IISE Transactions

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uiie21

Generating optimal robust continuous piecewise linear regression with outliers through combinatorial Benders decomposition

John Alasdair Warwicker & Steffen Rebennack

To cite this article: John Alasdair Warwicker & Steffen Rebennack (2022): Generating optimal robust continuous piecewise linear regression with outliers through combinatorial Benders decomposition, IISE Transactions, DOI: 10.1080/24725854.2022.2107249

To link to this article: https://doi.org/10.1080/24725854.2022.2107249

IISE TRANSACTIONS
https://doi.org/10.1080/24725854.2022.2107249

Generating optimal robust continuous piecewise linear regression with outliers through combinatorial Benders decomposition

John Alasdair Warwicker and Steffen Rebennack

Stochastic Optimization, Institute of Operations Research, Karlsruhe Institute of Technology, Karlsruhe, Germany

ABSTRACT

Using piecewise linear (PWL) functions to model discrete data has applications for example in healthcare, engineering and pattern recognition. Recently, mixed-integer linear programming (MILP) approaches have been used to optimally fit continuous PWL functions. We extend these formulations to allow for outliers. The resulting MILP models rely on binary variables and big-$\mathcal{M}$ constructs to model logical implications. The combinatorial Benders decomposition (CBD) approach removes the dependency on the big-$\mathcal{M}$ constraints by separating the MILP model into a master problem of the complicating binary variables and a linear sub problem over the continuous variables, which feeds combinatorial solution information into the master problem. We use the CBD approach to decompose the proposed MILP model and solve for optimal PWL functions. Computational results show that vast speedups can be found using this robust approach, with problem-specific improvements including smart initialization, strong cut generation and special branching approaches leading to even faster solve times, up to more than 12,000 times faster than the standard MILP approach.

ARTICLE HISTORY
Received 22 May 2021
Accepted 20 July 2022

KEYWORDS
Piecewise linear function; combinatorial Benders decomposition; mixed-integer linear programming (MILP); function fitting; outlier detection

1. Introduction

It is often necessary to model discrete data points with a continuous function; for example, to allow for predictions of the qualities of future data based on interpolation or extrapolation from the existing data. Fitting data points with a piecewise linear (PWL) function can be advantageous over using a non-linear continuous function (polynomial regression) or a single linear function (linear regression), since a PWL function consists of linear segments and can lead to more accurate modelling than a simple linear function. As well as modelling discrete data, PWL functions can also be used to model continuous functions. By approximating data (or non-linear functions) with PWL functions, complicated mixed-integer non-linear and non-convex programming problems can be (approximately) solved relatively quickly using standard mixed-integer linear programming (MILP) techniques (Feijoo and Meyer, 1998; Geißler *et al.* 2012; Rebennack and Kallrath, 2015; Rebennack, 2016a).

PWL functions are comprised of connected, *affine functions* (also referred to as *linear segments*) which intersect at *breakpoints*. PWL fitting is also known by different names in other disciplines, such as linear spline regression in statistics, or polyhedral function fitting in mathematics. Applications of fitting PWL functions to data have been seen in the fields of healthcare (Wagner *et al.*, 2002), pattern recognition (Chang, 1973), biomedical studies (Berman *et al.*, 1996), amongst others. In particular, Gunnerud and Foss (2010) presented a method for the real-time optimization of process systems within the field of engineering, where the non-linearities of the presented approach were modelled by a PWL function. This allowed a MILP model to be formulated, with error bounds on the optimal solution also being found.

Despite the (piecewise) linearity of PWL functions, there are still difficulties in computing them. Such difficulties include calculating optimal breakpoint locations. Ensuring the continuity of the linear segments presents a similar challenge (Chen and Wang, 2009).

A recent trend is approaching the problem of PWL function fitting from an optimization perspective. Various methods, ranging from dynamic programming (Bellman and Roth, 1969) and numerical approaches (Jupp, 1978), to heuristic approaches (Ertel and Fowlkes, 1976) and the R package *segmented* (Muggeo, 2003), have been implemented. Hakimi and Schmeichel (1991) presented an efficient, $\mathcal{O}(n)$-time algorithm to fit an optimal PWL function within some given maximum error tolerance $\epsilon > 0$. Approaches for minimizing the maximum error for a PWL function with a given number of breakpoints have also been presented by Hakimi and Schmeichel (1991) ($\mathcal{O}(n^2 \log n)$), Wang *et al.* (1993) ($\mathcal{O}(n^2)$) and Goodrich (1994) ($\mathcal{O}(n \log n)$).

Many MILP approaches for PWL fitting have also been presented in the literature. However, most disregard the continuity requirement in order to simplify the problem. Bertsimas and Shioda (2007) used MILP models for classification and to fit

CONTACT John Alasdair Warwicker john.warwicker@kit.edu

Supplemental data for this article is available online at https://doi.org/10.1080/24725854.2022.2107249.

discontinuous PWL functions to each class. Further approaches by Bertsimas and Mazumder (2014), Bertsimas and King (2016), and Bertsimas *et al.* (2016) similarly dismiss the continuity requirement. Goldberg *et al.* (2014) and Goldberg *et al.* (2021) include the continuity requirement using a non-convex, dynamic programming approach, and Toriello and Vielma (2012) use a non-convex, quadratically constrained approach. Toriello and Vielma (2012) also introduce a mixed-integer linear model to fit convex PWL functions to discrete data, which simplifies the problem significantly. The first exact approach to approximate continuous functions by PWL continuous functions was presented by Rebennack and Kallrath (2015).

Rebennack and Krasko (2020) recently introduced a MILP approach to PWL fitting for discrete data which does not require any assumptions on the convexity of the PWL function. This novel approach does not directly compute the breakpoints, yet ensures the continuity of the linear segments by allowing the breakpoints to fall as necessary between two datapoints. This modelling trick comes at the cost of using big-$\mathcal{M}$ constructs, which are known to yield weak LP-relaxations. They also constructed an exact algorithm to optimally fit PWL functions to univariate continuous functions. A similar MILP formulation for fitting PWL functions to discrete data and univariate functions has also been presented by Kong and Maravelias (2020). Again, the breakpoints are calculated implicitly, at the cost of a large number of big-$\mathcal{M}$ constructs and binary variables. The binary variables are used to assign data points to linear segments, as well as to designate the first and last point in each linear segment. Further binary variables are used to activate the continuity constraints. However, the formulation of Kong and Maravelias (2020) contains a larger number of variables and constraints, and has recently been shown to be less effective for most instances than the formulation of Rebennack and Krasko (2020) for fitting optimal continuous PWL functions (Warwicker and Rebennack, 2022).

The MILP model introduced by Rebennack and Krasko (2020) for fitting PWL functions to data only uses binary variables to assign data points to linear segments, and to model the change in gradient between consecutive linear segments. In this article, we update the model by implementing inbuilt outlier detection, whereby outlier data points are identified during the optimization process and implemented into the objective function. This is in contrast with other approaches for outlier detection within classification or regression problems, which typically employ two-stage models (see e.g., Chatzinakos *et al.*, 2016; Sudermann-Merx and Rebennack, 2021). In such models, outliers are often identified separately from the main optimization problem based on a given distance measure to the remainder of the data.

We use Benders decomposition (Benders, 1962) to look for speedups in the updated, robust formulation to mitigate the weakness presented by the presence of big-$\mathcal{M}$ constructs. The Benders decomposition approach allows for linear programming problems with a certain structure to be optimised via a divide-and-conquer approach (Rebennack, 2016b; Rahmaniani *et al.*, 2017). Benders decomposition also works for MILP models as long as the sub problems remain linear. Since the computational difficulty of solving an optimization problem increases with the size of the problem, iteratively solving sub problems can be more efficient than solving the main, monolithic problem. The sub problems feed information and constraints to the master problem. The given MILP model fits this structure well, since the problem can be separated into a *master* problem solved over the first set of variables (i.e., binary variables), and a *sub* problem over the continuous variables. Depending on the optimality (or infeasibility) of the sub problem, so-called *Benders cuts* are implemented into the master problem, which is re-solved until no more cuts are available. However, for the MILP we consider, we expect (and observe in experimental results) that the Benders cuts are weak due to the presence of big-$\mathcal{M}$ values and the number of constraints (Codato and Fischetti, 2006). Improvements to the Benders decomposition algorithm have been sought since its inception (see the survey by Rei *et al.* (2009) for an in-depth discussion).

In particular, problem-specific tailoring of Benders decomposition can lead to significant speedups (Lohmann and Rebennack, 2017; Rebennack *et al.*, 2020). For example, *combinatorial* Benders decomposition (CBD) is particularly effective for MILP problems with a large number of logical implications modelled through big-$\mathcal{M}$ constructs, since it aims to remove the dependency on the big-$\mathcal{M}$ constraints (Codato and Fischetti, 2006). The master problem contains the constraints which include the binary variables and feeds solution information into the sub problem, which tests for feasibility. On infeasibility, combinatorial cuts (with no big-$\mathcal{M}$ values present) relating to irreducible infeasible subsystems of the sub problem are added into the master problem. This way, the structure of the problem is exploited instead of weak cuts resulting from the loose big-$\mathcal{M}$ formulation. Codato and Fischetti (2006) performed experiments on two classes of MILP models, demonstrating that combinatorial cuts can lead to considerable speedups. Parallels can be drawn between CBD and the logic-based Benders decomposition approach of Hooker and Ottosson (2003), which is a more general case of CBD.

After testing a number of natural improvements to the CBD method, we develop a tailored CBD approach to the PWL fitting problem and compare it with the standard MILP approach with outliers. Using a series of real-world data sets with different characteristics, we present experimental comparisons between the default CBD model, the tailored CBD model, and the monolithic MILP approach.

This article has the following main contributions:

- We implement automatic outlier detection into the MILP model for optimal PWL function fitting introduced by Rebennack and Krasko (2020), such that a given number of data points are excluded from the final PWL function.
- We show that the standard Benders decomposition method when applied to the updated MILP model for PWL function fitting with outlier detection is inefficient with respect to solving the MILP model as they only produce cuts which invalidate the current solution.
- We use CBD to reformulate the MILP model. Using a state-of-the-art implementation making use of callbacks, we show the CBD approach far outperforms the standard MILP on a series of data sets with different characteristics. Large speedups are seen for functions where more breakpoints are required.
- We test a number of improvements to the CBD model, and implement the best performing ones within the

model for PWL function fitting with outliers. By increasing the quantity and quality of combinatorial cuts and using smart initialization strategies and branching rules, we show that this tailored model is even faster than the default CBD model and solves instances with a large number of breakpoints very quickly.

The rest of this article is structured as follows. In Section 2 we discuss the MILP approach of Rebennack and Krasko (2020) and our added outlier detection implementation in detail. In Section 3 we apply Benders decomposition to the problem. We use the CBD approach in Section 4 to discuss the combinatorial cuts and the CBD methodology for the MILP model, with improvements to this model discussed in Section 5. Experimental results are presented in Section 6, and we conclude with Section 7.

2. PWL function fitting

We begin this section by defining the important concepts related to PWL function fitting. Throughout this article we use the following notation: $[n]$ to denote the set $\{1, ..., n\}$.

Definition 1. (Rebennack and Krasko (2020)). *A continuous univariate function $p(x) : [\underline{X}, \overline{X}] \rightarrow \mathbb{R}$ with compact interval $[\underline{X}, \overline{X}]$ is called a continuous **PWL function**, if there exists a finite number B with $\underline{X} = r_1 < ... < r_b < r_{b+1} < ... < r_B = \overline{X}$, such that $p(x)$ is an affine function on $[r_b, r_{b+1}]$ for all $b \in [B-1]$. The r_b are called **breakpoints**, with B the number of breakpoints. For each $b \in [B-1]$, the function $p(x) : [r_b, r_{b+1}] \rightarrow \mathbb{R}$ is called a **linear segment**.*

For the problem of PWL function fitting of bivariate data, a set of I ordered tuples $(X_i, Y_i) \in \mathbb{R}^2$, $i \in [I]$, is given where $-\infty < \underline{X} = X_1 \leq ... \leq X_i \leq X_{i+1} \leq ... \leq X_I = \overline{X} < \infty$. The data tuples ($X_i$, Y_i) are the sorted values of a discrete function which takes inputs from $[\underline{X}, \overline{X}]$ and returns outputs from $\mathbb{R}$. We seek to model these data points by a PWL function $p(x) : [\underline{X}, \overline{X}] \rightarrow \mathbb{R}$ with (at most) a given number of breakpoints B. Since the resulting PWL function will be an approximation of the data, only $B \leq I$ is meaningful, while $B \ll I$ is typically seen in practice. Moreover, we seek an optimal PWL function which minimises some distance metric $d(\cdot, \cdot)$ between the data points and given PWL function.

For the construction of an optimal PWL function, it is necessary to allow the breakpoints to be free within the range $[\underline{X}, \overline{X}]$, i.e., the breakpoint locations are not limited to certain points, or pre-specified.

2.1. Existing model for PWL function fitting

For readibility and because the MILP formulation is central for this work, we repeat the formulation of Rebennack and Krasko (2020). This formulation does not explicitly solve for the location of the breakpoints, yet provides the linear segments. Each linear segment ($b \in [B-1]$) is defined by a gradient c_b and an intercept d_b (i.e., the affine function $b \in [B-1]$ has equation $y = c_b x + d_b$). If the binary variable $\delta_{i,b} = 1$, then the data point (X_i, Y_i) is associated with segment b of the PWL function (for $i \in [I]$ and $b \in [B-1]$). The breakpoints are then given by the intersection of consecutive linear segments (further information on how solution information can be extracted is found in Section A of the Appendix):

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

The objective function (1a) minimises the chosen distance metric. Throughout this article, we use the *maximum difference* metric (or ℓ_∞-norm), which minimises the maximum absolute distance between the data points and the PWL function (we aim to consider other distance metrics, such as the *sum of absolute differences* and *sum of squared differences*, in future work). It works by minimizing the absolute distance between the PWL function and the farthest outlier data point, without consideration to all the other data points. Note that the objective function does not contain any of the binary variables, with its value implicitly given by constraints (1b) and (1c), which evaluate the objective value for the given PWL function. The role of the big-$\mathcal{M}$ constructs is discussed in Section B of the Appendix.

Constraints (1d) ensure each data point is associated with exactly one linear segment. Constraints (1e)–(1g) ensure the ordering of all data points, such that if a point is associated with a certain segment, the previous point must either be associated with the previous or same segment ((1f) and (1g) ensure this for the first and last segments, respectively).

Constraints (1h)–(1m) ensure continuity of the PWL functions. If there is a breakpoint between data points X_i and X_{i+1} which are associated with linear segments b and $b+1$ respectively, then $\delta_{i,b} = \delta_{i+1,b+1} = 1$. Furthermore, the two segments will intersect at the breakpoint which lies between the two data points. That is,

$$X_i \leq \frac{d_{b+1} - d_b}{c_b - c_{b+1}} \leq X_{i+1}.$$

In this instance, either $\delta^+_{i,b}$ or $\delta^-_{i,b}$ is set to one depending on the value of the binary variable γ_b, which denotes whether the change in gradient between adjacent segments is positive or negative. In either case, the denominator will be distributed to avoid non-linearity, and the direction of the inequalities will change depending on the sign of the change in gradient. If $\gamma_b = 1$, then $\delta^+_{i,b} = 1$ and constraints (1j) and (1k) are activated, implying that the gradient decreases between the two consecutive linear segments (i.e., $c_b - c_{b+1} \geq 0$). Alternatively, if $\gamma_b = 0$, then $\delta^-_{i,b} = 1$ and constraints (1l) and (1m) are activated, implying the gradient increases (i.e., $c_b - c_{b+1} \leq 0$). Note that if all of the γ_b variables take the same value, then the PWL function is either convex or concave. Finally, constraints (1n)–(1r) give the domains for the decision variables (note that $\delta^+_{i,b}$ and $\delta^-_{i,b}$, while having domains $[0, 1]$, are not binary variables). The derivation of the variable domains are discussed further in Section B of the Appendix. We refer the reader to Rebennack and Krasko (2020) for a detailed derivation and explanation of formulation (1).

Using this formulation results in a convex MILP with $2BI - 4I + 3$ continuous variables and $B(I+1) - I - 2$ binary variables, and a total of $9BI - 7B - 13I + 12$ functional constraints. Solving formulation (1) for the given distance metric provides a globally optimal PWL for the associated set of I data points, with a given number of breakpoints $B \leq I$ (Rebennack and Krasko, 2020, Theorem 1). Recall that $B \leq I$ is required to avoid overfitting.

2.2. Outlier detection

In its current state, formulation (1) can be susceptible to outliers in the x-component (leverage points), as well as in the y-component. If there are outliers in the data set, the resulting PWL function will likely be heavily affected and far from optimal for the non-outlier data points. For our application, we aim to remove outlier data points from the resulting models. Post-processing outlier removal methods, such as the removal of a given number of data points with the farthest distance to their associated linear segment, can be implemented. However, it is likely that the model will have to be re-solved once the outliers have been removed in order to provide a better fit to the remaining data points. Furthermore, these data points will not necessarily always be outliers, and the resulting model may be better if other points were removed from the data. Alternative approaches, such as the total enumeration through all possible sets of outlier points, are likely to be inefficient.

Regarding pre-processing options for outlier detection (and elimination), we can identify a set of possible outliers before the model is solved. Such an approach was utilised by Sudermann-Merx and Rebennack (2021) for solving regression functions subject to the possible exclusion of pre-identified leverage points. In this case, possible outlier points are identified before the model is solved subject to the exclusion of a subset of these identified outlier points. The outlier points can be identified using statistical analyses; for example, identifying data points whose variables lie outside the interquartile range. Let $\mathbf{O}$ be the set of possible outlier points, ρ_i be $|\mathbf{O}|$ binary variables, each relating to a possible outlier point, where $\rho_i = 1$ means that data point $i \in \mathbf{O}$ is included in the fit (i.e., contributes to the objective function) and not an outlier, and let Q be the total number of outliers to be excluded. The following constraint was included in their model: $\sum_{i \in \mathbf{O}} \rho_i = |\mathbf{O}| - Q$.

For the application of solving regression functions, Sudermann-Merx and Rebennack (2021) implemented outlier detection into the objective function by multiplying the contribution of each possible outlier data point by the binary variable. In this instance, the resulting objective function is non-linear and is hence a relatively time-consuming addition.

In order to increase robustness and efficiency (and to avoid additional pre- or post-processing computational costs), we wish to implement outlier detection such that it is done simultaneously within the MILP model. Such a strategy is often utilised in linear regression models (Rousseeuw, 1984; Koenker and Bassett, 1985; Hawkins and Olive, 1999), and is often done using non-linear constraints. With the aim of preserving the linearity of the model, we can implement outlier detection into our formulation by slightly adjusting constraints (1b)–(1c). We use the binary variable ($\rho_i, i \in [I]$) to state that the data point X_i is included in the model (if $\rho_i = 1$), and hence not an outlier point, and we again state that we wish to omit Q outlier points.

$$\min \ \xi \tag{2a}$$

$$\text{s.t.} \quad Y_i - (c_b X_i + d_b) \leq \xi + M^1_i(2 - \delta_{i,b} - \rho_i) \quad \forall i \in [I]; \forall b \in [B-1] \tag{2b}$$

$$(c_b X_i + d_b) - Y_i \leq \xi + M^1_i(2 - \delta_{i,b} - \rho_i) \quad \forall i \in [I]; \forall b \in [B-1] \tag{2c}$$

$$\sum_{i=1}^{I} \rho_i = I - Q \tag{2d}$$

$$\rho_i \in \{0, 1\} \quad \forall i \in [I] \tag{2e}$$

$$(1\text{d}) - (1\text{r}) \tag{2f}$$

The objective function (2a) calculates the distance measure as usual. However, constraints (2b)–(2c) have been adjusted to only affect the calculation of ξ if the data point X_i is not an outlier. Otherwise, it does not affect the

objective function (since the data point is considered as an outlier). Constraint (2d) ensures that only Q data points are considered as outliers (note that even if *at most* Q outliers were being identified, the value of the objective function would not improve compared to a fixed value of Q), and constraint (2e) gives the domain of the binary variables.

Formulation (2) introduces I new binary variables in comparison to formulation (1), and one further functional constraint. Hence, formulation (2) has a total of $B(I+1)-2$ binary variables, $2BI-4I+3$ continuous variables, and $9BI-7B-13I+13$ functional constraints. We end the discussion on this model with the following remark, which gives the optimality of the found PWL function resulting from the MILP model.

Remark 1. *For a set of I ordered data points $(X_i, Y_i) \in \mathbb{R}^2$, $i = 1, ..., I$, bounds on the slope $[\underline{C}, \overline{C}]$ and intercept $[\underline{D}, \overline{D}]$, and a given number of breakpoints $B \leq I$ and a number of outlier points $Q \leq I$, optimal solutions of the MILP given by formulation (2) define globally optimal continuous PWL functions minimizing the given distance metric, with slope $\in [\underline{C}, \overline{C}]$ and intercept $\in [\underline{D}, \overline{D}]$.*

3. Model decomposition

Solving the MILP given by formulation (2) involves solving for $B(I+1)-2 = \mathcal{O}(BI)$ binary variables. Although it is often the case that $B \ll I$, branch-and-cut approaches are challenged even for medium-sized instances.

A natural decomposition for formulation (2) is to separate the model into a *master* problem consisting of the complicating variables, which can feed solution information into a *sub* problem. Such an idea is typically used when applying Benders decomposition (Benders, 1962). In this case, depending on the solution to the sub problem (i.e., optimality or infeasibility), cuts can be added to the master problem which is then re-solved. This process is repeated until optimality is found for the overall problem (and the solution is given by the distance metric in the final sub problem).

To implement this decomposition approach into the model, we use the following deconstruction of the MILP given by formulation (3), separating the binary variables and continuous variables. The master problem consists of all constraints containing only binary variables. Note that in the first iteration (without any induced cuts), the master problem is simply a feasibility problem, since the objective function of the monolithic MILP does not depend explicitly on the values of the binary variables (we use the objective function Z, which is affected by the cuts and provides a lower bound on the overall objective). Furthermore, since there are no constraints for the values of the γ_b variables, $b \in \{1, ..., B-2\}$, these can be chosen arbitrarily in the first iteration. The master problem is given in formulation (3):

$$\begin{aligned}
(\text{Master}) \quad \min \quad & Z \in \mathbb{R}^+ && && (3a)\\
\text{s.t} \quad & (1\text{d}) - (1\text{g}) && && (3b)\\
& \sum_{i=1}^{I} \rho_i = I - Q && && (3c)\\
& \textit{Induced cuts } (6) && && (3d)\\
& \delta_{i,b} \in \{0,1\} && \forall i \in [I];\ \forall b \in [B-1] && (3e)\\
& \gamma_b \in \{0,1\} && \forall b \in [B-2] && (3f)\\
& \rho_i \in \{0,1\} && \forall i \in [I] && (3g)
\end{aligned}$$

The objective of the master problem is given as Z, which is an auxiliary variable used to estimate the true value of the objective function through cuts. The cuts will remove a non-improving solution corresponding to the given assigment of binary variables in the master problem. Initially, we set the value of Z to zero as an initial lower bound on the true objective function value (since we know that the true objective function value is always non-negative in a feasible solution). Solving the master problem provides trial solutions of the binary variables. Let the optimal solution for the master problem after the most recent addition of cuts be given by $\hat{\delta}_{i,b}$ for $i \in [I]$, $b \in [B-1]$, $\hat{\rho}_i$ for $i \in [I]$, and $\hat{\gamma}_b$ for $b \in [B-2]$:

$$\begin{aligned}
(\text{Sub}(\hat{\delta}, \hat{\rho}, \hat{\gamma})) \quad \min \quad & \xi && && (4a)\\
\text{s.t.} \quad & \xi + (c_b X_i + d_b) \geq Y_i - M_i^1(2 - \hat{\delta}_{i,b} - \hat{\rho}_i) && \forall i \in [I];\ \forall b \in [B-1] && (4b)\\
& \xi - (c_b X_i + d_b) \geq -Y_i - M_i^1(2 - \hat{\delta}_{i,b} - \hat{\rho}_i) && \forall i \in [I];\ \forall b \in [B-1] && (4c)\\
& \delta^+_{i,b} \geq \hat{\delta}_{i,b} + \hat{\delta}_{i+1,b+1} + \hat{\gamma}_b - 2 && \forall i \in [I-1];\ b \in [B-2] && (4d)\\
& \delta^-_{i,b} \geq \hat{\delta}_{i,b} + \hat{\delta}_{i+1,b+1} + (1 - \hat{\gamma}_b) - 2 && \forall i \in [I-1];\ b \in [B-2] && (4e)\\
& (d_{b+1} - d_b) - X_i(c_b - c_{b+1}) - M_i^2 \delta^+_{i,b} \geq -M_i^2 && \forall i \in [I-1];\ b \in [B-2] && (4f)\\
& X_{i+1}(c_b - c_{b+1}) - (d_{b+1} - d_b) - M_{i+1}^2 \delta^+_{i,b} \geq -M_{i+1}^2 && \forall i \in [I-1];\ b \in [B-2] && (4g)\\
& X_i(c_b - c_{b+1}) - (d_{b+1} - d_b) - M_i^2 \delta^-_{i,b} \geq -M_i^2 && \forall i \in [I-1];\ b \in [B-2] && (4h)\\
& (d_{b+1} - d_b) - X_{i+1}(c_b - c_{b+1}) - M_{i+1}^2 \delta^-_{i,b} \geq -M_{i+1}^2 && \forall i \in [I-1];\ b \in [B-2] && (4i)\\
& \xi \in \left[\underline{M}^{\xi}, \overline{M}^{\xi}\right] && && (4j)\\
& d_b \in [\underline{D}_b, \overline{D}_b], c_b \in [\underline{C}_b, \overline{C}_b] && \forall b \in [B-1] && (4k)\\
& \delta^+_{i,b}, \delta^-_{i,b} \in [0,1] && \forall i \in [I-1];\ b \in [B-2] && (4l)
\end{aligned}$$

We present the rearranged sub problem consisting of only continuous variables in formulation (4) (which is a linear program), having moved any constant terms (including the fixed values for the binary variables) to the right-hand side of each constraint. Although the values of $\delta^{+}_{i,b}$ and $\delta^{-}_{i,b}$ are fixed to either zero or one in constraints (4d)–(4e), and hence lessen the effect of the big-$\mathcal{M}$ constants in constraints (4f)–(4i), they appear in the sub problem, since they are continuous variables (following from formulation (1) by Rebennack and Krasko (2020)). We note that they could also appear in the master problem, although preliminary experiments showed no tangible difference. Solving $\text{Sub}(\hat{\delta}, \hat{\rho}, \hat{\gamma})$ gives information leading to cuts to be added to the master problem (replacing constraint (3d)).

An advantage of solving for fixed binary variables is that there is a reduced dependence on the big-$\mathcal{M}$ constraints in the sub problem (i.e., many of the constraints (4b)–(4c) and (4f)–(4i) can essentially be ignored). Solving the sub problem gives an incumbent solution for the overall problem. Due to the setup of the problem, an optimal solution is always found for any feasible solution of binary variables, removing the need for feasibility cuts.

In order to apply Benders decomposition to the (bounded) sub problem and find the cuts, we consider its dual. We consider the dual variables $\alpha_{i,b,1/2}$, $\pi^{+/-}_{i,b}$, $\tau^{+/-}_{i,b,1/2}$ corresponding to constraints (4b)–(4c), (4d)–(4e), (4f)–(4i), respectively. The dual problem has the following objective function, a maximization problem with the coefficients of the dual variables given by the constants in the right-hand side of the rearranged sub problem given by formulation (4):

$$\begin{aligned}\max\Bigg\{ &\sum_{i=1}^{I}\sum_{b=1}^{B-1}\Big(\Big(Y_i - M_i^1\Big(2 - \hat{\delta}_{i,b} - \hat{\rho}_i\Big)\Big)\alpha_{i,b,1} \\ &+ \Big(-Y_i - M_i^1\Big(2 - \hat{\delta}_{i,b} - \hat{\rho}_i\Big)\Big)\alpha_{i,b,2}\Big) \\ &+ \sum_{i=1}^{I-1}\sum_{b=1}^{B-2}\Big(\Big(\hat{\delta}_{i,b} + \hat{\delta}_{i+1,b+1} + \hat{\gamma}_b - 2\Big)\pi^{+}_{i,b} \\ &+ \Big(\hat{\delta}_{i,b} + \hat{\delta}_{i+1,b+1} + (1 - \hat{\gamma}_b) - 2\Big)\pi^{-}_{i,b}\Big) \\ &- M_i^2\tau^{+}_{i,b,1} - M_{i+1}^2\tau^{+}_{i,b,2} - M_i^2\tau^{-}_{i,b,1} - M_{i+1}^2\tau^{-}_{i,b,2}\Bigg\}\end{aligned}$$

Without writing the dual problem out explicitly, we note that since the $\tau^{+/-}$ variables are bounded from below by zero, and have negative coefficients in the objective function, they will take a value of zero in an optimal dual solution. Hence, the maximization of the given objective function only involves maximizing for the coefficients of the α and $\pi^{+/-}$ variables. The coefficients of the α variables are only non-negative when the given data point is associated with the given segment, and is not an outlier (i.e., when $\delta_{i,b} + \rho_i = 2$), whereas the coefficients of the $\pi^{+/-}$ variables are only non-negative when a breakpoint exists between consecutive data points (i.e., when $\delta_{i,b} + \delta_{i+1,b+1} + \gamma_b - 2 = 1$ or $\delta_{i,b} + \delta_{i+1,b+1} + (1 - \gamma_b) - 2 = 1$). Hence, the cuts derived from the sub problem will involve the binary variables associated with non-outlier data points with the maximum absolute difference (in the former case), and at those associated with the breakpoints (in the latter case).

From this, we note that the Benders cuts will be optimality cuts of the form $Z \geq f(\delta, \rho, \gamma, \hat{\alpha}, \hat{\pi}^{+}, \hat{\pi}^{-})$, where f is an affine function of the master variables (i.e., δ, ρ, γ), and $(\hat{\alpha}, \hat{\pi}^{+}, \hat{\pi}^{-})$ represents the extreme points of the feasible region of the dual sub problem (in the projected space of the dual variables). These cuts will eliminate an infeasible $(Z, \delta, \rho, \gamma)$ solution for the master problem. That is, for a given objective function value to the master problem Z, alongside a set of binary variables forming that solution, the Benders cuts will state that the given objective function value Z is non-optimal, and that at least one of the binary variables must change in order to find an improved solution.

In experimental results, we found that the Benders decomposition approach applied to formulation (2) led to worse runtimes than the monolithic formulation. In particular, the runtime increases very quickly with the number of breakpoints. Therefore, we consider a more efficient way to find effective cuts for the master problem.

4. Implementing combinatorial cuts

The Benders decomposition approach relies on the sub problem feeding information about the given solution back to the master problem. The combinatorial Benders decomposition approach, introduced by Codato and Fischetti (2006), extends on this approach by returning combinatorial information on the variables in the master problem, making stronger cuts based on a smaller number of variables.

The combinatorial cuts are induced in the sub problem and added into the master problem on iterations in which the sub problem is infeasible. Since the objective function of formulation (2) relies only on the continuous variables, it cannot be accommodated explicitly into the master problem (only through an auxiliary variable which is used to estimate its value). However, by adding an inequality on the objective value into the sub problem of the form $\xi \leq \text{UB} - \epsilon$, where UB is the current incumbent value and ϵ a small, but positive, constant, we can induce infeasibility in the sub problem. Feasible iterations, which present an improvement of the current solution, update the value of the incumbent solution UB. Once the master problem becomes infeasible (due to the cuts), the optimal solution is defined by the set of variables providing the current incumbent solution.

At each iteration of the sub problem, we are looking to solve the following problem:

$$\begin{aligned} \min \quad & \xi && (5a)\\ \text{s.t.} \quad & \xi \leq \text{UB} - \epsilon && (5b)\\ & (4b) - (4l) && (5c)\end{aligned}$$

If the sub problem is infeasible, this suggests that at least one of the fixed binary variables must change in order to find a new feasible solution. This information can be fed into the master problem in the form of a cut. We refer to such a cut as a *solution elimination constraint*, since it ensures that the current solution cannot be found again by the master problem. In particular, this cut will imply that

the location of at least one breakpoint must change. We now explicitly formulate the solution elimination constraint. For $b \in [B-1]$, let the breakpoint b fall between data points X_{i_b} and X_{i_b+1} (i.e., $\hat{\delta}_{i_b,b} = \hat{\delta}_{i_b+1,b+1} = 1$) for the current given solution to the master problem (with fixed values $\hat{\delta}$, $\hat{\gamma}$, $\hat{\rho}$). The solution elimination constraint takes the following form:

$$\sum_{b=1}^{B-2}(2 - \delta_{i_b,b} - \delta_{i_b+1,b+1}) + \sum_{i\in[I]:\hat{\rho}_i=1}(1-\rho_i) + \sum_{b\in[B-2]:\hat{\gamma}_b=0}\gamma_b + \sum_{b\in[B-2]:\hat{\gamma}_b=1}(1-\gamma_b) \geq 1, \tag{6}$$

where we define $i_{B-1} = I$. First, we note that only changing $\delta_{i,b}$ variables that assign consecutive data points to adjacent segments will affect the calculation of the objective function, since the remaining values will be fixed in constraints (3c)–(3e). Second, we note that since any data point that is deemed an outlier has no effect on the calculation of the objective function, the constraint only eliminates the current assignment of non-outlier points. The cut is weak and very dense and implies that this approach is inefficient for this MILP, since it must cycle through every possible feasible assignment of data points to linear segments and selection of outliers.

In the context of mixed-integer non-linear programming (MINLP), such solution elimination constraints turned out to be helpful in designing a global optimization algorithm (Li *et al.*, 2011). However, an efficient solution algorithm can only be obtained when strengthening the master problem (which is a relaxation of the original problem), see Frank and Rebennack (2015).

In order to find stronger combinatorial cuts, Codato and Fischetti (2006) propose finding an *irreducible infeasible subsystem* (IIS) of the sub problem. An IIS of the sub problem given by formulation (6) is an infeasible subset of the constraints such that the removal of any one of the constraints in the IIS results in a feasible subsystem. A cut consisting of the need to change at least one of the binary variables appearing in the IIS (known as a *combinatorial Benders cut*) yields a much stronger cut than the standard solution elimination constraint, since many more potential solutions are declared infeasible.

Although the constraints (4f)–(4i) do not contain any binary variables, they can still contribute to the combinatorial Benders cut, since they are affected by the values of the binary variables in constraints (4d)–(4e). Hence, if any of the constraints (4f)–(4i) appear in the IIS, this implies that one (or both) of the constraints (4d)–(4e) (and their associated binary variables) are contributing to the infeasibility.

Let R denote the set of binary variables appearing in the constraints of the IIS (that is, the binary variables with non-zero coefficient in at least one constraint of the IIS). The combinatorial Benders cuts take the following form:

$$\sum_{\delta_{i,b}\in R:\hat{\delta}_{i,b}=1}(1-\delta_{i,b}) + \sum_{\rho_i\in R:\hat{\rho}_i=1}(1-\rho_i) + \sum_{\gamma_b\in R:\hat{\gamma}_b=1}(1-\gamma_b) + \sum_{\gamma_b\in R:\hat{\gamma}_b=0}\gamma_b \geq 1. \tag{7}$$

Note that only $\delta_{i,b}$ or ρ_i variables that are fixed to a value of one will cause infeasibility, since they do not affect the objective function if they are set to zero (due to the presence of the big-$\mathcal{M}$ constants in constraints (4b)–(4c)).

Implementing these combinatorial cuts into the master problem eliminates many suboptimal solutions in each iteration. In each iteration, the sub problem either updates the current objective value, or is infeasible due to the constraint (6b). Repeatedly solving infeasible sub problems for the given solution of the master problem induces more cuts, and eventually the master problem becomes infeasible. When the master problem becomes infeasible, the optimal solution to the model is given by the current value of UB, with the values for the variables given by the solution to the master and sub problems which led to this solution. Furthermore, it is possible to observe the gradual improvement in solution quality over time (for more information, see Section C of the Appendix). Overall, the process of CBD for our model (with the distance metric given by $d(\cdot,\cdot)$) is outlined in Algorithm 1, which returns an ϵ-optimal PWL function over the data points (X_i, Y_i), $i \in [I]$, with (at most) B breakpoints and Q outliers.

Algorithm 1 Combinatorial Benders Decomposition for PWL Regression with Outliers

```
Initialize UB = ∞, ε > 0; initialize some solution [δ̂, ρ̂, γ̂] of the master problem;
while master problem is feasible do
    Solve Sub(δ̂, ρ̂, γ̂);
    if sub problem is feasible then
        UB ← optimal objective function value of Sub(δ̂, ρ̂, γ̂);
        Add constraint d(·,·) ≤ UB − ε to the sub problem;
    end if
    Find (multiple) IISs of the sub problem;
    Add combinatorial Benders cuts (constraint (7)) to the master problem;
    Solve the master problem to give some new [δ̂, ρ̂, γ̂];
end while
Return UB.
```

5. Problem-specific model improvements

Due to the knowledge of the specific application we are considering (i.e., fitting PWL functions with outliers), we are able to implement improvements throughout.

5.1. Generating strong combinatorial cuts from the IIS

During the process of CBD (see Algorithm 1), finding an IIS leads to the creation of strong combinatorial cuts on the binary variables to be included in the master problem. If we are able to find multiple IISs for each infeasible sub problem, we can create different combinatorial cuts from each one, eliminating many possible solutions in each iteration. Since the number of IISs for a given system can be exponential in the size of the system (Chakravarti, 1994; Pfetsch,

2003), it is important that effective ones are computed efficiently.

In order to compute multiple IISs for each sub problem, we make use of the heuristic dual objective function $\sum_i w_i u_i$ with weights w_i, as suggested by Codato and Fischetti (2006) (see Section D of the Appendix). We initially set the weights all to one, and then iteratively set the weight of a constraint (which appears within the current IIS) one at a time to zero in order to find multiple IISs of the same infeasible system (until all the weights have been set to zero, or until an IIS can no longer be found). Furthermore, if we select which weights we reduce, and hence which constraints we no longer consider for the resulting IIS, we can improve the efficiency of the process. In experimental results in Section 6, we model this process by the use of the conflict refiner available within CPLEX.

The sub problem given by formulation (6) contains three possible groups of constraints that can appear in the IIS. Constraint (6b) always appears in the IIS, since that is the constraint that is causing the infeasibility. Additionally, further constraints are either of the form of constraints (4b)–(4c) or (4d)–(4e). If the constraints are of the form (4b)–(4c), this implies that either one $\delta_{i,b}$ variable or one ρ_i variable is causing the infeasibility. Hence, two variables will be present in the resulting cut. However, if the constraints are of the form (4d)–(4e), this implies that one of three variables (two $\delta_{i,b}$ variables and one γ_b variable) are causing the infeasibility, and hence it contributes three variables to the combinatorial cut. The latter cuts are weaker, since they suggest that either one of two $\delta_{i,b}$ variables must change, or a γ_b variable must change (note that there are no restrictions on the γ_b variables). In contrast, the former cuts suggest either one $\delta_{i,b}$ variable must change, or a ρ_i variable must change (for which there are further implications due to the knapsack-style constraint (3f)). In order to maximise the strength of the resulting combinatorial cuts, we can prioritise the removal of the weaker constraints from the IIS first (of the form (4d)–(4e)), so that stronger cuts can be found. Hence, we can reduce the weights of the heuristic dual objective function which correspond to the weaker cuts first. Thus, in each iteration, we create more cuts which are comparatively stronger.

5.2. Initialization

The initialization of the CBD method involves setting an initial value of the upper bound (UB). In the default method, we set $\text{UB} = +\infty$, which is immediately taken over by the value of the sub problem corresponding to the first found feasible master system. However, by fixing the initial value of UB to be lower (i.e., closer to the optimal solution), the system eliminates more infeasible solutions quickly and creates many strong combinatorial cuts. Hence, it may reach the optimal solution faster. Naturally, there must be a trade-off between the time required to find a reasonable initial solution and the improvement presented by implementing the given initialization. The R packages *segmented* and *strucchange* can be used in tandem to give approximations for breakpoint locations and initialized solutions (Muggeo, 2003). However, we note that these packages limit the number of breakpoints allowed for the PWL function (in preliminary experiments, we could not fit more than eight breakpoints), and hence cannot allow a fair comparison.

As a benchmark, we can initialise the model with the upper bound from the immediately smaller problem (i.e., with one fewer breakpoint and no outliers). When this information is not available, initializing the model with the upper bound from any smaller problems (if available), or through a simplified model where the breakpoints are assumed to be distributed evenly, may be beneficial.

5.3. Balancing the branching in the master problem

There are three types of binary variables in the master problem. The $\delta_{i,b}$ variables assign data points to linear segments, the ρ_i variables designate non-outlier points, and the γ_b variables model for the change in gradient between adjacent linear segments. For a given $i^\star \in [I]$ and $b^\star \in [B-1]$, standard branching rules on the $\delta_{i,b}$ binary variables lead to two leaf nodes in the tree, namely $\delta_{i^\star,b^\star} = 0$ and $\delta_{i^\star,b^\star} = 1$. Since branching on $\delta_{i^\star,b^\star} = 1$ has a huge impact compared with branching on $\delta_{i^\star,b^\star} = 0$, the branch-and-bound tree becomes very unbalanced (observe that such an effect is not present for the γ_b or ρ_i variables). Notably, setting $\delta_{i^\star,b^\star} = 1$ means that $\delta_{i,b} = 0$ for all $i < i^\star$ and all $b > b^\star$. Special branching rules can be implemented to exploit this special structure and decrease the impact of heavily unbalanced branching trees.

We consider a branching rule that takes advantage of the structure of the δ variables. Let there be a decision to branch on a given variable $\delta_{i^\star,b^\star}$, with indices $i^\star$ and $b^\star$. We assume $i^\star > 1$ and $b^\star > 1$, and implement standard branching otherwise. This gives rise to the following four sub problems depending on the values of other δ variables: (i) $\delta_{i^\star,b^\star} = 1$ and $\delta_{i^\star-1,b^\star-1} = 1$; this implies that $\delta_{i^\star,b^\star-1} = 0$ and $\delta_{i,b^\star} = 0$ for all $i < i^\star$; (ii) $\delta_{i^\star,b^\star} = 1$ and $\delta_{i^\star-1,b^\star} = 1$; (iii) $\delta_{i^\star,b^\star} = 0$ and $\delta_{i^\star,b^\star-1} = 1$; (iv) $\delta_{i^\star,b^\star} = 0$ and $\delta_{i^\star,b^\star-1} = 0$; this implies that $\delta_{i^\star-1,b^\star-1} = 0$. Naturally, the first branch is very strong as it implies the existence of a breakpoint between the data point $i^\star - 1$ and $i^\star$, for the join between the linear segments $b^\star - 1$ and $b^\star$. The second two branches contain at least one variable set to one which gives rise to a strong branch, despite no implication on the breakpoint locations. The final branch contains three variables set to zero. However, there is still some impact since neither the linear segment $b^\star - 1$ or $b^\star$ is assigned to data point $i^\star$. Depending on the fractional value of the data point $\delta_{i^\star,b^\star}$ and its surrounding variables, tailoring the branching direction using branch callbacks to lead to the strongest possible branch (with the most variables set to one) should lead to better runtimes and eliminate many weaker solutions where possible. Such a branching rule should also lead to less unbalanced branching trees.

6. Computational experiments

In order to assess the effectiveness of the combinatorial Benders decomposition approach to the PWL function fitting problem, we implemented the model in C++ and embedded it within IBM ILOG-CPLEX version 20.1.0. using standard solver settings. The experiments in this section were run on an Intel 3.00 GHz machine with 16 GB of RAM.

We implement the CBD approach within a branch-and-cut framework, with the combinatorial cuts being added as lazy cuts (see Section E of the Appendix). The IISs of the sub problem are found using the conflict refiner available within CPLEX. The conflict refiner works by assigning preferences to each constraint within the sub problem, analogous to the weights as described in Section D of the Appendix. In order to find multiple IISs of the same sub problem, all preferences are initially set to one (which indicates that the given constraint should be considered for the IIS). Once an IIS has been found, the preference corresponding to one of the constraints that appears in that IIS is set to -1 (indicating that the given constraint should not be considered when finding further IISs). The process repeats by finding further IISs of the same sub problem, consisting of constraints whose preference remains as one (ensuring that the found IISs are all different). Once no more IISs of the sub problem can be generated (i.e., an IIS of the constraints whose preference is still one), the process ends, and the combinatorial cuts relating to each IIS are found.

We run computational experiments on 10 bivariate data sets (with I data points) fitted with a PWL function with a given number of breakpoints, minimizing the maximum absolute difference between the data and the PWL function. The data sets are taken from real world data and present a variety of different landscapes. These data sets are often used for data fitting analyses, including PWL function fitting, regression analysis and time-series analysis. For further details on the data sets, see Section F of the Appendix.

6.1. Improvements from the default approach

We first consider a number of possible improvements that can be made to the default CBD approach, for finding optimal PWL functions with one outlier. In Section 5, we have seen that we can incorporate improvements in cut generation, initialization, and branching within the CBD model. We present comparative results with the default approach, assessing different implementations for each of these three improvements. The full complementary experimental results are found in Section G of the Appendix.

6.1.1. Generating strong cuts from the IIS

In Figure 1, we analyse the effect of allowing multiple IISs to be found in each iteration, leading to the generation of multiple combinatorial cuts. We compare the *default* CBD formulation with two formulations that allow multiple cuts. The two compared formulations differ in the order in which the preferences of the constraints (appearing in the IIS) are set to -1. That is, we present two methods to decide which constraints should not appear in future IISs, after an initial feasible IIS has been found.

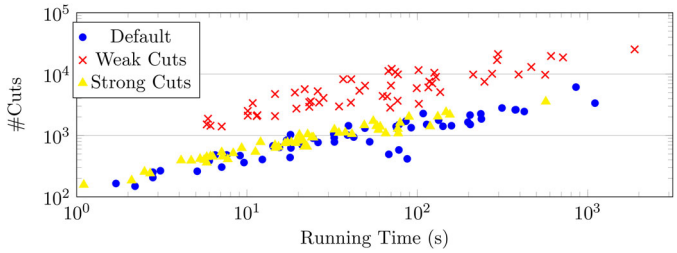

Figure 1. Runtime vs. number of cuts.

The first method prioritises the removal of constraints of the form of (4b)–(4c), which are strong constraints that would imply that one of two binary variables may be causing the infeasibility. After each feasible IIS has been found, the preference for one of these constraints is set to -1 (if such a constraint still appears in the IIS). This means that future IISs would not contain as many constraints of this type. We refer to this method as *Weak Cuts*. The second method prioritises the removal of constraints of the form (4d)–(4e), which would imply that one of three binary variables may be causing the infeasibility. This method ensures that cuts resulting from future IISs will be comparatively stronger, since they would suggest that the infeasibility is being caused by one of a smaller set of binary variables (in comparison with the *Weak Cuts* method). We refer to this method as *Strong Cuts*. Full experimental results are available in Section G.1 of the Appendix.

From Figure 1, we first note that the *Strong Cuts* approach is the fastest overall (from the experimental results in Section G.1 of the Appendix, we see that is it the fastest approach in 45 out of the 55 presented cases (i.e., 82%)). Although it produces a similar amount of cuts as the *Default* method, the faster runtimes indicate that the generated cuts are strong (which is backed up by the experimental results in the Appendix) and can eliminate many infeasible solutions in each iteration due to multiple cuts being produced. Typically, between 95 and 100% of the total cuts produced by each approach were the combinatorial cuts.

For the method implementing *Weak Cuts*, many more cuts were added overall. Furthermore, the experimental results in the Appendix suggest that these cuts were more dense, with each cut containing, on average, more variables than the other two approaches. This meant that many more incremental solutions were produced before the optimal solution was found, and suggests that this method is ineffective at removing non-

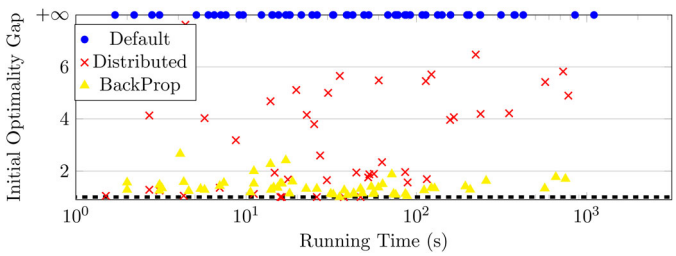

Figure 2. Runtime vs. initial optimality gap (calculated as a ratio to the optimal objective function value, denoted as the black dashed line).

optimal solutions in comparison with the *Strong Cuts* method. In particular, the results highlight that the quality of the generated cuts is more important that the quantity.

6.1.2. Smart initialization approaches

In Figure 2, we analyse whether improving the initialization of the upper bound can lead to speedups. The *BackProp* strategy uses the optimal solution from the smaller problem with no outliers (i.e., the optimal solution found by the CBD approach for a PWL function with $B - 1$ breakpoints) as the initialised solution. However, we note that this information may not always be available (and when it is, it may take a significant amount of time to acquire). Hence, we also consider an alternative strategy. The *distributed* strategy assumes an even distribution of data points to each segment of the PWL function (that is, each data point $i \in [I]$ is assigned to the segment $i \cdot B/I$, rounded as necessary), and takes the solution found by the sub problem as the initialised solution. Full experimental results are available in Section G.2 of the Appendix.

From Figure 2, we first note that while the *BackProp* approach is able to provide initial solutions close to the optimal objective function value, the *Distributed* approach can also provide a good approximation (especially if the data is uniformly distributed, which is seen in the experimental results in the Appendix). For the *Default* setting, the initialised upper bound is $+\infty$. We believe that heuristic approaches to initialization would also be beneficial if they are able to compute a feasible initial solution quickly.

Regarding the runtime of the CBD approach with each setting, we can see the *BackProp* approach is the fastest[1] (from the experimental results in Section G.2 of the Appendix, it is the fastest in 35 out of the 55 presented cases, i.e., 64%). Furthermore, the *Distributed* approach can also provide faster runtimes compared with the *Default* approach. Overall, we can see that providing good initial solutions can significantly improve the running time. However, the cost of acquiring the initialised solution should be considered against the gains in runtime. For future applications, we would consider implementing fast heuristic approaches to approximate the initial solution.

6.1.3. Branching approaches

In Figure 3, we compare the effectiveness of implementing different branching strategies throughout the optimization process on the *default* CBD formulation. The CPLEX parameter *VarSel* decides whether or not to branch on the maximum infeasibility (*MaxInfeas*) or the minimum infeasibility (*MinInfeas*). The *Special* branching rule is implemented using branch callbacks within CPLEX. When a decision to branch on a given variable is made, we tailor the branching to select the δ variable with maximum infeasibility (usually this variable has a fractional value of 0.5), and

[1]For the experimental comparison, we do not include the time taken to find the initial solutions in the presented runtimes for the *Distributed* and *BackProp* settings, as we are using these settings as a benchmark.

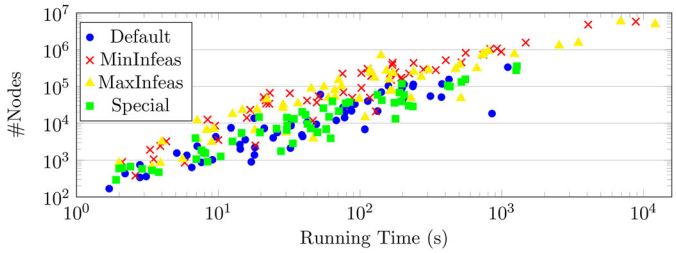

Figure 3. Runtime vs. number of branch-and-bound nodes.

lead the branching to the strongest possible branch, depending on the value of the surrounding variables when the branching decision is made (see the discussion in Section 5). For weaker branches, we ensure the branching tree remains balanced when adjacent δ variables are both infeasible (i.e., $\delta_{i,b}, \delta_{i,b+1} \in (0, 1)$). This is done by fixing one of the variables to one in each branch. We also limit one branch type for each variable, so as not to repeat the same branches in the search tree. Full experimental results are available in Section G.3 of the Appendix.

From Figure 3, we first note that the *MinInfeas* and *MaxInfeas* approaches are slower overall, and visit many more nodes in the branch-and-bound trees. That is, these branching approaches explore more possible solutions and suboptimal branches before finding the optimal solution, leading to faster runtimes. The *Default* and *Special* approaches visit consistently fewer nodes. Regarding the runtimes, the experimental results suggest a slight advantage for the *Special* approach (it is the fastest of the four approaches in 29 of the 55 presented cases, whereas the *Default* approach is the fastest in 21 cases). Furthermore, the *Special* approach leads to more balanced branch-and-bound trees by design, which can be advantageous for solving larger instances.

6.2. Tailored combinatorial Benders decomposition

We now consider an approach that combines the best performing approaches for each of the three improvements discussed in the previous subsection. The presented *Tailored* approach implements *Strong Cuts*, *BackProp* initialization, and *Special* branching from Figures 1, 2, and 3, respectively. We discuss the improvement from the tailored approach against the monolithic MILP approach and the default CBD approach for outlier detection (we present a comparison of the MILP approach with the Benders decomposition approach, and the tailored approach with the algorithmic approaches presented by Wang *et al.* (1993) and Goodrich (1994) in Section H of the Appendix). Recall that outliers are identified during the optimization process, and they do not contribute towards the final objective function value. We consider instances across the 10 data sets for between 6 and 20 breakpoints, and between zero and three outliers. The full detailed results of the comparative experiments are found in Section I of the Appendix, while comparative results for finding PWL functions without outliers are found in Section J of the Appendix. Runtime results for the default

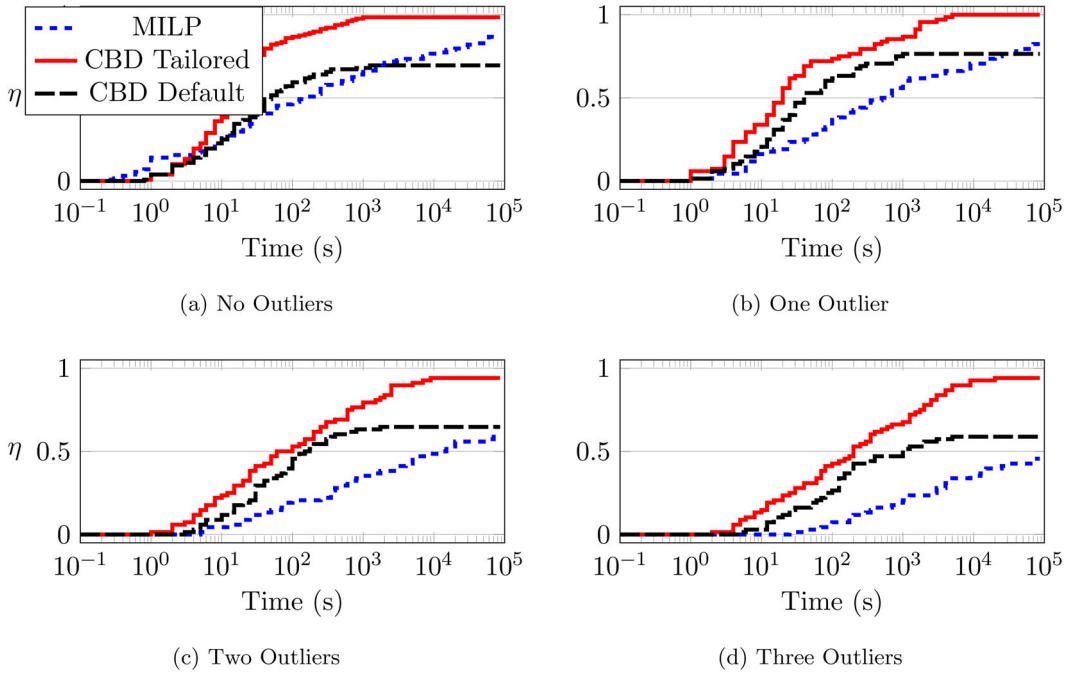

Figure 4. η refers to the fraction of instances solved to optimality within the given time.

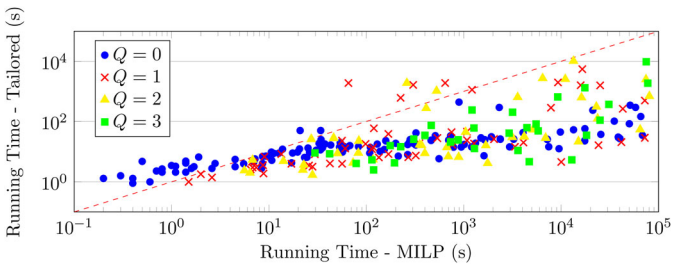

Figure 5. Comparative running times: MILP vs. Tailored.

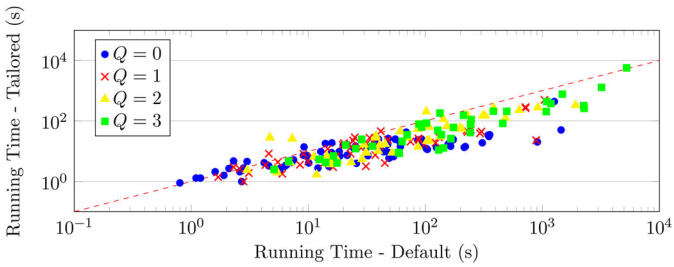

Figure 6. Comparative running times: Default vs. Tailored.

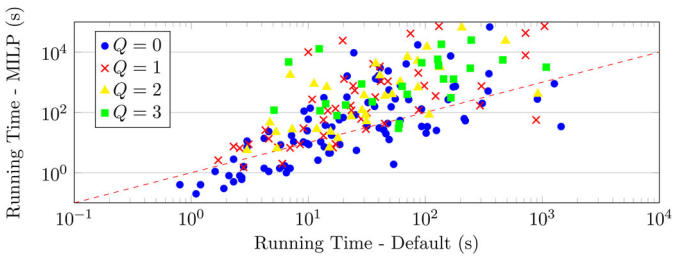

Figure 7. Comparative running times: MILP vs. Default.

CBD approach are found in Section K of the Appendix. We illustrate the results in Figures 4-7.

Figure 4 shows the fraction of instances that are solved within a given time limit for the three approaches for up to three outliers. As the number of outliers increases, we see a comparatively better performance of the tailored CBD method. For any time budget above 5 seconds, the tailored CBD approach is preferable for calculating optimal PWL functions without outliers, and is preferable in calculating optimal PWL functions with outliers for any given time budget. The default CBD approach performs well in comparison with the MILP approach; however, the improvements embedded within the tailored approach allow it to solve more instances within a given time budget across all problem sizes and instances.

Figures 5-7 show direct pairwise comparisons between the running times of the three approaches. For these comparisons, we have excluded any instances where either approach fails (either exceeding the time or memory limits). The line $y = x$ is shown on all figures. In particular, from Figure 5 and Figure 6, we see how the Tailored approach is overall faster than the MILP approach and the default CBD approach, respectively. From Figure 7, we see that the even the default CBD approach is preferable to the MILP approach, especially for instances with outliers.

7. Conclusions

We have implemented combinatorial Benders cuts into the recently presented MILP model used to fit PWL functions to discrete data. We first implemented inbuilt outlier detection into the MILP model, which exhibits a special structure containing a large amount of binary variables and big-$\mathcal{M}$ constraints, and hence the CBD approach fits well. The cuts found by the CBD approach can also be implemented into the standard MILP model.

By taking advantage of problem-specific knowledge, we showed that improvements in the CBD model can be found. These including implementing a larger quantity and quality of combinatorial cuts. Furthermore, implementing an

initialization approach that gives a good estimation on the optimal value of the objective function can also lead to faster solution times. The implementation of a special branching rule also proved effective. The tailored CBD model takes advantage of these improvements and leads to a greatly improved model.

We have shown empirically that the tailored CBD approach leads to vast speedups over the standard MILP approach, especially when there are a larger number of breakpoints and multiple outliers. For cases with nine or more breakpoints and two or more outliers, the CBD approach is faster in all tested instances. Furthermore, the CBD approach solves a larger percentage of the presented instances than the MILP approach when the time limit is 10 seconds or more. These results can be easily extended to model non-linear continuous functions with PWL functions more accurately.

Acknowledgments

We are very grateful for the detailed and thorough comments from the editor and two anonymous reviewers.

Funding

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) [Grant 445857709].

Notes on contributors

John Alasdair Warwicker completed a master's degree in Mathematics from Loughborough University and a PhD in Theoretical Computer Science from the University of Sheffield. Since 2019, he has been working as a research associate at the Institute of Operations Research at Karlsruhe Institute of Technology, in the chair of stochastic optimization. His research interests lie on the border of discrete mathematics, computer science and operations research. Specific areas of interest include formulations and decomposition approaches for mathematical programming models, and analyses of heuristics and hyper-heuristics.

Steffen Rebennack completed a degree in Mathematics at Heidelberg University, and a master's degree in Industrial & Systems Engineering at the University of Florida. He also obtained a PhD in industrial & systems engineering from the University of Florida, before working at the Colorado School of Mines as an assistant professor and an associate professor. Since 2017, he has been working as chair professor for the stochastic optimization group in the Institute of Operations Research at Karlsruhe Institute of Technology. His research interests include stochastic and large-scale global optimization problems, with a focus on applications in power systems analysis.

ORCID

John Alasdair Warwicker http://orcid.org/0000-0002-6274-2638

References

Bellman, R. and Roth, R. (1969) Curve fitting by segmented straight lines. *Journal of the American Statistical Association*, **64**(327), 1079–1084.

Benders, J.F. (1962) Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, **4**(1), 238–252.

Berman, N.G., Wong, W.K., Bhasin, S. and Ipp, E. (1996) Applications of segmented regression models for biomedical studies. *American Journal of Physiology-Endocrinology and Metabolism*, **270**(4), E723–E732.

Bertsimas, D. and King, A. (2016) OR forum—an algorithmic approach to linear regression. *Operations Research*, **64**(1), 2–16.

Bertsimas, D., King, A. and Mazumder, R. (2016) Best subset selection via a modern optimization lens. *The Annals of Statistics*, **44**(2), 813–852.

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

Bertsimas, D. and Shioda, R. (2007) Classification and regression via integer optimization. *Operations Research*, **55**(2), 252–271.

Chakravarti, N. (1994) Some results concerning post-infeasibility analysis. *European Journal of Operational Research*, **73**(1), 139–143.

Chang, C. (1973) Pattern recognition by piecewise linear discriminant functions. *IEEE Transactions on Computers*, **22**(9), 859–862.

Chatzinakos, C., Pitsoulis, L. and Zioutas, G. (2016) Optimization techniques for robust multivariate location and scatter estimation. *Journal of Combinatorial Optimization*, **31**(4), 1443–1460.

Chen, D.Z. and Wang, H. (2009) Approximating points by a piecewise linear function, in Y. Dong, D. Z. Du, O. Ibarra (eds.), *Algorithms and Computation*. ISAAC 2009. Lecture Notes in Computer Science, vol. 5878. Springer, Berlin, Heidelberg.

Codato, G. and Fischetti, M. (2006) Combinatorial Benders cuts for mixed-integer linear programming. *Operations Research*, **54**, 756–766.

Ertel, J.E. and Fowlkes, E.B. (1976) Some algorithms for linear spline and piecewise multiple linear regression. *Journal of the American Statistical Association*, **71**(355), 640–648.

Feijoo, B. and Meyer, R.R. (1988) Piecewise-linear approximation methods for nonseparable convex optimization. *Management Science*, **34**(3), 411–419.

Frank, S.M. and Rebennack, S. (2015) Optimal design of mixed AC–DC distribution systems for commercial buildings: A nonconvex generalized Benders decomposition approach. *European Journal of Operational Research*, **242**(3), 710–729.

Geißler, B., Martin, A., Morsi, A. and Schewe, L. (2012) Using piecewise linear functions for solving MINLPs, in J. Lee, S. Leyffer (eds.), *Mixed Integer Nonlinear Programming*. The IMA Volumes in Mathematics and its Applications, vol. 154. Springer, New York, NY.

Goldberg, N., Kim, Y., Leyffer, S. and Veselka, T.D. (2014) Adaptively refined dynamic program for linear spline regression. *Computational Optimization and Applications*, **58**(3), 523–541.

Goldberg, N., Rebennack, S., Kim, Y., Krasko, V. and Leyffer, S. (2021) MINLP formulations for continuous piecewise linear function fitting. *Computational Optimization and Applications*, **79**(1), 223–233.

Goodrich, M.T. (1994) Efficient piecewise-linear function approximation using the uniform metric, in *Proceedings of the Tenth Annual Symposium on Computational Geometry*, (SCG '94), pp. 322–331. Association for Computing Machinery, New York, NY, USA.

Gunnerud, V. and Foss, B. (2010) Oil production optimization—a piecewise linear model, solved with two decomposition strategies. *Computers & Chemical Engineering*, **34**(11), 1803–1812.

Hakimi, S. and Schmeichel, E. (1991) Fitting polygonal functions to a set of points in the plane. *CVGIP: Graphical Models and Image Processing*, **53**(2), 132 – 136.

Hawkins, D.M. and Olive, D. (1999) Applications and algorithms for least trimmed sum of absolute deviations regression. *Computational Statistics & Data Analysis*, **32**(2), 119–134.

Hooker, J. and Ottosson, G. (2003) Logic-based Benders decomposition. *Mathematical Programming*, **96**(1), 33–60.

Jupp, D.L.B. (1978) Approximation to data by splines with free knots. *SIAM Journal on Numerical Analysis*, **15**(2), 328–343.

Koenker, R. and Bassett, G. (1985) On Boscovich's estimator. *The Annals of Statistics*, **13**(4), 1625–1628.

Kong, L. and Maravelias, C.T. (2020) On the derivation of continuous piecewise linear approximating functions. *INFORMS Journal on Computing*, **32**(3), 531–546.

Li, X., Tomasgard, A. and Barton, P.I. (2011) Nonconvex generalized Benders decomposition for stochastic separable mixed-integer nonlinear programs. *Journal of Optimization Theory and Applications*, **151**, 343—389.

Lohmann, T. and Rebennack, S. (2017) Tailored Benders decomposition for a long-term power expansion model with short-term demand response. *Management Science*, **63**(6), 2027–2048.

Muggeo, V.M.R. (2003) Estimating regression models with unknown break-points. *Statistics in Medicine*, **22**(19), 3055–3071.

Pfetsch, M. (2003) The maximum feasible subsystem problem and vertex-facet incidences of polyhedra. Doctoral thesis, Technische Universität Berlin, Fakultät II - Mathematik und Naturwissenschaften, Berlin.

Rahmaniani, R., Crainic, T.G., Gendreau, M. and Rei, W. (2017) The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, **259**(3), 801–817.

Rebennack, S. (2016a) Computing tight bounds via piecewise linear functions through the example of circle cutting problems. *Mathematical Methods of Operations Research*, **84**(1), 3–57.

Rebennack, S. (2016b) Combining sampling-based and scenario-based nested Benders decomposition methods: application to stochastic dual dynamic programming. *Mathematical Programming*, **156**(1), 343–389.

Rebennack, S. and Kallrath, J. (2015) Continuous piecewise linear delta-approximations for bivariate and multivariate functions. *Journal of Optimization Theory and Applications*, **167**(1), 102–117.

Rebennack, S. and Krasko, V. (2020) Piecewise linear function fitting via mixed-integer linear programming. *INFORMS Journal on Computing*, **32**(2), 507–530.

Rebennack, S., Prokopyev, O.A. and Singh, B. (2020) Two-stage stochastic minimum s-t cut problems: Formulations, complexity and decomposition algorithms. *Networks*, **75**(3), 235–258.

Rei, W., Cordeau, J.-F., Gendreau, M. and Soriano, P. (2009) Accelerating Benders decomposition by local branching. *INFORMS Journal on Computing*, **21**(2), 333–345.

Rousseeuw, P.J. (1984) Least median of squares regression. *Journal of the American Statistical Association*, **79**(388), 871–880.

Sudermann-Merx, N. and Rebennack, S. (2021) Leveraged least trimmed absolute deviations. *OR Spectrum*.

Toriello, A. and Vielma, J.P. (2012) Fitting piecewise linear continuous functions. *European Journal of Operational Research*, **219**(1), 86–95.

Wagner, A.K., Soumerai, S.B., Zhang, F. and Ross-Degnan, D. (2002) Segmented regression analysis of interrupted time series studies in medication use research. *Journal of Clinical Pharmacy and Therapeutics*, **27**(4), 299–309.

Wang, D.P., Huang, N.F., Chao, H.S. and Lee, R.C.T. (1993) Plane sweep algorithms for the polygonal approximation problems with applications, in *International Symposium on Algorithms and Computation*, pp. 515–522. Springer, Berlin, Heidelberg.

Warwicker, J.A. and Rebennack, S. (2022) A comparison of two mixed-integer linear programs for piecewise linear function fitting. *INFORMS Journal on Computing*, **34**(2), 1042–1047.