \leq \xi + M_i^1(1-\delta_{i,b}) \quad \forall\, i \in [I];\ b \in [B]\\
& (c_b X_i + d_b) - Y_i \leq \xi + M_i^1(1-\delta_{i,b}) \quad \forall\, i \in [I];\ b \in [B]\\
& \xi \geq 0
\end{aligned}
$$

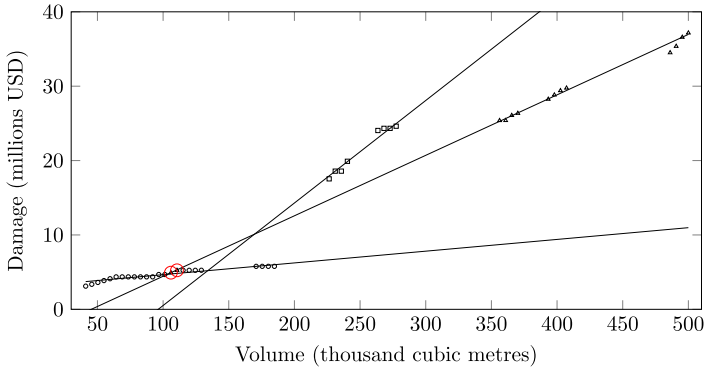


Fig. 1. An example of clusterwise linear regression (CLR) with three clusters on the *DebrisFlow* data set [28,34]. Different clusters are marked by different shapes.

This metric seeks to minimise the distance to the farthest outlier point, and hence is heavily susceptible to outliers in the data set. However, finding an optimal ξ with regards to this metric takes less time in comparison to the two previous metrics, and often provides good approximations to the data in the case of data sets without outliers.

We can also consider the *maximum absolute clusterwise difference* metric, which aims to minimise the maximum absolute difference from each cluster.

$$\begin{aligned} \min \quad & \sum_{b=1}^{B} \xi_b \\ \text{s.t.} \quad & Y_i - (c_b X_i + d_b) \leq \xi_b + M_i^1(1-\delta_{i,b}) \quad \forall\, i \in [I];\ b \in [B] \\ & (c_b X_i + d_b) - Y_i \leq \xi_b + M_i^1(1-\delta_{i,b}) \quad \forall\, i \in [I];\ b \in [B] \\ & \xi_b \geq 0 \qquad\qquad\qquad\qquad\qquad\qquad\quad \forall\, b \in [B] \end{aligned}$$

This metric is specific to CLR and CPWLR applications. In this setting, each data point for a given segment b is within a maximum distance of ξ_b to the associated linear segment (for $b \in [B]$). One advantage of this approach (alongside the maximum absolute difference metric) is that by translating each resulting linear segment $b \in [B]$ above and below by ξ_b, we can form an approximation of the interior of the convex hull formed by the data points associated to each cluster, and provide over- and underestimators for each cluster. Regarding the performance of this metric, it typically provides better approximations than the maximum absolute differences metric (although it is slightly slower), and any outlier data points will only affect the approximation of the cluster they are assigned to.

In order to present an application of CLR to a real world data set, Fig. 1 shows the result of formulation (1) (using the sum of absolute differences metric) applied to the *DebrisFlow* data set, which is taken from a case study of postfire debris flow hazard management after a wildfire [28,34]. The goal is to predict the cost of the damage based on the volume of postfire debris flow.

The result fits the data very well for the given number of clusters ($B = 3$). However, the two highlighted data points have been assigned to the triangle-marked cluster. That is, data points may be misclassified based on their proximity to other clusters, as visually it appears the two points are a more natural fit for the circle-marked cluster. We note that this effect is dependent on the selection of the objective function; in particular, objective functions that penalise ill fitting models harshly (e.g., sum of squared differences) will reduce such occurrences of possible misclassification. Since formulation (1) aims at providing an analytical description of the data set, any new data points can be used to refine the model (although it must be re-solved in each case).

2.3. Ordered clusterwise linear regression

The CLR model presented in formulation (1) does not take into account the ordering of the data points, and there are few limitations on the assignment of the data points to clusters. Hence, the solve times are very long. To combat the long solve times, we can implement an ordering of the data points such that adjacent data points are either in the same or adjacent clusters. Due to the inclusion of such symmetry-breaking constraints on the binary variables, there is a modelling advantage in that solutions can be found significantly quicker. Although this comes at the cost of solution quality, it also allows for the clusters to be more well-defined around a given area.

We implement the ordering constraints for the data points from the MILP model for continuous PWLR presented by Rebennack and Krasko [42]. Formulation (2) presents the first MILP model for ordered clusterwise linear regression

(oCLR), using the sum of absolute differences metric:

$$(2)\ \underline{\text{oCLR}}: \quad \min \quad \sum_{i=1}^{I} \xi_i \tag{2a}$$

$$\text{s.t.} \quad (1b)\text{–}(1d) \tag{2b}$$

$$\delta_{i+1,b+1} \leq \delta_{i,b} + \delta_{i,b+1} \qquad \forall\, i \in [I-1];\ b \in [B-1] \tag{2c}$$

$$\delta_{i+1,1} \leq \delta_{i,1} \qquad \forall\, i \in [I-1] \tag{2d}$$

$$\delta_{i,B} \leq \delta_{i+1,B} \qquad \forall\, i \in [I-1] \tag{2e}$$

$$(1e)\text{–}(1h) \tag{2f}$$

Constraints (2c)–(2e) ensure the ordering of the data points amongst the linear segments, ensuring that adjacent data points are either associated to the same linear segment, or consecutive linear segments. Constraints (2d) and (2e) ensure the same for the first and last data points. The remaining constraints match exactly those of formulation (1).

Fig. 2 shows the result of applying formulation (2) for oCLR (using the sum of absolute differences metric) to the real world *DebrisFlow* data set. We can see that the clusters are well-defined, as we can identify the first and last data point in each cluster. Furthermore, we do not have the issue which is experienced by the CLR formulation, whereby data points may be misclassified based on their proximity to other clusters. The *DebrisFlow* data set is modelled very well by oCLR; however, we note that other data sets may not be modelled so accurately. Furthermore, oCLR does not allow for accurate predictions from future data where the independent variable lies between existing clusters. Again, the model would have to be re-solved in such instances.

2.4. Piecewise linear regression

As opposed to fitting discontinuous linear functions to discrete data, there is an advantage to fitting a continuous piecewise linear (PWL) function. In particular, if we want to model the data as a function, using a PWL function as opposed to a polynomial function avoids the non-linearities associated with such models. Hence, they are computationally efficient. The continuity aspect of a PWL function can provide breakpoint information as to where different trends can occur.

Continuous PWL functions consist of linear segments which intersect at breakpoints. A variety of MILP models for piecewise linear regression (PWLR) have been presented in the literature, although the majority avoid the complicating continuity requirement [3–5]. In order to fit optimal PWL functions, it is necessary that the location of breakpoints is allowed to be placed freely, and not in fixed locations. Toriello and Vielma [48] introduced the first MILP model to fit optimal continuous PWL functions; however, the resulting PWL function is required to be convex:

$$\min \quad \sum_{i=1}^{I} \xi_i$$

$$\text{s.t.} \quad Y_i - (c_b X_i + d_b) \leq \xi_i + M_i^1(1-\delta_{i,b}) \qquad \forall\, i \in [I];\ b \in [B-1]$$
$$(c_b X_i + d_b) - Y_i \leq \xi_i \qquad \forall\, i \in [I];\ b \in [B-1]$$

$$\sum_{b=1}^{B-1} \delta_{i,b} = 1 \qquad \forall\, i \in [I]$$

$$c_b \leq c_{b+1} \qquad \forall\, b \in [B-2]$$
$$(1e)\text{ - }(1h)$$

This model takes advantage of the fact that the value of the resulting convex PWL function evaluated for any data point is maximal at the linear segment associated with that data point. That is, for y_i, the evaluation of the PWL function for data point i,

$$y_i \geq c_b X_i + d_b \qquad \forall\, i \in [I];\ b \in [B-1].$$

We present an application of this formulation to the *DebrisFlow* data set in Section B of the Appendix.

Recently, Rebennack and Krasko [42] presented the first MILP model to fit optimal continuous PWL functions without any requirements on convexity. Their model extends upon formulation (2) by implementing the continuity requirement between the linear segments. Firstly, they require that the data set be strictly ordered, i.e,;

$$-\infty < \underline{X} = X_1 < \cdots < X_i < X_{i+1} < \cdots < X_I = \overline{X} < \infty.$$

Furthermore, the following variables are introduced (note that a PWL function with B breakpoints and $B-1$ linear segments is being fitted):

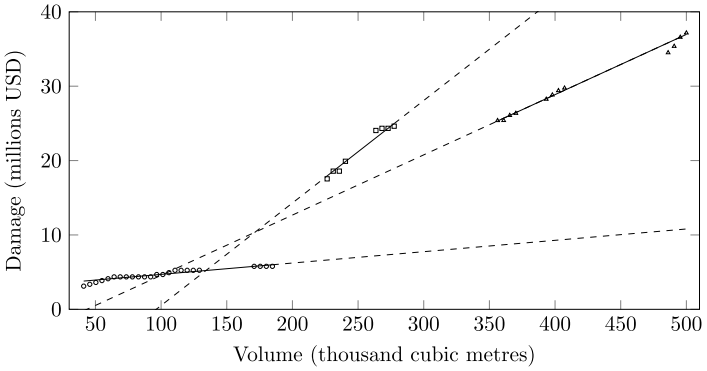


Fig. 2. An example of ordered clusterwise linear regression (oCLR) with three clusters on the *DebrisFlow* data set. Different clusters are marked by different shapes.

Continuous variables

- $\delta_{i,b}^{+/-}$, continuous variables taking values in [0, 1], and either $\delta_{i,b}^{+}$ or $\delta_{i,b}^{-}$ is set to 1 if a breakpoint exists between X_i and X_{i+1}.

Binary variables

- γ_b, set to 0 or 1 depending on the change in gradient between adjacent linear segments.

The following big-$\mathcal{M}$ constant is also introduced:

$$M_i^2 = \overline{D} - \underline{D} - X_i(\underline{C} - \overline{C}) \qquad \forall\, i \in [I].$$

The MILP model for PWLR by Rebennack and Krasko [42] is presented in formulation (3).

$$\begin{aligned}
(3)\ \underline{\text{PWLR}}: \quad \min \quad & \sum_{i=1}^{I} \xi_i && && (3a)\\
\text{s.t.} \quad & (1b)\text{–}(1d) && && (3b)\\
& (2c)\text{–}(2e) && && (3c)\\
& \delta_{i,b} + \delta_{i+1,b+1} + \gamma_b - 2 \le \delta_{i,b}^{+} && \forall\, i \in [I-1];\ b \in [B-2] && (3d)\\
& \delta_{i,b} + \delta_{i+1,b+1} + (1-\gamma_b) - 2 \le \delta_{i,b}^{-} && \forall\, i \in [I-1];\ b \in [B-2] && (3e)\\
& d_{b+1} - d_b \ge X_i(c_b - c_{b+1}) - M_i^2(1 - \delta_{i,b}^{+}) && \forall\, i \in [I-1];\ b \in [B-2] && (3f)\\
& d_{b+1} - d_b \le X_{i+1}(c_b - c_{b+1}) + M_{i+1}^2(1 - \delta_{i,b}^{+}) && \forall\, i \in [I-1];\ b \in [B-2] && (3g)\\
& d_{b+1} - d_b \le X_i(c_b - c_{b+1}) + M_i^2(1 - \delta_{i,b}^{-}) && \forall\, i \in [I-1];\ b \in [B-2] && (3h)\\
& d_{b+1} - d_b \ge X_{i+1}(c_b - c_{b+1}) - M_{i+1}^2(1 - \delta_{i,b}^{-}) && \forall\, i \in [I-1];\ b \in [B-2] && (3i)\\
& \gamma_b \in \{0, 1\} && \forall\, b \in [B-1] && (3j)\\
& \delta_{i,b}^{+/-} \in [0, 1] && \forall\, i \in [I-1];\ b \in [B-2] && (3k)\\
& (1e)\text{–}(1h) && && (3l)
\end{aligned}$$

Note firstly that constraints (1b)–(1c) and (1f)–(1h) are instead defined over the set of linear segments, i.e., $b \in [B-1]$.

The new constraints (3d)–(3i) ensure the continuity of the PWL function. Suppose there is a breakpoint between data point X_i and X_{i+1} connecting the linear segments b and $b+1$ (with equations $y = c_b x + d_b$ and $y = c_{b+1}x + d_{b+1}$, respectively). In this case, we have $\delta_{i,b} = \delta_{i+1,b+1} = 1$. In order for the two adjacent linear segments to be continuous, they must attain the same value at the breakpoint location r, where $X_i \le r \le X_{i+1}$. That is, for $c_b \ne c_{b+1}$,

$$\begin{aligned}
c_b r + d_b = c_{b+1} r + d_{b+1} \implies r &= \frac{d_{b+1} - d_b}{c_b - c_{b+1}}\\
\implies X_i \le \frac{d_{b+1} - d_b}{c_b - c_{b+1}} &\le X_{i+1}.
\end{aligned}$$

When multiplying through by the denominator, the direction of the inequalities changes depending on its sign. If $c_b - c_{b+1} > 0$ then $\gamma_b = 1$; otherwise $c_b - c_{b+1} < 0$ and $\gamma_b = 0$. The denominator is then distributed accordingly in

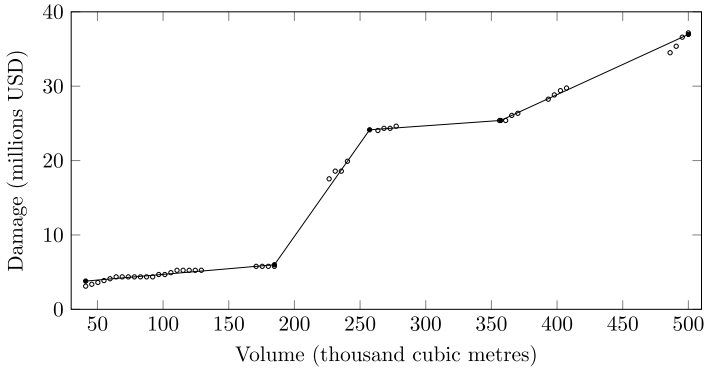


Fig. 3. An example of piecewise linear (PWL) function fitting with four linear segments (and five breakpoints) on the *DebrisFlow* data set.

constraints (3f)-(ei). In particular, depending on the value of the binary variable γ_b, either $\delta^+_{i,b}$ or $\delta^-_{i,b}$ is set to 1. If $\gamma_b = 1$ (by constraint (3d)), then $\delta^+_{i,b} = 1$ and constraints (3f)–(3g) are activated, implying that the gradient decreases between the two consecutive linear segments. Alternatively, if $\gamma_b = 0$ (by constraint (3e)), then $\delta^-_{i,b} = 1$ and constraints (3h)–(3i) are activated, implying the gradient increases. Note that if all of the γ_b variables take the same value, then the given PWL function is either convex or concave. In the case of fitting convex functions, formulation (3) reduces exactly to the model presented by Toriello and Vielma [48].

A similar MILP model to formulation (3) was recently presented by Kong and Maravelias [27], which includes binary variables denoting the first and last data point in each segment, and a novel formulation to ensure the continuity of the segments. A thorough comparison of the two MILP models showed that across three different distance metrics and a series of data sets with differing landscapes, the MILP formulation presented by Rebennack and Krasko [42] (formulation (3)) was faster to find the global optimum in the majority of cases, with fewer constraints and complicating binary variables [52]. Hence, we refer to formulation (3) as the canonical formulation for (continuous) PWLR within our framework.

Fig. 3 shows the result of applying formulation (3) for PWLR (using the sum of absolute differences metric) to the *DebrisFlow* data set.

The *DebrisFlow* data set has appeared often in the PWLR literature, as it is modelled very well by a PWL function. The breakpoint locations can allow us to identify where the relationship between the dependent and independent variables changes, in contrast to a polynomial function. In this particular example, the breakpoints represent volumes where the flow of debris starts damaging certain structures such as houses (e.g., as the volume increases past ∼190,000 cubic metres), or where the debris enters large open areas (e.g., as the volume increases past ∼250,000 cubic metres) [28].

3. Clusterwise piecewise linear regression

For the majority of clustering problems, the information returned simply classifies the data into clusters, and gives no further information about the data in each cluster (other than that they share some degree of similarity, and dissimilarity to other clusters). Hence, the CLR problem, which simultaneously fits a linear regression function to each cluster, is preferable, as it provides information that data in different clusters share different relationships. For clustered data which does not necessarily adhere to a strict linear relationship, performing PWLR on the clustered data can provide even more information about the trends in the data. The clusterwise piecewise linear regression (CPWLR) problem aims at classifying and performing PWLR on sets of data, to provide more detailed information about data which can cover various different trends and patterns.

In this section, we present the first MILP model for CPWLR. This model simultaneously clusters the data and fits each cluster with an optimal PWL function, depending on the given objective function. The MILP model builds upon the framework of formulations (1)–(3) and allows for more accurate modelling of the relationship of the data for the provided clusters. The aim of the model is to distribute B breakpoints between K clusters, such that the objective function (e.g., sum of absolute differences, where the absolute distance between each data point and its associated PWL function is summed) is minimised. This results in a total of $B - K$ linear segments distributed amongst the clusters.

For the following formulation, we are assuming that the number of clusters K (and the total number of breakpoints B) is given in advance. Furthermore, we assume no overlap between the clusters (that is, the clusters are separated by the variable on the x-axis). We leave work on removing these assumptions for future research.

We introduce the following variable:

Binary variables

- Z_b, set to 1 if breakpoint b is the last breakpoint in its cluster.

This formulation reuses many of the constraints seen in formulations (1)–(3). Since the presentation and implementation of this formulation is one of the main contributions of this paper, we present the full model.

$$
\begin{aligned}
(4)\quad \underline{\text{CPWLR}}: \quad \min \quad & \sum_{i=1}^{I} \xi_i && && (4a)\\
\text{s.t.} \quad & Y_i - (c_b X_i + d_b) \leq \xi_i + M_i^a(1-\delta_{i,b}) && \forall i \in [I];\ b \in [B-K] && (4b)\\
& (c_b X_i + d_b) - Y_i \leq \xi_i + M_i^a(1-\delta_{i,b}) && \forall i \in [I];\ b \in [B-K] && (4c)\\
& \sum_{b=1}^{B-K} \delta_{i,b} = 1 && \forall i \in [I] && (4d)\\
& \delta_{i+1,b+1} \leq \delta_{i,b} + \delta_{i,b+1} && \forall i \in [I-1];\ b \in [B-K-1] && (4e)\\
& \delta_{i+1,1} \leq \delta_{i,1} && \forall i \in [I-1] && (4f)\\
& \delta_{i,B-K} \leq \delta_{i+1,B-K} && \forall i \in [I-1] && (4g)\\
& \sum_{b=1}^{B-K-1} Z_b = K-1 && && (4h)\\
& \delta_{i,b} + \delta_{i+1,b+1} + \gamma_b - 2 \leq \delta_{i,b}^{+} + Z_b && \forall i \in [I-1];\ b \in [B-K-1] && (4i)\\
& \delta_{i,b} + \delta_{i+1,b+1} + (1-\gamma_b) - 2 \leq \delta_{i,b}^{-} + Z_b && \forall i \in [I-1];\ b \in [B-K-1] && (4j)\\
& d_{b+1} - d_b \geq X_i(c_b - c_{b+1}) - M_i^2(1-\delta_{i,b}^{+}) && \forall i \in [I-1];\ b \in [B-K-1] && (4k)\\
& d_{b+1} - d_b \leq X_{i+1}(c_b - c_{b+1}) + M_{i+1}^2(1-\delta_{i,b}^{+}) && \forall i \in [I-1];\ b \in [B-K-1] && (4l)\\
& d_{b+1} - d_b \leq X_i(c_b - c_{b+1}) + M_i^2(1-\delta_{i,b}^{-}) && \forall i \in [I-1];\ b \in [B-K-1] && (4m)\\
& d_{b+1} - d_b \geq X_{i+1}(c_b - c_{b+1}) - M_{i+1}^2(1-\delta_{i,b}^{-}) && \forall i \in [I-1];\ b \in [B-K-1] && (4n)\\
& \xi_i \geq 0 && \forall i \in [I] && (4o)\\
& c_b \in [\underline{C}_b, \overline{C}_b] && \forall b \in [B-K] && (4p)\\
& d_b \in [\underline{D}_b, \overline{D}_b] && \forall b \in [B-K] && (4q)\\
& \delta_{i,b} \in \{0,1\} && \forall i \in [I];\ b \in [B-K] && (4r)\\
& \gamma_b, Z_b \in \{0,1\} && \forall b \in [B-K-1] && (4s)\\
& \delta_{i,b}^{+}, \delta_{i,b}^{-} \in [0,1] && \forall i \in [I-1];\ b \in [B-K-1] && (4t)
\end{aligned}
$$

Constraints (4a)-(4g) remain the same as in formulation (3) (for PWLR). Constraints (4h)–(4j) ensure that the continuity requirement given by constraints (4k)–(4n) is not implemented between clusters. If $Z_b = 1$, this implies breakpoint $b \in [B-K]$ is the last in the given cluster. Hence, from constraints (4i)–(4j), it is not necessary to set either $\delta_{i,b}^{+}$ or $\delta_{i,b}^{-}$ to 1 and active the continuity constraints (4k)–(4n). Constraint (4h) ensures that the continuity requirement is only invalid $K-1$ times (only between adjacent clusters). Finally, constraints (4o)–(4t) give the domains of the continuous and binary variables.

The final breakpoint does not require an associated binary variable Z_b (since we know it is the last in its cluster), and we require that each cluster must contain at least two breakpoints to ensure that each cluster is modelled with at least a linear function. Currently, this is imposed by the choice of B and K by the user, where we require that $B \geq 2K$.

Fig. 4 shows the result of applying formulation (4) (using the sum of absolute differences metric) to the *DebrisFlow* data set, with the setting $B = 6$ and $K = 2$ (resulting in four linear segments). Although the fit is similar to that displayed by the PWL function in Fig. 3, we note that the data has also been separated into two distinct clusters. The first cluster is modelled by a simple linear regression function, while the second is modelled by a continuous PWL function, reflecting the staggered relationship between the variables. Any future variables which fall in between the clusters can be *assigned* to a certain cluster by extending the closest linear segments (to see which gives the most natural fit). However, due to the analytical nature of the presented CPWLR model, any variables lying outside the defined clusters are difficult to assign. For those falling between two clusters, comparing the fits can be done somewhat easily through analytical methods (e.g., comparing the distances to each cluster). Alternatively, the model can be re-run to account for new data points.

Compared to the models for fitting non-necessarily continuous (NNC) functions presented by Ngueveu [37] and Codsi et al. [9], formulation (4) does not require the breaks in continuity to occur instantaneously. This allows a distinct

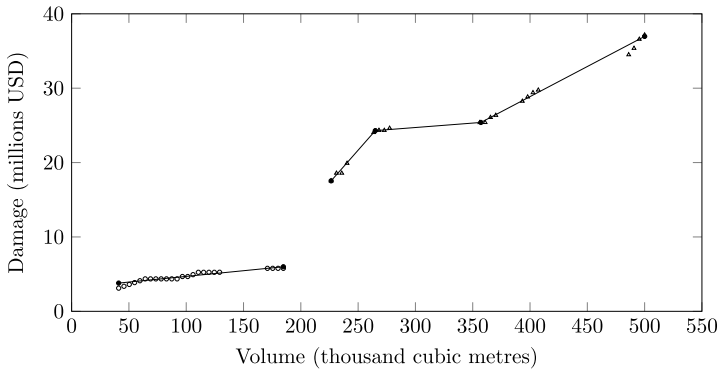


Fig. 4. An example of clusterwise piecewise linear regression with six breakpoints distributed over two clusters on the *DebrisFlow* data set. Different clusters are marked by different shapes.

Table 1
A comparison of the four MILP formulations for CLR [40], oCLR, PWLR [42] and CPWLR.

Model name	(1) CLR	(2) oCLR	(3) PWLR	(4) CPWLR
Reference	Park et al. [40]	This paper	Rebennack and Krasko [42]	This paper
Constraints	(1a)–(1h)	(2a)–(2f)	(3a)–(3l)	(4a)–(4t)
Model type	MILP	MILP	MILP	MILP
# Constraints	$2BI + I$	$B(3I - 1) + 2I - 1$	$B(9I - 7) - 13I + 12$	$B(9I - 7) - I(9 + 4K) + 7K + 6$
# Continuous variables	$2B + I$	$2B + I$	$2BI - 3I + 2$	$2BI - I(1 + 2K) + 2$
# Binary variables	BI	BI	$B(I + 1) - I - 2$	$B(I + 2) - K(2 + I) - 2$
# Big-$\mathcal{M}$ Constraints	$2BI$	$2BI$	$B(6I - 4) - 10I + 8$	$B(6I - 4) - I(6 + 4K) + 4K + 4$

separation between clusters (i.e., heterogeneity), which is sought after in clustering applications. Furthermore, we have seen that a number of distance metrics can be utilised by the model (see Section 2.2 for a discussion on the benefits of the different metrics), as opposed to the formulations for NNC functions requiring the use of the maximum absolute difference metric.

3.1. Model comparison

In Table 1, we compare the four models seen so far: formulations (1) for CLR, formulation (2) for oCLR, formulation (3) for PWLR, and formulation (4) for CPWLR. We consider the total number of functional constraints, big-$\mathcal{M}$ constraints, and continuous and binary variables.

Typically, we have that $I \gg B$ and $I \gg K$. For all formulations, an increase in the number of breakpoints and data points results in the increase of the number of constraints and variables (for realistic values of B and I; typically we consider $2 \leq B \leq 10$ and $I \geq 40$).

While the model for oCLR contains the same number of continuous and binary variables as the model for CLR, it also contains more constraints which limit the assignment of binary variables to clusters. However, this results in reduced solve times as many solutions for the binary variables are now infeasible.

For the model for CPWLR, we require $B \geq 2K$. For a large number of breakpoints (i.e., $B > 2K$), an increase in the number of clusters (for $K > 1$) results in a decrease in the number of constraints and variables. Hence, for a fixed number of breakpoints B, as the number of clusters increases (and as long as $B \geq 2K$ holds), formulation (4) results in fewer constraints and variables than formulation (3) for PWLR. This results in faster programs as the MILP model has fewer decisions to make (there are fewer complicating continuity requirements and more infeasible settings for the binary variables Z_b).

4. Improvements to the mixed-integer linear models

We have presented four mixed-integer linear models in formulations (1)–(4) for the modelling and clustering of bivariate discrete data. These four formulations share a special structure and many similar variables throughout (see Table 1). In particular, these structures include the large number of binary variables and logical implications modelled through big-$\mathcal{M}$ constraints. We can take advantage of this structure and use special MILP formulation techniques from the literature to improve the models [50]. In this section, we consider the implementation of outlier detection and the use of combinatorial Benders decomposition to improve the models. We use formulation (4) (for CPWLR) as the benchmark, yet we note that the implementations described here apply to each of the four formulations (CLR, oCLR and PWLR).

4.1. Outlier detection

Aside from the maximum absolute difference metrics, formulations (1)–(4) are largely robust to outliers on the y-axis. With regard to outliers on the x-axis (i.e., leverage points), it is possible that the clustering models assign each outlier to its own separate cluster (or to a cluster with the nearest data point, consisting of a linear function connecting the two). One methodology to avoid such small clusters is to include the constraint suggested by Park et al. [40], which requires each cluster to have a minimum number of associated data points (see Section 2.1). Alternatively, we can remove clusters which contain fewer than a given number of data points after the model has been completed. Other post-processing outlier removal methods, such as the removal of a given number of data points with the furthest distance to their associated linear segment (the value of ξ_i), can also be implemented. However, it is likely that the model has to be re-solved once the outliers are removed in order to provide a better fit to the remaining data points. Furthermore, this may not always result in optimal clusters or regression functions.

It is also possible to identify sets of possible outlier points before the model is solved. Sudermann-Merx and Rebennack [47] suggested to use statistical models to identify sets of possible outlier points before the model is solved, and then solve the model such that a subset of the possible outlier points is excluded from the final solution. The removed outlier points are found in tandem with the solution of the model. This implementation can be contained within formulation (4), requiring the use of $|O| < I$ new binary variables:

Binary variables

- ρ_i, set to 1 if data point X_i is included in the fit and hence not an outlier.

Let $\mathcal{O}$ be the set of possible outlier points, and let $\mathcal{Q}$ be the total number of outliers to be excluded. The objective function is presented as:

$$\min \sum_{i\in[I]\setminus\mathcal{O}} \xi_i + \sum_{i\in\mathcal{O}} \xi_i \cdot \rho_i,$$

with the added constraint

$$\sum_{i\in\mathcal{O}} \rho_i = |\mathcal{O}| - \mathcal{Q}.$$

In this instance, the objective function is non-linear and leads to increased solve times.

As opposed to these pre- and post-processing methods, we can also implement outlier detection within the MILP model. This strategy is often used in linear regression models using non-linear constraints [21,26,43]. Recent approaches have considered bounded loss functions, where the convexity of the model is sacrificed for scalability [10]. However, we wish to preserve the linearity of the model, and use the implementation presented by Warwicker and Rebennack [53] (similar models have been presented by Bertsimas and Shioda [6]). To this end, we adjust the constraints (4b)–(4c) (and the equivalent constraints in the other models) to calculate the contribution of the given data point only if it is a non-outlier point. We use the same binary variable (ρ_i) to state that the data point X_i is included in the model, and not an outlier point. The implementation we present in formulation (5) can be considered as a linearised version of the constraints introduced by Sudermann-Merx and Rebennack [47], where the set of possible outlier points $\mathcal{O}$ consists of the entire data set (i.e., $\mathcal{O} = I$).

$$\min \quad \sum_{i=1}^{I} \xi_i \tag{5a}$$

$$\text{s.t.} \quad Y_i - (c_b X_i + d_b) \le \xi_i + M_i^1(2 - \delta_{i,b} - \rho_i) \qquad \forall\, i \in [I];\ b \in [B-K] \tag{5b}$$

$$(c_b X_i + d_b) - Y_i \le \xi_i + M_i^1(2 - \delta_{i,b} - \rho_i) \qquad \forall\, i \in [I];\ b \in [B-K] \tag{5c}$$

$$\sum_{i=1}^{I} \rho_i = I - \mathcal{Q} \tag{5d}$$

$$\rho_i \in \{0, 1\} \qquad \forall\, i \in [I] \tag{5e}$$

(4d)–(4t) (5f)

The objective function (5a) calculates the objective function as normal. However, constraints (5b)–(5c) have been adjusted to only calculate the value ξ_i if the data point X_i ($i \in [I]$) is **not** an outlier and associated with the linear segment $b \in [B-K]$ (i.e., when $\rho_i = 1$ and $\delta_{i,b} = 1$, respectively). Otherwise, the value of ξ_i is not calculated (i.e., the constraint does not affect the calculation of ξ_i for the objective function). Constraint (5d) ensures that only $\mathcal{Q}$ data points are considered as outliers (i.e., $\rho_i = 0$ holds for only $\mathcal{Q}$ data points), and constraint (5e) gives the domains of the new binary variables. The remaining constraints match exactly that of formulation (4) (or the respective formulation). In particular, constraints (5b)–(5e) would replace constraints (1b)–(1c) in formulations (1)–(3), and constraints (4b)–(4c) in formulation (4), in order to implement inbuilt outlier detection.

Table 2
A comparison of the four MILP formulations for CLR [40], oCLR, PWLR [42] and CPWLR, with embedded outlier detection.

Model name	CLR^o	oCLR^o	PWLR^o	CPWLR^o
Reference	This paper	This paper	Warwicker and Rebennack [53]	This paper
# Constraints	$2BI + I + 1$	$B(3I - 1) + 2I$	$B(9I - 7) - 13I + 13$	$B(9I - 7) - I(9 + 4K) + 7K + 7$
# Continuous variables	$2B + I$	$2B + I$	$3BI + 2 - 4I$	$2BI - I(1 + 2K) + 2$
# Binary variables	$BI + I$	$BI + I$	$B(I + 1) - 2$	$B(I + 2) - K(2 + I) + I - 2$
# Big-$\mathcal{M}$ Constraints	$2BI$	$2BI$	$B(6I - 4) - 10I + 8$	$B(6I - 4) - I(6 + 4K) + 4K + 4$
Model type	MILP	MILP	MILP	MILP

Regarding the selection of the value for $\mathcal{Q}$, we note that as $\mathcal{Q}$ increases, the objective value will decrease, and lead to models that fit the non-outlier data points better. However, if $\mathcal{Q}$ is set too large, this can lead to loss of information. Hence, testing a number of different values and choosing that which yields a good compromise is advised for practical implementations. Alternatively, adjusting the objective function to account for loss of information if $\mathcal{Q}$ is set too high would lead to balanced solutions; however, identifying such an objective function is a difficult task (see e.g., [47]).

Furthermore, by implementing statistical models to identify possible sets of outliers, it would also be possible to implement formulation (5) using fewer than I new binary variables, and fewer constraints. Sudermann-Merx and Rebennack [47] suggested to consider possible outlier leverage points which are outside the interquartile range of the independent variable; a technique which extends for higher-dimension problems. Further statistical tests could be implemented after an initial estimation of the regression functions to identify possible outlier points with higher accuracy; we leave this problem for future work.

In Table 2, we summarise the current implementation of outlier detection (presented in formulation (5)) for the four MILP formulations within our framework. These models retain their linearity. Each implementation contains an additional binary variable for each data point and an additional constraint. Naturally, the increased number of variables leads to longer solve times, while the removal of outlier data points leads to solutions that better fit the remaining non-outlier data points. The identification of possible sets of outlier points before the model is solved would lead to the addition of fewer variables, yet remove the absolute guarantee of globally optimal solutions.

4.2. Combinatorial Benders decomposition

Combinatorial Benders decomposition (CBD) was introduced by Codato and Fischetti [8] as a tool to remove the dependency of MILPs on logical constraints modelled by big-$\mathcal{M}$ inequalities and complicating binary variables. By separating monolithic MILPs into a master problem consisting of the constraints containing the complicating binary variables, and a sub problem consisting of continuous variables, the goal is that information can be shared between the two decomposed problems. Solutions of the master problem are fed as fixed binary variables into the sub problem, in which an infeasibility is either present or induced. An irreducible infeasible subsystem (IIS) of this sub problem is then used to identify which of the fixed binary variables are causing the infeasibility. In order to find an improving solution, at least one of these fixed binary variables (whose values were present in the constraints appearing in the IIS) must be changed. This information is added back to the master problem in the form of a combinatorial cut, which is then resolved. This process continues until the master problem is infeasible (or the optimality gap is closed), at which point the optimal solution has been found.

Combinatorial Benders cuts are typically added within a branch-and-cut framework [8]. This is a natural choice for a large problem and is much more efficient than repeatedly solving the complicated master problem containing the binary variables in a cyclical framework. At each node of the decision tree, the incumbent solution to the master problem is fed into the sub problem, which is solved to optimality. If the sub problem is infeasible, combinatorial cuts are added into the master problem and the branching continues until more feasible solutions are found. The master problem is never fully solved — once it becomes infeasible, the best upper bound for the sub problem relates to the optimal solution of the overall problem. The advantage of this approach is that all feasible solutions to the binary master problem encountered during the search are verified; however, this means the linear sub problem may be required to be solved many times (this is usually solved quickly by state-of-the-art solvers).

Formulations (1)–(5) contain many complicating binary variables and big-$\mathcal{M}$ constraints. Hence, we can apply CBD to each of the monolithic formulations in order to find speedups and take advantage of their special structure. CBD works best when the objective function contains only few of the inherent continuous variables, so we present the application of CBD for the maximum absolute difference metric only.

The application of CBD on formulation (3) for PWLR (with embedded outlier detection) was shown by Warwicker and Rebennack [53] to be very effective. In particular, it was able to find speedups of up to more than 10,000 times in comparison to the monolithic MILP model. Their implementation benefitted from problem-specific implementations, such as a special branching rule, smart initialisation strategies and the inclusion of strong combinatorial cuts. Formulation (4) (and the inclusion of outlier detection embedded within formulation (5)) contains many of the same constraints and fits within the same framework. Therefore, we expect that CBD is effective also for the CPWLR problem. Formulation (2) for ordered CLR also contains many of the same ordering constraints, so we expect CBD also to be effective on this problem.

However, formulation (1) lacks the ordering constraints on the binary variables, so we expect it to be less effective here. Nevertheless, there may be some improvements due to the removal of the dependency on the big-$\mathcal{M}$ constraints.

To demonstrate how CBD is applied within our framework, we present the reformulation of formulation (4) (for CPWLR), which has been decomposed into a master problem and a sub problem. The master problem consists of constraints which contain only the complicating binary variables. Since none of the binary variables appear in the objective problem, the master problem is a feasibility problem.

$$\underline{\text{MASTER}}: \quad \min \quad \mathcal{X} \in \mathbb{R}^{+} \tag{6a}$$

$$\text{s.t.} \quad \sum_{b=1}^{B-K} \delta_{i,b} = 1 \qquad \forall i \in [I] \tag{6b}$$

$$\delta_{i+1,b+1} \leq \delta_{i,b} + \delta_{i,b+1} \qquad \forall i \in [I-1];\ b \in [B-K-1] \tag{6c}$$

$$\delta_{i+1,1} \leq \delta_{i,1} \qquad \forall i \in [I-1] \tag{6d}$$

$$\delta_{i,B-K} \leq \delta_{i+1,B-K} \qquad \forall i \in [I-1] \tag{6e}$$

$$\sum_{b=1}^{B-K-1} Z_b = K-1 \tag{6f}$$

$$\textit{Combinatorial Benders cuts}\ (8) \tag{6g}$$

$$\delta_{i,b} \in \{0,1\} \qquad \forall i \in [I];\ b \in [B-K] \tag{6h}$$

$$\gamma_b, Z_b, \in \{0,1\} \qquad \forall b \in [B-K-1] \tag{6i}$$

Since the monolithic formulation does not contain any binary variables in the objective function, we instead look to optimise some real-valued variable $\mathcal{X}$, which is an auxiliary variable used to estimate the true value of the objective function through cuts. Initially, we set its value to 0 as an initial lower bound on the true objective function value (since we know that the true objective function value is always nonnegative in a feasible solution). Constraint (6g) contains the combinatorial Benders cuts, which provide combinatorial information on infeasible solutions to the master problem.

After the master problem is solved (i.e., a feasible solution has been identified), it provides fixed binary variables to feed into the sub problem ($\hat{\delta}_{i,b}$, $\hat{\gamma}_b$ and $\hat{Z}_b$). If the sub problem is solved to optimality, infeasibility is then induced by an added constraint based on the current upper bound on the newly found optimal solution (UB); otherwise, the sub problem is infeasible based on the current fixed binary variables. In either case, the infeasibility provides combinatorial information on which of the binary variables should be altered.

We now present the sub problem, in which the constraints have been rearranged such that the right hand side contains only constant values (and no variables).

$$\underline{\text{SUB}}(\hat{\delta},\hat{\gamma},\hat{Z}): \quad \min \quad \xi \tag{7a}$$

$$\text{s.t.} \quad \xi \leq \text{UB} - \epsilon \tag{7b}$$

$$\xi + (c_b X_i + d_b) \geq -Y_i - M_i^1(1-\hat{\delta}_{i,b}) \qquad \forall i \in [I];\ b \in [B-K] \tag{7c}$$

$$\xi - (c_b X_i + d_b) \geq Y_i - M_i^1(1-\hat{\delta}_{i,b}) \qquad \forall i \in [I];\ b \in [B-K] \tag{7d}$$

$$\delta^{+}_{i,b} \geq \hat{\delta}_{i,b} + \hat{\delta}_{i+1,b+1} + \hat{\gamma}_b - \hat{Z}_b - 2 \qquad \forall i \in [I-1];\ b \in [B-K-1] \tag{7e}$$

$$\delta^{-}_{i,b} \geq \hat{\delta}_{i,b} + \hat{\delta}_{i+1,b+1} + (1-\hat{\gamma}_b) - \hat{Z}_b - 2 \qquad \forall i \in [I-1];\ b \in [B-K-1] \tag{7f}$$

$$(d_{b+1} - d_b) - X_i(c_b - c_{b+1}) - M_i^2 \delta^{+}_{i,b} \geq -M_i^2 \qquad \forall i \in [I-1];\ b \in [B-K-1] \tag{7g}$$

$$X_{i+1}(c_b - c_{b+1}) - (d_{b+1} - d_b) - M_{i+1}^2 \delta^{+}_{i,b} \geq -M_{i+1}^2 \qquad \forall i \in [I-1];\ b \in [B-K-1] \tag{7h}$$

$$X_i(c_b - c_{b+1}) - (d_{b+1} - d_b) - M_i^2 \delta^{-}_{i,b} \geq -M_i^2 \qquad \forall i \in [I-1];\ b \in [B-K-1] \tag{7i}$$

$$(d_{b+1} - d_b) - X_{i+1}(c_b - c_{b+1}) - M_{i+1}^2 \delta^{-}_{i,b} \geq -M_{i+1}^2 \qquad \forall i \in [I-1];\ b \in [B-K-1] \tag{7j}$$

$$\xi \geq 0 \tag{7k}$$

$$d_b \in [\underline{D}_b, \overline{D}_b] \qquad \forall b \in [B-K] \tag{7l}$$

$$c_b \in [\underline{C}_b, \overline{C}_b] \qquad \forall b \in [B-K] \tag{7m}$$

$$\delta^{+}_{i,b}, \delta^{-}_{i,b} \in [0,1] \qquad \forall i \in [I-1];\ b \in [B-K-1] \tag{7n}$$

Constraint (7b) induces infeasibility into the model if the solution to the objective function is not improved (if it is improved, the model is run again with the new value of UB until infeasibility is induced). The value of ε should be sufficiently small, but greater than 0 (in experimental results in Section 5, we use a value of 0.001). This is to ensure that a strictly improving solution is sought in each sub problem. An IIS of the continuous sub problem is then sought which provides combinatorial information regarding the fixed binary variables. In order to find IISs for the experimental results

we present, we used the conflict refined inbuilt within CPLEX, which allows us to identify the constraints that are causing the infeasibility. We discuss this in more detail in Section 5.4.

In particular, the (fixed) binary variables appearing in the IIS are known to lead to an infeasible solution, so it is necessary for at least one of them to change in order to find a new improving solution. The combinatorial cuts (constraint (6g)) take the following form. Let R denote the set of binary variables appearing in the constraints of the IIS (that is, the fixed binary variables with non-zero coefficients appearing in at least one constraint of the IIS).

$$\sum_{\delta_{i,b}\in R:\hat{\delta}_{i,b}=1}(1-\delta_{i,b})+\sum_{Z_b\in R:\hat{Z}_b=0}Z_b+\sum_{\gamma_b\in R:\hat{\gamma}_b=1}(1-\gamma_b)+\sum_{\gamma_b\in R:\hat{\gamma}_b=0}\gamma_b\geq 1. \tag{8}$$

The knapsack-style constraints on the $\delta_{i,b}$ variables (constraint (6b)), as well as constraints (7c)–(7d) which ensure that infeasibility is only possible when $\hat{\delta}_{i,b}=1$, imply that changing any of these variables that are fixed to 1 will result in a new structure of $\delta_{i,b}$ values, and a new solution. Further, the knapsack-style constraints on the Z_b variables (constraint (6f)), alongside constraints (7e)–(7f) which ensure that infeasibility is only possible when $\hat{Z}_b=0$, imply that changing any of these variables that are fixed to 0 will result in a new structure of Z_b variables, and a new solution. There are no such limitations on the γ_b variables, and so any γ_b variables appearing in the IIS should be changed.

When the master problem becomes infeasible (due to the combinatorial cuts), the solution pertaining to the current upper bound (UB) is the ϵ-optimal solution. A full pseudocode of the CBD approach for CPWLR is presented in Section C of the Appendix.

Tailored improvements taking advantage of problem-specific knowledge can also be implemented into each of the CBD models to find speedups [53]. Firstly, for each infeasible sub problem, there can exist multiple different IISs, which each give rise to a different combinatorial cut (of the form of constraint (8)). Finding each of these IISs, and engineering them such that stronger cuts are found, leads to more stronger cuts in each iteration, eliminating many more infeasible solutions and speeding up the optimisation process. For example, if a constraint of the form (7b)–(7c) occurs in the IIS, this means that one $\delta_{i,b}$ variable is causing infeasibility, and should be changed in the resulting combinatorial cut. On the other hand, if a constraint of the form (7g)–(7j) occurs in the IIS, this means one of two $\delta_{i,b}$ variables, or a γ_b variable, or a Z_b variable is causing the infeasibility (implicitly through constraints (7e)–(7f)), and at least one of them should be changed in the resulting combinatorial cut. The former cuts are stronger, since they state one particular variable should change, which impacts the solution through the constraints (6b)–(6e). However, the latter cuts do not give such precise information. Hence, prioritising the removal of weaker cuts from the IIS first will lead to stronger cuts being found.

Secondly, smart initialisation techniques based on results from previous solutions (or heuristic approximations), which give approximate values for the starting value of UB, have been shown to lead to faster runtimes. For example, approximating the clusters beforehand and running simple linear regression on each cluster would lead to a valid initial upper bound. Finally, tailored branching rules, which take advantage of the special structure of the $\delta_{i,b}$ variables, have also been shown to lead to faster runtimes and can save memory during the optimisation process [53].

For CLR, the master problem only consists of constraint (1d) and the combinatorial cuts, while the sub problem contains constraints (1b)–(1c) with the fixed values of the δ variables. For oCLR, the ordering constraints (2c)–(2e) are also present in the master problem, which immediately eliminates many possible infeasible solutions. The implementation of CBD for PWLR has been discussed in detail by Warwicker and Rebennack [53].

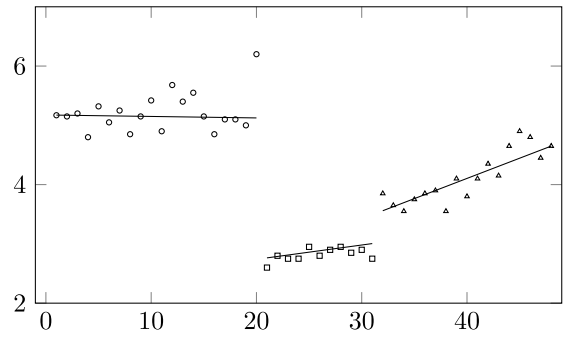
5. Computational results

In order to assess the effectiveness of each of the formulations (1)–(4) and the improvements presented in Section 4, we implemented each of the models in C++ embedded within IBM ILOG-Cplex version 20.0.1., using standard solver settings (unless explicitly stated otherwise). The experiments in this section were run on an Intel 3.00 GHz machine with 16 GB of RAM and 6 cores.

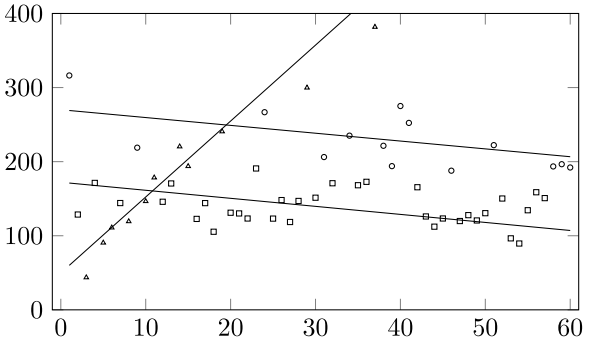
5.1. Data sets

We ran computational experiments on five bivariate data sets, each taken from real world data and presenting a variety of different function landscapes. These data sets are often used for data fitting analyses, including CLR, PWLR and clustering.

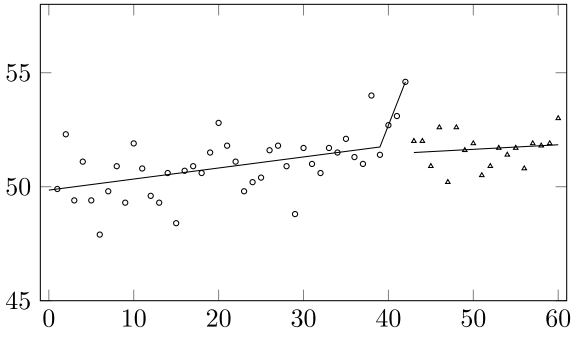
1. *DebrisFlow* (Figs. 1–4) - Data showing the expected economic damage of a real world location as a function of the volume of the debris flow ($I=44$) [28,34].
2. *Medication* (Fig. 5(a)) — Data showing the mean number of monthly prescriptions of a certain medicine, before and after a policy change ($I=48$) [51].
3. *DailyDemand* (Fig. 5(b)) — Data showing the number of non-urgent daily orders for a logistics company ($I=60$) [16].
4. *NHTemp* (Fig. 5(c)) — Data showing the mean annual temperature in New Haven, a city in the USA ($I=60$) [35].
5. *Paperweight* (Fig. 5(d)) — A large data set showing a measure of paper density over time from a large paper manufacturer ($I=231$) [31].



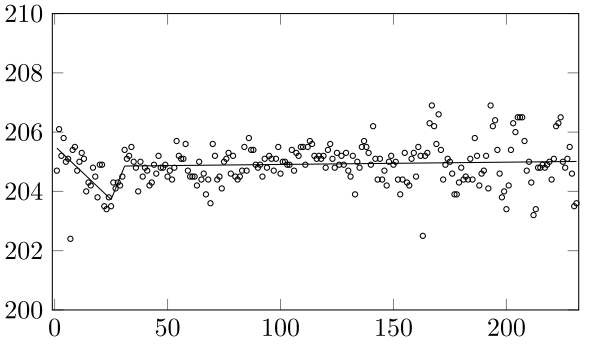
(a) Medication ($I = 48$) fitted with oCLR with three linear segments.



(b) DailyDemand ($I = 60$) fitted with CLR with three linear segments.



(c) NHTemp ($I = 60$) fitted with CPWLR with three linear segments distributed over two clusters.



(d) Paperweight ($I = 231$) fitted with PWLR with three linear segments.

Fig. 5. Bivariate data sets modelling using the Sum of Absolute Differences metric. Different clusters are marked by different shapes.

5.2. Runtime results for the monolithic formulations

We first present runtime results for the monolithic formulations (1)–(4) without inbuilt outlier detection or the use of combinatorial Benders decomposition.

CLR and oCLR

In Table 3, we present runtime results for CLR (formulation (1)) over the five data sets, using the maximum absolute differences metric (noting that similar results hold for the other distance metrics discussed in Section 2.1). We present, for each result, the runtime (in seconds) and the optimal solution found by the model.

From Table 3, we see how the runtime increases dramatically as the number of linear segments increases. Due to the lack of restrictions on the binary variables, there are a large number of feasible solutions. Naturally, as the number of segments increases, so does the accuracy of the model. We believe that heuristic approaches may be preferable for finding sufficiently good solutions for CLR models with a large number of segments.

In Table 4, we present comparative results for CLR with its ordered variant (oCLR). In particular, we present the number of segments required by oCLR in order to match the objective function value found by the CLR model (presented in Table 3). We further present the runtime required to find such a solution, and the speedup (as a ratio of CLR:oCLR).

Notably, from Table 4, we note that the structure of the data can lead to vastly different results. For data sets which present an *ordered* pattern (such as the *DebrisFlow* and *Medication* data sets), oCLR is able to improve upon (or match) the runtime required by CLR in every case. For CLR fitting 7 segments across these two data sets, oCLR is respectively over 11,000 and over 14,000 times faster to find a solution of the same (or better) objective function value, despite requiring more linear segments.

For the data sets which do not present a noticeable ordered pattern (such as the *DailyDemand*, *NHTemp* and *Paperweight* data sets), oCLR requires significantly more linear segments to find a solution of similar quality as CLR. Although these solutions may be found faster (for CLR with larger numbers of segments), the large amount of segments required by oCLR suggests that the fitting is somewhat random (and can be classed as overfitting).

Table 3
Runtime results for CLR — Maximum absolute difference.

CLR: Linear segments		2	3	4	5	6	7	8
Debris flow	Runtime	0.1	0.1	0.5	8.4	71.7	15,351.5	[a]
($I = 44$)	Solution	2.18	0.78	0.41	0.27	0.19	0.12	[a]
Medication	Runtime	0.1	0.2	3.3	23.2	362.0	76,599.1	[a]
($I = 48$)	Solution	0.57	0.41	0.23	0.16	0.11	0.09	[a]
DailyDemand	Runtime	0.1	0.3	13.3	79.9	393.1	[a]	
($I = 60$)	Solution	87.23	47.27	35.14	25.29	17.11	[a]	
NHTemp	Runtime	0.1	0.7	16.3	26.2	87.2	50,192.2	[a]
($I = 60$)	Solution	1.21	0.82	0.54	0.40	0.30	0.24	[a]
Paperweight	Runtime	0.2	16.2	115.9	4074.9	[a]		
($I = 231$)	Solution	1.08	0.71	0.51	0.38	[a]		

[a]Indicates the 86 400 s time limit has been exceeded.

Table 4
Runtime results for oCLR to match the results from Table 3 — Maximum absolute difference.

oCLR: Linear segments		2	3	4	5	6	7
	Segments required	2	3	6	8	8	12
Debris flow	Solution	2.18	0.78	0.33	0.19	0.19	0.079
($I = 44$)	Runtime	0.1	0.1	0.1	0.3	0.3	1.3
	Improvement	1.00	1.00	5.00	28.00	239.00	11,808.85
	Segments required	2	5	10	12	17	18
Medication	Solution	0.57	0.39	0.19	0.16	0.11	0.083
($I = 48$)	Runtime	0.1	0.2	1.0	1.3	7.5	5.3
	Improvement	1.00	1.00	3.30	17.85	48.27	14,452.66
	Segments required	5	13	17	21	23	
DailyDemand	Solution	86.6	44.8	33.2	23.3	16.1	
($I = 60$)	Runtime	0.2	3.0	21.1	13.9	11.1	
	Improvement	0.50	0.10	0.63	5.75	35.41	
	Segments required	10	16	19	20	24	28
NHTemp	Solution	1.15	0.73	0.53	0.37	0.28	0.23
($I = 60$)	Runtime	1.6	8.1	31.7	26.5	22.9	23.0
	Improvement	0.06	0.09	0.51	0.99	3.81	2182.27
	Segments required	13	19	33	48		
Paperweight	Solution	0.96	0.71	0.50	0.37		
($I = 231$)	Runtime	31.2	131.3	2560.0	10,117.8		
	Improvement	0.01	0.12	0.05	0.40		

While the performance difference is dependent on the objective function itself (which does not explicitly measure the suitability of the presented model), we note that oCLR is much faster due to the enforced structure on the binary variables. However, this can come at a loss of problem information, especially when the data set does not provide an ordered pattern. In general, we suggest that practitioners should only implement oCLR if the data set being modelled has such an ordering (or if such a pattern can be quantified), due to the significant speedups presented by the model.

PWLR and CPWLR

We now compare the solution quality and runtime of the formulations for PWLR (formulation (3)) and CPWLR (formulation (4)). Although the overall aims of PWLR and CPWLR are different, we can still compare the quality of the formulations, as they are both assigning data to linear segments. Naturally, while PWLR does not attempt to cluster the data, and ensures continuity of the linear segments throughout, it takes longer to find a solution for a given number of linear segments. CPWLR, however, must decide on the locations of the clusters and how to optimally assign piecewise linear segments to the clusters. PWLR acts exactly as the CPWLR model for fitting one cluster.

Firstly, in Table 5, we present the runtime results and the optimal solutions found by the PWLR model (formulation (3)), in order to benchmark the quality of the solutions and the efficiency of the model. We use the sum of absolute differences metric. Note that the number of linear segments is one fewer than the number of breakpoints in the final PWLR function.

As the number of linear segments increases, so does the complexity of the model and the runtime required to find the optimal solution. PWLR can be considered as a variant of CPWLR, where only one cluster is sought. If a clustering model is not required, PWLR is advantageous over standard linear regression models, as it can model the change in the relationship between the variables. Further experimental results benchmarking the performance of formulation (3) across different distance metrics have been presented by Rebennack and Krasko [42],Warwicker and Rebennack [52] and Warwicker and Rebennack [53].

Table 5
Runtime results for PWLR — Sum of absolute differences.

PWLR: Linear segments		4	5	6	7	8	9	10	11	12	13	14
DebrisFlow	Runtime	0.3	0.9	6.5	144.7	1938.9	263.6	1084.0	1635.9	4621.4	43,620.6	[a]
($I = 44$)	Solution	10.75	8.85	7.20	6.20	4.55	3.90	2.71	2.05	1.73	1.40	[a]
Medication	Runtime	0.9	5.2	20.0	414.9	835.5	5799.3	56,526.7	[a]			
($I = 48$)	Solution	8.79	7.86	7.03	6.69	6.04	5.70	5.30	[a]			
DailyDemand	Runtime	3.1	22.8	727.9	1829.0	51,874.8	[a]					
($I = 60$)	Solution	2295	2166	2003	1828	1713	[a]					
NHTemp	Runtime	5.0	58.1	1660.0	11,419.3	[a]						
($I = 60$)	Solution	41.92	40.66	38.80	36.88	[a]						
Paperweight	Runtime	945.7	43,201.6	[a]								
($I = 231$)	Solution	110.77	107.77	[a]								

[a]Indicates the 86 400 s time limit has been exceeded.

Table 6a
Runtime results for CPWLR — Sum of absolute differences.

CPWLR: Linear segments		4			5			6			7		
Clusters		2	3	4	2	3	4	2	3	4	2	3	4
DebrisFlow	Runtime	0.3	0.2	0.1	0.5	0.6	0.6	6.2	3.2	0.8	42.2	7.2	4.0
($I = 44$)	Solution	9.12	8.87	8.72	7.21	6.48	6.41	6.21	4.56	4.44	4.55	3.93	3.31
Medication	Runtime	0.6	0.4	0.2	3.7	3.3	0.7	18.1	8.0	11.0	76.6	37.4	21.8
($I = 48$)	Solution	7.86	6.76	6.65	6.76	6.42	6.02	6.42	5.77	5.67	5.77	5.43	5.03
DailyDemand	Runtime	4.0	1.3	0.4	15.0	7.1	5.8	39.8	44.2	32.2	560.0	142.5	105.6
($I = 60$)	Solution	2201	2078	2078	2016	1925	1865	1846	1786	1691	1734	1612	1589
NHTemp	Runtime	4.2	3.1	0.7	27.0	21.6	9.7	283.6	82.2	69.4	2926.1	1247.8	861.7
($I = 60$)	Solution	40.81	39.87	38.70	39.35	37.94	36.44	37.24	35.89	34.33	35.39	33.83	32.87
Paperweight	Runtime	144.7	75.0	21.7	15,836.4	2239.6	812.9	[a]	17,807.9	29,264.1	[a]	[a]	[a]
($I = 231$)	Solution	108.71	106.37	105.24	104.49	102.23	101.35	[a]	99.13	97.21	[a]	[a]	[a]

[a]Indicates the 86 400 s time limit has been exceeded.

Table 6b
Runtime results for CPWLR — Sum of absolute differences.

CPWLR: Linear segments		8			9			10			11		
Clusters		2	3	4	2	3	4	2	3	4	2	3	4
DebrisFlow	Runtime	184.1	25.4	21.6	114.2	125.2	72.0	304.6	104.7	337.2	1574.9	950.8	966.4
($I = 44$)	Solution	3.91	2.72	2.65	2.71	2.31	2.07	2.07	1.84	1.77	1.80	1.68	1.47
Medication	Runtime	610.5	305.5	134.5	4751.3	1564.7	1376.8	54,532.3	22,185.3	9093.1	[a]	[a]	50,128.4
($I = 48$)	Solution	5.43	5.03	4.76	5.03	4.76	4.43	4.76	4.43	4.16	[a]	[a]	3.90
DailyDemand	Runtime	4641.7	563.8	383.6	31,091.5	5515.3	4537.6	77,189.8	36,088.6	52,246.2			
($I = 60$)	Solution	1602	1531	1478	1521	1424	1358	1414	1343	1277			
NHTemp	Runtime	64,460.2	14,406.4	8235.2	[a]	[a]	42,838.6						
($I = 60$)	Solution	33.47	32.37	31.23	[a]	[a]	29.36						

[a]Indicates the 86 400 s time limit has been exceeded.

In Table 6a–6b, we present the respective runtime results and optimal solutions for the CPWLR model (formulation (4)). For each number of linear segments, we present results for the model selecting between two and four clusters. Note that the respective results for one cluster can be taken from Table 5.

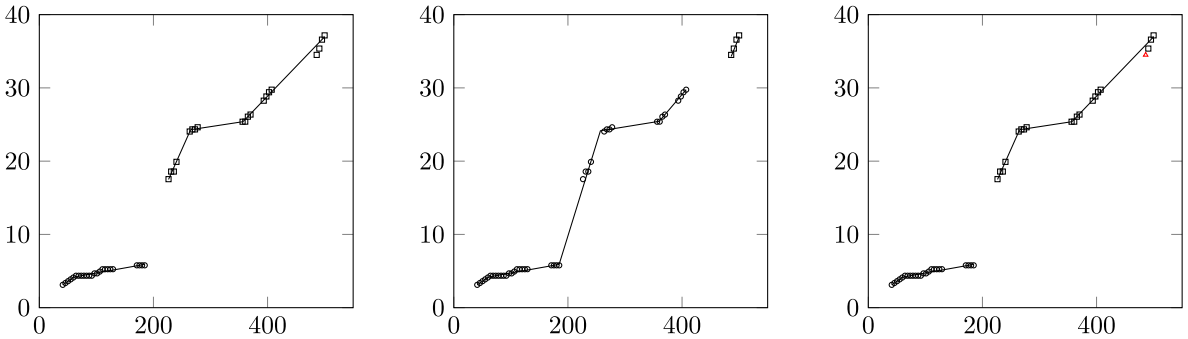
We can see that as the number of linear segments increases, so does the complexity of the model and hence, the runtime required to find the optimal solution. Counterintuitively, perhaps, is that for a given number of linear segments, increasing the number of clusters required by the model decreases the complexity of the model as well as improving the solution. We have seen in Table 1 that as the number of clusters K increases, the number of constraints and variables decreases.

As a general guide, we believe that CPWLR can be more effective than PWLR, since it removes some of the complicating continuity requirements, while still modelling the change in the relationship between the variables of the data. Naturally, if the number of clusters is too large, this can lead to overfitted data. The model for CPWLR requires the number of breakpoints and clusters as input. If such information is available, or able to be calculated efficiently through heuristic or hierarchical techniques (see e.g., [36]), then CPWLR can be very effective.

Table 7
Runtime results for CPWLR with outliers – Sum of absolute differences – 2 Clusters.

CPWLR: Linear segments		4			5			6			7		
Outliers		1	2	3	1	2	3	1	2	3	1	2	3
DebrisFlow	Runtime	2.6	6.7	27.8	7.1	73.4	171.8	48.0	250.1	3083.8	169.1	2021.4	12,942.4
($I = 44$)	Solution	7.79	6.81	6.11	5.87	4.89	4.31	5.07	3.91	3.54	3.91	3.05	2.47
Medication	Runtime	10.3	42.8	480.5	120.0	958.3	7163.4	563.3	7017.8	43,405.1	2434.3	84,364.1	[a]
($I = 48$)	Solution	6.65	6.11	5.65	6.20	5.75	5.31	5.67	5.12	4.69	5.21	4.78	[a]
DailyDemand	Runtime	41.0	596.8	2813.8	332.6	2234.7	10,381.0	646.6	12,374.3	67,044.6	13,128.2	54,055.8	[a]
($I = 60$)	Solution	1935	1747	1613	1777	1584	1446	1667	1470	1327	1539	1373	[a]
NHTemp	Runtime	99.2	1435.8	11,132.7	1619.4	20,566.7	[a]	6954.0	[a]	[a]	[a]	[a]	[a]
($I = 60$)	Solution	38.29	35.94	33.54	36.59	34.38	[a]	34.69	[a]	[a]	[a]	[a]	[a]
Paperweight	Runtime	[a]											
($I = 231$)	Solution	[a]											

[a]Indicates the 86 400 s time limit has been exceeded.



(a) 2 Clusters, 5 Segments, 0 Outliers OF Value = 7.206

(b) 2 Clusters, 6 Segments, 0 Outliers OF Value = 6.212.

(c) 2 Clusters, 5 Segments, 1 Outlier OF Value = 5.872.

Fig. 6. Illustration on the *DebrisFlow* data set using the sum of absolute differences metric showing the removal of an outlier point can affect the objective function value by more than the inclusion of an additional linear segment. Different clusters (and outlier points) are marked by different shapes.

5.3. Outlier detection

We now analyse the effect of the implementation of inbuilt outlier detection (formulation (5)) into the MILP model for CPWLR (formulation (4)). Due to the increase in the number of binary variables, as the number of outliers increases, so does the runtime required to find the optimal solution. We note that the results presented here also reflect the performance of the implementation of outlier detection into formulations (1)–(3).

Table 7 shows the effect of including between 1 and 3 outliers into the model for CPWLR (Table 6a shows the respective results with 0 outliers). We analyse over the five data sets, each modelled with two clusters, using the sum of absolute differences metric. We note that similar results hold for the other distance metrics.

We can see that the complexity and runtime increases as the number of outliers increases. However, increasing the number of outliers in the formulation can dramatically improve the solution quality (with regards to the chosen objective function). In many cases, for each extra outlier included, the value of the objective function decreases by more than the maximum absolute difference (i.e., $\max_i\{\xi_i\}$). That is, not taking the outlier points into account when fitting the remaining data points leads to models that fit the remaining data points better, as opposed to solving for all data points and removing those contributing the most to the objective function.

Although these data sets do not necessarily contain outliers, the decrease in objective function value is noticeable for increased numbers of outliers, and can have a greater impact on the solution quality than the inclusion of more linear segments. We illustrate an example of this phenomenon in Fig. 6. Regarding the selection of the number of outlier points to remove, we advise testing for a number of different values, and choosing that with the best tradeoff between solution quality and potential loss of information. Of course, testing for many values can lead to longer solution times; hence, decomposition approaches can also be implemented in tandem with inbuilt outlier detection.

5.4. Combinatorial Benders decomposition

We now analyse the effect of the implementation of combinatorial Benders decomposition (formulations (6)–(7)) on the MILP model for CPWLR, to see if speedups can be found. Combinatorial Benders decomposition has already been shown

Table 8a
Runtime results for CPWLR with combinatorial Benders decomposition — Maximum absolute difference.

CPWLR: Linear segments		6			7			8			9		
Clusters		2	3	4	2	3	4	2	3	4	2	3	4
DebrisFlow	Monolith	0.8	0.6	0.4	2.1	1.0	0.8	2.6	2.1	2.3	7.4	4.2	2.0
($I = 44$)	CBD	2.1	2.3	2.0	4.9	3.7	3.3	3.8	6.8	5.1	6.7	7.5	5.3
Speedup		0.38	0.26	0.20	0.43	0.27	0.24	0.68	0.31	0.45	1.10	0.56	0.38
Solution		0.42	0.39	0.33	0.39	0.30	0.29	0.30	0.26	0.19	0.26	0.19	0.16
Medication	Monolith	1.8	0.8	0.6	5.0	5.2	1.3	13.6	2.9	2.0	8.1	8.1	11.0
($I = 48$)	CBD	1.5	1.9	1.6	1.9	3.8	1.9	4.8	4.7	4.5	4.1	5.5	5.6
Speedup		1.20	0.42	0.38	2.63	1.37	0.68	2.83	0.62	0.44	1.98	1.47	1.96
Solution		0.39	0.37	0.33	0.37	0.33	0.27	0.33	0.27	0.27	0.27	0.27	0.25
DailyDemand	Monolith	1.6	2.1	1.2	3.6	3.4	2.8	14.3	13.0	10.9	91.2	25.9	21.4
($I = 60$)	CBD	2.3	3.3	2.6	2.7	3.6	3.3	4.6	4.4	10.0	9.0	19.4	8.3
Speedup		0.70	0.64	0.46	1.33	0.94	0.85	3.11	2.95	1.09	10.13	1.34	2.58
Solution		86.6	80.1	78.8	80.1	78.8	75.3	78.8	75.3	71.1	75.3	71.1	66.5
NHTemp	Monolith	31.2	5.1	1.0	28.4	7.4	3.3	80.4	29.3	13.9	147.5	327.3	222.0
($I = 60$)	CBD	83.3	8.3	2.3	34.6	8.8	5.2	14.8	11.3	7.0	14.6	15.5	14.9
Speedup		0.37	0.61	0.43	0.82	0.84	0.63	5.43	2.59	1.99	10.10	21.12	14.90
Solution		1.89	1.43	1.40	1.43	1.40	1.38	1.40	1.38	1.36	1.38	1.36	1.33
Paperweight	Monolith	56.7	25.2	9.0	1195.2	3383.4	301.3	[a]	49,324.4	34,047.3			
($I = 231$)	CBD	115.2	33.4	8.0	406.8	141.0	242.9	*	2732.1	1156.9			
Speedup		0.49	0.75	1.13	2.94	24.00	1.24		18.05	29.43			
Solution		1.65	1.61	1.55	1.60	1.54	1.53		1.52	1.49			

[a]Indicates the 86 400 s time limit has been exceeded.
*Appears for the Paperweight data set, CBD, 8 segments, 2 clusters.

to be effective for formulation (3) for PWLR [53], when using the maximum absolute difference metric. We implement CBD in the model within the same way, using lazy callbacks and the conflict refiner within CPLEX to calculate multiple IISs. In particular, the conflict refiner works by assigning preferences to each constraint within the sub problem. All preferences are initially set to 1 (meaning the given constraint should be considered for the IIS). In order to find multiple IISs for a given sub problem (leading to multiple cuts), we set the preference of one of the constraints appearing in the initial IIS to -1 (meaning it should no longer be considered when finding further IISs) and find a new IIS. Once no more IISs of the sub problem can be generated, the process ends, and the combinatorial cuts relating to each IIS are added to the master problem.

Within CPLEX, we further removed the presolve for the master problem and set the number of threads to 1 (enforcing sequential operation).

Note that, in contrast to the monolithic formulation, the CBD approach does not currently yield similar results for the other metrics — we leave this problem for future work.

Table 8a–8b shows a comparison between the monolithic MILP for CPWLR (formulation (4)) and the implementation of CBD (formulation (6)–(7)), over the five data sets. We use the maximum absolute difference metric, and consider the effect for varying numbers of linear segments and clusters.

We firstly note that the runtime required by the monolithic approach using the maximum absolute difference metric is much faster in comparison to using the sum of absolute differences metric (shown in Table 6a–6b). Hence, we are able to analyse models with a much larger number of linear segments within the presented time limit. However, since only smaller data sets are considered, the solution quality does not increase too much when considering such a large number of segments. In particular, we see only small improvements in the objective function value for each further segment included, while the increase in runtime is noticeable.

We can further see that combinatorial Benders decomposition can have a pronounced effect on the runtime required to find an optimal solution to the CPWLR problem. In particular, as the problem complexity increases (by increasing the total number of linear segments), the time required by the monolithic CPWLR model (formulation (4)) increases at a much faster rate than that of the CBD model. While the MILP model is faster for smaller problems (such as *DebrisFlow* with fewer than 10 segments) — this changes as the number of linear segments increases. For 11 of the presented cases, we see that implementing CBD can lead to speedups of over 100 times that of the original formulation, up to a maximum speedup of over 780.

Fig. 7 presents the fraction of instances from Table 8a–8b that are solved within a given time limit, for between 2 and 4 clusters. The CBD approach is able to solve more instances within a time limit of 4 s, 8 s and 6 s for modelling the data with 2, 3, and 4 clusters respectively. In each case, it is clear that CBD presents a considerable advantage for any time

Table 8b
Runtime results for CPWLR with combinatorial Benders decomposition — Maximum absolute difference.

CPWLR: Linear segments		10			11			12			13		
Clusters		2	3	4	2	3	4	2	3	4	2	3	4
DebrisFlow	Monolith	13.0	3.2	4.3	18.0	12.1	13.5	72.2	67.9	11.5	39.6	23.3	49.1
($I = 44$)	CBD	9.2	6.1	4.5	11.1	6.7	8.5	8.7	11.3	7.2	9.1	14.6	13.6
Speedup		1.41	0.52	0.96	1.62	1.81	1.59	8.30	6.01	1.60	4.35	1.60	3.61
Solution		0.19	0.16	0.16	0.16	0.16	0.14	0.16	0.14	0.12	0.14	0.10	0.089
Medication	Monolith	29.9	54.0	13.9	93.0	120.6	443.4	246.2	876.5	817.5	1705.3	2571.8	4527.6
($I = 48$)	CBD	6.0	6.6	5.4	9.5	10.5	18.2	12.0	15.7	16.3	14.8	17.5	23.0
Speedup		4.98	8.18	2.57	9.79	11.49	24.36	20.52	55.83	50.15	115.22	146.96	196.85
Solution		0.27	0.25	0.24	0.25	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.21
DailyDemand	Monolith	94.2	110.6	49.9	425.6	257.5	363.4	1916.4	2346.3	7775.3	1995.5	7023.7	8239.2
($I = 60$)	CBD	23.0	43.8	19.0	39.8	27.9	24.9	31.6	27.9	30.9	35.1	42.3	71.0
Speedup		4.10	2.53	2.63	10.69	9.23	14.59	60.65	84.10	251.63	56.85	166.04	116.05
Solution		71.1	66.5	65.4	66.5	65.4	64.0	65.4	64.0	63.0	64.0	62.9	61.6
NHTemp	Monolith	958.3	2213.0	313.7	8033.7	9939.8	1304.4	31,531.6	9444.8	157.4	75,944.1	11,618.6	1644.2
($I = 60$)	CBD	16.2	17.3	28.6	28.1	39.7	79.5	40.0	101.0	36.5	96.3	132.3	36.0
Speedup		59.15	127.92	10.97	285.90	250.37	16.41	788.29	93.51	4.31	788.62	87.82	45.67
Solution		1.35	1.33	1.31	1.33	1.31	1.29	1.31	1.29	1.17	1.29	1.17	1.15

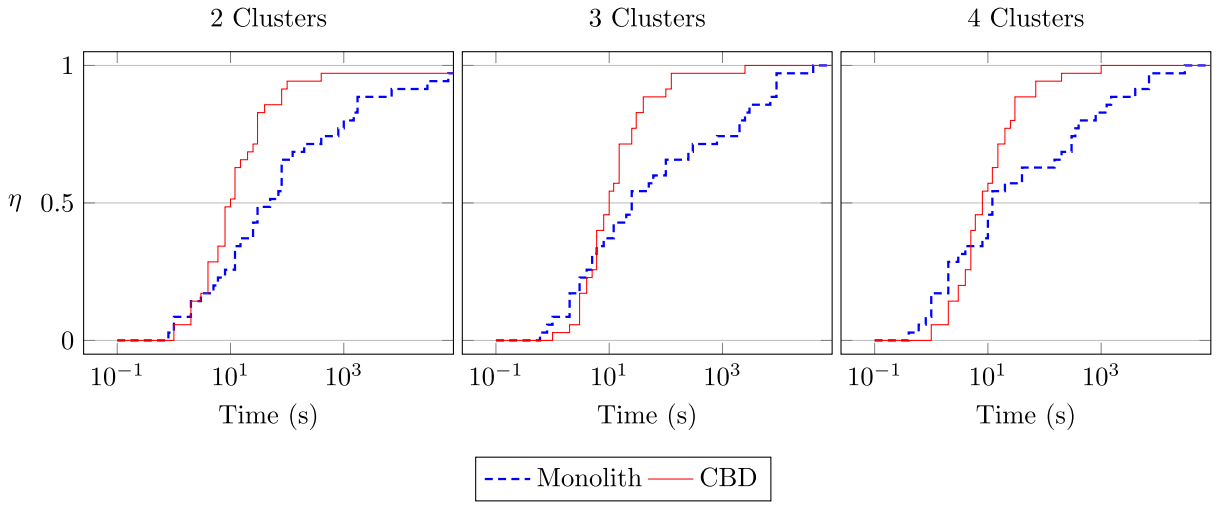


Fig. 7. η refers to the fraction of instances solved to optimality within the given time.

limit above 10 s. In particular, as the problem being solved becomes harder (in terms of complexity), the advantages of the CBD approach become more evident.

6. Conclusion

Fitting discrete data using regression functions allows users to predict and identify patterns and trends within the data. Furthermore, clustering allows the data to be classified such that different clusters present different information.

We have presented an innovative framework of mixed-integer linear models for regression and clustering problems, including two models from the literature and two original formulations. The four formulations, for clusterwise linear regression (CLR), ordered clusterwise linear regression (oCLR), piecewise linear regression (PWLR) and clusterwise piecewise linear regression (CPWLR) serve different purposes. All four problems can be modelled with a mixed-integer linear program, making use of binary variables and logical implications modelled by big-$\mathcal{M}$ constraints. Hence, each of the four models shares a special structure, which can be enhanced. We have shown that outlier detection can be implemented into the models, eliminating the need for pre- or post-processing. Furthermore, we have shown that combinatorial Benders decomposition can be used to speed up the models by up to 800 times.

For future work, we aim to improve the formulation for CPWLR by removing the ordering of the clusters and allowing them to overlap. This involves a large reformulation, yet we hope this first work on the topic allows for further developments and understanding of the models we have presented. We further plan to extend the application of combinatorial Benders decomposition to the sum of absolute differences metric, allowing speedups to be found for more accurate data analyses.

Data availability

Data will be made available on request.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 445857709.

Appendix. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.dam.2023.03.010.

References

[1] E. Angün, A. Altınoy, A new mixed-integer linear programming formulation for multiple responses regression clustering, in: International Conference on Control, Decision and Information Technologies, CoDIT '19, 2019, pp. 1634–1639.
[2] R. Bellman, R. Roth, Curve fitting by segmented straight lines, J. Amer. Statist. Assoc. 64 (327) (1969) 1079–1084.
[3] D. Bertsimas, A. King, OR forum—An algorithmic approach to linear regression, Oper. Res. 64 (1) (2016) 2–16.
[4] D. Bertsimas, A. King, R. Mazumder, Best subset selection via a modern optimization lens, Ann. Statist. 44 (2) (2016) 813–852.
[5] D. Bertsimas, R. Mazumder, Least quantile regression via modern optimization, Ann. Statist. 42 (6) (2014) 2494–2525.
[6] D. Bertsimas, R. Shioda, Classification and regression via integer optimization, Oper. Res. 55 (2) (2007) 252–271.
[7] E. Camponogara, L.F. Nazari, Models and algorithms for optimal piecewise-linear function approximation, Math. Probl. Eng. 2015 (2015).
[8] G. Codato, M. Fischetti, Combinatorial Benders' cuts for mixed-integer linear programming, Oper. Res. 54 (2006) 756–766.
[9] J. Codsi, S.U. Ngueveu, B. Gendron, LinA: A Faster Approach to Piecewise Linear Approximations Using Corridors and Its Application to Mixed-Integer Optimization, Technical Report, 2021.
[10] R. Collobert, F. Sinz, J. Weston, L. Bottou, Trading convexity for scalability, in: Proceedings of the 23rd International Conference on Machine Learning, 2006, pp. 201–208.
[11] R.A. da Silva, F.d.A. de Carvalho, Weighted clusterwise linear regression based on adaptive quadratic form distance, Expert Syst. Appl. 185 (2021) 115609.
[12] W.S. DeSarbo, W.L. Cron, A maximum likelihood methodology for clusterwise linear regression, J. Classification 5 (1988) 249–282.
[13] W. DeSarbo, R. Oliver, A. Rangaswamy, A simulated annealing methodology for clusterwise linear regression, Psychometrika 54 (4) (1989) 707–736.
[14] A. Duguet, C. Artigues, L. Houssin, S.U. Ngueveu, Properties, extensions and application of piecewise linearization for Euclidean norm optimization in R 2, J. Optim. Theory Appl. 195 (2) (2022) 418–448.
[15] A. Duguet, S.U. Ngueveu, Piecewise linearization of bivariate nonlinear functions: minimizing the number of pieces under a bounded approximation error, in: Combinatorial Optimization: 7th International Symposium, ISCO 2022, Virtual Event, May 18–20, 2022, Revised Selected Papers, Springer, 2022, pp. 117–129.
[16] R.P. Ferreira, A. Martiniano, A. Ferreira, A. Ferreira, R.J. Sassi, Study on daily demand forecasting orders using artificial neural network, IEEE Lat. Am. Trans. 14 (3) (2016) 1519–1525.
[17] S. Frank, I. Steponavice, S. Rebennack, Optimal power flow: a bibliographic survey I, Energy Syst. 3 (3) (2012) 221–258.
[18] N. Goldberg, Y. Kim, S. Leyffer, T.D. Veselka, Adaptively refined dynamic program for linear spline regression, Comput. Optim. Appl. 58 (3) (2014) 523–541.
[19] M.T. Goodrich, Efficient piecewise-linear function approximation using the uniform metric, Discrete Comput. Geom. 14 (4) (1995) 445–462.
[20] S. Hakimi, E. Schmeichel, Fitting polygonal functions to a set of points in the plane, CVGIP, Graph. Models Image Process. 53 (2) (1991) 132–136.
[21] D.M. Hawkins, D. Olive, Applications and algorithms for least trimmed sum of absolute deviations regression, Comput. Statist. Data Anal. 32 (2) (1999) 119–134.
[22] C. Hennig, Models and methods for clusterwise linear regression, in: W. Gaul, H. Locarek-Junge (Eds.), Classification in the Information Age, 1999, pp. 179–187.
[23] H. Imai, M. Iri, An optimal algorithm for approximating a piecewise linear function, J. Inf. Process. 9 (3) (1986) 159–162.
[24] K. Joki, A.M. Bagirov, N. Karmitsa, M.M. Mäkelä, S. Taheri, Clusterwise support vector linear regression, European J. Oper. Res. 287 (1) (2020) 19–35.
[25] M. Khadka, A. Paz, A. Singh, Generalised clusterwise regression for simultaneous estimation of optimal pavement clusters and performance models, Int. J. Pavement Eng. 21 (9) (2020) 1122–1134.
[26] R. Koenker, G. Bassett, On Boscovich's estimator, Ann. Statist. 13 (4) (1985) 1625–1628.
[27] L. Kong, C.T. Maravelias, On the derivation of continuous piecewise linear approximating functions, INFORMS J. Comput. 32 (3) (2020) 531–546.
[28] V. Krasko, S. Rebennack, Two-stage stochastic mixed-integer nonlinear programming model for post-wildfire debris flow hazard management: Mitigation and emergency evacuation, European J. Oper. Res. 263 (1) (2017) 265–282.
[29] S. Lloyd, Least squares quantization in PCM, IEEE Trans. Inform. Theory 28 (2) (1982) 129–137.
[30] Z. Luo, E.Y. Chou, Pavement condition prediction using clusterwise regression, Transp. Res. Rec. J. Transp. Res. Board 1974 (1) (2006) 70–77.
[31] J.F. Macgregor, T.J. Harris, The exponentially weighted moving variance, J. Qual. Technol. 25 (2) (1993) 106–118.
[32] A. Magnani, S.P. Boyd, Convex piecewise-linear fitting, Opt. Eng. 10 (1) (2009) 1–17.
[33] A. Martin, M. Möller, S. Moritz, Mixed integer models for the stationary case of gas network optimization, Math. Program. 105 (2) (2006) 563–582.
[34] K. McCoy, V. Krasko, P. Santi, D. Kaffine, S. Rebennack, Minimizing economic impacts from post-fire debris flows in the western United States, Nat. Hazards 83 (1) (2016) 149–176.

[35] D. McNeil, Interactive Data Analysis: A Practical Primer, John Wiley & Sons, 1977.
[36] F. Murtagh, P. Contreras, Algorithms for hierarchical clustering: an overview, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 2 (1) (2012) 86–97.
[37] S.U. Ngueveu, Piecewise linear bounding of univariate nonlinear functions and resulting mixed integer linear programming-based solution methods, European J. Oper. Res. 275 (3) (2019) 1058–1071.
[38] S.U. Ngueveu, C. Artigues, N. Absi, S. Kedad-Sidhoum, Lower and upper bounds for scheduling energy-consuming tasks with storage resources and piecewise linear costs, J. Heuristics (2022) 1–28.
[39] S.U. Ngueveu, C. Artigues, P. Lopez, Scheduling under a non-reversible energy source: An application of piecewise linear bounding of non-linear demand/cost functions, Discrete Appl. Math. 208 (2016) 98–113.
[40] Y.W. Park, Y. Jiang, D. Klabjan, L. Williams, Algorithms for generalized clusterwise linear regression, INFORMS J. Comput. 29 (2) (2017) 301–317.
[41] V. Piccialli, A.M. Sudoso, A. Wiegele, SOS-SDP: an exact solver for minimum sum-of-squares clustering, INFORMS J. Comput. 34 (4) (2022) 2144–2162.
[42] S. Rebennack, V. Krasko, Piecewise linear function fitting via mixed-integer linear programming, INFORMS J. Comput. 32 (2) (2020) 507–530.
[43] P.J. Rousseeuw, Least median of squares regression, J. Amer. Statist. Assoc. 79 (388) (1984) 871–880.
[44] H. Späth, Algorithm 39: Clusterwise linear regression, Computing 22 (1979) 367–373.
[45] H. Späth, Correction to algorithm 39: Clusterwise linear regression, Computing 26 (1981) 275.
[46] S.M. Stigler, Gauss and the invention of least squares, Ann. Statist. (1981) 465–474.
[47] N. Sudermann-Merx, S. Rebennack, Leveraged least trimmed absolute deviations, OR Spectrum 43 (2021) 809–834.
[48] A. Toriello, J.P. Vielma, Fitting piecewise linear continuous functions, European J. Oper. Res. 219 (1) (2012) 86–95.
[49] B. Tunga, A hybrid algorithm with cluster analysis in modelling high dimensional data, Discrete Appl. Math. 235 (2018) 161–168.
[50] J.P. Vielma, Mixed integer linear programming formulation techniques, SIAM Rev. 57 (1) (2015) 3–57.
[51] A.K. Wagner, S.B. Soumerai, F. Zhang, D. Ross-Degnan, Segmented regression analysis of interrupted time series studies in medication use research, J. Clin. Pharm. Ther. 27 (4) (2002) 299–309.
[52] J.A. Warwicker, S. Rebennack, A comparison of two mixed-integer linear programs for piecewise linear function fitting, INFORMS J. Comput. 34 (2) (2022) 1042–1047.
[53] J.A. Warwicker, S. Rebennack, Generating optimal robust continuous piecewise linear regression with outliers through combinatorial Benders decomposition, IISE Trans. (2022) http://dx.doi.org/10.1080/24725854.2022.2107249.
[54] R. Xu, D.C. Wunsch, Clustering, John Wiley & Sons, Ltd, 2008.
[55] L. Yang, S. Liu, S. Tsoka, L.G. Papageorgiou, Mathematical programming for piecewise linear regression analysis, Expert Syst. Appl. 44 (2016) 156–167.