ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, IL 60439

# Adaptively Refined Dynamic Program for Linear Spline Regression

**Noam Goldberg, Youngdae Kim, Sven Leyffer, and Thomas Veselka**

Mathematics and Computer Science Division

Preprint ANL/MCS-P3040-0912

September 25, 2012

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

# Adaptively Refined Dynamic Program for Linear Spline Regression

Noam Goldberg[*] Youngdae Kim[†] Sven Leyffer[‡] Thomas D. Veselka[§]

September 25, 2012

**Abstract**

The linear spline regression problem is to determine a piecewise linear function for estimating a set of given points while minimizing a given measure of misfit or error. This is a classical problem in computational statistics and operations research; dynamic programming was proposed as a solution technique more than forty years ago by Bellman and Roth (1969). The algorithm requires a discretization of the solution space to define a grid of candidate breakpoints. This paper proposes an adaptive refinement scheme for the grid of candidate breakpoints in order to allow the dynamic programming method to scale for larger instances of the problem. We evaluate the quality of solutions found on small instances compared with optimal solutions determined by a novel integer programming formulation of the problem. We also consider a generalization of the linear spline regression problem to fit multiple curves that share breakpoint horizontal coordinates, and we extend our method to solve the generalized problem. Computational experiments verify that our nonuniform grid construction schemes are useful for computing high-quality solutions for both the single-curve and two-curve linear spline regression problem.

**Keywords:** Piecewise regression, least squares, change point detection, dynamic programming, mixed integer programming.

## 1 Problem Description and Survey of Literature

The linear spline regression problem, given a dataset $D = (x \;\; y) \in \mathbb{R}^{n\times 2}$, is to determine a continuous piecewise linear function with $m$ line segments that best fits the data as measured by a given error function $e : D \to \mathbb{R}$. The data consists of an *independent variable* vector $x \in \mathbb{R}^n$ and a *dependent variable* vector $y \in \mathbb{R}^n$; without losing generality it is assumed that $x_1 \le \cdots \le x_n$. Continuity implies that every pair of incident line segments must intersect at a breakpoint. Also, note that the methods herein generalize for the case that $x$ is a matrix but there is a single ordered dimension (e.g., time) in which the function is piecewise linear with multiple pieces and otherwise the function is linear.

The estimated function has a set of $m + 1$ breakpoints denoted by $b = [b_{ij}] \in \mathbb{R}^{(m+1)\times 2}$. The breakpoints must be determined within a domain of interest $I \subset \mathbb{R}$; typically $I = [x_1, x_n]$ or $I = [0, x_n]$. The error over the line segments for given data $D$ is measured by one of the following:

Absolute error:
$$e_1(D, b) = \sum_{j=1}^{m} \sum_{\substack{i=1,\dots,n:\\ b_{j1}\le x_i\le b_{j+1,1}}} \left| y_i - \left( \frac{b_{j+1,2} - b_{j,2}}{b_{j+1,1} - b_{j1}} (x_i - b_{j1}) + b_{j1} \right) \right|. \tag{1}$$

Sum of squared error:
$$e_2(D, b) = \sum_{j=1}^{m} \sum_{\substack{i=1,\dots,n:\\ b_{j1}\le x_i\le b_{j+1,1}}} \left[ y_i - \left( \frac{b_{j+1,2} - b_{j2}}{b_{j+1,1} - b_{j1}} (x_i - b_{j1}) + b_{j1} \right) \right]^2. \tag{2}$$

Maximum of absolute errors:
$$e_\infty(D, b) = \sum_{j=1}^{m} \max_{\substack{i=1,\dots,n:\\ b_{j1}\le x_i\le b_{j+1,1}}} \left| y_i - \left( \frac{b_{j+1,2} - b_{j2}}{b_{j+1,1} - b_{j1}} (x_i - b_{j1}) + b_{j1} \right) \right|. \tag{3}$$

[*]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, noamgold@mcs.anl.gov.
[†]Department of Computer Sciences, University of Wisconsin-Madison, youngdae@cs.wisc.edu.
[‡]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, leyffer@mcs.anl.gov.
[§]Decision and Information Sciences, Argonne National Laboratory, Argonne, IL 60439, tdveselka@anl.gov.

Bellman and Roth [3] proposed a dynamic programming algorithm for minimizing (3) over a given (uniform) grid of candidate breakpoints. Guthery [6] proposed an efficient dynamic program to fit a segmented curve that is not required to be a continuous function in order to minimize (2). Ertel and Flowlkes [4] developed heuristics to solve both problems. More recently, in the econometrics community, Bai and Perron [2] revisited the problem considered by Guthery. Also, Aronov et al. [1] suggest an approximation algorithm for the problem considered herein and similar to that of [3] with a runtime complexity that is a high degree polynomial in the number of data points; hence it may not be efficient in practice. The problem's computational complexity remains an open problem according to [1]. In recent work, Toridello and Viemla [8] proposed integer programming formulations for a related curve-fitting problem.

In the following section we propose an integer programming formulation for the linear spline regression problem. In Section 3 we give an overview of the dynamic programming method. In Section 4 we consider two approaches to constructing tractable grids of candidate breakpoints for applying the dynamic program: the first is a static scheme, and the second is a dynamic scheme that adaptively refines the grid based on a current solution. We conclude with numerical experiments in Section 6.

## 2 A Mixed Integer Programming Formulation

In this section, we consider a mixed-integer programming (MIP) formulation for the problem. The following tables define the variables and constants of the formulation.

| Variable | Meaning |
|---|---|
| $\beta_j$ | The slope of the $j$-th line segment, i.e., $\beta_j = \frac{b_{j2}-b_{j-1,2}}{b_{j1}-b_{j-1,1}}$. |
| $\alpha_j$ | The intercept of the $j$-th line segment, i.e., $\alpha_j = b_{j-1,2} + \beta_j(b_{j1} - b_{j-1,1})$. |
| $\phi_{ij}$ | Indicating whether the error of the $i$-th data point should should contribute to the error of the $j^{\text{th}}$ line segment, i.e., $\phi_{ij} = \begin{cases} 0 & \text{if } b_{j-1,1} \le x_i \le b_{j1} \\ 1 & \text{otherwise} \end{cases}$ |
| $\xi_i$ | $\xi_i = \lvert y_i - (\beta_j x_i + \alpha_j)\rvert$, the difference between $y$ value of the $i$th data point and the estimated $y$ value of the line segment. |

| Constant | Meaning |
|---|---|
| $M_1, M_2, M_3$ | Large ("big-M") constants – to satisfy the associated inequalities for $i = 1, \dots, n$. For example, it suffices to set $M_1 \ge \max_{\substack{i,j,k=1,\dots,n \\ i\neq j}} \left\lvert y_k - y_j - \frac{y_i-y_j}{x_i-x_j}(x_k - x_j)\right\rvert$. |

We first consider a nonconvex nonlinear formulation for the general problem:

$$\underset{\beta,\phi,\xi,b}{\text{minimize}} \quad \sum_{i=1}^{n} \xi_i^2 \tag{4a}$$

$$\text{subject to} \quad |y_i - b_{j2} - \beta_j x_i| \le \xi_i + M_1\phi_{ij} \qquad i = 1,\dots,n, \quad j = 1,\dots,m \tag{4b}$$

$$\sum_{j=1}^{m} \phi_{ij} = m - 1 \qquad i = 1,\dots,n \tag{4c}$$

$$- M_2\phi_{ij} \le b_{j1} - x_i \qquad i = 1,\dots,n, \quad j = 1,\dots,m \tag{4d}$$

$$- M_2\phi_{ij} \le x_i - b_{j-1,1} \qquad i = 1,\dots,n, \quad j = 2,\dots,m \tag{4e}$$

$$b_{j1} = \frac{b_{j+1,2} - b_{j2}}{\beta_j - \beta_{j+1}} \qquad j = 1,\dots,m-1 \tag{4f}$$

$$\phi_{ij} \in \{0,1\} \qquad i = 1,\dots,n, \quad j = 1,\dots,m. \tag{4g}$$

Here we minimize the error loss function $e_2(D,\cdot)$. Note that by replacing the objective function with other loss functions such as (1) or (3), we can instead consider, respectively, the minimization of the sum of absolute errors, or the minimization of the maximum of absolute errors. The constraints (4b) imply that for $i = 1,\dots,n$, $\xi_i$ measures the deviation of $y_i$ and $\alpha_j + \beta_j x_i$ for the segment $j$ to which data point $i$ is assigned; in other words, if $\phi_{ij} = 0$, then the constraints (4b) require that $\xi_i = |y_i - (\beta_j x_i + \alpha_j)|$. Otherwise,

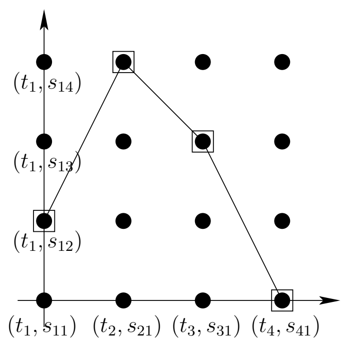


Figure 1: Example of a uniform grid $\Omega = \bigcup_{i=1}^{4} \bigcup_{j=1}^{4} \{(t_i, s_{ij})\}$.

the constraints are automatically satisfied by using the sufficiently large constant $M_1$. The constraints (4d)-(4e) force each data point to be assigned to exactly one of the $m$ line segments and its error to be respectively evaluated by using that line segment.

Assuming that the fitted curve is either concave or convex, we can rewrite (4) as a convex mixed integer quadratic program (MIQP). The MIQP formulation is obtained from (4) by substituting the $b_{j1}$ variables in (4d) and (4e) and replacing $M_2$ by a constant $M_3$ satisfying $M_3 > (\beta_j - \beta_{j+1})M_2$ for $j = 1, \ldots, m$; if the function to be estimated is Lipschitz with constant $L$, then we may let

$$M_3 = L \cdot M_2 \geq \max_{j=1,\ldots,m} \beta_j M_2 \geq \max_{j=1,\ldots,m} \{\beta_j - \beta_{j+1}\} M_2.$$

The resulting formulation is as follows:

$$\begin{aligned}
&\underset{\alpha,\beta,\phi,\xi}{\text{minimize}} && \sum_{i=1}^{n} \xi_i^2 && && \text{(5a)}\\
&\text{subject to} && |y_i - (\beta_j x_i + \alpha_j)| \leq \xi_i + M_1 \phi_{ij} && \text{for } i = 1,\ldots,n,\ j = 1,\ldots,m && \text{(5b)}\\
& && \sum_{j=1}^{m} \phi_{ij} = m - 1 && \text{for } i = 1,\ldots,n && \text{(5c)}\\
& && -M_3\phi_{ij} \leq \alpha_j - \alpha_{j+1} - (\beta_{j+1} - \beta_j)x_i && \text{for } i = 1,\ldots,n,\ j = 1,\ldots,m-1 && \text{(5d)}\\
& && -M_3\phi_{ij} \leq \alpha_j - \alpha_{j-1} + (\beta_j - \beta_{j-1})x_i && \text{for } i = 1,\ldots,n,\ j = 2,\ldots,m && \text{(5e)}\\
& && \beta_{j+1} \leq \beta_j && \text{for } j = 1,\ldots,m-1 && \text{(5f)}\\
& && \phi_{ij} \in \{0,1\} && \text{for } i = 1,\ldots,n,\ j = 1,\ldots,m. && \text{(5g)}
\end{aligned}$$

The constraints (5d)-(5e) follow from (4d)-(4e) given that $\beta_j \geq \beta_{j+1}$ for $j = 1, \ldots, m$ (when the function to be estimated is a concave piecewise linear function). Note that if instead of (5f) we required that $\beta_j \leq \beta_{j+1}$ for all $j$ and replaced the inequalities in (5d)-(5e) with the reverse inequalities, then (5) could be used to estimate a convex piecewise linear function. The constraint (5f) ensures that the resulting continuous piecewise linear function is concave.

Note that an MIQP formulation for a related problem has been proposed by Toriello and Viemla [8]. A key difference between their formulation and (5) is that it does not require the estimated curve to correspond to a function over the domain defined by the data points; in other words, in [8] the error is computed as a function of the vertical distances from the data points to the closest line segments, whereas in our case a set of points between adjacent breakpoints must always be assigned to the same line segment. We use formulation (5) as a benchmark for numerical results in Section 6.

Next we review the dynamic programming method for solving this problem.

## 3 Dynamic Programming for Linear Spline Regression

Bellman and Roth [3] proposed dynamic programming (DP) for determining a continuous piecewise linear function that minimizes the error function $e_\infty(\cdot)$. The Bellman optimality equations of the DP are expressed for a given grid system consisting of a finite number of candidate breakpoints over which the error is minimized [3]. Figure 1 shows an example of the uniform grid system considered in [3] along with a piecewise linear solution curve with three line segments.

We now write the Bellman equations of the linear spline regression problem DP [3]. First, let $\Omega \in \mathbb{R}^2$ denote a finite set of candidate breakpoints. For a positive integer $p$ let $t \in \mathbb{R}^p$ denote the vector of breakpoint $x$-coordinates of a grid $\Omega$, with $t_1 < \cdots < t_p$. Define $S : \mathbb{R} \to 2^n$, where $S(\bar{x})$ is the set of indexes that share the same $x$-coordinate value $\bar{x}$. For example, in Figure 1, $S(t_1) = \{1, 2, 3, 4\}$. For convenience we also write the breakpoints as $\Omega = \bigcup_{i=1}^{p} \{(t_i, s_{ij}) \mid j \in S(t_i)\} \in \mathbb{R}^2$; we assume that $s_{i1} < \cdots < s_{i|S(t_i)|}$ for $i = 1, \ldots, p$.

For $k, l \in \{1, \dots, p\}$ with $t_k < t_i$, $l \in S(t_k)$ and $j \in S(t_i)$, we denote the error of segment $(t_k, s_{kl}), (t_i, s_{ij})$ by $E(t_k, s_{kl}, t_i, s_{ij})$; for the squared error function $e_2(\cdot)$ it is given by

$$E(t_k, s_{kl}, t_i, s_{ij}) = \sum_{\substack{r=1,\dots,n:\\ t_k \le x_r \le t_i}} \left\{ y_r - \left( \frac{s_{ij} - s_{kl}}{t_i - t_k} (x_r - t_k) + s_{kl} \right) \right\}^2 .$$

Let $F_{\widehat{m}}(i,j)$ denote the optimal error of $\widehat{m}$ line segments when the right breakpoint of the last segment is $(t_i, s_{ij})$. Then, letting $\widehat{m} \in \{2, \dots, m\}$, the optimality equations of the DP are given by:

$$F_{\widehat{m}}(i,j) = \min_{k=\widehat{m},\dots,i-1, l \in S(t_k)} \{E(t_k, s_{kl}, t_i, s_{ij}) + F_{\widehat{m}-1}(k,l)\} \text{ for } i = \widehat{m}+1, \dots, p-m+\widehat{m}, j \in S(t_i). \tag{6a}$$

$$F_1(i,j) = \min_{l \in S(t_1)} E(t_1, s_{1l}, t_i, s_{ij}) \text{ for } i = 2, \dots, p-m+1, j \in S(t_i). \tag{6b}$$

The method requires the evaluation of $F_1(\cdot), \dots, F_m(\cdot)$ to determine an optimal continuous piecewise linear function with $m$ line segments given $\Omega$. To obtain the optimal solution value using (6), we solve for

$$\min_{j \in S_{t_p}} F_m(p,j). \tag{7}$$

To determine the optimal solution, we must also store the breakpoints $(t_k, s_{k\ell})$ at which the minimum of $F_{\widehat{m}}(t_i, s_{ij})$ is attained for $\widehat{m} = 1, \dots, m$, for each $i \in \{1, \dots, p\}$ and each $j \in S(t_i)$.

## 4 Constructing a Tractable Grid for Dynamic Programming

The density of the grid directly affects the performance of the DP method. A denser grid with many candidate breakpoints yields lower error values, but this result is at the expense of significant degradation of runtime performance. Table 1 shows that lower error values could be achieved by increasing the number of candidate breakpoints, although, at the expense of significant increases in the runtime. For example, with a total of 10,000 candidate breakpoints, we could lower the error value by 1.9% compared with 500 candidate breakpoints. However, the runtime increased nearly 400-fold.

Table 1: Running time vs. lower error values for the UCI MPG dataset [5] with $n = 398$ and $m = 5$.

| $\lvert\Omega\rvert$ | Time (sec) | Objective Value |
|---:|---:|---:|
| 500 | 0.056 | 6,830.611 |
| 1,000 | 0.201 | 6,812.490 |
| 5,000 | 4.981 | 6,718.609 |
| 10,000 | 19.917 | 6,697.479 |

If the grid has $|S(t_i)| = q$ for $i = 1, \dots, p$, then letting $N = |\Omega| = pq$, the number of candidate breakpoints is $O(N)$. If the segment errors are precomputed, then the DP running time is bounded by $O(N^2 m)$; for a detailed analysis see Appendix A. Hence, it is essential to construct a grid that will be tractable in terms of running time satisfactory in terms of the quality of the solutions.

Bellman and Roth [3] do not consider a particular grid system. In the example introduced in [3], a uniform grid is assumed, as depicted in Figure 1. For low-accuracy solutions a static uniform grid may be adequate. In order to improve on the accuracy, however, a finer grid may be required, which may not be tractable using the uniform grid approach.

Figure 2 shows that there are many cases in which the range of useful $y$-coordinate values of candidate breakpoints can be significantly narrowed down to lie within close proximity to the given data. Based on this observation we suggest two types of techniques for constructing (nonuniform) grid systems. In Section 4.1, we describe a grid system which generates a fixed number of candidate breakpoints near data points. In Section 4.2 we consider an adaptive construction of a grid. First we solve using a coarse grid. Then, we generate additional candidate breakpoints that lie within a neighborhood of the current solution, and we iterate until a termination condition is satisfied.

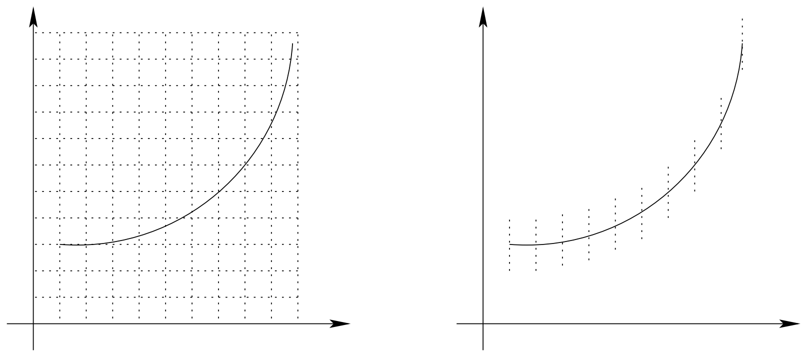


Figure 2: Uniform grid vs. nonuniform grid. In the uniform grid, many of the candidate breakpoints do not help much in approximating the given curve, depicted as a solid line. In the nonuniform grid, however, we generate candidate breakpoints that are expected to improve the approximation of this curve.

## 4.1 DP-static: Placing a Fixed Number of Candidate Breakpoints near the Data Points

Here we propose a scheme for constructing a nonuniform grid of candidate breakpoints; we call this scheme DP-static. The input of the DP-static scheme consists of the dataset $D$, an integer $p$, and an odd integer $q$.

DP-static generates $p$ equally spaced $x$-coordinate values $t_1 < \cdots < t_p$ that subdivide the interval $[\min_i x_i, \max_i x_i]$ into $(p-1)$ sub-intervals. The values $t_1, \ldots, t_p$ constitute the $x$-coordinate values of the candidate breakpoints.

For each $t_i$ we determine

$$k \in \underset{r=1,\ldots,n}{\operatorname{argmin}} \{|x_r - t_i| \mid x_r \le t_i\} \qquad \text{and} \qquad \ell \in \underset{r=1,\ldots,n}{\operatorname{argmin}} \{|x_r - t_i| \mid t_i \le x_r\}.$$

Then, for $i = 1, \ldots, n$ we compute $s_{i1}, \ldots, s_{iq}$. Since $q$ is odd, it follows that $\lceil \frac{q}{2} \rceil$ is the median index of $s_{i1}, \ldots, s_{iq}$. For $c = \frac{\max_i y_i - \min_i y_i}{2n}$ and $j = 1, \ldots, q$ we let

$$s_{ij} = s_{i\lceil \frac{q}{2} \rceil} + \left(j - \left\lceil \frac{q}{2} \right\rceil\right) c.$$

Note that $s_{i\lceil \frac{q}{2} \rceil} = \frac{y_k + y_l}{2}$. The resulting breakpoints are the $\bigcup_{i=1,\ldots,p} \{(t_i, s_{i1}), \ldots, (t_i, s_{iq})\}$. An example of the resulting grid is depicted on the right of Figure 2.

## 4.2 DP-adaptive: Adaptive Refinement

The statically determined grid may not be adequate when the data points do not lie close enough to the optimal breakpoints. When a region of data space requires a larger number of candidate breakpoints in order to obtain adequate solutions, it may be useful to employ an iterative procedure based on the current solution. Also, through an adaptive refinement scheme it may be possible to attain competitive error values by using a sparser grid and smaller number of candidate breakpoints.

Based on this intuition, we propose an adaptive refinement scheme. The algorithm is initialized with a coarse grid. At each iteration, we apply the DP algorithm to the current grid and adaptively refine only those rectangles that intersect the DP solution curve. Algorithm 1 describes our simple procedure for refining a subset of the rectangles, subdividing each into four subrectangles by halving each one of its sides. Figure 3 shows an example that applies a few steps of the adaptive refinement algorithm. At the first iteration we start with a large rectangle with four candidate breakpoints (denoted by circles) corresponding to the rectangle's vertices. After applying the DP, we refine the rectangles that intersect the line segments, by creating more candidate breakpoints at the center of the rectangles as well as midpoints of each rectangle's sides. The algorithm repeats the refinements step until a termination criterion is met. Algorithm 2 implements the DP-adaptive scheme.

We use $\{[(t_k, s_{kl}), (t_i, s_{ij})]\}$ with $t_k < t_i$ and $s_{kl} < s_{ij}$ to denote a rectangle with bottom left point $(t_k, s_{kl})$ and top right point $(t_i, s_{ij})$. Algorithm 2 is typically initialized with a rectangle set consisting of a single rectangle:

$$R \leftarrow \left\{ \left[ \left( \min_{i=1,\ldots,n} x_i, \min_{i=1,\ldots,n} y_i \right), \left( \max_{i=1,\ldots,n} x_i, \max_{i=1,\ldots,n} y_i \right) \right] \right\}. \tag{8}$$

In the following for a finite $C = \{p_1, p_2, \ldots\} \subseteq \mathbb{R}^2$ assume $p_{11} \le p_{21} \le \cdots \le p_{|C|1}$, and define

$$\mathcal{L}(C) = \{\lambda p_i + (1-\lambda) p_{i+1} \mid i = 1, \ldots, |C|, 0 \le \lambda \le 1\};$$

$\mathcal{L}(C)$ corresponds to the line segments that connect the points in $C$ that are pairwise adjacent along the $x$-axis.

**Algorithm 1** refineGrid($R, \hat{R}, \Omega$), refine a subset of rectangles of a given grid

```
Input: A set of rectangles $R$, current grid $\Omega$, subset of rectangles $\widehat{R}$ to be refined
1: for each $[(t_k, s_{kl}), (t_i, s_{ij})] \in \widehat{R}$ do
2:     $r_1 \leftarrow [(t_k, s_{kl}), \left(\frac{t_k+t_i}{2}, \frac{s_{kl}+s_{ij}}{2}\right)]$, $r_2 \leftarrow [\left(t_k, \frac{s_{kl}+s_{ij}}{2}\right), \left(\frac{t_k+t_i}{2}, s_{ij}\right)]$
3:     $r_3 \leftarrow [\left(\frac{t_k+t_i}{2}, \frac{s_{kl}+s_{ij}}{2}\right), (t_i, s_{ij})]$, $r_4 \leftarrow [\left(\frac{t_k+t_i}{2}, s_{kl}\right), \left(t_i, \frac{s_{kl}+s_{ij}}{2}\right)]$.
4:     $R \leftarrow R \cup \{r_1, r_2, r_3, r_4\} \setminus \{[(t_k, s_{kl}), (t_i, s_{ij})]\}$
5:     $b_1 \leftarrow \left(t_k, \frac{s_{kl}+s_{ij}}{2}\right)$, $b_2 \leftarrow \left(\frac{t_k+t_i}{2}, s_{kl}\right)$, $b_3 \leftarrow \left(\frac{t_k+t_i}{2}, \frac{s_{kl}+s_{ij}}{2}\right)$, $b_4 \leftarrow \left(\frac{t_k+t_i}{2}, s_{ij}\right)$
6:     $b_5 \leftarrow \left(t_i, \frac{s_{kl}+s_{ij}}{2}\right)$.
7:     $\Omega \leftarrow \Omega \cup \{b_1, b_2, b_3, b_4, b_5\}$.
8: end for
Output: $R, \Omega$.
```

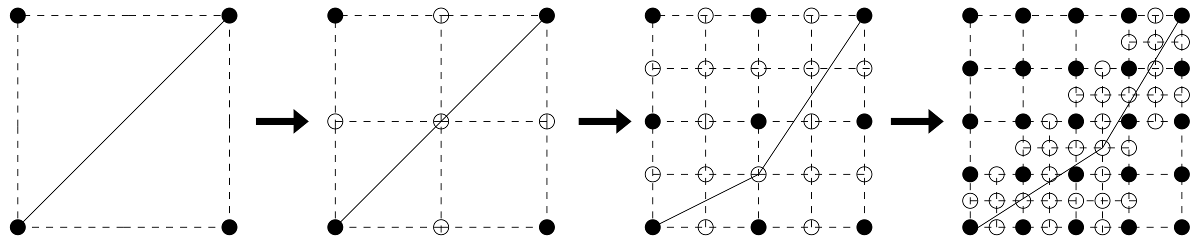


Figure 3: Adaptive refinement steps to find a continuous piecewise linear function with two line segments. Unfilled circles denote newly added candidate breakpoints at each iteration. The graph of the piecewise linear function in each step consists of the line segments connecting the circles. The curve minimizes the error with respect to the current grid consisting of both filled and unfilled circles. For the sake of clarity, the data points are not displayed in this diagram.

**Algorithm 2** Adaptive refinement for a single curve (DP-adaptive)

```
Input: Dataset $D \subset \mathbb{R}^2$, the number of line segments $m$, iteration limit $T$,
    initial grid $\Omega \subset \mathbb{R}^2$, and rectangle set $R$
1: for $k = 1, \ldots, T$ do
2:     $m' \leftarrow \min\{m, k\}$
3:     Evaluate (7) on $\Omega$ and let $C \subseteq \Omega$ denote the optimal solution.
4:     $\widehat{R} \leftarrow \{[(t_i, s_{ij}), (t_k, s_{kj})] \in R \mid \operatorname{conv}\{(t_i, s_{ij}), (t_i, s_{kj}), (t_k, s_{ij}), (t_k, s_{kj})\} \cap \mathcal{L}(C) \neq \emptyset\}$
5:     $(R, \Omega) \leftarrow$ refineGrid$(R, \hat{R}, \Omega)$
6: end for
Output: $C$
```

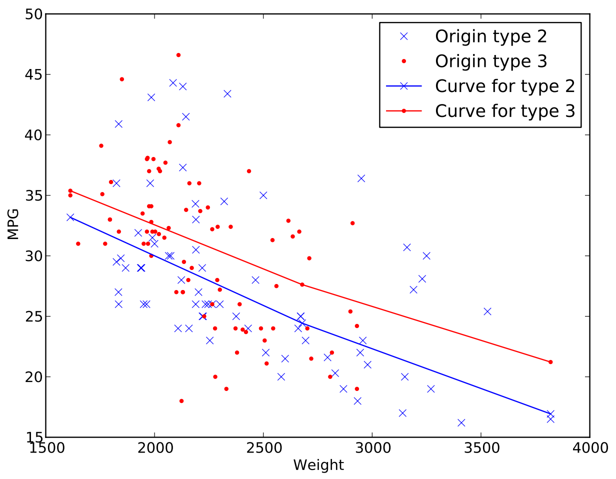


Figure 4: Least squares fit of two piecewise linear functions to a partition of the UCI MPG data [5] into domestic and foreign made cars. Note that each breakpoint of the two piecewise linear functions share the same $x$-coordinate value.

## 5 Extension of the Problem: Fitting Multiple Curves that Share a Breakpoint Coordinate

The data points may be associated with one or more categories. For example, as in Figure 4 one may consider domestic vs. foreign vehicles for estimating gas consumption. To each category we may choose to fit a distinct curve. However, the breakpoints' horizontal coordinates, also known as change-points or timestamps for time series data [7], may be required to be equal over all categories. For example, with financial data the change-point may correspond to an event such as a market crash, and each category may correspond to a different financial index.

In the following, we consider an extension of the dyanamic program and grid construction methods for multiple curves. To simplify the presentation, we limit the current discussion to two curves, although Algorithm 3 describes the general method for $v \geq 2$ curves. Let $n^w$ denote the number of data points in part $w$ of the data, for $w \in W \overset{\text{Def}}{=} \{1,\ldots,v\}$, and so that $\sum_{w\in W} n^w = n$. Denote the corresponding data $D^w = (x^w \;\; y^w) \in \mathbb{R}^{n^w \times 2}$ for $w \in W$. As for the single curve case, a grid of candidate breakpoints is constructed in order to apply the DP method. We generate distinct sets of candidate breakpoints, each of which is used to approximate one of the curves. We follow the same procedure to generate the candidate breakpoints for each curve as described in Sections 4.1 and 4.2 except that the candidate breakpoints of each curve must share the same $x$-coordinate values.

### 5.1 Dynamic Programming for Multiple Curves

Now we write the Bellman optimality equations for the multiple-curve problem. For a curve $w \in W$ and each pair of $x$-coordinate values $(t_k, t_i)$ with $t_k < t_i$, we have a $v$-tuples of line segments

$$\left\{ \times_{w\in W}(t_k, s^w_{k\ell}, t_i, s^w_{ij}) \;\middle|\; \ell \in S^w(t_k), j \in S^w(t_i) \right\};$$

for each index pair of $x$-coordinates $(k,i)$ there are $\prod_{w\in W} |S^w(t_k)| \times |S^w(t_i)|$ line segments. For example, with $v = 2$ there are $\left(|S^1(t_k)| \times |S^1(t_i)|\right)\left(|S^2(t_k)| \times |S^2(t_i)|\right)$ possible combinations of line segments. For convenience, for a pair of indices $(k,i) \in \{1,\ldots,p\} \times \{1,\ldots,p\}$ with $k \neq i$, $l_w \in S^w(t_k)$ and $j_w \in S^w(t_i)$ for each $w \in W$, we write a $v$-tuple of segments as $(t_k, \times_{w\in W} s^w_{kl_w}, t_i, \times_{w\in W} s^w_{ij_w})$. The segment $v$-tuple error is then defined as

$$E(t_k, \times_{w\in W} s^w_{kl_w} t_i, \times_{w\in W} s^w_{ij_w}) = \sum_{w\in W} E(t_k, s^w_{kl_w}, t_i, s^w_{ij_w}).$$

For $\widehat{m} = 2, \ldots, m$ the optimality equations of the generalized multiple curve problem are

$$F_{\widehat{m}}(i,j) = \min_{\substack{k=\widehat{m},\ldots,i-1,\\ l_w \in S^w(t_k):\\ w\in W}} \left\{ E(t_k, \times_{w\in W} s^w_{il_w}, t_i, \times_{w\in W} s^w_{ij_w}) + F_{\widehat{m}-1}(k, \times_{w\in W} l_w) \right\} \quad \text{for } i = \widehat{m}+1, \ldots, p-m+1, \quad j_w \in S^w(t_i) : w \in W. \tag{9a}$$

$$F_1(i,j) = \min_{l_w \in S^w(t_1): w\in W} E(t_1, \times_{w\in W} s^w_{1l_w}, t_i, \times_{w\in W} s^w_{ij_w}) \quad \text{for } i = 2, \ldots, p-m+1, \quad j_w \in S^w(t_i) : w \in W. \tag{9b}$$

If $S^w(t_i) \leq q$ for $w \in W$ and $i = 1, \ldots, p$, then the running time of evaluating (9) is $O(p^2 q^{2v} m)$. The running time analysis and proof are given in Appendix A.

## 5.2 Grid Construction for Multiple Curves

For DP-static we determine a single set of $t_i$'s by subdividing the entire range of data (including both sets) into $(q-1)$ subintervals of equal lengths. To generate a distinct set $S^w(t_i)$ for each $i$ and each curve $w$, we apply a procedure similar to the one described in Section 4.1 and using the same rectangle refinement subroutine given by Algorithm 1.

Algorithm 3 describes the extension of Algorithm 2 to the case of multiple curves. For the initial grid we typically use four

**Algorithm 3** Adaptive refinement for multiple curves

```
Input: Datasets $D_w \subset \mathbb{R}^2$, for $w \in W$, the number of line segments $m$, iteration limit $T$,
    initial grid $\Omega_w$, and rectangles $R_w$ for $w \in W$.
1: for $k = 1, \ldots, T$ do
2:     $m' \leftarrow \min\{m, k\}$
3:     Evaluate $\min_{j \in \times_{w\in W} S^w(t_{2^k-1})} F_{m'}(2^{k-1}, j)$ on $\Omega_1, \ldots, \Omega_v$ and let $C_1 \subseteq \Omega_1, \ldots, C_v \subseteq \Omega_v$ denote the
       optimal solutions.
4:     $\widehat{R}_w \leftarrow \{[(t_i, s_{ij}), (t_k, s_{kj})] \in R_w \mid \text{conv}\{(t_i, s_{ij}), (t_i, s_{kj}), (t_k, s_{ij}), (t_k, s_{kj})\} \cap \mathcal{L}(C_w) \neq \emptyset\}$ for $w \in W$
5:     refineGrid($R_w, \hat{R}_w, \Omega_w$) for $w = 1, \ldots, v$
6: end for
Output: $C_w \in \Omega_w$ for $w \in W$.
```

candidate breakpoints for each $\Omega_w$ for $w = 1, \ldots, v$. Accordingly, for $w = 1, \ldots, v$, we initialize

$$R_w \leftarrow \left\{ \left[ (\min_{w'=1,\ldots,v} \min_{i=1,\ldots,n^w} x^w_i, \min_{i=1,\ldots,n^w} y^w_i), (\max_{w'=1,\ldots,v} \max_{i=1,\ldots,n^w} x^w_i, \max_{i=1,\ldots,n^w} y^w_i) \right] \right\}.$$

After the initialization, the algorithm follows the same procedure described in Algorithm 2 for refinement while maintaining the distinct sets of candidate breakpoints for each curve. As the two curves are initialized with a single set of $t_i$'s and the refinement maintains a single set of $t_i$'s for both curves thereafter, it follows that the breakpoints of the two solution curves will share the same set of x-coordinate values.

Algorithm 3 implements the adaptive refinement procedure for multiple curves. In particular, with $v = 2$, we have an adaptive refinement algorithm for two curves. The computational experiments of the current paper, discussed in Section 6, focus on the cases of $v = 1, 2$.

## 6 Numerical Results

In this section we evaluate the performance of our proposed techniques. We experimented with six datasets including standard benchmark UCI data [5] and hydrological data of water reservoirs in Colorado. Table 2 lists the datasets which we used: The first three datasets are derived from the Auto MPG Dataset [5], with different columns as the independent variable. The fourth and fifth datasets include hydrological data for estimating a reservoir's volume based on water elevation measurements. The sixth dataset is a hydrological time series.

Table 2: Datasets used in the computational experiments.

| **Dataset** | $n$ | **Description** |
|---|---|---|
| (weight,MPG) | 398 | Vehicle weight as the independent variable used to explain MPG [5]. |
| (horsepower, MPG) | 392 | Vehicle horsepower as the independent variable used to explain MPG [5]. |
| (displacement, MPG) | 398 | Vehicle engine displacement as the independent variable used to explain MPG [5]. |
| (elevation, volume) | 50 | A subset of a dataset used to estimate the volume of the Morrow Point reservoir as a function of the water elevation. |
| (elevation, volume) | 834 | A dataset used to estimate the volume of the Morrow Point reservoir as a function of the water elevation. |
| (time, elevation) | 1216 | A dataset of water elevation measurements in the Blue Mesa reservoir over 1216 time periods during the years 2008-2011. |

The DP method and Algorithm 2 were implemented in C++. All experiments were run in serial using Intel Q8400 2.66 GHz CPUs with a 2 MB cache size.

## 6.1 Comparison of the DP Solution Quality vs. MIP

We now compare the solution quality of the proposed DP variants with the solution of the MIP. In this section we limit our consideration to a fixed setting of the parameters of DP-static and DP-adaptive and focus on comparing the solution quality with respect to the optimal solution. The MIP formulations were solved using CPLEX 12.04. A time limit of one hour was applied to all runs. Table 3 compares the running times and solution values of DP with error loss function $e_2(\cdot)$ with optimal solutions computed by (5). Here, the initial rectangle of DP-adaptive is set to be three times the height of the default setting of (8) in order to allow for a larger range of y-intercepts that seemed to be required with $m = 1$. The CPLEX relative optimality gap is set to 1%, and we set $M_1 = M_3 = 1e5$. Table 4 compares the running times and solution values of DP with error loss function $e_1(\cdot)$ using optimal solutions that are computed by (5) with (5a) replaced by $\sum_{i=1}^{n} \xi_i$. Both tables indicate that in most cases the solution quality of the DP methods closely approximates the optimal solution. An exception is the DP-static method with $m = 1$. However, overall DP-adaptive shows an advantage, albeit small, over DP-static in terms of the solution quality. Tables 3 and 4 also display the running times of the three methods. The MIQP can only be solved up to $m = 2$. As shown in Table 4, the MILP can be solved with more breakpoints, but the table also shows that the running times rapidly increase to exceed 50 minutes with 4 breakpoints.

Table 3: Computational results for (elevation,volume), $n = 50$, (#xGrid,#yGrid)=(100,5) for DP-static, and $T = 9$ for DP-adaptive.

| $m$ | MIQP | | DP-Static | | | DP-Adaptive | | |
|---|---|---|---|---|---|---|---|---|
| | Time (sec) | Objective | Time (sec) | Objective | % Error | Time (sec) | Objective | % Error |
| 1 | 0.01 | 4,677.89 | 0.020 | 18,720.035 | 300.18 | 0.744 | 4,696.153 | 0.39 |
| 2 | 1.31 | 289.878 | 0.022 | 371.700 | 28.22 | 2.053 | 295.099 | 1.80 |
| 3 | Time Limit | | 0.021 | 58.096 | | 3.276 | 60.571 | |

Table 4: Computational results for (elevation,volume), $n = 50$, (#xGrid,#yGrid)=(100,5) for DP-static, and $T = 9$ for DP-adaptive.

| $m$ | MILP | | DP-Static | | | DP-Adaptive | | |
|---|---|---|---|---|---|---|---|---|
| | Time (sec) | Objective | Time (sec) | Objective | % Error | Time (sec) | Objective | % Error |
| 1 | 0.46 | 405.000 | 0.020 | 855.400 | 111.21 | 0.712 | 405.527 | 0.13 |
| 2 | 2.41 | 100.083 | 0.021 | 120.972 | 20.87 | 2.125 | 100.569 | 0.49 |
| 3 | 193.31 | 43.179 | 0.024 | 44.439 | 2.92 | 3.211 | 45.035 | 4.30 |
| 4 | 2934.81 | 18.997 | 0.025 | 26.874 | 41.46 | 6.263 | 25.981 | 36.76 |

## 6.2 DP-static vs. DP-adaptive

We now discuss experiments with larger datasets and larger $m$ values to compare our two schemes for constructing tractable grids of candidate breakpoints. In our experiments we first ran DP-static and then DP-adaptive. DP-adaptive was run for sufficiently many iterations so that its objective value (or error) matches that of DP-static. In Tables 5-10, the last column shows the number of iterations of DP-adaptive that were needed to achieve the indicated results.

Tables 6-8 show that typically as $m$ increases, the running time of DP-adaptive may increases in order to match and exceed the quality of the DP-static solution. This is explained by the fact that DP-static allows the breakpoints to be located at or near the y-coordinate of the actual data points. When $m = n$, this heuristic in fact yields the optimal solution with an optimal objective value (i.e., error) of 0. When $m$ is small the quality of this heuristic is poor; but as $m$ is increased while $n$ is held constant, the quality of the heuristic typically improves. Table 6 shows a clear performance advantage of DP-adaptive over DP-static in terms of the running times that are needed to exceed the error performance of the DP-static alternative. Table 7 shows that with a denser grid the quality of DP-static's solutions slightly improves but DP-adaptive easily exceeds the quality of DP-static's solutions within a fraction of the running time.

In Tables 9–10 it becomes apparent that as $m \geq 6$ the running time required of DP-adaptive can significantly increase relative to DP-static; as mentioned this is due to the improved performance of the DP-static heuristic for larger values of $m$ relative to fixed values of $n$. However, for a wide range of values of $m$, DP-adaptive remains faster than DP-static while providing a superior solution quality. In Table 9 in cases in which the running time of DP-adaptive exceeds that of DP-static, the improvement in solution quality, on the other hand, also tends to be significant (ranging from 8%-40%).

Figure 5 graphically displays the solutions of the linear spline regression experiments. The solutions of DP-static and DP-adaptive seem to substantially differ for the noisier MPG datasets. For the less noisy water-elevation datasets it appears that the estimated curves are quite close even when the estimated breakpoints do not coincide. Finally, Figure 6 displays an example of the grid that is evaluated by DP-static compared with the grid that is evaluated by DP-adaptive. The figure shows that the DP-Adaptive algorithm evaluates points that seem to be sparsely dispersed around the final solution curve but a little more densely distributed within a close proximity to the curve. The DP-static grid on the other hand is very densely distributed around the datapoints (and final solution curve).

Table 5: (weight,MPG) dataset, $n = 398$, (#xGrid,#yGrid)=(100,5) for DP-static.

| $m$ | DP-Static | | | DP-Adaptive | | | |
|---|---|---|---|---|---|---|---|
| | Time (sec) | Objective | $\|\Omega\|$ | Time (sec) | Objective | $\|\Omega\|$ | $T$ |
| 2 | 0.055 | 7,031.140 | 500 | 0.010 | 7,019.904 | 138 | 5 |
| 3 | 0.055 | 6,980.827 | 500 | 0.012 | 6,970.296 | 150 | 5 |
| 4 | 0.056 | 6,915.180 | 500 | 0.014 | 6,783.982 | 165 | 5 |
| 5 | 0.057 | 6,830.611 | 500 | 0.014 | 6,783.670 | 165 | 5 |

Table 6: (horsepower,MPG) dataset, $n = 392$, (#xGrid,#yGrid)=(100,5) for DP-static.

| $m$ | DP-Static | | | DP-Adaptive | | | |
|---|---|---|---|---|---|---|---|
| | Time (sec) | Objective | $\|\Omega\|$ | Time (sec) | Objective | $\|\Omega\|$ | $T$ |
| 2 | 0.054 | 14,129.924 | 500 | 0.0002 | 12,210.006 | 9 | 2 |
| 3 | 0.054 | 8,192.229 | 500 | 0.002 | 7,694.997 | 60 | 4 |
| 4 | 0.055 | 7,467.350 | 500 | 0.012 | 7,176.181 | 147 | 5 |
| 5 | 0.055 | 7,378.477 | 500 | 0.012 | 7,162.479 | 147 | 5 |

Table 7: (horsepower,MPG) dataset, $n = 392$, (#xGrid,#yGrid)=(200,20) for DP-static.

| $m$ | DP-Static | | | DP-Adaptive | | | |
|---|---|---|---|---|---|---|---|
| | Time (sec) | Obj Value | $\|\Omega\|$ | Time (sec) | Obj Value | $\|\Omega\|$ | $T$ |
| 2 | 2.9391 | 12,250.488 | 4000 | 0.0002 | 12,210.006 | 9 | 2 |
| 3 | 2.9780 | 7,737.867 | 4000 | 0.002 | 7,694.997 | 60 | 4 |
| 4 | 3.0164 | 7,306.523 | 4000 | 0.012 | 7,176.181 | 147 | 5 |
| 5 | 3.0555 | 7,191.656 | 4000 | 0.012 | 7,162.479 | 147 | 5 |

Table 8: (displacement,MPG) dataset, $n = 398$, (#xGrid,#yGrid)=(100,5) for DP-static.

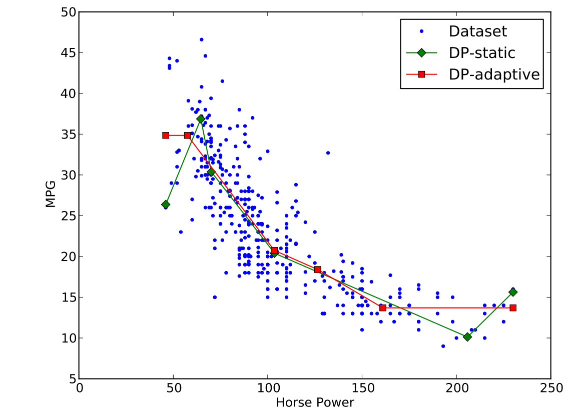
| $m$ | DP-Static | | | DP-Adaptive | | | |
|---|---|---|---|---|---|---|---|
| | Time (sec) | Objective | $\|\Omega\|$ | Time (sec) | Objective | $\|\Omega\|$ | $T$ |
| 2 | 0.050 | 8,811.040 | 500 | 0.002 | 7,808.003 | 54 | 4 |
| 3 | 0.051 | 7,328.727 | 500 | 0.050 | 7,092.556 | 328 | 6 |
| 4 | 0.055 | 6,817.053 | 500 | 0.073 | 6,811.856 | 406 | 6 |
| 5 | 0.058 | 6,741.545 | 500 | 0.346 | 6,593.420 | 855 | 7 |

Table 9: (elevation,storage) dataset, $n = 834$, (#xGrid,#yGrid)=(200,5) for DP-static.

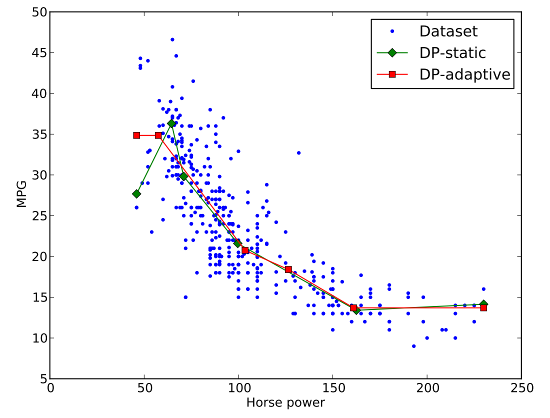
| $m$ | DP-Static | | | DP-Adaptive | | | |
|---|---|---|---|---|---|---|---|
| | Time (sec) | Objective | $\|\Omega\|$ | Time (sec) | Objective | $\|\Omega\|$ | $T$ |
| 2 | 0.352 | 71,814,118,000 | 1000 | 0.002 | 25,628,928,000 | 57 | 4 |
| 3 | 0.356 | 13,302,963,000 | 1000 | 0.016 | 5,620,973,400 | 153 | 5 |
| 4 | 0.358 | 3,630,796,100 | 1000 | 0.016 | 2,054,457,400 | 152 | 5 |
| 5 | 0.362 | 1,230,120,200 | 1000 | 0.082 | 664,427,240 | 393 | 6 |
| 6 | 0.363 | 546,859,270 | 1000 | 0.086 | 406,544,860 | 397 | 6 |
| 7 | 0.365 | 221,637,540 | 1000 | 0.101 | 221,608,790 | 431 | 6 |
| 8 | 0.369 | 100,619,500 | 1000 | 5.259 | 60,009,459 | 2960 | 8 |
| 9 | 0.373 | 47,851,391 | 1000 | 5.569 | 44,004,352 | 3046 | 8 |
| 10 | 0.376 | 24,115,875 | 1000 | 35.878 | 21,359,717 | 7370 | 9 |

Table 10: (time,elevation) dataset, $n = 1216$, (#xGrid,#yGrid)=(200,5) for DP-static.

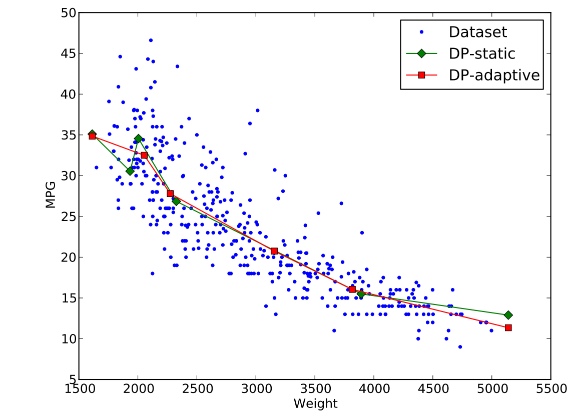
| $m$ | DP-Static | | | DP-Adaptive | | | |
|---|---|---|---|---|---|---|---|
| | Time (sec) | Objective | $\|\Omega\|$ | Time (sec) | Objective | $\|\Omega\|$ | $T$ |
| 2 | 0.473 | 541.8554 | 1000 | 0.003 | 285.8593 | 54 | 4 |
| 3 | 0.477 | 127.3319 | 1000 | 0.015 | 66.5020 | 137 | 5 |
| 4 | 0.485 | 28.4243 | 1000 | 0.073 | 17.4254 | 333 | 6 |
| 5 | 0.483 | 14.6801 | 1000 | 0.083 | 11.6842 | 352 | 6 |
| 6 | 0.489 | 9.7675 | 1000 | 0.083 | 8.368 | 352 | 6 |
| 7 | 0.491 | 5.0368 | 1000 | 3.387 | 4.587 | 2276 | 8 |
| 8 | 0.494 | 4.2606 | 1000 | 3.569 | 4.034 | 2312 | 8 |
| 9 | 0.497 | 3.7497 | 1000 | 4.250 | 3.551 | 2504 | 8 |
| 10 | 0.500 | 3.0380 | 1000 | 4.323 | 2.998 | 2515 | 8 |



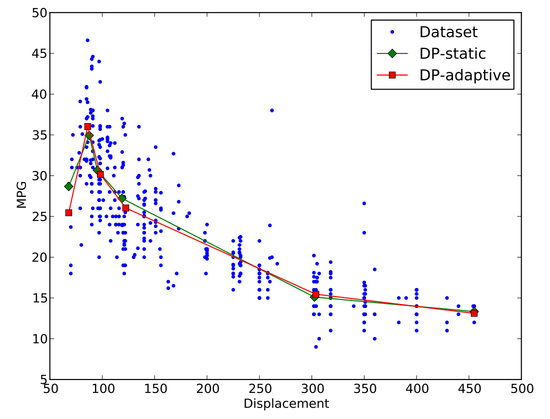
(a) (horsepower,MPG), $m = 5$, (#xGrid,#yGrid)=(200,5)



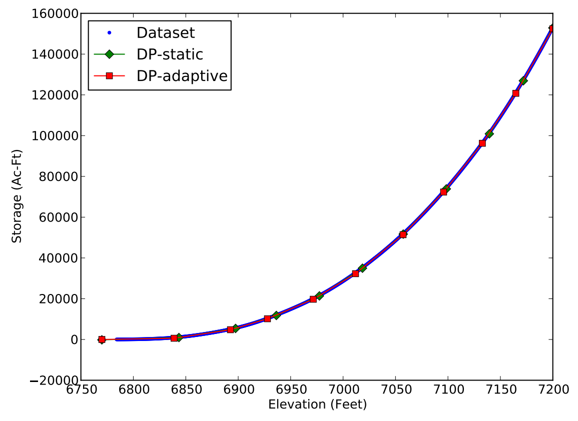
(b) (horsepower,MPG), $m = 5$, (#xGrid,#yGrid)=(200,20)



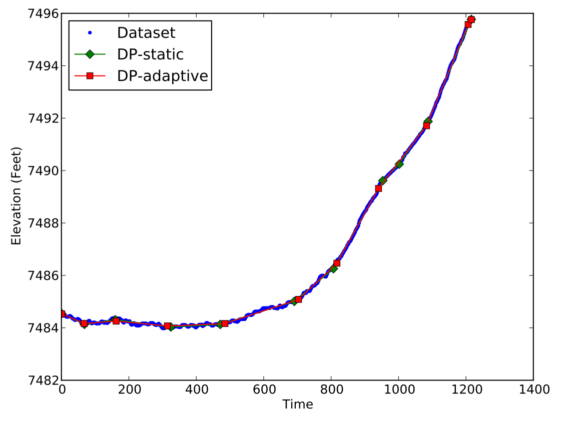
(c) (weight,MPG), $m = 5$



(d) (displacement,MPG), $m = 5$



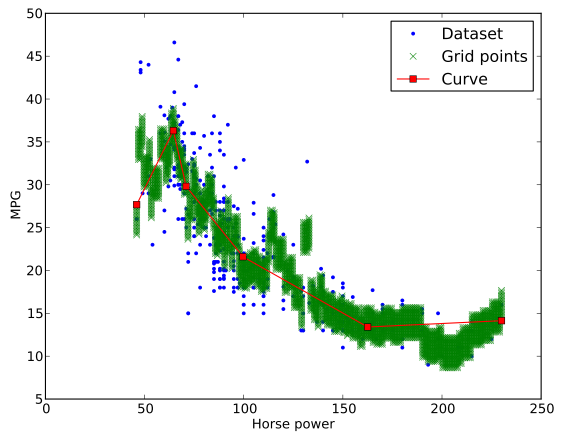
(e) (elevation,storage), $m = 10$



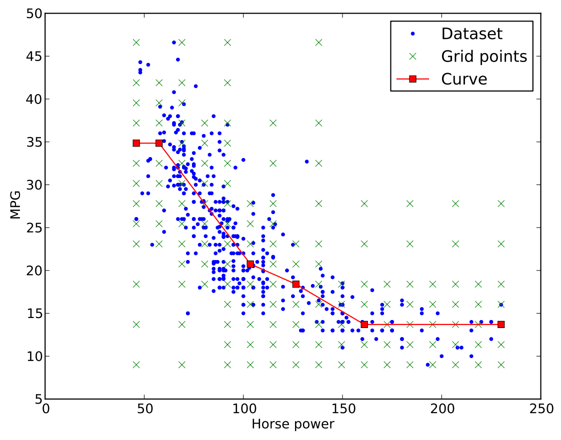
(f) (time,elevation), $m = 10$

Figure 5: Graphical representation of numerical results of DP-static and DP-adaptive on several datasets. Figure 5(b) show the results for the same dataset and setting as Figure 5(a) but with DP-static using a denser static. Also, note that both methods result in similar solutions in Figure 5(e) where the approximating curve is convex.

Finally, Figure 6 shows the static grid next to the grid resulting from adaptive refinement.



(a) DP-Static



(b) DP-Adaptive

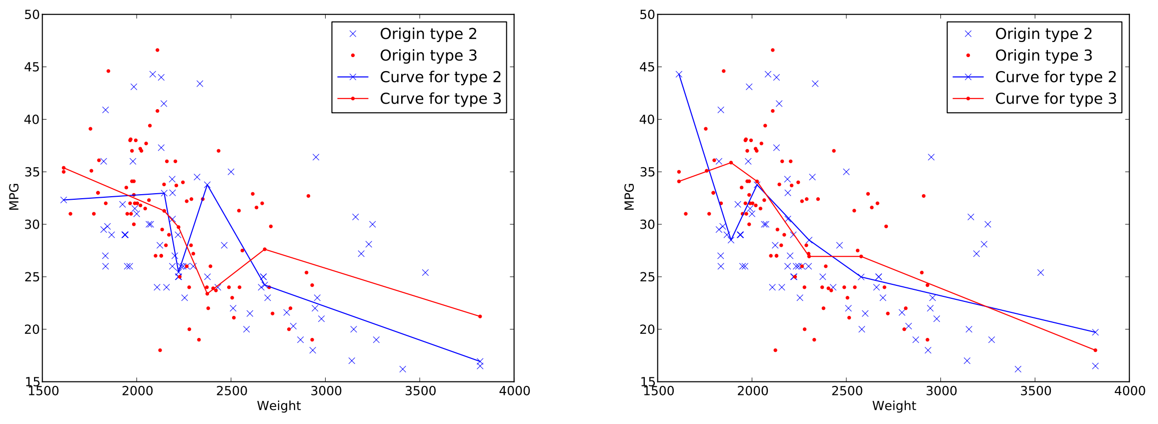
Figure 6: Comparison of the dense static grid and the adaptively refined grid for the (horsepower,MPG) dataset.

## 6.3 DP-Static vs. DP-Adaptive with Two Curves

We now compare the performance of the static grid and adaptive refinement for the extended problem of determining two curves that share breakpoints with identical horizontal coordinates. In Table 11, the column $(|\Omega_1|, |\Omega_2|)$ indicates the total number of candidate breakpoints generated for the first and second curves, respectively. The results of the table indicate that DP-adaptive is competitive in terms of the running time required to obtain a given solution quality. The running time of DP-adaptive significantly increases, however, for $m = 5$ with $T = 6$. The improved solution quality however is also significant in this case and Figure 7 displays the different estimated curves associated with the DP-static and DP-adaptive solutions.

Table 11: Comparison of DP-static and the DP-adaptive Algorithm 3 with $\nu = 2$ (two curves) using the (weight,MPG) dataset. Here $D_1$ consists of vehicles of origin type 2 ($|D_1| = 70$), and $D_2$ consists of vehicles of origin type 3 ($|D_2| = 79$). Also, (#xGrid,#yGrid)=(30,5) for DP-static.

| $m$ | DP-Static | | | DP-Adaptive | | | |
|---|---|---|---|---|---|---|---|
| | Time (sec) | Objective | $\|\Omega_1\|, \|\Omega_2\|$ | Time (sec) | Objective | $\|\Omega_1\|, \|\Omega_2\|$ | $T$ |
| 2 | 0.039 | 4431.550 | (150,150) | 0.003 | 4363.912 | (22,22) | 3 |
| 3 | 0.040 | 4230.532 | (150,150) | 0.064 | 4071.355 | (66,57) | 4 |
| 4 | 0.041 | 4108.698 | (150,150) | 0.064 | 4043.569 | (66,57) | 4 |
| 5 | 0.042 | 4033.527 | (150,150) | 1.168 | 3864.417 | (175,158) | 5 |



(a) DP-static

(b) DP-adaptive

Figure 7: Graphical representation of numerical results of DP-static and DP-adaptive for the UCI MPG dataset [5]. Here $D_1$ contains the subset of vehicles with origin type 2, $D_2$ contains the subset of vehicles with origin type 3, and $m = 5$.

## 7 Conclusion

We have shown the efficacy of using nonuniform grids for dynamic programming in solving linear spline regression. In particular, we devise an adaptive refinement scheme for solving linear spline regression with an arbitrary location of breakpoints. We compared our methods with the optimal solutions computed using a novel MIP formulation of the problem. The advantage of the adaptive refinement scheme is particularly apparent for (difficult) instances whose points are well approximated using $m \geq 2$ line segments where $m$ remains small relative to $n$. When $m$ is large relative to $n$, an alternative heuristic that determines a small set of candidate breakpoints near the data points is competitive in terms of solution quality and speed.

Finally, we have extended the DP algorithm and our grid construction approach to fit multiple piecewise linear curves that share the breakpoint horizontal coordinates (or change points). We also find the proposed adaptive refinement scheme useful in this setting.

## Acknowledgements

We thank Jed Brown for comments and Ashutosh Mahajan for pointing us to [2].

## A Analysis of the Running Time of the Dynamic Program

**Proposition 1.** *Suppose that for $w \in W = \{1, \ldots, v\}$, and $i = 1, \ldots, p$ we have $S^w(t_i) \leq q$. Then, if we assume $E(\cdot)$ is given, the running time of the DP* (9) *is $O(p^2 q^{2v} m)$.*

*Proof.* The DP requires the evaluation of $F_{\widehat{m}}(i, j)$ for $\widehat{m} = 1, \ldots, m$, $i = 1, \ldots, p$, and $j \in S(t_i)$.

First, $F_1(i, j)$ is evaluated for $i = 2, \ldots, p - m + 1$ and $j \in \times_{w \in W} S^w(t_i)$ and each such evaluation is $O(1)$ assuming that $E$ is given. For each $i \in \{1, \ldots, p - (m - 1)\}$, $|S^w(t_i)| \leq q$ for each $w \in W$, so that $q^{2v}$ operations are used to enumerate elements of $S^w(t_1) \times S^w(t_i)$ in order to determine $F_1(i, j) = \min_{l_w \in S^w(t_1): w \in W} E(t_1, \times_{w \in W} s^w_{1 l_w}, t_i, \times_{w \in W} s^w_{i j_w})$, for $j \in \times_{w \in W} S^w(t_i)$ and $w \in W$. Hence, in total at most $(p-m)q^{2v}$ evaluations are required to determine $F_1(i, j)$ for $i = 1, \ldots, p-m+1$ and $j \in \times_{w \in W} S^w(i)$.

Next, for $\widehat{m} \in \{2, \ldots, m\}$, $F_{\widehat{m}}(i, j)$ is evaluated for $i \in I \overset{\text{Def}}{=} \{\widehat{m}+1, \ldots, p-m+\widehat{m}\}$ and $j \in \times_{w \in W} S^w(i)$. Note that $|I| = p - m$. For a given $i \in I$ and $j \in S(t_i)$, $F_{\widehat{m}}(i, j)$ evaluates $F_{\widehat{m}-1}(k, \cdot)$ for $k \in \{\widehat{m} + 1, \ldots, i - 1\}$. For each pair $(k, i) \in I \times I$ with $k \neq i$, and $w \in W$, at most $q^2$ evaluations are required. Then, the number of segment combinations over $w \in W$ is $q^{2|W|} = q^{2v}$. There are at most $\binom{p-m}{2}$ such pairs (where $k \neq i$), so that at most $\frac{(p-m)(p-m-1)}{2} q^{2v}$ evaluations are required to compute $F_{\widehat{m}}(i, j)$ for $i = \widehat{m}, \ldots, p - m + \widehat{m}$ and $j \in \times_{w \in W} S^w(t_i)$.

Hence, the number of operations is bounded by

$$(p-m)q^{2v} + (m-1)\frac{(p-m)(p-m-1)}{2}q^{2v} = (p-m)q^2\left(\frac{(m+1)+(m-1)(p-m)}{2}\right)$$
$$\in O((p-m)^2 q^{2v} m) \subseteq O(p^2 q^{2v} m). \qquad \square$$

If $v = 1$ then the total running time may be dominated by the computation of the segment errors $E(\cdot)$. If $N$ is the total number of breakpoints then note that the running time of computing the segment errors is bounded by $O(N^2 n)$. As (9) with $v = 1$ reduces to (6), the following corollary establishes the running time of evaluating (6) following the precomputation of $E(\cdot)$.

**Corollary 1.** *Suppose that for $i = 1, \ldots, p$ we have $S^w(t_i) \leq q$ and let $N = pq$. Then, if we assume $E(\cdot)$ is given, the running time of the DP* (6) *is $O(N^2 m)$.*

## References

[1] B. Aronov, T. Asanov, N. Katoh, and K. Mehlhorn. Polyline fitting of planar points under min-sum criteria. *International Journal of Computational Geometry and Applications*, 16(2&3):97–116, 2006.

[2] J. Bai and P. Perron. Computation and analysis of multiple structural change models. *Journal of Applied Econometrics*, 18:1–22, 2003.

[3] R. Bellman and R. Roth. Curve fitting by segmented straight lines. *Journal of the American Statistical Association*, 64:1079–1084, 1969.

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

[5] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[6] S.B. Guthery. Partition regression. *Journal of the American Statistical Association*, 348:945–947, 1974.

[7] A. Kehagias, E. Nidelkou, and V. Petridis. A dynamic programming segmentation procedure for hydrological and environmental time series. *Stochastic Environmental Research and Risk Assessment*, 20(1):77–94, 2005.

[8] A. Toriello and J.P. Vielma. Fitting piecewise linear continuous functions. *European Journal of Operational Research*, 219(1):89–95, 2012.