数理解析研究所講究録
第1676巻 2010年 223-229

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

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

Graduate School of Systems end 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 ω-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 $f : S \subset \mathbb{R}^n \to \mathbb{R}^1$ be a continuous, nondecreasing function, i.e., for any $\mathbf{x}^1, \mathbf{x}^2 \in S$,

$$\mathbf{x}^1 \leq \mathbf{x}^2 \Rightarrow f(\mathbf{x}^1) \leq f(\mathbf{x}^2).$$

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

$$\left| \begin{array}{ll} \text{maximize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{Ax} \leq \mathbf{b}, \end{array} \right. \tag{1}$$

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

$$D = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{Ax} \leq \mathbf{b}\},$$

*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:

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

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

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

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

$$\mathrm{P}_{M^0} \left| \begin{array}{ll} \text{maximize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in D \cap M^0. \end{array} \right.$$

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

$$\bigcup_{k \in \mathscr{K}} M^k = M^0, \quad M^k \bigcap M^\ell = \emptyset \text{ if } k \neq \ell \text{ and } k, \ell \in \mathscr{K}, \tag{2}$$

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

$$\mathrm{P}_{M^k} \left| \begin{array}{ll} \text{maximize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in D \cap M^k. \end{array} \right.$$

Each M^k is either fathomed, or else branched with the branches being added to $\mathscr{M}$. 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),

$$P_M \left| \begin{array}{ll} \text{maximize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in D \cap M, \end{array} \right.$$

where

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

Associated with P_M, we define an auxiliary problem

$$\left| \begin{array}{ll} \text{minimize} & \max\{t_j - x_j \mid j = 1, \ldots, n\} \\ \text{subject to} & \mathbf{x} \in D \\ & x_j \leq t_j, \quad j = 1, \ldots, n, \end{array} \right. \tag{3}$$

which is equivalent to a linear programming problem

$$Q_M \left| \begin{array}{ll} \text{minimize} & z \\ \text{subject to} & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & 0 \leq t_j - x_j \leq z, \quad j = 1, \ldots, n. \end{array} \right.$$

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

5 Branching operation

Given an optimal solution $(\bar{\mathbf{x}}, \bar{z})$ 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 $\bar{\mathbf{x}}$.

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

Proof. (a) For any $\mathbf{x} \in M$, we have $\mathbf{x} \leq \mathbf{t}$ and so $f(\mathbf{x}) \leq f(\mathbf{t})$. 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, $\mathbf{x} \leq \mathbf{t} \leq \bar{\mathbf{x}}$, and we have $f(\mathbf{x}) \leq f(\bar{\mathbf{x}})$.

(b) Suppose there exists a point $\mathbf{x} \in D \cap M$. Then $\mathbf{s} \leq \mathbf{x}$, 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 $(\mathbf{x}, z)$ is a solution to Q_M and we have $z < t_j - x_j$ for some j. □

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 $\bar{\mathbf{x}} \notin M$. If $\bar{z} \leq 0$ and $\bar{\mathbf{x}} \in M$, then $\bar{\mathbf{x}}$ is an optimal solution to P_M and we need not further search M.

Suppose then that the following holds for some index j:

$$0 < \bar{z} < t_j - s_j, \tag{4}$$

and let

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

where $\mathbf{e} \in \mathbb{R}^n$ 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 $x_j = \omega_j$ int 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 (ω, j).

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

6 Bounding operation

Because f is a nondecreasing function and $M = (\mathbf{s}, \mathbf{t}]$, the values $f(\mathbf{s})$ and $f(\mathbf{t})$ 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 $\mathbf{x}^*$, then*

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

where

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

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

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

$$0 < \bar{z} < t_j - s_j, \tag{5}$$

or

$$\bar{z} \geq t_j - s_j. \tag{6}$$

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

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

For any j that satisfies (5), it is clear that $M_j^- = (\mathbf{s}, \mathbf{v}_j]$ and therefore

$$f(\mathbf{x}) \leq f(\mathbf{v}_j), \quad \forall \mathbf{x} \in M_j^-,$$

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

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

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

$$\begin{aligned} M' &= \bigcap_{j=1}^{n} (M \setminus M_j^-) \\ &\subset \bigcup_{j=1}^{n} M_j^+ \qquad (8) \\ &= (\omega, \mathbf{t}], \end{aligned}$$

where (8) follows from (7). Let $\mathbf{s}' = \omega$ and $\mathbf{t}' = \mathbf{t}$ so that $M' = (\mathbf{s}', \mathbf{t}']$.

Solving $\mathrm{P}_{M'}$ we obtain an optimal solution $(\bar{\mathbf{x}}', \bar{z}')$. But $\mathrm{P}_{M'}$ is the same problem as P_M because $\mathbf{t}' = \mathbf{t}$, so $\bar{z}' = \bar{z}$, which means that

$$\mathbf{t}' - \bar{z}'\mathbf{e} = \mathbf{t} - \bar{z}\mathbf{e} = \omega = \mathbf{s}'.$$

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

7 Prototype algorithm

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

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

```
algorithm prototype_rectangle_bb
begin
    calculate s^0, t^0; M^0 := (s^0, t^0 9];
    M := {M^0}; α := f(s^0); β^0 := f(t^0);
    while z̄ > ε
        select a rectangle M = (s, t] ∈ M; M := M \ {M};
                                                    /* Bounding operation */
        let (x̄, z̄) be an optimal solution to Q_M;
        if α < f(x̄ then begin α := f(x̄; x* := x̄ end;
        if z̄ < max{t_j − s_j | j = 1,...,n} then begin
            calculate β^M := max{f(v_j) | j = 1,...,n}, an upper bound of M;
            if β^M > α then begin
                if α < f(s) then α := f(s^k);
                                                    /* Branching operation */
                let i be an index satisfying both z̄ = t_i − x̄_i and s_i < x̄_i < t_i;
                calculate M_i^− and M_i^+, the children of M generated via (ω, i);
                M := M ∪ {M_i^−, M_i^−};
                                                    /* Pruning operation */
                M := M \ {M | β^M ≤ α};
            end
        end
    end
end;
```

Table 1: CPU seconds taken to find an optimal solution.

		f		
		$\sum e^{x_j}$	$\sum x_j^3$	$\sum \log x_j$
n	2	0.144	3.293	0.982
	3	1.671	42.717	7.498
	4	24.392	79.942	20.781
	5	62.929	93.442	67.274

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

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

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

[1] Octave, http://www.gnu.org/software/octave/.

[2] Rubinov, A, H.Tuy, and H.Mays, "Algorithm for a monotonic global optimization", *Optimization* **49** (2001), 205–221.

[3] Tuy, H., "Monotonic optimization: problems and solution approaches", *SIAM Journal on Optimization* **11** (2000), 464–494.

[4] Tuy, H., F.Al-Khayyal, and P.T.Thach, "Monotonic optimization: branch and cut methods", in *Essays and Surveys in Global Optimization*, Springer (2005), 39–78.