Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol.1, No.2, pp. - , May 2005

A SIMPLICIAL ALGORITHM WITH TWO-PHASE BOUNDING OPERATION FOR A CLASS OF CONCAVE MINIMIZATION PROBLEMS

Takahito Kuno* and Hidetoshi Nagai

Dedicated to Professor Hiroshi Konno on his 65th birthday.

Abstract: In this paper, we develop a simplicial branch-and-bound algorithm with two-phase bounding operation for solving a class of concave minimization problems to which many of problems with low rank nonconvexity reduce. In the first phase of the bounding operation, we enlarge the feasible set of each linear programming relaxed problem to facilitate application of some procedures for improving the efficiency. In the second phase, we tighten the lower bound deteriorated by this enlargement, using a Lagrangian relaxation. Computational results indicate that the proposed algorithm is promising, compared with a standard simplicial branch-and-bound algorithm.

Key words: *global optimization, concave minimization, low-rank nonconvexity, branch-and-bound algorithm, Lagrangian relaxation*

Mathematics Subject Classification: *90-08, 90C26, 90C30, 90C57*

1 Introduction

In this paper, we develop a branch-and-bound algorithm for solving a class of concave minimization problems to which many of problems with low rank nonconvexity [7] reduce. The major feature of this class is that the variables involved in the objective function are a small fraction of the whole variables. As a typical example, let us consider the linear multiplicative program [14, 6, 8, 10]:

$$\left|\begin{array}{ll} \text{minimize} & \prod_{i=1}^{r}(\mathbf{c}_i^{\mathsf{T}}\mathbf{y}+c_{i0}) \\ \text{subject to} & \mathbf{B}\mathbf{y}\leq\mathbf{b},\quad \mathbf{y}\geq\mathbf{0}, \end{array}\right. \tag{1.1}$$

where $\mathbf{c}_i^{\mathsf{T}}\mathbf{y}+c_{i0}>0$ for any feasible solution $\mathbf{y}$. In general, the number r of affine functions in the objective is assumed to be far less than the dimensionality of (1.1). If we introduce a vector $\mathbf{x}=(x_1,\ldots,x_r)^{\mathsf{T}}$ of auxiliary variables, (1.1) reduces to a concave minimization problem in our target class:

$$\left|\begin{array}{lll} \text{minimize} & \sum_{i=1}^{r}\log(x_i) & \\ \text{subject to} & -x_i+\mathbf{c}_i^{\mathsf{T}}\mathbf{y}\leq -c_{i0}, & i=1,\ldots,r \\ & \mathbf{B}\mathbf{y}\leq\mathbf{b}, & (\mathbf{x},\mathbf{y})\geq\mathbf{0}. \end{array}\right. \tag{1.2}$$

*The author was partially supported by the Grand-in-Aid for Scientific Research (C)(2) 15560048 from the Japan Society for the Promotion of Science.

Another example is the production-transportation problem [9, 11, 16, 17]. This is a kind of network flow problem and minimizes the sum of concave production and linear transportation costs. If we move the transportation cost function to the set of constraints by means of an auxiliary variable, the objective function is left with only the concave production cost and auxiliary variable. For various other examples that can undergo similar transformations to this class of problems, the readers are referred to the textbook on low-rank nonconvex structures [7].

If the objective function is separable into a sum of univariate functions like (1.2), problems of this class can be solved rather efficiently using the rectangular branch-and-bound algorithm [3, 8, 9]. To deal with a wider range of problems, we do not assume the separability in this paper, but tailor the simplicial branch-and-bound algorithm [4] to suit the class and to facilitate application of some procedures for improving the efficiency. In Section 2, after giving the problem settings, we will review the basic workings of the simplicial branch-and-bound algorithm and explore difficulties in its implementation. In Section 3, to overcome those difficulties, we will modify the linear programming relaxed problem to be solved in the bounding operation, by enlarging the feasible set. This modification does not affect the convergence property of the algorithm but naturally deteriorates the quality of the lower bound on the optimal value. To prevent rapid growth of the branching tree, we will propose the second bounding operation based on a Lagrangian relaxation in Section 4, and give a detailed description of the algorithm incorporating two bounding operations. Computational results of comparison with the standard simplicial branch-and-bound algorithm are reported in Section 5. Lastly, we will discuss some remaining issues to be resolved in the future, in Section 6.

2 Problem Settings and the Simplicial Algorithm

Let $D \subset \mathbb{R}^r$ be an open convex set and $f : D \to \mathbb{R}$ a concave function. The problem we consider in this paper is a concave minimization over a polyhedral set:

$$\left| \begin{array}{lll} \text{minimize} & z = & f(\mathbf{x}) \\ \text{subject to} & & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array} \right. \tag{2.1}$$

where $\mathbf{A} \in \mathbb{R}^{m \times r}$, $\mathbf{B} \in \mathbb{R}^{m \times (n-r)}$, $\mathbf{b} \in \mathbb{R}^m$ and $1 \leq r \leq n$. Let us denote the feasible set and its projection onto the space of $\mathbf{x}$, respectively by

$$S = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b},\ (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}\}, \quad X = \{\mathbf{x} \in \mathbb{R}^r \mid \exists \mathbf{y}, (\mathbf{x}, \mathbf{y}) \in S\}.$$

Using these notation, (2.1) can be embedded in $\mathbb{R}^r$:

$$\mathrm{P} \left| \begin{array}{lll} \text{minimize} & z = & f(\mathbf{x}) \\ \text{subject to} & & \mathbf{x} \in X. \end{array} \right.$$

We assume that S is bounded and has a nonempty interior. The same is then true for the projection X; and so we have $v = \max\{\sum_{j=1}^r x_j \mid \mathbf{x} \in X\}$. We assume the domain D of f large enough to include the set $\{\mathbf{x} \in \mathbb{R}^r \mid 0 \leq x_j \leq v, j = 1, \ldots, r\}$.

Unless the objective function is separable, the most often used solution method for concave minimization is the simplicial branch-and-bound algorithm [4, 5, 15]. Let us run through its basic workings on our problem P.

Overview of the Standard Simplicial Algorithm

In the simplicial branch-and-bound algorithm, we first need to define an r-simplex Δ^1 including X. For the value v defined above, let $\mathbf{v}_i^1 = v\mathbf{e}_i$ for $i = 1, \ldots, r$, where $\mathbf{e}_i \in \mathbb{R}^r$ is the ith unit vector. Then Δ^1 is given by the convex hull of $\mathbf{v}_1^1, \ldots, \mathbf{v}_r^1$ and $\mathbf{v}_{r+1}^1 = \mathbf{0}$. Since X is just a subset of Δ^1, problem P is equivalent to the following with $\Delta = \Delta^1$:

$$\mathrm{P}(\Delta) \left| \begin{array}{lll} \text{minimize} & z = & f(\mathbf{x}) \\ \text{subject to} & & \mathbf{x} \in X \cap \Delta. \end{array} \right.$$

Then, as we subdivide Δ^1 into smaller simplices Δ^j, $j \in \mathcal{L}$, satisfying

$$\bigcup_{j \in \mathcal{L}} \Delta^j = \Delta^1, \quad \mathrm{int}(\Delta^j) \cap \mathrm{int}(\Delta^k) = \emptyset \ \text{ if } \ j \neq k,$$

we solve each subproblem $\mathrm{P}(\Delta^j)$ of $\mathrm{P}(\Delta^1)$ with a feasible set $X \cap \Delta^j$, where $\mathrm{int}(\cdot)$ represents the set of interior points. Since $\mathrm{P}(\Delta^j)$ is of the same class as the initial $\mathrm{P}(\Delta^1)$, we cannot solve it in a direct manner. Instead, the following recursive method of three steps is used:

Let $\mathcal{L} := \{1\}$ and $k := 1$. Repeat Steps 1–3 until $\mathcal{L} = \emptyset$.

Step 1. Take an appropriate index j_k out of $\mathcal{L}$ and let $\Delta := \Delta^{j_k}$.

Step 2 (bounding operation). Compute a lower bound z^k on the optimal value $z(\Delta)$ of $\mathrm{P}(\Delta)$. If $f(\mathbf{x}^*) - z^k \leq \epsilon$ for the best feasible solution $\mathbf{x}^*$ to P obtained so far, discard Δ from further consideration.

Step 3 (branching operation). Otherwise, divide the simplex Δ into two subsimplices Δ^{2k}, Δ^{2k+1} and add their indices to $\mathcal{L}$. Let $k := k + 1$.

In this description, $\epsilon \geq 0$ is a given tolerance, and $z(\Delta)$ is regarded as $+\infty$ if $X \cap \Delta = \emptyset$. When the set $\mathcal{L}$ comes to be empty, we have a globally ϵ-optimal solution $\mathbf{x}^*$ to problem P. However, if the value of ϵ is set to zero, the simplicial branch-and-bound algorithm does not terminate in general, and generates an infinite sequence of nested simplices $\{\Delta^{k_\ell} \mid \ell = 1, 2, \ldots\}$ such that

$$\Delta^{k_1} \supset \Delta^{k_2} \supset \cdots, \quad X \cap \left(\bigcap_{\ell=1}^{\infty} \Delta^{k_\ell} \right) \neq \emptyset.$$

Even in the case where $\epsilon > 0$, to terminate the algorithm in finite steps, we have to subdivide Δ^1 in a way that makes $\cap_{\ell=1}^{\infty} \Delta^{k_\ell}$ a singleton. The simplest subdivision rule to ensure this exhaustiveness is *bisection*. We select the longest edge of $\Delta = \mathrm{conv}(\{\mathbf{v}_1, \ldots, \mathbf{v}_{r+1}\})$, say $\mathbf{v}_p$–$\mathbf{v}_q$, and divide it at a fixed ratio of $\alpha \in (0, 1/2]$, where $\mathrm{conv}(\cdot)$ is the convex hull. Letting $\mathbf{v} = (1 - \alpha)\mathbf{v}_p + \alpha\mathbf{v}_q$, then Δ^{2k} and Δ^{2k+1} are given as follows

$$\Delta^{2k} = \mathrm{conv}(\{\mathbf{v}_i \mid i \neq p\} \cup \{\mathbf{v}\}), \quad \Delta^{2k+1} = \mathrm{conv}(\{\mathbf{v}_i \mid i \neq q\} \cup \{\mathbf{v}\}).$$

If $\epsilon > 0$ and the bisection rule is adopted, we can obtain an ϵ-optimal solution to P after a finite number of steps, using either of the usual selection rules at Step 1:

Depth first. The set $\mathcal{L}$ is maintained as a list of *stack*. An index j_k is taken from the top of $\mathcal{L}$; and $2k$, $2k + 1$ are put back to the top at Step 3.

Best bound. The set $\mathcal{L}$ is maintained as a list of *priority queue*. An index j_k of least z^k is taken out of $\mathcal{L}$.

Linear Programming Relaxation at Step 2

The most time-consuming step in the simplicial branch-and-bound algorithm is Step 2. As is well known, the efficiency of algorithms of this kind depends largely on this bounding operation. At Step 2, to compute a lower bound z^k, we usually replace the objective function f of $\mathrm{P}(\Delta)$ by its convex envelope g on Δ and solve a problem:

$$\overline{\mathrm{P}}(\Delta) \left| \begin{array}{lll} \text{minimize} & z = & g(\mathbf{x}) \\ \text{subject to} & & \mathbf{x} \in X \cap \Delta. \end{array} \right.$$

The convex envelope g is an affine function which agrees with f at $r+1$ vertices of Δ. Since Δ is given by the vertices, we can easily determine the value of g at any point $\mathbf{x} \in \Delta$ if we have $\mathbf{x}$ as a convex combination of $\mathbf{v}_i$, $i = 1, \ldots, r+1$:

$$\mathbf{x} = \sum_{i=1}^{r+1} \mathbf{v}_i \xi_i, \quad \sum_{i=1}^{r+1} \xi_i = 1, \quad \boldsymbol{\xi} = (\xi_1, \ldots, \xi_{r+1})^{\mathsf{T}} \geq \mathbf{0}. \tag{2.2}$$

By the concavity of f, we immediately have

$$g(\mathbf{x}) = \sum_{i=1}^{r+1} f(\mathbf{v}_i)\xi_i \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in \Delta. \tag{2.3}$$

Substituting (2.2) into $\overline{\mathrm{P}}(\Delta)$, we see that $\overline{\mathrm{P}}(\Delta)$ is equivalent to a linear program of $n+1$ variables:

$$\left| \begin{array}{lll} \text{minimize} & z = & \mathbf{f}^{\mathsf{T}}\boldsymbol{\xi} \\ \text{subject to} & & \mathbf{AV}\boldsymbol{\xi} + \mathbf{By} \leq \mathbf{b} \\ & & \mathbf{e}^{\mathsf{T}}\boldsymbol{\xi} = 1, \quad (\boldsymbol{\xi}, \mathbf{y}) \geq \mathbf{0}, \end{array} \right. \tag{2.4}$$

where $\mathbf{e} \in \mathbb{R}^{r+1}$ is an all-ones vector and

$$\mathbf{f} = [f(\mathbf{v}_1), \ldots, f(\mathbf{v}_{r+1})]^{\mathsf{T}}, \quad \mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_{r+1}]. \tag{2.5}$$

Obviously, (2.4) has an optimal solution $(\overline{\boldsymbol{\xi}}, \overline{\mathbf{y}})$ if and only if $X \cap \Delta \neq \emptyset$. Then we may set z^k to

$$\overline{z} = \begin{cases} \mathbf{f}^{\mathsf{T}}\overline{\boldsymbol{\xi}} & \text{if } X \cap \Delta \neq \emptyset \\ +\infty & \text{otherwise.} \end{cases}$$

Note that when $X \cap \Delta \neq \emptyset$, we have a feasible solution $\overline{\mathbf{x}}$ to the subproblem $\mathrm{P}(\Delta)$, and to the target problem P, by letting $\overline{\mathbf{x}} = \mathbf{V}\overline{\boldsymbol{\xi}}$. We can therefore update the incumbent $\mathbf{x}^*$ with $\overline{\mathbf{x}}$ if necessary.

Certainly, the linearized subproblem $\overline{\mathrm{P}}(\Delta)$ is far easier to solve than $\mathrm{P}(\Delta)$. However, since the number of subproblems generated in the course of iterating Steps 1–3 is an exponential in r, in the worst case, we cannot obtain an optimal solution nor an ϵ-optimal solution to P within a practical amount of time if we solve each $\overline{\mathrm{P}}(\Delta)$ from scratch. In the rectangular and combinatorial branch-and-bound algorithms, one can solve linearized subproblems successively using sensitivity analysis of the simplex method, or using specialized algorithms if the original problem has some favorable structure. Unfortunately, such a procedure does not work well on $\overline{\mathrm{P}}(\Delta)$ because

(a) each (2.4) associated with $\overline{\mathrm{P}}(\Delta)$ has a different constraint matrix, and

(b) no (2.4) inherits the structure of the original problem (2.1).

Due to (a), we cannot utilize the optimal solution to $\overline{\mathrm{P}}(\Delta^{j_{k-1}})$ as the initial solution in solving $\overline{\mathrm{P}}(\Delta^{j_k})$ through sensitivity analysis, because it might be neither feasible nor dual feasible for (2.4) associated with $\overline{\mathrm{P}}(\Delta^{j_k})$. Moreover, due to (b), even if the original problem (2.1) has a network structure for instance, we cannot apply any network flow algorithms to each $\overline{\mathrm{P}}(\Delta)$. To overcome these difficulties, we need to add some new twists to the relaxation of each subproblem $\mathrm{P}(\Delta)$ at Step 2.

3 Modified Linear Programming Relaxation

One way of sweeping away both difficulties (a) and (b) is to replace the constraint $\mathbf{x} \in \Delta$ in $\overline{\mathrm{P}}(\Delta)$ by a simple bounding constraint on $\mathbf{x}$. Let

$$\left.\begin{array}{l} s_j = \min\{v_{ij} \mid i = 1, \ldots, r+1\} \\ t_j = \max\{v_{ij} \mid i = 1, \ldots, r+1\} \end{array}\right\} \quad j = 1, \ldots, r, \tag{3.1}$$

where v_{ij} denotes the jth component of $\mathbf{v}_i$. Also let

$$\Gamma(\Delta) = \{\mathbf{x} \in \mathbb{R}^r \mid \mathbf{s} \leq \mathbf{x} \leq \mathbf{t}\},$$

where $\mathbf{s} = (s_1, \ldots, s_r)^\mathsf{T}$ and $\mathbf{t} = (t_1, \ldots, t_r)^\mathsf{T}$. Then we have $\Delta \subset \Gamma(\Delta)$. Instead of $\overline{\mathrm{P}}(\Delta)$, we may solve the following to obtain a lower bound z^k at Step 2:

$$\widetilde{\mathrm{P}}(\Delta) \left| \begin{array}{lll} \text{minimize} & z = & g(\mathbf{x}) \\ \text{subject to} & & \mathbf{x} \in X \cap \Gamma(\Delta). \end{array}\right.$$

However, if we transform the variables $\mathbf{x}$ into $\boldsymbol{\xi}$ via (2.2), we would face the same difficulties as (a) and (b). Here, we try another way to draw an explicit linear programming representation of $\widetilde{\mathrm{P}}(\Delta)$.

Suppose that the convex envelope g of f is given by $\mathbf{c}^\mathsf{T}\mathbf{x} + c_{r+1}$, where $\mathbf{c} \in \mathbb{R}^r$ and $c_{r+1} \in \mathbb{R}$. Since g agrees with f at $r+1$ vertices of Δ, the following equations hold:

$$\mathbf{c}^\mathsf{T}\mathbf{v}_i + c_{r+1} = f(\mathbf{v}_i), \quad i = 1, \ldots, r+1.$$

Note that $\mathbf{v}_i$'s are affinely independent if Δ is generated according to the bisection rule. By adding an all-ones vector $\mathbf{e}^\mathsf{T}$ to $\mathbf{V}$ given in (2.5) as the $(r+1)$st row, we have a nonsingular matrix:

$$\mathbf{U} = \begin{bmatrix} \mathbf{V} \\ \mathbf{e}^\mathsf{T} \end{bmatrix},$$

and for $\mathbf{f}$ in (2.5) we have

$$[\mathbf{c}^\mathsf{T}, c_{r+1}] = \mathbf{f}^\mathsf{T}\mathbf{U}^{-1}.$$

Thus, g is specified as $g(\mathbf{x}) = \mathbf{c}^\mathsf{T}\mathbf{x} + c_{r+1}$, and $\widetilde{\mathrm{P}}(\Delta)$ is represented explicitly as a linear program:

$$\left| \begin{array}{lll} \text{minimize} & z = & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{subject to} & & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \quad \mathbf{s} \leq \mathbf{x} \leq \mathbf{t}, \quad \mathbf{y} \geq \mathbf{0}. \end{array}\right. \tag{3.2}$$

If $X \cap \Gamma(\Delta) \neq \emptyset$, then (3.2) has an optimal solution $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})$. The following can be an alternative for the lower bound z^k:

$$\widetilde{z} = \begin{cases} \mathbf{c}^\mathsf{T}\widetilde{\mathbf{x}} + c_{r+1} & \text{if } X \cap \Gamma(\Delta) \neq \emptyset \\ +\infty & \text{otherwise.} \end{cases}$$

Lemma 3.1 *Among $\widetilde{z}$, $\overline{z}$ and the optimal value $z(\Delta)$ of $P(\Delta)$, there is a relationship:*

$$\widetilde{z} \leq \overline{z} \leq z(\Delta).$$

Proof. Immediately follows from (2.3) and the inclusion relation of the feasible sets of $\widetilde{\mathrm{P}}(\Delta)$ and $\overline{\mathrm{P}}(\Delta)$. □

It should be emphasized that each (3.2) has the same set of constraints as the original (2.1), except for the bounding constraint $\mathbf{s} \leq \mathbf{x} \leq \mathbf{t}$, though associated with a different subproblem $\mathrm{P}(\Delta)$. The bounding constraint can be treated almost the same way as the usual nonnegative constraint in the simplex and network flow algorithms [1, 2]. Therefore, when (2.1) has some favorable structure, we can exploit it in solving each (3.2). In general [2], we can generate an optimal solution $(\widetilde{\mathbf{x}}^k, \widetilde{\mathbf{y}}^k)$ to (3.2) associated with $\widetilde{\mathrm{P}}(\Delta^{j_k})$ from the preceding $(\widetilde{\mathbf{x}}^{k-1}, \widetilde{\mathbf{y}}^{k-1})$ in two steps: (i) restore the feasibility of $(\widetilde{\mathbf{x}}^{k-1}, \widetilde{\mathbf{y}}^{k-1})$ for $\widetilde{\mathrm{P}}(\Delta^{j_k})$ with dual pivoting operations, and (ii) reestablish the optimality of the resulting feasible basic solution with primal pivoting operations. Since $(\widetilde{\mathbf{x}}^{k-1}, \widetilde{\mathbf{y}}^{k-1})$ violates only the bounding constraint, step (i) requires a very few pivoting operations. If step (i) fails, both $\widetilde{\mathrm{P}}(\Delta^{j_k})$ and $\mathrm{P}(\Delta^{j_k})$ are infeasible.

Convergence Property When Using $\widetilde{\mathrm{P}}(\Delta)$

Any optimal solution $\widetilde{\mathbf{x}}$ to $\widetilde{\mathrm{P}}(\Delta)$ obtained by solving (3.2) is obviously feasible for the target problem P; and hence we can update the incumbent $\mathbf{x}^*$ with $\widetilde{\mathbf{x}}$ if necessary. However, $\widetilde{\mathbf{x}}$ might be infeasible for $\mathrm{P}(\Delta)$ and can satisfy

$$f(\widetilde{\mathbf{x}}) < \widetilde{z}, \tag{3.3}$$

unlike the optimal solution $\overline{\mathbf{x}}$ of $\overline{\mathrm{P}}(\Delta)$. If (3.3) holds, Δ contains no feasible solution better than $\widetilde{\mathbf{x}}$, because $\widetilde{z}$ is a lower bound of f on $X \cap \Delta$, and we can discard Δ from further consideration. In addition to this, we can discard Δ if

$$f(\widetilde{\mathbf{x}}) < f(\mathbf{v}_i), \quad i = 1, \ldots, r+1. \tag{3.4}$$

Recall that f is concave and achieves the minimum on Δ at some vertex. In other words, the minimum of $f(\mathbf{v}_i)$'s is another lower bound on the value $z(\Delta)$ of $\mathrm{P}(\Delta)$ which minimizes f on $X \cap \Delta \subset \Delta$. Let us define the lower bound z^k at Step 2 as

$$z^k = \max\{\tilde{z}, \min\{f(\mathbf{v}_1), \ldots, f(\mathbf{v}_{r+1})\}\}. \tag{3.5}$$

If neither (3.3) nor (3.4) holds, the simplicial branch-and-bound algorithm might generate an infinite sequence of nested simplices $\{\Delta^{k_\ell} \mid \ell = 1, 2, \ldots\}$ in Δ when ϵ is zero. Even in that case, we can show that z^{k_ℓ} defined in (3.5) tends to $f(\widetilde{\mathbf{x}}^{k_\ell})$ as $\ell \to \infty$, because $\Gamma(\Delta^{k_\ell})$ shrinks to a single point as Δ^{k_ℓ} does if the bisection rule is adopted. When $\epsilon > 0$, it follows from this property that $f(\mathbf{x}^*) - z^k \leq \epsilon$ holds for some $k = k_\ell$ before Δ^{k_ℓ} becomes a point.

Lemma 3.2 *Suppose that $\{\Delta^{k_\ell} \mid \ell = 1, 2, \ldots\}$ is an infinite sequence of nested simplices generated by bisection. Then we have*

$$\lim_{\ell \to \infty} \left(f(\widetilde{\mathbf{x}}^{k_\ell}) - z^{k_\ell} \right) = 0. \tag{3.6}$$

Proof. For the sequence $\{\Delta^{k_\ell} \mid \ell = 1, 2, \dots\}$ we can assume that

$$f(\mathbf{v}^{k_\ell}) \le z^{k_\ell} < f(\widetilde{\mathbf{x}}^{k_\ell}), \tag{3.7}$$

where $\mathbf{v}^{k_\ell}$ is a vertex of Δ^{k_ℓ} minimizing f. Let $\{\mathbf{v}\} = \cap_{\ell=1}^{\infty} \Delta^{k_\ell}$. Then we see from the definition of $\Gamma(\Delta^{k_\ell})$ that $\cap_{\ell=1}^{\infty} \Gamma(\Delta^{k_\ell}) = \{\mathbf{v}\}$. Since both $\mathbf{v}^{k_\ell}$ and $\widetilde{\mathbf{x}}^{k_\ell}$ belong to $\Gamma(\Delta^{k_\ell})$, we have $\mathbf{v}^{k_\ell} \to \mathbf{v}$ and $\widetilde{\mathbf{x}}^{k_\ell} \to \mathbf{v}$ as $\ell \to \infty$. Moreover, by the continuity of f we have

$$\lim_{\ell \to \infty} f(\mathbf{v}^{k_\ell}) = \lim_{\ell \to \infty} f(\widetilde{\mathbf{x}}^{k_\ell}) = f(\mathbf{v}),$$

which, together with (3.7), implies (3.6). □

Some Issues to Resolve in Relaxation $\widetilde{\mathrm{P}}(\Delta)$

Lemma 3.2 guarantees that the simplicial branch-and-bound still works if we use the relaxation $\widetilde{\mathrm{P}}(\Delta)$ instead of the usual $\overline{\mathrm{P}}(\Delta)$ in the bounding operation of Step 2. Before applying it in practice, however, we need to resolve two issues.

First, $\widetilde{\mathrm{P}}(\Delta)$ requires one to compute the inverse of the $(r+1) \times (r+1)$-matrix $\mathbf{U}$ every iteration to determine the objective function $\mathbf{c}^{\mathsf{T}}\mathbf{x}$ of (3.2). This is the main reason why the representation $\mathbf{c}^{\mathsf{T}}\mathbf{x} + c_{r+1}$ of g has been avoided in the past. However, this is not a big challenge if we adopt the depth first rule at Step 1. Because most $\mathbf{U}$'s are different from their predecessors in only one column, we can update the inverse of the kth matrix $\mathbf{U}^k$ from $(\mathbf{U}^{k-1})^{-1}$ almost always in time $O(r)$ using the *rank-one update* [2]. Suppose that $\mathbf{U}^{k-1}$ and $\mathbf{U}^k$ are the same except for the pth column. Let $\mathbf{u}_p^k = [(\mathbf{v}_p^k)^{\mathsf{T}}, 1]^{\mathsf{T}}$ denote the pth column of $\mathbf{U}^k$ and let

$$\mathbf{w} = (\mathbf{U}^{k-1})^{-1}\mathbf{u}_p^k, \quad \mathbf{E} = \mathbf{I} + (\mathbf{e}_p - \mathbf{w})\mathbf{e}_p^{\mathsf{T}}/w_p,$$

where $\mathbf{I}$ denotes the identity matrix and w_p is the pth component of $\mathbf{w}$. Then we have

$$(\mathbf{U}^k)^{-1} = \mathbf{E}(\mathbf{U}^{k-1})^{-1}. \tag{3.8}$$

Note that $\mathbf{E}$ is an eta matrix with nonzero off-diagonal elements in pth column. Since $\mathbf{v}_p^k$ is a convex combination $(1-\alpha)\mathbf{v}_p^{k-1} + \alpha\mathbf{v}_q^{k-1}$ of two vertices of $\Delta^{j_{k-1}}$, we have $w_p = 1 - \alpha$, $w_q = \alpha$ and the other components of $\mathbf{w}$ equal to zeros. As a result, the pth column of $\mathbf{E}$ has only two nonzero entries $1/(1-\alpha)$ and $-\alpha/(1-\alpha)$ in the p and qth rows, respectively. We see from (3.8) that the inverse of $\mathbf{U}^k$ is yielded if we replace only the p and qth rows of $(\mathbf{U}^{k-1})^{-1}$ by their affine combinations.

The second issue is much more serious. As shown in Lemma 3.1, the lower bound $\widetilde{z}$ yielded by $\widetilde{\mathrm{P}}(\Delta)$ is inferior to $\overline{z}$; and the difference is not expected to be so small. Although it is somewhat tightened to z^k by $f(\mathbf{v}_i)$'s in (3.5), the essential reason for introducing their minimum is merely to guarantee convergence of the algorithm by Lemma 3.2. Therefore, the branching tree when using $\widetilde{\mathrm{P}}(\Delta)$ might grow more rapidly than when using $\overline{\mathrm{P}}(\Delta)$. To prevent the rapid growth of the branching tree, we have to introduce a full-scale procedure for tightening $\widetilde{z}$.

4 Algorithm Using Two-Phase Bounding Operation

For $\widetilde{z}$ yielded by $\widetilde{\mathrm{P}}(\Delta)$, let $G = \{\mathbf{x} \in \mathbb{R}^r \mid g(\mathbf{x}) \ge \widetilde{z}\}$. Since $X \cap \Delta$ is a subset of this half space G, no feasible solution to $\mathrm{P}(\Delta)$ is lost if we add $\mathbf{x} \in G$ to $\mathrm{P}(\Delta)$ as a constraint. The

resulting problem is then equivalent to

$$\left|\begin{array}{lll} \text{minimize} & z = & f(\mathbf{x}) \\ \text{subject to} & & \mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y} \leq \mathbf{b}, \quad (\mathbf{x},\mathbf{y}) \geq \mathbf{0} \\ & & \mathbf{x} \in \Delta, \quad \mathbf{c}^{\mathsf{T}}\mathbf{x} \geq \widetilde{z} - c_{r+1}. \end{array}\right. \tag{4.1}$$

In the preceding section, we have relaxed the objective function f and the constraint $\mathbf{x} \in \Delta$. Instead, we try relaxing $\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y} \leq \mathbf{b}$ here, by introducing a Lagrangian multiplier $\boldsymbol{\lambda} \in \mathbb{R}^m$. Then we have

$$\mathrm{L}(\Delta;\boldsymbol{\lambda}) \left|\begin{array}{lll} \text{minimize} & z = & f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathsf{T}}(\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}-\mathbf{b}) \\ \text{subject to} & & \mathbf{x} \in \Delta, \quad \mathbf{y} \geq \mathbf{0}, \quad \mathbf{c}^{\mathsf{T}}\mathbf{x} \geq \widetilde{z} - c_{r+1}, \end{array}\right.$$

by noting $\mathbf{x} \geq \mathbf{0}$ for any $\mathbf{x} \in \Delta$. If $\mathbf{c}^{\mathsf{T}}\mathbf{v}_i < \widetilde{z} - c_{r+1}$, or $f(\mathbf{v}_i) < \widetilde{z}$ equivalently, for each vertex $\mathbf{v}_i$ of Δ, then $\mathrm{L}(\Delta;\boldsymbol{\lambda})$ is infeasible. In that case, the hyperplane $\partial G = \{\mathbf{x} \in \mathbb{R}^r \mid g(\mathbf{x}) = \widetilde{z}\}$ separates Δ and X; and we can discard Δ because $\mathrm{P}(\Delta)$ is infeasible.

Suppose that $\mathrm{L}(\Delta;\boldsymbol{\lambda})$ has an optimal solution $(\mathbf{x}(\boldsymbol{\lambda}), \mathbf{y}(\boldsymbol{\lambda}))$ and denote the value $f(\mathbf{x}(\boldsymbol{\lambda}))+ \boldsymbol{\lambda}^{\mathsf{T}}(\mathbf{A}\mathbf{x}(\boldsymbol{\lambda}) + \mathbf{B}\mathbf{y}(\boldsymbol{\lambda}) - \mathbf{b})$ by $z(\boldsymbol{\lambda})$. As is well-known (see e.g. [12]), we have

$$z(\boldsymbol{\lambda}) \leq z(\Delta), \quad \forall \boldsymbol{\lambda} \geq \mathbf{0}.$$

However, to use $\mathrm{L}(\Delta;\boldsymbol{\lambda})$ as a procedure for tightening $\widetilde{z}$, we need to fix the value of $\boldsymbol{\lambda}$ appropriately so that $z(\boldsymbol{\lambda}) > \widetilde{z}$ holds.

Lagrangian Relaxation $\mathrm{L}(\Delta;\boldsymbol{\lambda})$ and its Solution

Since the structure of $\mathrm{L}(\Delta;\boldsymbol{\lambda})$ is similar to $\mathrm{P}(\Delta)$, we can relax it into a linear program as in the same way as we have obtained $\widetilde{\mathrm{P}}(\Delta)$. Let us replace f and Δ by g and $\Gamma(\Delta)$, respectively, in $\mathrm{L}(\Delta;\boldsymbol{\lambda})$, and further drop the constraint $\mathbf{c}^{\mathsf{T}}\mathbf{x} \geq \widetilde{z} - c_{r+1}$. Then we have

$$\phi(\boldsymbol{\lambda}) = \min\{(\mathbf{c}^{\mathsf{T}} + \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{A})\mathbf{x} + \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{B}\mathbf{y} - \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{b} \mid \mathbf{s} \leq \mathbf{x} \leq \mathbf{t},\ \mathbf{y} \geq \mathbf{0}\},$$

where $\mathbf{s}$ and $\mathbf{t}$ are defined in (3.1). The right-hand side can also be thought of as a Lagrangian relaxation of (3.2), i.e., problem $\widetilde{\mathrm{P}}(\Delta)$. As long as $\boldsymbol{\lambda}$ satisfies $\boldsymbol{\lambda}^{\mathsf{T}}\mathbf{B} \geq \mathbf{0}$, the value $\phi(\boldsymbol{\lambda})$ is finite and coincides with

$$\psi(\boldsymbol{\lambda}) = \max\{\mathbf{s}^{\mathsf{T}}\boldsymbol{\mu} - \mathbf{t}^{\mathsf{T}}\boldsymbol{\nu} - \mathbf{b}^{\mathsf{T}}\boldsymbol{\lambda} \mid \boldsymbol{\mu} - \boldsymbol{\nu} = \mathbf{c} + \mathbf{A}^{\mathsf{T}}\boldsymbol{\lambda},\ (\boldsymbol{\mu},\boldsymbol{\nu}) \geq \mathbf{0}\}$$

by the duality theorem of linear programming. Therefore, we have

$$\max\{\phi(\boldsymbol{\lambda}) \mid \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{B} \geq \mathbf{0}, \boldsymbol{\lambda} \geq \mathbf{0}\} = \max\{\psi(\boldsymbol{\lambda}) \mid \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{B} \geq \mathbf{0}, \boldsymbol{\lambda} \geq \mathbf{0}\}.$$

Note that the right-hand side of this equation can be rewritten as

$$\left|\begin{array}{lll} \text{maximize} & z = & -\mathbf{b}^{\mathsf{T}}\boldsymbol{\lambda} + \mathbf{s}^{\mathsf{T}}\boldsymbol{\mu} - \mathbf{t}^{\mathsf{T}}\boldsymbol{\nu} \\ \text{subject to} & & \mathbf{A}^{\mathsf{T}}\boldsymbol{\lambda} - \boldsymbol{\mu} + \boldsymbol{\nu} = -\mathbf{c} \\ & & \mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda} \geq \mathbf{0}, \quad (\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu}) \geq \mathbf{0}, \end{array}\right. \tag{4.2}$$

which is the dual problem of (3.2). Let $(\widetilde{\boldsymbol{\lambda}}, \widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\nu}})$ be an optimal solution to (4.2).

Lemma 4.1 *There is a relationship:*

$$\phi(\boldsymbol{\lambda}) + c_{r+1} \leq \widetilde{z}, \quad \forall \boldsymbol{\lambda} \geq \mathbf{0}.$$

where the equality holds if $\boldsymbol{\lambda} = \widetilde{\boldsymbol{\lambda}}$.

The dual optimal solution $(\widetilde{\boldsymbol{\lambda}}, \widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\nu}})$ is generated as a byproduct in solving the primal problem $\widetilde{\mathrm{P}}(\Delta)$. We adopt $\widetilde{\boldsymbol{\lambda}}$ as the Lagrangian multiplier of $\mathrm{L}(\Delta; \boldsymbol{\lambda})$. As is easily seen, $\mathrm{L}(\Delta; \boldsymbol{\lambda})$ can be decomposed into

$$\begin{array}{l|lll} \mathrm{L}_x(\Delta; \boldsymbol{\lambda}) & \text{minimize} & z_x = & f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathsf{T}}(\mathbf{A}\mathbf{x} - \mathbf{b}) \\ & \text{subject to} & & \mathbf{x} \in \Delta \cap G, \\ \mathrm{L}_y(\Delta; \boldsymbol{\lambda}) & \text{minimize} & z_y = & \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{B}\mathbf{y} \\ & \text{subject to} & & \mathbf{y} \geq \mathbf{0}. \end{array}$$

When $\boldsymbol{\lambda} = \widetilde{\boldsymbol{\lambda}}$, the latter problem has an obvious optimal solution $\mathbf{y}(\widetilde{\boldsymbol{\lambda}}) = \mathbf{0}$ because $(\widetilde{\boldsymbol{\lambda}}, \widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\nu}})$ is an optimal solution to (4.2) and $\mathbf{B}^{\mathsf{T}}\widetilde{\boldsymbol{\lambda}} \geq \mathbf{0}$ holds. Thus, for an optimal solution $\mathbf{x}(\widetilde{\boldsymbol{\lambda}})$ to $\mathrm{L}_x(\Delta; \widetilde{\boldsymbol{\lambda}})$ the optimal value of $\mathrm{L}(\Delta; \widetilde{\boldsymbol{\lambda}})$ is given by

$$z(\widetilde{\boldsymbol{\lambda}}) = f(\mathbf{x}(\widetilde{\boldsymbol{\lambda}})) + \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}(\mathbf{A}\mathbf{x}(\widetilde{\boldsymbol{\lambda}}) - \mathbf{b}).$$

Theorem 4.2 *Among $z(\widetilde{\boldsymbol{\lambda}})$, $\widetilde{z}$ and the value $z(\Delta)$ of $P(\Delta)$, there is a relationship:*

$$\widetilde{z} \leq z(\widetilde{\boldsymbol{\lambda}}) \leq z(\Delta), \tag{4.3}$$

where the first inequality holds strictly if $\mathbf{x}(\widetilde{\boldsymbol{\lambda}}) \notin \{\mathbf{v}_1, \ldots, \mathbf{v}_{r+1}\}$ and f is strictly concave on Δ.

Proof. Since $(\widetilde{\boldsymbol{\lambda}}, \widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\nu}})$ is an optimal solution to (4.2) and $\mathbf{s} \leq \mathbf{v}_i \leq \mathbf{t}$ for each $\mathbf{v}_i$, we have

$$\begin{aligned} f(\mathbf{v}_i) + \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}(\mathbf{A}\mathbf{v}_i - \mathbf{b}) &= \mathbf{c}^{\mathsf{T}}\mathbf{v}_i + c_{r+1} + \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}(\mathbf{A}\mathbf{v}_i - \mathbf{b}) \\ &= \widetilde{\boldsymbol{\mu}}^{\mathsf{T}}\mathbf{v}_i - \widetilde{\boldsymbol{\nu}}^{\mathsf{T}}\mathbf{v}_i - \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}\mathbf{b} + c_{r+1} \\ &\geq \widetilde{\boldsymbol{\mu}}^{\mathsf{T}}\mathbf{s} - \widetilde{\boldsymbol{\nu}}^{\mathsf{T}}\mathbf{t} - \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}\mathbf{b} + c_{r+1} \\ &= \psi(\widetilde{\boldsymbol{\lambda}}) + c_{r+1} = \phi(\widetilde{\boldsymbol{\lambda}}) + c_{r+1}. \end{aligned}$$

By the concavity of f and Lemma 4.1, the point $\mathbf{x}(\widetilde{\boldsymbol{\lambda}})$ in Δ must satisfy

$$z(\widetilde{\boldsymbol{\lambda}}) = f(\mathbf{x}(\widetilde{\boldsymbol{\lambda}})) + \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}(\mathbf{A}\mathbf{x}(\widetilde{\boldsymbol{\lambda}}) - \mathbf{b}) \geq \phi(\widetilde{\boldsymbol{\lambda}}) + c_{r+1} = \widetilde{z}.$$

Suppose $\mathbf{x}(\widetilde{\boldsymbol{\lambda}}) \notin \{\mathbf{v}_1, \ldots, \mathbf{v}_{r+1}\}$. Then $\mathbf{x}(\widetilde{\boldsymbol{\lambda}})$ lies among vertices of $\Delta \cap \partial G$; and we have $\mathbf{x}(\widetilde{\boldsymbol{\lambda}}) = (1-\beta)\mathbf{v}_p + \beta\mathbf{v}_q$ for some $\mathbf{v}_p$, $\mathbf{v}_q$ and $\beta \in (0, 1)$. Therefore, the following holds:

$$z(\widetilde{\boldsymbol{\lambda}}) > (1-\beta)[f(\mathbf{v}_p) + \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}(\mathbf{A}\mathbf{v}_p - \mathbf{b})] + \beta[f(\mathbf{v}_q) + \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}(\mathbf{A}\mathbf{v}_q - \mathbf{b})] \geq \widetilde{z},$$

if f is strictly concave on Δ. □

Since $\widetilde{z}$ might coincide with $\overline{z}$, e.g., when $\widetilde{\mathbf{x}} \in \Delta$, the bound $z(\widetilde{\boldsymbol{\lambda}})$ can be superior even to $\overline{z}$. Although $\mathrm{L}_x(\Delta; \widetilde{\boldsymbol{\lambda}})$ yielding $z(\widetilde{\boldsymbol{\lambda}})$ is a concave minimization problem, we can solve it in polynomial time if the value of f is given by oracle. Since the objective function is concave, $\mathbf{x}(\widetilde{\boldsymbol{\lambda}})$ is a vertex of $\Delta \cap G$. The number of its vertices is, however, $O(r^2)$ at most. We need only to check the objective function value at the intersection of ∂G with each edge $\mathbf{v}_i$–$\mathbf{v}_j$ of Δ such that $\mathbf{v}_i \in \mathrm{int}(G)$ and $\mathbf{v}_j \notin G$, as well as at each $\mathbf{v}_i \in G$

Description of the Modified Algorithm

Now, recall the three basic steps of the simplicial branch-and-bound algorithm given in Section 2. The bounding operation of Step 2 we propose consists of two phases:

Step 2.1. Solve $\widetilde{\mathrm{P}}(\Delta)$ and compute z^k defined in (3.5). If $f(\mathbf{x}^*) - z^k \leq \epsilon$ for the incumbent $\mathbf{x}^*$, discard Δ from further consideration.

Step 2.2. If $f(\mathbf{x}^*) - z^k > \epsilon$, solve $\mathrm{L}_x(\Delta; \widetilde{\boldsymbol{\lambda}})$ and compute $z(\widetilde{\boldsymbol{\lambda}})$. If $f(\mathbf{x}^*) - z(\widetilde{\boldsymbol{\lambda}}) \leq \epsilon$, then discard Δ from further consideration.

The following is the detailed description of our simplicial branch-and-bound algorithm for solving problem P:

```
algorithm 2PHASE_BB
begin
  compute v := max{∑_{j=1}^r x_j | x ∈ X} and let Δ^1 := conv({v e_1, ..., v e_r, 0});
  L := {1}; z* := +∞; k := 1;
  while L ≠ ∅ do begin
    select j_k ∈ L and let L := L \ {j_k}; Δ := Δ^{j_k};                    /* Step 1 */
    let v_1, ..., v_{r+1} denote the vertices of Δ and let V := [v_1, ..., v_{r+1}];
    U := [V^T, e]^T; [c^T, c_{r+1}] := [f(v_1), ..., f(v_{r+1})]U^{-1};
    solve P~(Δ) of minimizing g(x) = c^T x + c_{r+1};                      /* Step 2.1 */
    if P~(Δ) is feasible then begin
      let x~^k be an optimal solution to P~(Δ) and z~ := g(x~^k);
      if f(x~^k) < z* then update z* := f(x~^k) and x* := x~^k;
      z^k := max{z~, min{f(v_1), ..., f(v_{r+1})}};
      if f(x*) - z^k > ε then begin                                        /* Step 2.2 */
        define L_x(Δ; λ~) for a dual optimal solution (λ~, μ~, ν~) to P~(Δ);
        compute an optimal solution x(λ~) to L_x(Δ; λ~) and the value z(λ~);
        if f(x*) - z(λ~) > ε then begin                                    /* Step 3 */
          select the longest edge v_p–v_q of Δ and let v := (1 - α)v_p + αv_q for a fixed
          α ∈ (0, 1/2];
          Δ^{2k} := conv({v_i | i ≠ p} ∪ {v}); Δ^{2k+1} := conv({v_i | i ≠ q} ∪ {v});
          L := L ∪ {2k, 2k + 1}
        end
      end
    end;
    k := k + 1
  end
end;
```

Theorem 4.3 *When $\epsilon = 0$, the sequence $\{\widetilde{\mathbf{x}}^k \mid k = 1, 2, \dots\}$ generated by algorithm 2PHASE_BB with the best-bound rule has accumulation points, each of which is a globally optimal solution to problem P.*

Proof. When the algorithm terminates in finite time, the assertion is obvious. Suppose that it does not terminate and generates an infinite sequence of nested simplices $\{\Delta^{k_\ell} \mid \ell = 1, 2, \dots\}$. Since the best-bound rule is adopted, we have

$$z^{k_\ell} \leq z^j \leq z(\Delta^j), \quad \forall j \in \mathcal{L},$$

at the k_ℓth iteration. Recall that $z(\Delta^j)$ is the optimal value of subproblem $\mathrm{P}(\Delta^j)$ and $\min\{z(\Delta^j) \mid j \in \mathcal{L}\}$ is equal to the value $z(\Delta^1)$ of the target P. Therefore, we have

$$z^{k_\ell} \leq z(\Delta^1) \leq f(\widetilde{\mathbf{x}}^{k_\ell}), \quad \ell = 1, 2, \ldots.$$

However, by Lemma 3.2, we have $f(\widetilde{\mathbf{x}}^{k_\ell}) - z^{k_\ell} \to 0$ as $\ell \to \infty$. This implies that $f(\widetilde{\mathbf{x}}^{k_\ell}) \to z(\Delta^1)$ as $\ell \to \infty$. □

Corollary 4.4 *When $\epsilon > 0$, algorithm 2PHASE_BB with either of the depth-first and best-bound rules terminates after a finite number of iterations and yields $\mathbf{x}^*$ as a globally ϵ-optimal solution to problem P.*

Proof. If the algorithm does not terminates, it generates an infinite sequence of nested simplices $\{\Delta^{k_\ell} \mid \ell = 1, 2, \ldots\}$ such that

$$f(\mathbf{x}^{k_\ell}) - z^{k_\ell} \geq f(\mathbf{x}^*) - z^{k_\ell} > \epsilon > 0, \quad \ell = 1, 2, \ldots.$$

However, $f(\widetilde{\mathbf{x}}^{k_\ell}) - z^{k_\ell} \to 0$ as $\ell \to \infty$, which is a contradiction. □

5 Numerical Experiment

Let us report numerical results of having compared computer codes of 2PHASE_BB and the standard simplicial branch-and-bound algorithm using only the relaxation $\overline{\mathrm{P}}(\Delta)$. We refer to them here, as 2phase and standard, respectively. The test problem we solved is a concave quadratic minimization problem of the form:

$$\left| \begin{array}{ll} \text{minimize} & -(1/2)\mathbf{x}^\mathsf{T}\mathrm{Q}^\mathsf{T}\mathrm{Q}\mathbf{x} - \omega\mathbf{d}^\mathsf{T}\mathbf{y} \\ \text{subject to} & \mathbf{A}'\mathbf{x} + \mathbf{B}'\mathbf{y} \leq \mathbf{b}', \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array} \right. \tag{5.1}$$

where $\mathrm{Q} \in \mathbb{R}^{r' \times r'}$, $\mathbf{A}' \in \mathbb{R}^{m' \times r'}$, $\mathbf{B}' \in \mathbb{R}^{m' \times (n'-r')}$, $\mathbf{b}' \in \mathbb{R}^{m'}$, $\mathbf{d} \in \mathbb{R}^{n'-r'}$ and ω is a positive weight. The matrix $\mathrm{Q} = [q_{ij}]$ was generated so as to have two nonzero entries in each row, i.e., $(q_{ii}, q_{i,i+1})$ for $i = 1, \ldots, r'-1$, and $(q_{r'1}, q_{r'r'})$, where $q_{ii} = q_{r'r'} = 1.0$ and the rest were drawn randomly from the uniform distribution on $[0.0, 1.0]$. Then $\mathrm{Q}^\mathsf{T}\mathrm{Q}$ has three nonzero entries at most in each row. Also, each component of $\mathbf{d}$ was a uniformly random number in $[0.0, 1.0]$. To make the feasible set bounded, $\mathbf{b}'$ was an all-ones vector and each component in the last row of $[\mathbf{A}', \mathbf{B}']$ was fixed at $1.0/n'$. Other components were all random numbers in $[-0.5, 1.0]$, where the percentages of zeros and negative numbers were about 20% and 10%, respectively. Selecting various sets of parameters (m', n', r', ω), we solved ten instances of (5.1) for each set using 2phase and standard on a Linux workstation (Linux 2.4.21, Itanium2 processor 1.3GHz).

Computer Codes

Both codes 2phase and standard were written using GNU Octave (version 2.1.50) [13], a Matlab-like computational tool, in accordance with the depth-first rule. The tolerance ϵ was fixed at 10^{-4}. To adjust the form of (5.1) to (2.1), we introduced an additional variable ζ and applied the code 2phase to

$$\left| \begin{array}{lll} \text{minimize} & -(1/2)\mathbf{x}^\mathsf{T}\mathrm{Q}^\mathsf{T}\mathrm{Q}\mathbf{x} - \omega\zeta & \\ \text{subject to} & \mathbf{A}'\mathbf{x} + \mathbf{B}'\mathbf{y} \leq \mathbf{b}', & (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \\ & \zeta - \mathbf{d}^\mathsf{T}\mathbf{y} \leq 0, & \zeta \geq 0. \end{array} \right. \tag{5.2}$$

where we should note $\zeta \geq 0$ because $\mathbf{d} \geq \mathbf{0}$. The size (m, n, r) of (5.2) is therefore equal to $(m'+1, n'+1, r'+1)$. As for standard, we applied it directly to (5.1) because it uses only the relaxed problem $\overline{\mathrm{P}}(\Delta)$, which can be written [5] as

$$\left|\begin{array}{ll} \text{minimize} & (\mathbf{f}')^{\mathsf{T}}\boldsymbol{\xi} - \omega \mathbf{d}^{\mathsf{T}}\mathbf{y} \\ \text{subject to} & \mathbf{A}'\mathbf{V}'\boldsymbol{\xi} + \mathbf{B}'\mathbf{y} \leq \mathbf{b}' \\ & \mathbf{e}^{\mathsf{T}}\boldsymbol{\xi} = 1, \quad (\boldsymbol{\xi}, \mathbf{y}) \geq \mathbf{0}, \end{array}\right.$$

where $\mathbf{V}' = [\mathbf{v}_1, \ldots, \mathbf{v}_{r'+1}]$ and $\mathbf{f}' = [f(\mathbf{v}_1), \ldots, f(\mathbf{v}_{r'+1})]^{\mathsf{T}}$ for $r'+1$ vertices $\mathbf{v}_j$'s of $\Delta \subset \mathbb{R}^{r'}$. As the subdivision rule of Δ, bisection of ratio $\alpha = 1/2$ was adopted in 2phase, but not in standard, because we found in our preliminary experiment that the convergence of standard with the bisection rule is too slow to be compared with 2phase. Instead, we took the way to bisect the longest edge of the minimal face of Δ which contains an optimal $\overline{\mathbf{x}} = \mathbf{V}'\overline{\boldsymbol{\xi}}$ of $\overline{\mathrm{P}}(\Delta)$. Although this subdivision rule does not guarantee the convergence, standard incorporating it terminated for every tested instance of (5.1) and generated the same output as 2phase with the usual bisection rule.

Numerical Results

In Figures 5.1–5.4, line plots are given for comparing the behavior of 2phase with that of standard when the size of constraint matrix $[\mathbf{A}', \mathbf{B}']$ was fixed at $(m', n') = (40, 80)$. The solid and broken lines represent the results of 2phase and standard, respectively.

Figure 5.1 shows the variation in the average number of branching operations required by each code when ω was fixed at 5.0 and r' was increased from 16 to 32. We see that the dominance between 2phase and standard is reversed around $r' = 25$, and can confirm that the second phase of the bounding operation using the Lagrangian relaxation $\mathrm{L}_x(\Delta; \widetilde{\boldsymbol{\lambda}})$ works properly. The variations in the average CPU seconds are plotted in Figure 5.2. The code 2phase surpasses standard in computational time at every r', which we can understand the problem (3.2) associated with $\widetilde{\mathrm{P}}(\Delta)$ is easy enough to cancel out the inferiority of 2phase in the number of branching operations for $r' < 25$. In our preliminary experiments, we removed the second-phase procedure from 2phase and tried to solve the same set of instances using the resulting code, named 1phase. It performed well when $r' < 25$, just as 2phase did, but failed to terminate in 10^5 branching operations, on one instance with each $r' = 26, 28$, four instances with $r' = 30$ and three instances with $r' = 32$. This implies that the second-phase bears a crucial role in 2PHASE_BB.

Figures 5.3 and 5.4 show the variations in the average number of branching operations and CPU seconds, respectively, required by each code when r' was fixed at 20 and ω was changed in $\{3.0, 3.5, 4.0, 5.0, 7.0, 10.0, 20.0\}$. Unfortunately, both codes are very sensitive to changes in ω, especially when $\omega < 5$. Nevertheless, 2phase needs considerably less branching operations than standard when $\omega \leq 4$, which is totally due to the tight lower bound $z(\widetilde{\boldsymbol{\lambda}})$ computed in the second phase of the bounding operation. This, together with the ease of solution to (3.2), yields the significant advantage of 2phase against standard in computational time when $\omega < 10$. Incidentally, 1phase failed to terminate in 10^5 branching operations, on seven instances with $\omega = 3.0$ and three instances with $\omega = 3.5$.

From the above observation, we can expect that 2PHASE_BB has potential for solving much larger scale problems than the standard algorithm can, unless the concavity part has a lot of weight in the objective function. We therefore tested the code 2phase on (5.1) of size (m', n') from $(60, 120)$ to $(300, 200)$ with ω fixed at 5.0. The number of nonlinear variables r' was set from 20% to 50% of the whole variables, i.e., the maximum size of (m', n', r') was $(300, 200, 100)$. The computational results are listed in Table 5.1, in which # and

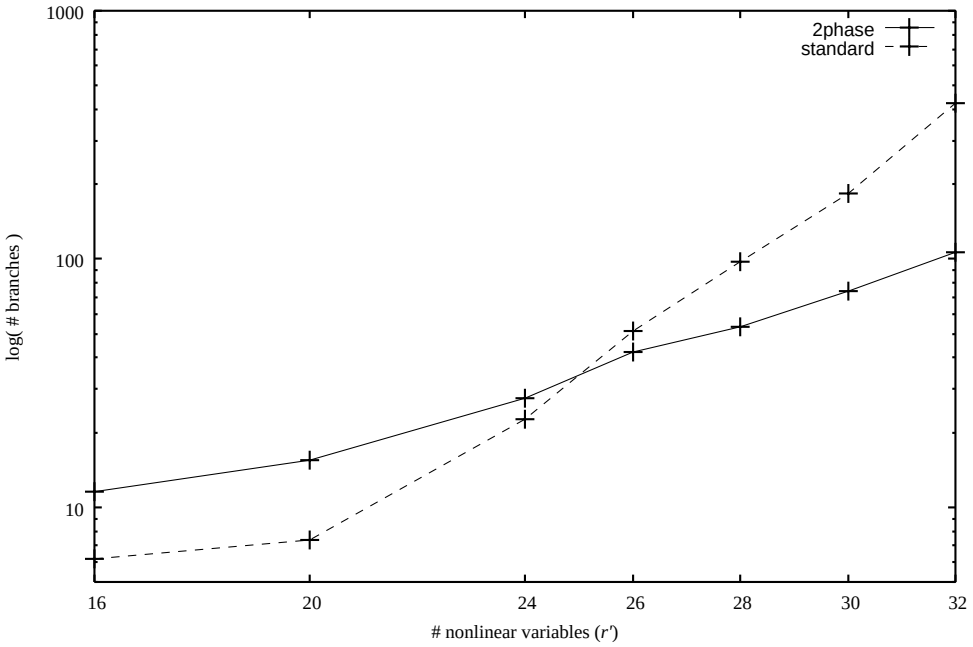

Figure 5.1: Numbers of branching operations when $(m', n', \omega) = (40, 80, 5.0)$.

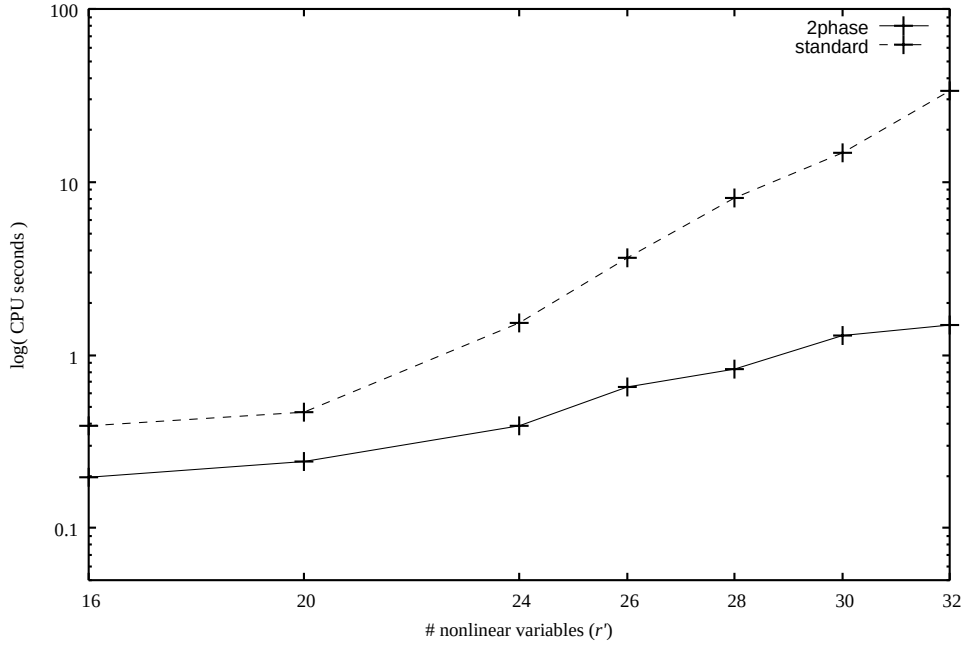

Figure 5.2: CPU seconds when $(m', n', \omega) = (40, 80, 5.0)$.

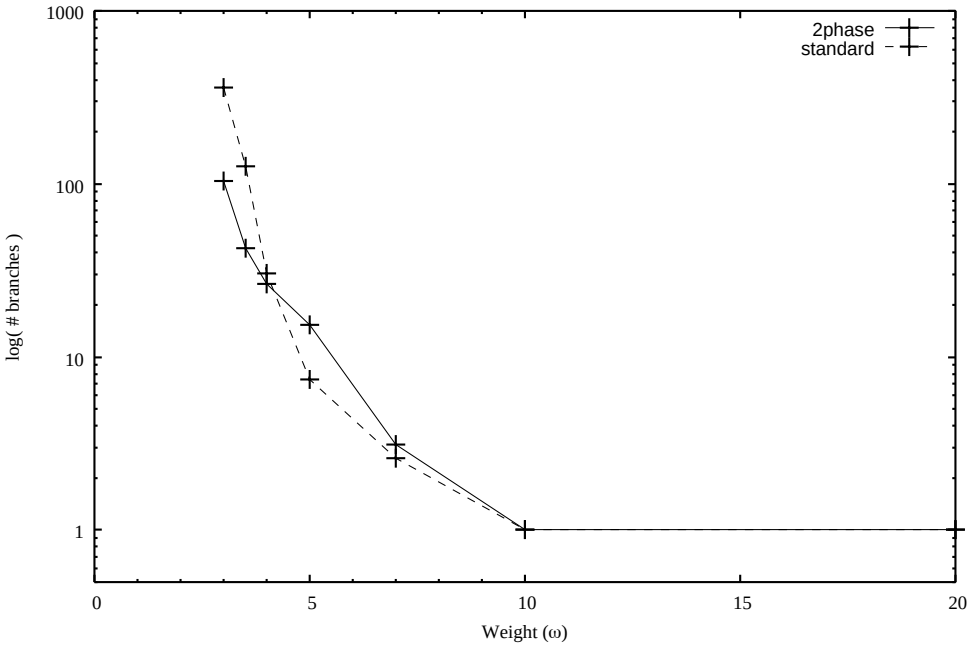

Figure 5.3: Numbers of branching operations when $(m', n', r') = (40, 80, 20)$.

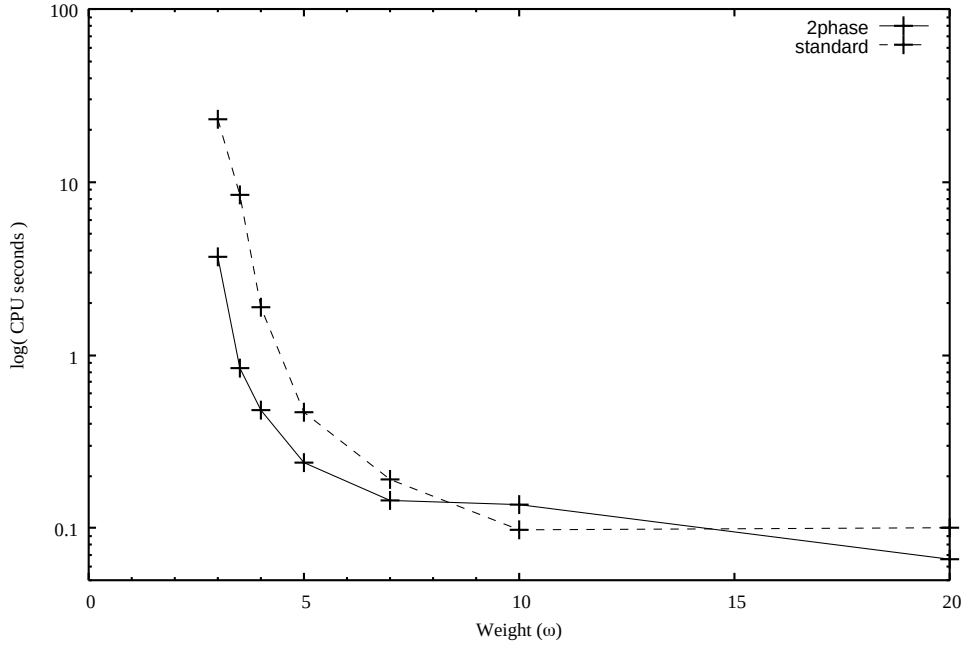

Figure 5.4: CPU seconds when $(m', n', r') = (40, 80, 20)$.

Table 5.1: Computational results of 2phase when $\omega = 5.0$.

	$r' = 0.2n'$		$r' = 0.3n'$		$r' = 0.4n'$		$r' = 0.5n'$	
$m' \times n'$	#	*time*	#	*time*	#	*time*	#	*time*
60×120	23.2	0.646	41.0	1.309	91.0	4.030	141.9	10.56
180×120	17.0	2.646	54.4	5.311	49.2	6.554	141.3	18.34
80×160	15.8	1.156	55.0	3.376	134.9	12.06	238.3	42.29
240×160	8.0	7.854	77.2	20.08	117.4	33.19	229.5	80.40
100×200	22.0	2.526	54.8	6.117	129.0	23.97	256.1	89.83
300×200	26.6	21.83	66.6	41.55	135.4	81.56	200.2	170.8

time indicate the average number of branching operations and CPU seconds, respectively, required by 2phase for each (m', n', r'). We see from this table that the number of branching operations increases rather mildly as m' and n' increase, in contrast to the case of r'. The similar tendency can be observed in the CPU seconds. We could solve still larger scale problems by elaborating the computer code of algorithm 2PHASE_BB, as long as the number r' of nonlinear variables is about 30% of the whole.

6 Conclusion and Future Issues

We have developed a simplicial branch-and-bound algorithm for solving a low-rank concave minimization problem (2.1). The major feature of this problem is that the variables involved in the objective function are only a part of the whole. In the bounding operation of the algorithm, we have proposed to enlarge the feasible set of each linear programming relaxed problem, in order to facilitate application of specialized algorithms and sensitivity analysis of the simplex method. Furthermore, to tighten the lower bound deteriorated by this enlargement of the feasible set, we have proposed the second bounding operation based on a Lagrangian relaxation. We have seen in the preceding section that both operations work very well and the algorithm has potential for solving much larger scale problems than the existing algorithm can solve.

To further expand the versatility of the algorithm, we need to resolve two issues in the future. Low-rank concave minimization problems can certainly be transformed into the form of (2.1). However, many of such transformations destroy the structure of the constraint, like the ones from (1.1) to (1.2) and from (5.1) to (5.2), and can take away from the devices in the first phase of our bounding operation. Another issue is on the subdivision rule. Even though bisection works reasonably well in our algorithm compared with in the standard algorithm, its performance is still far from satisfactory. In the meanwhile, we need to try out a variety of subdivision rules and hybrids of them to accelerate the convergence. For these issues, we will report the details elsewhere.

References

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice Hall, N.J, 1993.

[2] V. Chvátal, *Linear Programming*, Freeman, N.Y., 1983.

[3] J.E. Falk and R.M. Soland, An algorithm for separable nonconvex programming problems, *Management Sci.* 15 (1969) 550–569.

[4] R. Horst, An algorithm for nonconvex programming problems, *Math. Program.* 10 (1976) 312–321.

[5] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*, 2nd ed., Springer-Verlag, Berlin, 1993.

[6] H. Konno and T. Kuno, Linear multiplicative programming, *Math. Program.* 56 (1992) 51–64.

[7] H. Konno, P.T. Thach and H. Tuy, *Optimization on Low Rank Nonconvex Structures*, Kluwer Academic Publishers, Dordrecht, 1997.

[8] T. Kuno, A finite branch-and-bound algorithm for linear multiplicative programming, *Comput. Optim. Appl.* 20 (2001) 119–135.

[9] T. Kuno and T. Utsunomiya, A Lagrangian based branch-and-bound algorithm for production-transportation problems, *J. Global Optim.* 18 (2000) 59–73.

[10] T. Kuno, Y. Yajima and H. Konno, An outer approximation method for minimizing the product of several convex functions on a convex set, *J. Global Optim.* 3 (1993) 325–335.

[11] H. Nagai and T. Kuno, A simplicial branch-and-bound algorithm for production-transportation problems with inseparable concave production cost, Technical Report ISE-TR-04-197, University of Tsukuba (Ibaraki, 2004), *J. Oper. Res. Soc. Japan* to appear.

[12] G.L Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, John Willey and Sons, N.Y., 1988.

[13] Octave Home Page, http://www.octave.org/.

[14] H.S. Ryoo and N.V. Sahinidis, Global optimization of multiplicative programs, *J. Global Optim.* 26 (2003) 387–418.

[15] H. Tuy., *Convex Analysis and Global Optimization*, Kluwer Academic Publishers, Dordrecht, 1998.

[16] H. Tuy, N.D. Dan and S. Ghannadan, Strongly polynomial time algorithms for certain concave minimization problems on networks, *Oper. Res. Lett.* 14 (1993) 99–109.

[17] H. Tuy, S. Ghannadan, A. Migdalas and P. Värbrand, Strongly polynomial algorithm for a concave production-transportation problem with a fixed number of nonlinear variables, *Math. Program.* 72 (1996) 229–258.

Manuscript received 26 November 2004
revised 28 January 2005
accepted for publication 3 March 2005

Takahito Kuno
Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan
E-mail address: takahito@cs.tsukuba.ac.jp

Hidetoshi Nagai
Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan
E-mail address: nagai@syou.cs.tsukuba.ac.jp