Abstract. In this paper, the monotonic regression problem (MR) is considered. We have recently generalized for MR the well-known Pool-Adjacent-Violators algorithm (PAV) from the case of completely to partially ordered data sets. The new algorithm, called GPAV, combines both high accuracy and low computational complexity which grows quadratically with the problem size. The actual growth observed in practice is typically far lower than quadratic. The fitted values of the exact MR solution compose blocks of equal values. The GPAV approximation to this solution has also a block structure. We present here a technique for refining blocks produced by the GPAV algorithm to make the new blocks much closer to those in the exact solution. This substantially improves the accuracy of the GPAV solution and does not deteriorate its computational complexity. The computational time for the new technique is approximately triple the time of running the GPAV algorithm. Its efficiency is demonstrated by results of our numerical experiments.

Key words: Monotonic regression, Partially ordered data set, Pool-adjacent-violators algorithm, Quadratic programming, Large scale optimization, Least distance problem.

1 Introduction

The monotonic regression problem (MR), which is also known as the isotonic regression problem, deals with an ordered data set of observations. We focus on partially ordered data sets, because in this case, in contrast to completely ordered sets, there are no efficient algorithms for solving large scale MR problems.

The MR problem has important statistical applications in physics, chemistry, medicine, biology, environmental science etc. (see [2, 23]). It is present also in operations research (production planning, inventory control, multi-center location etc.) [13, 15, 24] and signal processing [22, 25]. All these problems are often a kind of a monotonic data fitting problem, which is addressed in Section 4 where

* This work was supported by the Swedish Research Council.
we use it for generating test problems. The most challenging of the applied MR problems are characterized by a very large value of the number of observations denoted here by $n$. For such large-scale problems, it is of great practical value to develop algorithms whose complexity does not rise with $n$ too rapidly.

To formulate the MR problem, we introduce the following notations. The vector of observed values is denoted by $Y \in \mathbb{R}^n$. The partial order is expressed here with the use of a directed acyclic graph $G(N, E)$, where $N = \{1, 2, \ldots, n\}$ is a set of nodes and $E$ is a set of edges. Each node is associated with one observation, and each edge is associated with one monotonicity relation as described below. In the MR problem, we must find among those vectors $u \in \mathbb{R}^n$ preserving the monotonicity of the partially ordered data set, the one, which is the closest to $Y$ in the least-squares sense. It can be formulated as follows. Given $Y$, $G(N, E)$ and a strictly positive vector of weights $w \in \mathbb{R}^n$, find the vector of fitted values $u^* \in \mathbb{R}^n$ that solves the problem:

$$\min \sum_{i=1}^{n} w_i(u_i - Y_i)^2$$

s.t. $u_i \leq u_j$ $\forall (i, j) \in E$

It can be viewed as a problem of minimizing the weighted distance from the vector $Y$ to the set of feasible points which is a convex cone. This strictly convex quadratic programming problem has a unique optimal solution.

The conventional quadratic programming algorithms (see [19]) can be used for solving the general MR problem only in the case of moderate values of $n$, up to few hundred.

There are some algorithms especially developed for solving this problem. They can be divided into the two separate groups, namely, exact and approximate MR algorithms.

The most efficient and the most widely used of the exact algorithms is the Pool-Adjacent-Violators (PAV) algorithm [1, 14, 16]. Although it has a very low computational complexity, namely $O(n)$ [11], the area of its application is severely restricted by the complete order. In this case, the graph is just a path and the monotonicity constraints in (1) takes the simple form:

$$u_1 \leq u_2 \leq \ldots \leq u_n.$$

In [20], the PAV algorithm was extended to a more general, but still restricted, case of the rooted tree type of the graph defining the monotonicity. The computational complexity of this exact algorithm is $O(n \log n)$.

The minimum lower set algorithm [5, 6] is known to be the first exact algorithm designed for solving partially ordered MR problems. If the order is complete, its complexity is $O(n^2)$. In the partial order case, its complexity is unknown, but it is expected to grow with $n$ much more rapidly than quadratically.

The best known computational complexity of the exact algorithms, which are able to solve partially ordered MR problems, is $O(n^4)$. It refers to an algorithm introduced in [15, 24]. This algorithm is based on solving the dual problem to (1) by solving $O(n)$ minimal flow problems. The resulting growth of computational requirements in proportion to $n^4$ becomes excessive for large $n$. 


The isotonic block class with recursion (IBCR) algorithm was developed in [4] for solving partially ordered MR problems. It is an exact algorithm. Its computational complexity is unknown, but according to [21], it is bounded below by $O(n^3)$. In practice, despite of this estimate, it is the fastest among the exact algorithms. This is the reason why we use it in our numerical experiments to compare the performance of algorithms.

Perhaps, the most widely used inexact algorithms for solving large-scale partially ordered MR problems are based on simple averaging techniques [17, 18, 26]. They can be easily implemented and have a relatively low computational burden, but the quality of their approximation to $u^*$ is very case-dependent and furthermore, the approximation error can be too large (see [7]).

In [7, 8], we generalized the well-known Pool-Adjacent-Violators algorithm (PAV) from the case of completely to partially ordered variables. The new algorithm, called GPAT, combines both low computational complexity $O(n^2)$ and high accuracy. The computational time grows in practice less rapidly with $n$ than in this worst-case estimate. The GPAT solution is feasible, and it is optimal if to regard its active constraints as equalities. The corresponding active set induces a partitioning of $N$ into connected subsets of nodes, called blocks, which are obtained after excluding from $E$ the edges representing the non-active constraints. We present here a block refinement technique which substantially improves the accuracy of the GPAT solution, while the overall computational complexity remains $O(n^2)$. Its run time is between twice and triple the time of running GPAT.

Since the GPAT and IBCR algorithms coincide with the PAV algorithm when the order is complete, they both can be viewed as its generalizations. Although they have much in common, the main difference is that the first of them is an approximate algorithm, while the second one is exact. Moreover, according to the reported here result of numerical experiments, the GPAT is much faster than the IBCR, and the difference in the computational time grows rapidly with $n$.

The paper is organized as follows. The block refinement technique is introduced in Section 2. This technique is illustrated with a simple example in Section 3. The results of our numerical experiments are presented and discussed in Section 4. In Section 5, we draw conclusions about the performance of the block refinement technique and discuss future work.

2 Block Refinement Technique

Our block refinement technique is based on the GPAT algorithm. To present this algorithm, we will use the definitions and notations from [8]. Let

$$i^- = \{j \in N : (j, i) \in E\}$$

denote the set of all immediate predecessors for node $i \in N$. The connected subset of nodes $B \subset N$ is called a block if, for any $i, j \in B$, all the nodes in all the undirected paths between $i$ to $j$ belong to $B$. The block $B_i$ is said to be an immediate predecessor for $B_j$, or adjacent to $B_j$, if there exist $k \in B_i$ and $l \in B_j$.
such that \( k \in I^- \). Let \( B_i^- \) denote the set of all blocks adjacent to block \( B_i \). We associate each block with one of its nodes, which is called the head node. If \( i \) is the head node for some block, we denote this block by \( B_i \). The set of all head nodes is denoted by \( H \). The set of blocks \( \{ B_i \}_{i \in H} \), where \( H \subseteq N \), is called a block partitioning of \( N \) if
\[
\bigcup_{i \in H} B_i = N
\]
and
\[
B_i \cap B_j = \emptyset, \quad \forall i \neq j, \quad i, j \in H.
\]
Let \( W_k \) denote the weight of the block \( B_k \). It is computed by the formula
\[
W_k = \sum_{i \in B_k} w_i.
\]

The GPAV algorithm produces a block partitioning of \( N \). It returns also a solution \( u \in \mathbb{R}^n \) which is uniquely defined by the block partitioning as follows. If node \( i \) belongs to a block \( B_k \), the corresponding component \( u_i \) of the solution equals the block common value:
\[
U_k = \frac{\sum_{i \in B_k} w_i Y_i}{W_k}.
\]

This algorithm treats the nodes \( N \) or, equivalently, the observations in a consecutive order. Any topological order of \( N \) is acceptable, but the accuracy of the resulted solution depends on the choice (see [8]). We assume that the nodes \( N \) have been sorted topologically. The GPAV algorithm creates initially the singletone blocks \( B_i = \{i\} \) and sets \( B_i^- = i^- \) for all the nodes \( i \in N \). Subsequently it operates with the blocks only. It treats them in the order consistent with the topological sort, namely, \( B_1, B_2, \ldots, B_n \). When at iteration \( k \) the block \( B_k \) is treated, its common value (2) is compared with those of its adjacent blocks. While there exists an adjacent violator of the monotonicity, the block \( B_k \) absorbs the one responsible for the most severe violation. The common value \( U_k \) and the lists of adjacent blocks are updated accordingly.

The outlined GPAV algorithm can be formally presented as follows.

**Algorithm 1 (GPAV)**

*Given: vectors \( w, Y \in \mathbb{R}^n \) and a directed acyclic graph \( G(N, E) \) with topologically sorted nodes.*

Set \( H = N \).

For all \( i \in N \), set \( B_i = \{i\} \), \( B_i^- = i^- \), \( U_i = Y_i \) and \( W_i = w_i \).

For \( k = 1, 2, \ldots, n \), do:

While there exists \( i \in B_k^- \) such that \( U_i \geq U_k \), do:

Find \( j \in B_k^- \) such that \( U_j = \max\{U_i : \ i \in B_k^- \} \).

Set \( H = H \setminus \{j\} \).

Set \( B_k^- = B_j^- \cup B_k^- \setminus \{j\} \).
For all \( k \) output as \( u \) the MR problem:

\[ \min \sum_{i=1}^{n} w_i (u_i^B - Y_i^B)^2 \]
\[ \text{s.t. } u_i^B \leq u_j^B \quad \forall (i,j) \in E^B \]

where \( Y_i^B = -Y_i \) and \( E^B = \{(i,j) : (j,i) \in E\} \). Note that the optimal solution to this problem equals \(-u^*\). The backward mode of GPAV returns \( u^B \) and \( \{B_k^B\}_{k \in H^B} \). In this mode, one can use the inverse of the topological order used by the forward mode.

In our block refinement technique, it is assumed that two or more approximate solutions to problem (1) are available. They could result from applying any approximate algorithms, for instance, the forward and backward modes of Algorithm GPAV in combination with one or more topological orders. We will refer to them as old solutions.

We denote the vector of the component-wise average of the old solutions by \( v \), i.e. if there are available two old solutions, \( u' \) and \( u'' \), then \( v = (u' + u'')/2 \). The vector \( v \) is feasible in problem (1), and it can be viewed as a new approximation to its solution. It induces a block partitioning of the nodes \( N \). The new blocks are denoted by \( B_k^{new} \) where \( k \) belongs to the new set of head nodes \( H^{new} \). Let \( h(j) \) denote the head element of the new block which contains node \( j \), i.e. if \( j \in B_k^{new} \) then \( h(j) = k \).

It can be seen that the new blocks are, roughly speaking, nonempty intersections of the old blocks. Thus, the new ones results from a certain splitting of the old ones. The use of the vector \( v \) will allow us to simplify the construction of \( \{B_k^{new}\}_{k \in H^{new}} \). This idea is presented by the following algorithm.

**Algorithm 2 (SPLIT)**

Given: vector \( v \in \mathbb{R}^n \) and a directed acyclic graph \( G(N,E) \) with topologically sorted nodes.

Set \( H^{new} = \emptyset \) and \( E^{new} = \emptyset \).

For \( i = 1, 2, \ldots, n \), do:

If there exists \( j \in i^- \) such that \( v_j = v_i \),

then for \( k = h(j) \), set \( B_k^{new} = B_k^{new} \cup \{i\} \),

else set \( H^{new} = H^{new} \cup \{i\} \) and \( B_k^{new} = \{i\} \).

For all \( j \in i^- \) such that \( h(j) \neq h(i) \) & \( (h(j),h(i)) \notin E^{new} \), do:

set \( E^{new} = E^{new} \cup (h(j), h(i)) \).

This algorithm returns not only the new blocks \( \{B_k^{new}\}_{k \in H^{new}} \), but also the set of directed edges \( E^{new} \subset H^{new} \times H^{new} \). The new blocks are represented in the new directed acyclic graph \( G(H^{new}, E^{new}) \) by their head elements.
It can be easily seen that a topological order in the new graph can be obtained by sorting the nodes \( H_{\text{new}} \) in increasing order of the values \( v_k \).

Using the notations

\[
    w_{k}^{\text{new}} = \sum_{i \in B_{k}^{\text{new}}} w_i, \quad Y_{k}^{\text{new}} = \sum_{i \in B_{k}^{\text{new}}} y_i w_i / w_{k}^{\text{new}},
\]

where \( k \in H_{\text{new}} \), we can formulate for the new graph the new MR problem:

\[
    \min \sum_{k \in H_{\text{new}}} w_{k}^{\text{new}} (u_{k}^{\text{new}} - Y_{k}^{\text{new}})^2 \quad \text{s.t.} \quad u_{i}^{\text{new}} \leq u_{j}^{\text{new}} \quad \forall (i,j) \in E_{\text{new}}
\]

Since the number of unknowns \( u_{k}^{\text{new}} \) is, typically, less than \( n \) and \( |E_{\text{new}}| < |E| \), this problem is smaller, in its size, than the original problem (1). One can apply here, for instance, Algorithm GPAV. The resulting refined blocks yields an approximate solution to (1) which provides in practice a very high accuracy (see Section 4). Furthermore, if the blocks \( \{B_{k}^{\text{new}}\}_{k \in H_{\text{new}}} \) are the same as those induced by \( u^{*} \), then the optimal solution to problem (5) produces \( u^{*} \).

In our numerical experiments, we use the following implementation of the block refinement technique.

**Algorithm 3 (GPAVR)**

Given: vectors \( w, Y \in \mathbb{R}^n \) and a directed acyclic graph \( G(N, E) \) with topologically sorted nodes.

1. Use the forward and backward modes of Algorithm GPAV to produce two approximations, \( u^F_1 \) and \( -u^B_1 \), to \( u^* \).
2. Run Algorithm SPLIT with \( v = (u^F_1 - u^B_1)/2 \).
3. For all \( k \in H_{\text{new}} \), compute \( w_{k}^{\text{new}} \) and \( Y_{k}^{\text{new}} \) by formula (4).
4. Use \( v \) to sort topologically the nodes \( H_{\text{new}} \).
5. Use Algorithm GPAV to find \( u^{\text{new}} \) which solves approximately the new MR problem (5).
6. For all \( k \in H_{\text{new}} \) and for all \( i \in B_{k}^{\text{new}} \), set \( u_i = u_{k}^{\text{new}} \).

The block refinement technique is illustrated in the next section with a simple example.

It is not difficult to show that the computational complexity of Algorithm GPAVR is estimated, like for Algorithm GPAV, as \( O(n^2) \). Indeed, Step 1 involves two runs of Algorithm GPAV, which does not break the estimate. The computational burden of Step 2 is proportional to the number of edges in \( E \) which does not exceed \( n^2 \). The number of arithmetic operations on Steps 3 and 6 grows in proportion to \( n \). The computational complexity of the sorting on Step 4 is estimated as \( O(n \log(n)) \). On Step 5, Algorithm GPAV is applied to the graph \( G(H_{\text{new}}, E_{\text{new}}) \) with the number of nodes which does not exceed \( n \). This finally proves the desired estimate.
3 Illustrative Example

Consider the MR problem defined in Fig. 1. The weights $w_i = 1$, $i = 1, 2, 3, 4$. The vector $u^* = (0, -6, 6, 0)$ is the optimal solution to this problem. It induces the optimal block partitioning: $\{1, 4\}, \{2\}, \{3\}$ (indicated by the dashed lines).

One can see that the nodes are already topologically sorted. The forward mode of Algorithm GPAV produces the block partitioning: $B_F^2 = \{2\}$, $B_F^4 = \{1, 3, 4\}$ and the corresponding approximate solution $u_F = (2, -6, 2, 2)$.

Fig. 2 defines the MR problem (3) for the backward mode.

---

Fig. 1. The graph $G(N, E)$ and observed responses $Y$.

Fig. 2. The graph $G(N, E^B)$ and observed responses $Y^B$. 
In this case, the topological order is 4, 3, 2, 1. The backward mode of Algorithm GP AV produces the block partitioning: \( B_1^B = \{1, 2, 4\} \), \( B_2^B = \{3\} \) and the corresponding approximate solution \( u^B = (2, 2, -6, 2) \).

The nonempty intersections (dashed lines) of the forward mode blocks (dotted lines) and the backward mode blocks (solid lines) are shown in Fig. 3.

![Fig. 3. The old blocks (solid and dotted lines) and their splitting which yields the new blocks (dashed lines).](image)

The same splitting of the old blocks is provided by the input vector \( v = (u_i^F - u_i^B)/2 = (0, -2, 2, 0) \) of Algorithm SPLIT. This algorithm produces \( H_{new} = \{1, 2, 3\} \), \( B_1^{new} = \{1, 4\} \), \( B_2^{new} = \{2\} \), \( B_3^{new} = \{3\} \). The new MR problem (5) is defined by Fig. 4.

![Fig. 4. The new graph \( G(H^{new}, E^{new}) \) and observed responses \( Y^{new} \).](image)

The topological sort for the new graph is, obviously, 2, 1, 3. After applying Algorithm GP AV to solving the new MR problem, we obtain \( u_2^{new} = -6 \), \( u_1^{new} = 0 \), \( u_3^{new} = 6 \). Step 6 of Algorithm GP AVR yields the vector \( u = (0, -6, 6, 0) \) which is optimal in the original MR problem. In general, it is not guaranteed that the GP AVR solution is optimal.

4 Numerical Results

We use here test problems of the same type as in our earlier paper [8]. They originate from the monotonic data fitting problem which is one of the most
common types of applied MR problems. In the monotonic data fitting, it is assumed that there exists an unknown response function \( y(x) \) of \( p \) explanatory variables \( x \in \mathbb{R}^p \). It is supposed to be monotonic in the sense that
\[
y(x') \leq y(x''), \quad \forall x' \preceq x'', \quad x', x'' \in \mathbb{R}^p,
\]
where \( \preceq \) is a component-wise \( \leq \)-type relation. Instead of the function \( y(x) \), we have available a data set of \( n \) observed explanatory variables
\[
X_i \in \mathbb{R}^p, \quad i = 1, \ldots, n,
\]
and the corresponding observed responses
\[
Y_i \in \mathbb{R}, \quad i = 1, \ldots, n.
\]
The function and the data set are related as follows
\[
Y_i = y(X_i) + \varepsilon_i, \quad i = 1, \ldots, n, \quad (6)
\]
where \( \varepsilon_i \) is an observation error. In general, if the relation \( X_i \preceq X_j \) holds, this does not imply that \( Y_i \leq Y_j \), because of this error.

The relation \( \preceq \) induces a partial order on the set \( \{X_i\}_{i=1}^n \). The order can be presented by a directed acyclic graph \( G(N, E) \) in which node \( i \in N \) corresponds to the \( i \)-th observation, and the presence if edge \( (i, j) \) in \( E \) means that \( X_i \preceq X_j \). This graph is unique if all redundant relations are eliminated. We call edge \( (i, j) \), and also the corresponding monotonicity relations \( X_i \preceq X_j \) and \( u_i \leq u_j \), redundant if there is a directed path from \( i \) to \( j \). Redundant edges, if removed, leave the feasible set in (1) unchanged.

In monotonic data fitting, one must construct a monotonic response surface model \( u(x) \) whose values \( u(X_i) \) are as close as possible to the observed responses \( Y_i \) for all \( i = 1, \ldots, n \). Denoting
\[
u_i = u(X_i), \quad i = 1, \ldots, n \quad (7)
\]
and using the sum of squares as the distance function, one can reformulate this problem as the MR problem (1). In the numerical experiments, we set the weights \( w_i = 1 \) for all \( i = 1, \ldots, n \).

In the experiments, we restrict our attention to the case of \( p = 2 \) for the following reasons. Suppose that, for the two vectors \( X_i \) and \( X_j \) in \( \mathbb{R}^p \), neither \( X_i \preceq X_j \) nor \( X_j \preceq X_i \) holds, i.e. they are incomparable. Then if to delete (or disregard) in these vectors one and the same component, the reduced vectors may become comparable in \( \mathbb{R}^{p-1} \). On the other hand, if two vectors in \( \mathbb{R}^p \) are comparable, no deletion of their component is able to break this relation. This means that, for any fixed number of multivariate observations \( n \), the number of monotonic relations \( X_i \preceq X_j \) attains its maximum value when \( p = 2 \).

For our test problems, we use two types of functions \( y(x) \) of two explanatory variables, namely, linear and nonlinear.
Our nonlinear function is given by the formula
\[ y_{\text{nonlin}}(x) = f(x_1) + f(x_2), \quad (8) \]
where
\[ f(t) = \begin{cases} \sqrt[3]{t}, & t \leq 0, \\ t^3, & t > 0. \end{cases} \quad (9) \]
This function is shown in Fig. 5.

![Graph](image)

**Fig. 5.** Nonlinear function \( y(x) \) defined by (8)–(9).

Our choice of the linear test problems is inspired by the observation that the optimal values \( u^*_i \) that correspond to a local area of values of \( x \) depend mostly on the local behavior of the response function \( y(x) \) and on the values of the observation errors in this area. Due to the block structure of \( u^* \), these local values of \( u^*_i \) typically do not change if to perturb the function values \( y(x) \) in distant areas. Therefore, we assume that the local behavior can be well imitated by linear local models.

For the linear models, we consider the following two functions
\[ y_{\text{lin}1}(x) = 0.1x_1 + 0.1x_2, \quad y_{\text{lin}2}(x) = x_1 + x_2. \quad (10) \]
They model slower and faster monotonic increase, respectively.

The nonlinear function combines the considered types of behavior. In addition, depending on the side from which \( x_1 \) or \( x_2 \) approaches zero, the function value changes either sharply, or remarkably slowly.
For the numerical experiments, samples of \( n = 10^2 \), \( n = 10^3 \) and \( n = 10^4 \) observations \( \{X_i\}_{i=1}^n \) were generated with the use of the independent uniform distribution of the explanatory variables in the interval \([-2, 2]\). The error terms \( \varepsilon_i \) in (6) were independent and normally distributed with mean zero and variance one.

It should be emphasized that, in our numerical experiments, the variance of the error \( \varepsilon_i \) is comparable with the function values \( y(X_i) \). Such observations, with a high level of noise, were deliberately chosen for checking the performance of the algorithms in this difficult case.

We call preprocessing the stage at which MR problem (1) is generated. The observed explanatory variables \( \{X_i\}_{i=1}^n \) are involved in formulating this problem implicitly; namely, via the partial order intrinsic in this data set of vectors. Given \( \{X_i\}_{i=1}^n \), we generate a directed acyclic graph \( G(N, E) \) with all the redundant edges removed. The adjacency-matrix representation [10] is used for the graph. The preprocessing is accomplished by a topological sorting of the nodes \( N \).

We used MATLAB for implementing the algorithms GPAV, GPAVR and IBCR. The implementations are based on one of the topological sorts studied in [8], namely, (NumPred). By this sorting, the nodes in \( N \) are sorted in ascending order of the number of their predecessors. The (NumPred) is applied in Algorithms GPAV and IBCR to the graph \( G(N, E) \), as well as on Step 1 of Algorithm GPAVR to the graphs \( G(N, E) \) and \( G(N, E^B) \) in the forward and backward modes, respectively.

The numerical results presented here were obtained on a PC running under Windows XP with a Pentium 4 processor (2.8 GHz, 1GB RAM).

We evaluate performance of the algorithms GPAV and GPAVR by comparing the relative error

\[
e(u_A) = \frac{\varphi(u_A) - \varphi(u^*)}{\varphi(u^*)},
\]

where \( \varphi(u_A) \) is the objective function value in (1) obtained by Algorithm A, and \( \varphi(u^*) \) is the optimal value provided by IBCR.

In [8], it was shown that

\[
\frac{\|u_A - u^*\|}{\|Y - u^*\|} \leq \sqrt{e(u_A)},
\]

which means that \( \sqrt{e(u_A)} \) provides an upper estimate for the relative distance between the approximate solution \( u_A \) and the optimal solution \( u^* \).

Tables 1 and 2 summarize the performance data obtained for the algorithms. For \( n = 10^2 \) and \( n = 10^3 \), we present the average values for 10 MR problems generated as described above. For \( n = 10^4 \), we report the results of solving only one MR problem, because it takes about 40 minutes of CPU time to solve one such large-scale problem with the use of IBCR. The sign ‘—’ corresponds in the tables to the case when IBCR failed to solve problem within 6 hours.

The number of constraints (\#constr.) reported in Table 1 are aimed at indicating how difficult the quadratic programming problem (1) is for the conventional optimization methods when \( n \) is large.
Table 1. Relative error $e(u_A) \cdot 100\%$ for $A = \text{GPAV, GPAVR}$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Model</th>
<th>$n = 10^2$</th>
<th>$n = 10^3$</th>
<th>$n = 10^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#constr. = 322</td>
<td>#constr. = 5497</td>
<td>#constr. = 78170</td>
</tr>
<tr>
<td>GPAV</td>
<td>lin1</td>
<td>0.98</td>
<td>0.77</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>lin2</td>
<td>2.79</td>
<td>2.43</td>
<td>2.02</td>
</tr>
<tr>
<td></td>
<td>nonlin</td>
<td>3.27</td>
<td>5.66</td>
<td>11.56</td>
</tr>
<tr>
<td>GPAVR</td>
<td>lin1</td>
<td>0.01</td>
<td>0.07</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>lin2</td>
<td>0.08</td>
<td>0.12</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td>nonlin</td>
<td>0.00</td>
<td>0.17</td>
<td>0.46</td>
</tr>
</tbody>
</table>

Table 2. Computational time in seconds

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Model</th>
<th>$n = 10^2$</th>
<th>$n = 10^3$</th>
<th>$n = 10^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPAV</td>
<td>lin1</td>
<td>0.02</td>
<td>0.76</td>
<td>89.37</td>
</tr>
<tr>
<td></td>
<td>lin2</td>
<td>0.01</td>
<td>0.71</td>
<td>93.76</td>
</tr>
<tr>
<td></td>
<td>nonlin</td>
<td>0.01</td>
<td>0.67</td>
<td>87.51</td>
</tr>
<tr>
<td>GPAVR</td>
<td>lin1</td>
<td>0.05</td>
<td>1.67</td>
<td>234.31</td>
</tr>
<tr>
<td></td>
<td>lin2</td>
<td>0.05</td>
<td>1.60</td>
<td>197.06</td>
</tr>
<tr>
<td></td>
<td>nonlin</td>
<td>0.04</td>
<td>1.58</td>
<td>192.08</td>
</tr>
<tr>
<td>IBCR</td>
<td>lin1</td>
<td>0.21</td>
<td>129.74</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>lin2</td>
<td>0.09</td>
<td>5.07</td>
<td>2203.10</td>
</tr>
<tr>
<td></td>
<td>nonlin</td>
<td>0.08</td>
<td>6.68</td>
<td>3448.94</td>
</tr>
</tbody>
</table>
The tables show that GPAVR substantially improves the accuracy of the GPAV solution, while its run time is between twice and triple the time of running GPAV. They also demonstrate the limited abilities of IBCR.

5 Conclusions and Future Work

To the best of our knowledge, GPAV is the only practical algorithm able to produce sufficiently accurate solution to very large scale MR problems with partially ordered observations. Up till now, there has not been found any practical algorithm capable of solving such large scale MR problems with as high accuracy as the accuracy provided by the introduced here block refinement technique combined with the GPAV algorithm. This can be viewed as the main contribution of the paper.

In this paper, we focused on solving the MR problem (1) which is related to the first stage of constructing a monotonic response model $u(x)$ of an unknown monotonic response function $y(x)$. The second stage accomplishes the construction of a model $u(x)$, which is a monotonic function and interpolates, in accordance with (7), the fitted response values $u_i$, $i = 1, \ldots, n$, obtained in the first stage.

The quality of the obtained monotonic response model $u(x)$ depends not only on the accuracy of solving problem (1), but also on the interpolation methods used in the second stage. Among the existing main approaches to solving the monotonicity-preserving interpolation problem one can recognize the following three.

One approach [27, 28] involves minimizing some measure of smoothness over a convex cone of smooth functions which are monotone. The drawback of this approach is that the solutions must be found by solving constrained minimization problems, and the solutions are generally somewhat complicated, nonlocal and nonpiecewise polynomial functions.

The second approach is to use a space of piecewise polynomials defined over a partition of the interpolation domain, usually into triangles. Up until now, this approach has been studied only in the case where the data is given on a grid (see e.g. [3, 9]).

The third approach [12] is based on creating girded data from the scattered data. This approach is not suitable for large number of data points $n$, because the number of grid nodes grows with $n$ as $n^p$ and becomes easily unacceptably large, even for the bivariate case ($p = 2$).

Bearing in mind the importance of the second stage and the shortcomings of the existing approaches, we plan to develop efficient monotonicity-preserving methods for interpolation of scattered multivariate data.

References


