<table>
<thead>
<tr>
<th>Title</th>
<th>A rectangular branch-and-bound algorithm for solving a monotonic optimization problem (Mathematical Programming in the 21st Century : Algorithms and Modeling)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Buckland, Paul K.; Kuno, Takahito; Tsushima, Iori</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (2010), 1676: 223-229</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2010-04</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/141248">http://hdl.handle.net/2433/141248</a></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
</tbody>
</table>

Kyoto University
A rectangular branch-and-bound algorithm for solving a monotonic optimization problem

Paul K. Buckland, Takahito Kuno*, and Iori Tsushima

Graduate School of Systems and Information Engineering
University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan

1 Introduction

We consider a class of optimization problem, where the function being optimized is monotonic in an arbitrary number of dimensions, and the feasible region is a polytope, i.e., a closed polyhedral set. When every constraint function is monotonic, i.e., the coefficients are all non-negative, the problem is called a monotonic optimization problem, for which Tuy et.al. have developed a series of algorithms based on rectangular branch-and-bound with \( \omega \)-subdivision [2, 3, 4]. Without assuming monotonicity of constraint functions, we propose here another type of algorithm, based on rectangular branch-and-bound with bisection, and provide some numerical results.

2 Problem setup

Let \( f : S \subset \mathbb{R}^n \rightarrow \mathbb{R}^1 \) be a continuous, nondecreasing function, i.e., for any \( x^1, x^2 \in S \),

\[
    x^1 \leq x^2 \Rightarrow f(x^1) \leq f(x^2).
\]

The problem we wish to consider is to maximize \( f \) on a polytope,

\[
\begin{align*}
    \text{maximize} & \quad f(x) \\
    \text{subject to} & \quad Ax \leq b,
\end{align*}
\]

where \( A \in \mathbb{R}^{m \times n} \) and \( b \in \mathbb{R}^m \). Let us denote the feasible set by

\[
D = \{ x \in \mathbb{R}^n \mid Ax \leq b \}.
\]

*This author was partially supported by the Grant-in-Aid for Scientific Research (B) 20310082 from the Japan Society for the Promotion of Sciences.
which we assumed to be bounded and have a nonempty interior. We also assume that the domain $S$ of $f$ is large enough to contain $D$ in its interior.

3 Algorithm overview

As $D$ is assumed to be bounded, we can compute upper and lower bounds of $x_j$ on $D$, using any algorithm for linear programming:

$$s_j^0 < \min\{x_j \mid x \in D\}, \quad t_j^0 = \max\{x_j \mid x \in D\}, \quad j = 1, \ldots, n.$$ 

Let us denote the rectangle with corner points $s^0$ and $t^0$ by

$$M^0 = (s^0, t^0] = (s_1^0, t_1^0] \times \cdots \times (s_n^0, t_n^0].$$

Clearly, $D$ is a subset of $M^0$, so (1) is equivalent to

$$P_{M^0} \begin{array}{l}
\text{maximize} \\
\text{subject to}
\end{array} f(x) \quad x \in D \cap M^0.$$ 

The rectangular branch-and-bound algorithm we propose subdivides $M^0$ into a set of rectangles $\mathcal{M} = \{M^k \mid k \in \mathcal{K}\}$ satisfying

$$\bigcup_{k \in \mathcal{K}} M^k = M^0, \quad M^k \cap M^\ell = \emptyset \text{ if } k \neq \ell \text{ and } k, \ell \in \mathcal{K},$$

(2)

where $M^k = (s^k, t^k]$, and calculates lower and upper bounds of an optimal solution of each $P_{M^k}$, where $P_{M^k}$ is defined as

$$P_{M^k} \begin{array}{l}
\text{maximize} \\
\text{subject to}
\end{array} f(x) \quad x \in D \cap M^k.$$ 

Each $M^k$ is either fathomed, or else branched with the branches being added to $\mathcal{M}$. The process continues until either an optimal solution to (1) is found, or a solution is found that is within a predetermined tolerance of an optimal solution.

4 Auxiliary problem

To perform both branching and bounding operations, we first calculate a solution to an auxiliary problem.
Let $M$ be any rectangle in $\mathcal{M}$ and consider a subproblem of (reftarget),

\[
\begin{array}{ll}
P_M & \text{maximize } f(x) \\
\text{subject to } & x \in D \cap M,
\end{array}
\]

where

\[ M = (s,t] = (s_1,t_1] \times \cdots \times (s_n,t_n], \quad s_j < t_j, \quad j = 1, \ldots, n. \]

Associated with $P_M$, we define an auxiliary problem

\[
\begin{array}{ll}
\text{minimize} & \max\{t_j - x_j \mid j = 1, \ldots, n\} \\
\text{subject to} & x \in D \\
& x_j \leq t_j, \quad j = 1, \ldots, n,
\end{array}
\]

which is equivalent to a linear programming problem

\[
\begin{array}{ll}
\text{minimize} & z \\
\text{subject to} & Ax \leq b, \\
& 0 \leq t_j - x_j \leq z, \quad j = 1, \ldots, n.
\end{array}
\]

Since $D$ is nonempty and bounded, $Q_M$ has an optimal solution $(\bar{x}, \bar{z})$, and $\bar{x}$ naturally solves (3).

5 Branching operation

Given an optimal solution $(\bar{x}, \bar{z})$ to $Q_M$, there are three possibilities:

- $\bar{z} \leq 0$,
- $\bar{z} \geq t_j - s_j$ for $j = 1, \ldots, n$, or
- $0 < \bar{z} < t_j - s_j$ for some $j$.

Proposition 5.1.

(a) If $\bar{z} \leq 0$, then $M$ contains no feasible solution of (1) better than $\bar{x}$.

(b) If $\bar{z} \geq t_j - s_j$ for $j = 1, \ldots, n$, then $D \cap M = \emptyset$.

Proof. (a) For any $x \in M$, we have $x \leq t$ and so $f(x) \leq f(t)$. We also have $t_j - \bar{x}_j \leq \bar{z}$ for all $j$, so $\bar{z} \leq 0$ implies that $t_j - \bar{x}_j \leq 0$ for all $j$. Hence, $x \leq t \leq \bar{x}$, and we have $f(x) \leq f(\bar{x})$.

(b) Suppose there exists a point $x \in D \cap M$. Then $s \leq x$, so $t_j - x_j < t_j - s_j$ for all $j$. Let $z = \max\{t_j - x_j \mid j = 1, \ldots, n\}$, then $(x, z)$ is a solution to $Q_M$ and we have $z < t_j - x_j$ for some $j$. \[\square\]
Proposition 5.1 tells us we do not need to search $M$ for an optimal solution of (1) if $\bar{z} \geq t_j - s_j$ for $j = 1, \ldots, n$, or if both $\bar{z} \leq 0$ and $\bar{x} \notin M$. If $\bar{z} \leq 0$ and $\bar{x} \in M$, then $\bar{x}$ is an optimal solution to $P_M$ and we need not further search $M$.

Suppose then that the following holds for some index $j$:

$$0 < \bar{z} < t_j - s_j.$$  \hfill (4)

and let

$$\omega = t - \bar{z}e,$$

where $e \in \mathbb{R}^n$ is the all-ones vector. For an arbitrary index $j$ satisfying (4), we have $s_j < \omega_j < t_j$. Therefore, we can divide $M$ along $x_j = \omega_j$ into two rectangles

$$M_j^- = (s_1, t_1] \times \cdots \times (s_{j-1}, t_{j-1}] \times (s_j, \omega_j] \times (s_{j+1}, t_{j+1}] \times \cdots \times (s_n, t_n]$$

$$M_j^+ = (s_1, t_1] \times \cdots \times (s_{j-1}, t_{j-1}] \times (\omega_j, t_j] \times (s_{j+1}, t_{j+1}] \times \cdots \times (s_n, t_n].$$

where we refer to $M_j^-$ and $M_j^+$ as children of $M$ generated via $(\omega, j)$.

This procedure provides us with a branching operation. Removing $M$ and inserting $M_j^-$ and $M_j^+$ into $\mathcal{M}$ satisfies (2).

6 Bounding operation

Because $f$ is a nondecreasing function and $M = [s, t]$, the values $f(s)$ and $f(t)$ provide lower and upper bounds respectively of an optimal solution of $P_M$. We can, however, calculate a better upper bound.

**Proposition 6.1.** If $P_M$ has an optimal solution $x^*$, then

$$f(s) \leq f(x^*) \leq \max\{f(v_j) \mid j = 1, \ldots, n\},$$

where

$$v_j = (t_1, \ldots, t_{j-1}, \omega_j, t_{j+1}, \ldots, t_n)^T.$$

**Proof.** The lower bound $f(s)$ follows from the definition of $M$ and the fact that $f$ is a nondecreasing function.

If $\bar{z} \leq 0$, then $t \leq v_j$ for all $j$, and so $f(x) \leq f(t) \leq f(v_j)$ for all $v_j$ and $x \in M$. If $\bar{z} > 0$, then for each $j$ we have either

$$0 < \bar{z} < t_j - s_j,$$  \hfill (5)

or

$$\bar{z} \geq t_j - s_j.$$  \hfill (6)
For each $j$ that satisfies (5), we define $M_j^-$ as in Section 5,

$$M_j^- = (s_1, t_1] \times \cdots \times (s_{j-1}, t_{j-1}] \times (s_j, \omega_j] \times (s_{j+1}, t_{j+1}] \times \cdots \times (s_n, t_n],$$

where $\omega = t - \bar{z}e$, and for each $j$ that satisfies (6), we let

$$M_j^- = \emptyset.$$

For either case, we define $M_j^+$ as in Section 5,

$$M_j^+ = (s_1, t_1] \times \cdots \times (s_{j-1}, t_{j-1}] \times (\omega_j, t_j] \times (s_{j+1}, t_{j+1}] \times \cdots \times (s_n, t_n].$$

Note that

$$M_j^- \cup M_j^+ \supset M \quad \text{and} \quad M_j^- \cap M_j^+ = \emptyset. \quad (7)$$

For any $j$ that satisfies (5), it is clear that $M_j^- = (s, v_j]$ and therefore

$$f(x) \leq f(v_j), \quad \forall x \in M_j^-,$$

and since $M_j^- = \emptyset$ for all other $j$, we have

$$f(x) \leq \max\{f(v_j) \mid j = 1, \ldots, n\}, \quad \forall x \in \bigcup_{j=1}^{n} M_j^-.$$

To complete the proof, we show that the set $M \setminus \bigcup_{j=1}^{n} M_j^-$ does not contain any feasible points of $P_M$. Let

$$M' = M \setminus \bigcup_{j=1}^{n} M_j^-.$$

Then

$$M' = \bigcap_{j=1}^{n} (M \setminus M_j^-) \subset \bigcup_{j=1}^{n} M_j^+ = (\omega, t],$$

where (8) follows from (7). Let $s' = \omega$ and $t' = t$ so that $M' = (s', t']$. Solving $P_{M'}$ we obtain an optimal solution $(\bar{x}', F)$. But $P_{M'}$ is the same problem as $P_M$ because $t' = t$, so $\bar{z}' = \bar{z}$, which means that

$$t' - \bar{z}'e = t - \bar{z}e = \omega = s'.$$
Therefore, \( \vec{z}' = t' - s' \), so \( \vec{z}' = t'_j - s'_j \) for all \( j = 1, \ldots, n \), and by Proposition 5.1 we have \( D \cap M' = \emptyset \). \( \square \)

7 Prototype algorithm

We are now ready to state a prototype algorithm. In the pseudocode that follows, we use the notation:

- \( \mathcal{M} \): set of \( M^k \) yet to be fathomed.
- \( \beta^k \): upper bound of an optimal solution to \( P_{M^k} \).
- \( \alpha \): maximum of the the lower bounds of optimal solutions to \( P_{M^k} \) where each \( M^k \) has been bounded.
- \( x^* \): current best solution to (1).
- \( \epsilon \): given positive tolerance.

algorithm prototype_rectangle_bb
begin
   calculate \( s^0, t^0; M^0 := (s^0, t^0) \);
   \( \mathcal{M} := \{ M^0 \}; \alpha := f(s^0); \beta^0 := f(t^0) \);
   while:\( \epsilon \) > 0
      select a rectangle \( M = (s, t) \in \mathcal{M}; \mathcal{M} := \mathcal{M} \setminus \{ M \} \);
      let \( (\overline{x}, \overline{z}) \) be an optimal solution to \( Q_M \);
      if \( \alpha < f(\overline{x}) \) then begin \( \alpha := f(\overline{x}); x^* := \overline{x} \) end;
      if \( \overline{z} < \max\{t_j - s_j | j = 1, \ldots, n\} \) then begin
         calculate \( \beta^M := \max\{f(v_j) | j = 1, \ldots, n\} \), an upper bound of \( M \);
         if \( \beta^M > \alpha \) then begin
            if \( \alpha < f(s) \) then \( \alpha := f(s^k) \);
            /* Branching operation */
            let \( i \) be an index satisfying both \( \overline{z} = t_i - \overline{x}_i \) and \( s_i < \overline{x}_i < t_i \);
            calculate \( M_i^- \) and \( M_i^+ \), the children of \( M \) generated via \( (\omega, i) \);
            \( \mathcal{M} := \mathcal{M} \cup \{ M_i^- , M_i^+ \} \);
            /* Pruning operation */
            \( \mathcal{M} := \mathcal{M} \setminus \{ M | \beta^M \leq \alpha \} \);
         end
      end
   end
end;
Table 1: CPU seconds taken to find an optimal solution.

<table>
<thead>
<tr>
<th>n</th>
<th>[ e^{x_j} ]</th>
<th>[ x_j^3 ]</th>
<th>[ \log x_j ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.144</td>
<td>3.293</td>
<td>0.982</td>
</tr>
<tr>
<td>3</td>
<td>1.671</td>
<td>42.717</td>
<td>7.498</td>
</tr>
<tr>
<td>4</td>
<td>24.392</td>
<td>79.942</td>
<td>20.781</td>
</tr>
<tr>
<td>5</td>
<td>62.929</td>
<td>93.442</td>
<td>67.274</td>
</tr>
</tbody>
</table>

8 Numerical results

We ran the prototype algorithm on some instances of optimizing three nonlinear functions over a set of randomly generated polytopes of dimension 2, 3, 4, and 5. The three functions are

\[ \sum_{j=1}^{n} e^{x_j}, \sum_{j=1}^{n} x_j^3, \text{ and } \sum_{j=1}^{n} \log x_j. \]

The algorithm performed in GNU Octave v3.2 [1] for Microsoft Windows, on a computer with a 2.8 GHz Intel Core 2 Duo with 2 GB of memory. The results are presented in Table 1. Since this experiment is preliminary, we cannot draw any conclusion. But the time take to find optimal solutions increases significantly as the number of dimensions increases, and so we have to make numerous improvements in the algorithm.

9 Closing comments

We have presented a prototype branch-and-bound algorithm for solving a certain class of monotonic optimization problem. Further consideration is now required to address the significant increase in time take to solve as the number of dimensions increases. One possibility for addressing this problem is to implement sensitivity analysis, as successive problems Q_M differ by only one linear constraint.

Convergence of the algorithm will be shown in a future publication on the topic.

References


