A Simplicial Branch-and-Bound Algorithm Conscious of Special Structures in Concave Minimization Problems

Takahito Kuno[*]and Hidetoshi Nagai[†]

Graduate School of System and Information Engineering

University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan

July 2005

Abstract

In this paper, we develop a simplicial branch-and-bound algorithm for generating globally optimal solutions to concave minimization problems with low rank nonconvex structures. We propose to remove all additional constraints imposed on the usual linear programming relaxed problem. Therefore, in each bounding operation, we solve a linear programming problem whose constraints are exactly the same as the target problem. Although the lower bound worsens as a natural consequence, we offset this weakness by using an inexpensive bound tightening procedure based on Lagrangian relaxation. After giving a proof of the convergence, we report a numerical comparison with existing algorithms.

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

1 Introduction

In this paper, we develop a branch-and-bound algorithm to globally solve a class of concave minimization problems, to which many of low rank nonconvex structured problems reduce. Let us consider a concave function F defined on $\mathbb{R}^n$ and suppose the nonconvexity rank [9] is $r << n - r$, or the linearity [17] is $n - r >> r$. It is known [9] that $F(\mathbf{D}_x\mathbf{x} + \mathbf{D}_y\mathbf{y})$ is affine for some orthogonal matrix $\mathbf{D} = [\mathbf{D}_x, \mathbf{D}_y] \in \mathbb{R}^{n\times n}$ with $\mathbf{D}_x \in \mathbb{R}^{n\times r}$ when the value of each component of $\mathbf{x}$ is fixed. The matrix $\mathbf{D}$ is referred to as a *certificate* for the nonconvexity rank of F. If we try to minimize such a kind of concave functions, we can move the affine part of the objective function into the set of constraints by means of an auxiliary variable. The resulting problem has a characteristic structure that the variables involved in the objective function are a small fraction of the

[*]The author was partially supported by the Grand-in-Aid for Scientific Research (C) 17560050 from the Japan Society for the Promotion of Sciences. E-mail: takahito@cs.tsukuba.ac.jp

[†]E-mail: nagai@syou.cs.tsukuba.ac.jp

whole variables, e.g., in the case associated with the above F, we have a total of $n+1$ variables but $r+1$ among them in the objective function. Although it might not be easy to identify $\mathbf{D}$ in general, there are a number of cases with obvious certificates. A typical example is the production-transportation problem [12, 14, 20, 21]. This is a class of minimum concave-cost flow problems and minimizes the sum of concave production and linear transportation costs on a bipartite network (see Example 3.1 in Section 3).

Even if the objective function is not concave, we can reduce the problem to our intended class in some cases. Let us consider the linear multiplicative programming problem [8, 10, 13, 18]:

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

where $\mathbf{a}_i^{\mathsf{T}}\mathbf{y}+a_{i0}\geq 0$ for any feasible solution $\mathbf{y}\in\mathbb{R}^{n-r}$. Although the objective function of (1.1) is not concave but pseudoconcave [2]. we can reduce it to a concave minimization problem by introducing a vector $\mathbf{x}=(x_1,\ldots,x_r)^{\mathsf{T}}$ of auxiliary variables:

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

In general, the number r of affine functions in the objective function of (1.1) is assumed to be far less than the dimensionality $n-r$ of $\mathbf{y}$. Therefore, the variables involved in the objective function of (1.2) are again a small portion of the whole variables.

If the objective function is separable into a sum of univariate functions like (1.2), problems of our class can be solved rather efficiently using the rectangular branch-and-bound algorithm [4, 10, 12]. To deal with a wider range of problems, we do not assume the separability throughout this paper. We then tailor the simplicial branch-and-bound algorithm [5] 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 standard simplicial branch-and-bound algorithm. In Section 3, we will explore some difficulties in the implementation of existing bound procedures and propose to simplify the linear programming relaxation to be solved at each iteration. This simplification still guarantees the convergence property but deteriorates the quality of the lower bound on the optimal value. To prevent rapid growth of the branching tree, we will develop an additional bounding procedure based on Lagrangian relaxation in Section 4. Lastly, we will close the paper with a report of computational comparison among the proposed algorithm and two existing ones in Section 5.

2 Problem settings and the simplicial algorithm

Let f be a continuously differentiable concave function defined on an open convex set in a subspace $\mathbb{R}^r$ of $\mathbb{R}^n$ ($r\leq n$). The problem we consider in this paper is of minimizing the function f over a polyhedron in $\mathbb{R}^n$:

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

where $\mathbf{A} \in \mathbb{R}^{m \times r}$, $\mathbf{B} \in \mathbb{R}^{m \times (n-r)}$ and $\mathbf{b} \in \mathbb{R}^m$. Let us denote the polyhedron and its projection onto the subspace $\mathbb{R}^r$, respectively, by

$$\begin{aligned} W &= \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b},\ (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}\} \\ X &= \{\mathbf{x} \in \mathbb{R}^r \mid \exists \mathbf{y},\ (\mathbf{x}, \mathbf{y}) \in W\}. \end{aligned}$$

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

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

which we refer to as the master problem. We assume that W is nonempty and bounded. The same is then true for the projection X; and so we have

$$v = \max\{\mathbf{e}^\mathsf{T}\mathbf{x} \mid \mathbf{x} \in X\},$$

where $\mathbf{e} \in \mathbb{R}^r$ is the all-ones vector. We also assume the domain of f large enough to include an r-simplex

$$\Delta^1 = \{\mathbf{x} \in \mathbb{R}^r \mid \mathbf{e}^\mathsf{T}\mathbf{x} \leq v,\ \mathbf{x} \geq \mathbf{0}\}.$$

Unless the objective function f is separable, the simplicial branch-and-bound algorithm [4, 11] is a standard method for locating a globally optimal solution of (2.1), or equivalently of P. In this algorithm, while subdividing $\Delta^1 \supset X$ into smaller simplices Δ^i, $i \in \mathcal{L}$, such that

$$\bigcup_{i \in \mathcal{L}} \Delta^i = \Delta^1, \quad \text{int}(\Delta^p) \cap \text{int}(\Delta^q) = \emptyset \ \text{ if } p \neq q,$$

we solve subproblems of the master problem P one after another. The feasible set of each subproblem is restricted by Δ^i; and we need to solve the following with $\Delta = \Delta^i$ for every $i \in \mathcal{L}$:

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

This problem belongs to the same class of concave minimization problems as P and cannot be solved directly. Therefore, subproblems are recursively processed according to three basic steps:

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

Step 1 (subproblem selection). Take an appropriate index i_k out of $\mathcal{L}$ and let $\Delta := \Delta^{i_k}$.

Step 2 (bounding operation). Compute a lower bound z^k on the optimal value $z(\Delta)$ of P(Δ). If $f(\mathbf{x}^*) \leq z^k$ 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 $k := k + 1$.

If $X \cap \Delta = \emptyset$, then $z(\Delta)$ is thought of as $+\infty$. When $\mathcal{L}$ eventually becomes empty in this process, we see that the current incumbent $\mathbf{x}^*$ is an optimal solution to the master problem P. However, the set would not be empty in general, and the algorithm generates an infinite sequence of simplices $\{\Delta^{k_\ell} \mid \ell = 1, 2, \dots\}$ such that

$$\Delta^{k_1} \supset \Delta^{k_2} \supset \cdots, \quad X \cap \left(\bigcap_{\ell=1}^{\infty} \Delta^{k_\ell} \right) \neq \emptyset. \tag{2.2}$$

To guarantee the finiteness of the algorithm, we have to introduce a tolerance $\epsilon > 0$ to the backtracking criterion $f(\mathbf{x}^*) \leq z^k$ of Step 2 as follows:

$$f(\mathbf{x}^*) - z^k \leq \epsilon, \quad \text{or } f(\mathbf{x}^*) - z^k \leq \epsilon |f(\mathbf{x}^*)|, \tag{2.3}$$

and besides subdivide Δ^1 in an exhaustive manner that makes $\cap_{\ell=1}^{\infty} \Delta^{k_\ell}$ a singleton. The simplest exhaustive subdivision rule is *bisection*. We may select the longest edge of Δ and divide it at a fixed ratio of $\alpha \in (0, 1/2]$. In fact, this can be done easily if Δ is given as the convex hull $\Delta = \text{conv}(\{\mathbf{v}_1, \dots, \mathbf{v}_{r+1}\})$ of its $r+1$ vertices $\mathbf{v}_1, \dots, \mathbf{v}_{r+1}$. Suppose $\mathbf{v}_p$–$\mathbf{v}_q$ is the longest edge of Δ. Letting $\mathbf{v} = (1-\alpha)\mathbf{v}_p + \alpha \mathbf{v}_q$, then we have

$$\Delta^{2k} = \text{conv}\left(\{\mathbf{v}_j \mid j \neq p\} \cup \{\mathbf{v}\}\right), \quad \Delta^{2k+1} = \text{conv}\left(\{\mathbf{v}_j \mid j \neq q\} \cup \{\mathbf{v}\}\right).$$

Note that the initial simplex Δ^1 has vertices $\mathbf{0}$, $v\mathbf{e}_1$, ..., $v\mathbf{e}_r$, where $\mathbf{e}_j \in \mathbb{R}^r$ is the jth unit vector. Thus, starting from $\Delta^1 = \text{conv}(\{\mathbf{0}, v\mathbf{e}_1, \dots, v\mathbf{e}_r\})$, we can generate Δ^i for all $i \in \mathcal{L}$ in a well-defined way. If the bisection rule is adopted and $\epsilon > 0$, we can obtain an approximate 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 i_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 i_k of least z^k is taken out of $\mathcal{L}$.

The most time-consuming step in the simplicial branch-and-bound algorithm is the bounding operation of Step 2. In the next section, we will discuss troublesome issues with Step 2 faced by existing algorithms and their resolution in treating our target problem (2.1)

3 Linear programming relaxations

At Step 2 of the usual simplicial branch-and-bound algorithm, we replace the objective function of P(Δ) by its convex envelope g on Δ and solve a relaxed problem:

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

The convex envelope g is an affine function which agrees with f at the $r+1$ 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 $\mathbf{x}$ is given as a convex combination of $\mathbf{v}_j$, $j = 1, \dots, r+1$:

$$\mathbf{x} = \sum_{j=1}^{r+1} \zeta_j \mathbf{v}_j, \quad \sum_{j=1}^{r+1} \zeta_j = 1, \quad \boldsymbol{\zeta} = (\zeta_1, \dots, \zeta_{r+1})^{\mathsf{T}} \geq \mathbf{0}. \tag{3.1}$$

By the concavity of f, we have

$$g(\mathbf{x}) = \sum_{j=1}^{r+1} \zeta_j f(\mathbf{v}_j) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in \Delta. \tag{3.2}$$

Substituting (3.1) into Q(Δ), we have an equivalent linear programming problem of $n+1$ variables:

$$\left| \begin{array}{lll} \text{minimize} & w = & \mathbf{f}^{\mathsf{T}} \boldsymbol{\zeta} \\ \text{subject to} & & \mathbf{A}\mathbf{V}\boldsymbol{\zeta} + \mathbf{B}\mathbf{y} = \mathbf{b} \\ & & \mathbf{e}^{\mathsf{T}}\boldsymbol{\zeta} = 1, \quad (\boldsymbol{\zeta}, \mathbf{y}) \geq \mathbf{0}, \end{array} \right. \tag{3.3}$$

where

$$\mathbf{f} = [f(\mathbf{v}_1), \dots, f(\mathbf{v}_{r+1})]^{\mathsf{T}}, \quad \mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_{r+1}]. \tag{3.4}$$

Obviously, (3.3) has an optimal solution $(\boldsymbol{\zeta}^\circ, \mathbf{y}^\circ)$ if and only if $X \cap \Delta \neq \emptyset$. Since the inequality in (3.2) holds, we can set the lower bound z^k to

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

When $X \cap \Delta \neq \emptyset$, we also have a feasible solution $\mathbf{x}^\circ$ to the subproblem P(Δ), and hence to the master problem P, by letting $\mathbf{x}^\circ = \mathbf{V}\boldsymbol{\zeta}^\circ$. We can therefore update the incumbent $\mathbf{x}^*$ with $\mathbf{x}^\circ$ if necessary.

The troublesome issues are

(a) each (3.3) associated with Q(Δ) has a different set of constraints, and

(b) no (3.3) inherits the structure of the target problem (2.1).

Despite the vast number of (3.3)'s we have to solve before convergence, the solutions to previous ones are of little use in solving the current one, because they might be neither feasible nor dual feasible, due to (a). Moreover, even if the target problem (2.1) has some favorable structure, like network flow, (b) prevents us from applying efficient algorithms to (3.3). These issues, however, have been resolved partly in [11], as will be seen below.

3.1 Modified relaxation proposed in [11]

In [11], Kuno and Nagai have relaxed the constraint $\mathbf{x} \in \Delta$ of Q(Δ) into a bounding constraint $\mathbf{s} \leq \mathbf{x} \leq \mathbf{t}$. Component of the vectors $\mathbf{s}, \mathbf{t} \in \mathbb{R}^r$ are defined as

$$\left. \begin{array}{l} s_i = \min\{v_{ij} \mid j = 1, \dots, r+1\} \\ t_i = \max\{v_{ij} \mid j = 1, \dots, r+1\} \end{array} \right\} \quad i = 1, \dots, r,$$

where v_{ij} denotes the ith component of $\mathbf{v}_j$. Let

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

Then their alternative to $\mathrm{Q}(\Delta)$ is written as follows:

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

They have also proposed to abandon the variable transformation (3.1). Instead, the convex envelope $g(\mathbf{x}) = \mathbf{c}^\mathsf{T}\mathbf{x} + c_{r+1}$ is identified by solving a system of linear equations:

$$\mathbf{c}^\mathsf{T}\mathbf{v}_j + c_{r+1} = f(\mathbf{v}_j), \quad j = 1, \ldots, r+1, \tag{3.5}$$

and $\overline{\mathrm{Q}}(\Delta)$ is solved as a linear programming problem:

$$\left| \begin{array}{lll} \text{minimize} & w = & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{subject to} & & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b}, \quad \mathbf{s} \leq \mathbf{x} \leq \mathbf{t}, \quad \mathbf{y} \geq \mathbf{0}, \end{array} \right. \tag{3.6}$$

where

$$[\mathbf{c}^\mathsf{T}, c_{r+1}] = \mathbf{f}^\mathsf{T} \begin{bmatrix} \mathbf{V} \\ \mathbf{e}^\mathsf{T} \end{bmatrix}^{-1}. \tag{3.7}$$

If $X \cap \Gamma(\Delta) \neq \emptyset$, then (3.6) has an optimal solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$. The lower bound z^k can be set to

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

because $\overline{w} \leq w^\circ$ holds by the inclusion relation between the feasible sets of $\overline{\mathrm{Q}}(\Delta)$ and $\mathrm{Q}(\Delta)$.

Except for the bounding constraint $\mathbf{s} \leq \mathbf{x} \leq \mathbf{t}$, each (3.6) associated with $\overline{\mathrm{Q}}(\Delta)$ shares constraints. We can solve the current (3.6) using an optimal solution to the preceding one as the initial solution. Since the solution violates only the bounding constraint at worst, it regains the feasibility and optimality in a very few pivoting operations of the dual and primal simplex algorithms. Also, if (2.1) has some favorable structures, (3.6) inherits them. Unfortunately, however, it is not that (3.6) inherits the structure of the original low-rank nonconvex problem behind the target (2.1).

Example 3.1 Let us consider the production-transportation problem mentioned in Section 1:

$$\left| \begin{array}{lll} \text{minimize} & z = & \displaystyle\sum_{i \in M} \sum_{j \in N} a_{ij} y_{ij} + f(\mathbf{x}) \\ \text{subject to} & & \displaystyle\sum_{j \in N} y_{ij} = x_i, \quad i \in M \\ & & \displaystyle\sum_{i \in M} y_{ij} = b_j, \quad j \in N \\ & & (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array} \right. \tag{3.8}$$

where $M = \{1, \ldots, r\}$, $N = \{r+1, \ldots, m\}$, $\mathbf{x} = (x_i \mid i \in M)$ and $\mathbf{y} = (y_{ij} \mid i \in M,\ j \in N)$. We assume that the production cost f is a nonlinear and concave function on the feasible set, and that the unit transportation cost a_{ij} is nonnegative for each $i \in M$ and $j \in N$. If the amount of production x_i is constant for each $i \in M$, then (3.8) is an ordinary Hitchcock problem and can be solved in polynomial time using a special-purpose algorithm for network flow (see e.g., [1]). Introducing an additional variable $\xi \geq 0$, we have the same form as (2.1):

$$\left|\begin{array}{lrl} \text{minimize} & z = & \xi + f(\mathbf{x}) \\ \text{subject to} & & \xi - \sum_{i \in M} \sum_{j \in N} a_{ij} y_{ij} = 0 \\ & & x_i - \sum_{j \in N} y_{ij} = 0, \quad i \in M \\ & & \sum_{i \in M} y_{ij} = b_j, \quad j \in N \\ & & (\xi, \mathbf{x}, \mathbf{y}) \geq \mathbf{0}. \end{array}\right. \tag{3.9}$$

The linear programming representation of $\overline{\mathrm{Q}}(\Delta)$ associated with (3.9) is then as follows:

$$\left|\begin{array}{lrl} \text{minimize} & w = & \mathbf{c}_0 \xi + \mathbf{c}^{\mathsf{T}} \mathbf{x} \\ \text{subject to} & & \xi - \sum_{i \in M} \sum_{j \in N} a_{ij} y_{ij} = 0 \\ & & x_i - \sum_{j \in N} y_{ij} = 0, \quad i \in M \\ & & \sum_{i \in M} y_{ij} = b_j, \quad j \in N \\ & & s_0 \leq \xi \leq t_0, \quad \mathbf{s} \leq \mathbf{x} \leq \mathbf{t}, \quad \mathbf{y} \geq \mathbf{0}. \end{array}\right. \tag{3.10}$$

This is not a Hitchcock problem any longer, nor even a network flow problem. To solve (3.10), we have to use a general-purpose algorithm. ■

In addition to this, we have another difficulty with this modification. To obtain the objective function of (3.6), we need to solve the linear system (3.5) for $[\mathbf{c}^{\mathsf{T}}, c_{r+1}]$. If we adopt the depth-first rule at Step 1, it can be done in $O(r)$ almost always, as shown in [11]. However, its solution becomes numerically unstable as Δ grows smaller, and might fail to be computed in the worst case, due to rounding errors.

3.2 New relaxation resolving all difficulties

Both difficulties involved in the above modification can be swept away using two kinds of relaxation in combination.

Let us denote by $\phi(\Delta)$ the diameter of Δ, e.g.,

$$\phi(\Delta) = \min\{\|\mathbf{v}_p - \mathbf{v}_q\| \mid p = 1, \ldots, r;\ q = p+1, \ldots, r+1\}, \tag{3.11}$$

and assume that (3.5) can be solved with sufficient precision if $\phi(\Delta) \geq \delta$ for some number $\delta > 0$. While $\phi(\Delta) \geq \delta$, we drop the bounding constraint $\mathbf{x} \in \Gamma(\Delta)$ from $\overline{\mathrm{Q}}(\Delta)$ and solve

$$\widetilde{\mathrm{Q}}_g(\Delta) \left|\begin{array}{lrl} \text{minimize} & w = & g(\mathbf{x}) \\ \text{subject to} & & \mathbf{x} \in X. \end{array}\right.$$

If $\phi(\Delta)$ becomes smaller than δ, we further replace the objective function g by a simpler underestimating function of f. For this purpose, we first compute the gradient vector $\mathbf{d} = \nabla f(\mathbf{u})$ of f at the centroid $\mathbf{u} = \sum_{j=1}^{r+1} \mathbf{v}_j/(r+1)$ of Δ. Let

$$d_{r+1} = \min\{f(\mathbf{v}_j) - \mathbf{d}^\mathsf{T}\mathbf{v}_j \mid j = 1, \ldots, r+1\}, \tag{3.12}$$

and let

$$h(\mathbf{x}) = \mathbf{d}^\mathsf{T}\mathbf{x} + d_{r+1}.$$

From (3.12) and the concavity of f, we see that

$$h(\mathbf{x}) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in \Delta, \tag{3.13}$$

where the equality holds at some $\mathbf{v}_j$ fixing d_{r+1}. We then solve the following:

$$\widetilde{\mathrm{Q}}_h(\Delta) \left| \begin{array}{lll} \text{minimize} & w = & h(\mathbf{x}) \\ \text{subject to} & & \mathbf{x} \in X. \end{array} \right.$$

The problem to be solved depends on the size of Δ; but in any case it is equivalent to a linear programming problem of the same form:

$$\left| \begin{array}{lll} \text{minimize} & w = & \boldsymbol{\theta}^\mathsf{T}\mathbf{x} \\ \text{subject to} & & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array} \right. \tag{3.14}$$

where

$$[\boldsymbol{\theta}^\mathsf{T}, \theta_{r+1}] = \begin{cases} [\mathbf{c}^\mathsf{T}, c_{r+1}] & \text{if } \phi(\Delta) \geq \delta \\ [\mathbf{d}^\mathsf{T}, d_{r+1}] & \text{otherwise.} \end{cases}$$

Since X is nonempty, (3.14) has an optimal solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ and we have the following lower bound on the value of $\mathrm{P}(\Delta)$:

$$\tilde{w} = \boldsymbol{\theta}^\mathsf{T}\tilde{\mathbf{x}} + \theta_{r+1}.$$

Proposition 3.1 *Among the values $\tilde{w}$, $\overline{w}$, w° and $z(\Delta)$ exist relationships:*

$$\tilde{w} \leq \overline{w} \leq w^\circ \leq z(\Delta). \tag{3.15}$$

Proof: Let us show the first inequality. If $[\boldsymbol{\theta}^\mathsf{T}, \theta_{r+1}] = [\mathbf{c}^\mathsf{T}, c_{r+1}]$, then it follows from the inclusion relation between the feasible sets X of $\widetilde{\mathrm{Q}}_g(\Delta)$ and $X \cap \Gamma(\Delta)$ of $\overline{\mathrm{Q}}(\Delta)$. Recall that the objective function g of both problems is a convex envelope of f, i.e., a maximal convex function underestimating f on Δ. Therefore, we have

$$h(\mathbf{x}) \leq g(\mathbf{x}), \quad \forall \mathbf{x} \in \Delta,$$

which proves the case where $[\boldsymbol{\theta}^\mathsf{T}, \theta_{r+1}] = [\mathbf{d}^\mathsf{T}, d_{r+1}]$. ■

We see from this proposition that $\tilde{w}$ can serve as z^k at Step 2, though inferior to w° and $\overline{w}$. Problem (3.14) yielding $\tilde{w}$, however, has the redeeming feature that the constraints are exactly the same as those of (2.1). Whichever of $\widetilde{\mathrm{Q}}_g(\Delta)$ and $\widetilde{\mathrm{Q}}_h(\Delta)$ we need to solve, we can use an optimal solution to the preceding (3.14) as the initial feasible basic solution and start the primal simplex algorithm immediately. The most important thing is that (3.14) inherits not only the structure of (2.1) but also that of the original problem behind it.

Example 3.2 Again, consider the problem (3.9) which is reduced from the production-transportation problem (3.8). Associated with (3.9), we have the following linear programming representation of $\widetilde{\mathrm{Q}}_g(\Delta)$ and $\widetilde{\mathrm{Q}}_h(\Delta)$:

$$\left|\begin{array}{lll} \text{minimize} & w = & \theta_0\xi + \boldsymbol{\theta}^\mathsf{T}\mathbf{x} \\ \text{subject to} & & \xi - \sum_{i\in M}\sum_{j\in N} a_{ij}y_{ij} = 0 \\ & & x_i - \sum_{j\in N} y_{ij} = 0, \quad i \in M \\ & & \sum_{i\in M} y_{ij} = b_j, \quad j \in N \\ & & (\xi, \mathbf{x}, \mathbf{y}) \geq \mathbf{0}. \end{array}\right. \tag{3.16}$$

If we substitute the first constraint into the objective function and eliminate ξ, then (3.16) reduces to

$$\left|\begin{array}{lll} \text{minimize} & w = & \theta_0 \sum_{i\in M}\sum_{j\in N} a_{ij}y_{ij} + \boldsymbol{\theta}^\mathsf{T}\mathbf{x} \\ \text{subject to} & & \sum_{j\in N} y_{ij} = x_i, \quad i \in M \\ & & \sum_{i\in M} y_{ij} = b_j, \quad j \in N \\ & & (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array}\right. \tag{3.17}$$

which is a transshipment problem, a generalization of the Hitchcock problem, and can be solved efficiently using the network simplex algorithm or an appropriate polynomial-time algorithm for network flow (see [1] for details). ■

Example 3.3 Let us consider a more general example:

$$\left|\begin{array}{lll} \text{minimize} & z = & F(\mathbf{x}, \mathbf{y}) \\ \text{subject to} & & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array}\right. \tag{3.18}$$

where F is a concave function of nonconvexity rank $r < n$. Since the objective function can be written as

$$F(\mathbf{D}_x\mathbf{x} + \mathbf{D}_y\mathbf{y}) = f(\mathbf{x}) + \mathbf{a}^\mathsf{T}\mathbf{y} + a_0$$

for some $a_0 \in \mathbb{R}$, $\mathbf{a} \in \mathbb{R}^{n-r}$ and a certificate $\mathbf{D} = [\mathbf{D}_x, \mathbf{D}_y] \in \mathbb{R}^{n\times n}$ with $\mathbf{D}_x \in \mathbb{R}^{n\times r}$ [9], problem (3.18) is equivalent to

$$\left|\begin{array}{lll} \text{minimize} & z = & \xi + f(\mathbf{x}) \\ \text{subject to} & & \xi - \mathbf{a}^\mathsf{T}\mathbf{y} = a_0 \\ & & [\mathbf{A}, \mathbf{B}]\,\mathbf{D}_x\mathbf{x} + [\mathbf{A}, \mathbf{B}]\,\mathbf{D}_y\mathbf{y} = \mathbf{b} \\ & & \mathbf{D}_x\mathbf{x} \geq \mathbf{0}, \quad \mathbf{D}_y\mathbf{y} \geq \mathbf{0}. \end{array}\right. \tag{3.19}$$

Note that we can rearrange the constants into the form of (2.1) using appropriate transformations. The linear programming representation of $\widetilde{\mathrm{Q}}_g(\Delta)$ and $\widetilde{\mathrm{Q}}_h(\Delta)$ associated with (3.19) is then as follows:

$$\left|\begin{array}{lll} \text{minimize} & w = & \theta_0\xi + \boldsymbol{\theta}^\mathsf{T}\mathbf{x} \\ \text{subject to} & & \xi - \mathbf{a}^\mathsf{T}\mathbf{y} = a_0 \\ & & [\mathbf{A}, \mathbf{B}]\,\mathbf{D}_x\mathbf{x} + [\mathbf{A}, \mathbf{B}]\,\mathbf{D}_y\mathbf{y} = \mathbf{b} \\ & & \mathbf{D}_x\mathbf{x} \geq \mathbf{0}, \quad \mathbf{D}_y\mathbf{y} \geq \mathbf{0}. \end{array}\right.$$

This problem is equivalent to the following with the same set of constraints as (3.18):

$$\left|\begin{array}{lll} \text{minimize} & w = & (\boldsymbol{\theta}^{\mathsf{T}}, \theta_0 \mathbf{a}^{\mathsf{T}})\mathbf{D}^{-1} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \\ \text{subject to} & & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}. \end{array}\right. \tag{3.20}$$

■

4 Revised simplicial algorithm

The use of $\widetilde{\mathrm{Q}}_g(\Delta)$ and $\widetilde{\mathrm{Q}}_h(\Delta)$ in combination has yet another advantage over the existing linear programming relaxations, other than those we have seen in the preceding section. In $\overline{\mathrm{Q}}(\Delta)$, the constraint associated with Δ is left as $\mathbf{x} \in \Gamma(\Delta)$ to guarantee the convergence of the algorithm. Although neither $\widetilde{\mathrm{Q}}_g(\Delta)$ nor $\widetilde{\mathrm{Q}}_h(\Delta)$ has such an additional constraint, the objective function of $\widetilde{\mathrm{Q}}_h(\Delta)$ dominating the convergence behavior enables us to prove it without any extra effort. On the other hand, however, both $\widetilde{\mathrm{Q}}_g(\Delta)$ and $\widetilde{\mathrm{Q}}_h(\Delta)$ have the obvious drawback that their value $\widetilde{w}$ is inferior to w° and $\overline{w}$ as the lower bound z^k on the value of $\mathrm{P}(\Delta)$. Before proceeding to the convergence analysis, let us discuss a procedure, based on Lagrangian relaxation, for tightening $\widetilde{w}$. In [11], it has been reported that a similar procedure works well for tightening $\overline{w}$.

4.1 Lagrangian relaxation for tightening $\widetilde{w}$

For the lower bound $\widetilde{w}$, let

$$G = \{\mathbf{x} \in \mathbb{R}^r \mid \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} \geq \widetilde{w} - \theta_{r+1}\},$$

where $[\boldsymbol{\theta}^{\mathsf{T}}, \theta_{r+1}] = [\mathbf{c}^{\mathsf{T}}, c_{r+1}]$ if $\widetilde{w}$ is yielded by $\widetilde{\mathrm{Q}}_g(\Delta)$; otherwise, $[\boldsymbol{\theta}^{\mathsf{T}}, \theta_{r+1}] = [\mathbf{d}^{\mathsf{T}}, d_{r+1}]$. Since $X \cap \Delta$ is a subset of G, no feasible solution to the subproblem $\mathrm{P}(\Delta)$ is lost if we add $\mathbf{x} \in G$ as a constraint. The resulting problem is then equivalent to

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

Introducing a Lagrangian multiplier $\boldsymbol{\lambda} \in \mathbb{R}^m$ for the constraint $\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b}$, we have a problem:

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

by noting $\mathbf{x} \geq \mathbf{0}$ for any $\mathbf{x} \in \Delta$. If $\boldsymbol{\theta}^{\mathsf{T}}\mathbf{v}_j < \widetilde{w} - \theta_{r+1}$ for each vertex $\mathbf{v}_j$ of Δ, then $\mathrm{L}(\Delta; \boldsymbol{\lambda})$ is infeasible. In that case, the hyperplane $\partial G = \{\mathbf{x} \in \mathbb{R}^r \mid \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} = \widetilde{w} - \theta_{r+1}\}$ separates Δ from X; and we can discard Δ because it can never contain an optimal solution to the master problem P. In the rest of this subsection, we assume

$$\exists j \in \{1, \ldots, r+1\}, \quad \boldsymbol{\theta}^{\mathsf{T}}\mathbf{v}_j \geq \widetilde{w} - \theta_{r+1}. \tag{4.2}$$

Let $(\mathbf{x}(\boldsymbol{\lambda}), \mathbf{y}(\boldsymbol{\lambda}))$ be an optimal solution to $\mathrm{L}(\Delta; \boldsymbol{\lambda})$ and let

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

Then $w(\boldsymbol{\lambda})$ is a lower bound on $z(\Delta)$ for any $\boldsymbol{\lambda}$, as is well known (see e.g. [15]). The question lies in how we should fix the value of $\boldsymbol{\lambda}$ in $\mathrm{L}(\Delta; \boldsymbol{\lambda})$ inexpensively so that $w(\boldsymbol{\lambda}) > \widetilde{w}$ holds. To answer this, let $\boldsymbol{\lambda}$ be constant and consider a linear programming problem:

$$\left| \begin{array}{lll} \text{minimize} & w = & (\boldsymbol{\theta}^{\mathsf{T}} - \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{A})\mathbf{x} - \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{B}\mathbf{y} + \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{b} \\ \text{subject to} & & (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array} \right. \tag{4.3}$$

which is obtained from $\mathrm{L}(\Delta; \boldsymbol{\lambda})$ by dropping $\mathbf{x} \in \Delta$, $\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} \geq \widetilde{w} - \theta_{r+1}$, and by replacing f with its underestimating function g or h. If $\boldsymbol{\theta}^{\mathsf{T}} - \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{A} \geq \mathbf{0}$ and $\boldsymbol{\lambda}^{\mathsf{T}}\mathbf{B} \leq \mathbf{0}$, then (4.3) has a trivial optimal value $\boldsymbol{\lambda}^{\mathsf{T}}\mathbf{b}$. These conditions simultaneously ensure the feasibility of the dual problem of (4.3):

$$\left| \begin{array}{lll} \text{maximize} & w = & \mathbf{b}^{\mathsf{T}}\boldsymbol{\lambda} \\ \text{subject to} & & \mathbf{A}^{\mathsf{T}}\boldsymbol{\lambda} \leq \boldsymbol{\theta} \\ & & \mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda} \leq \mathbf{0}. \end{array} \right. \tag{4.4}$$

If we think of $\boldsymbol{\lambda}$ as a vector of variables, (4.4) is the dual problem of the linear programming problem (3.14) introduced in the previous section. In other words, the value of $\boldsymbol{\lambda}$ maximizing the optimal value of (4.3) is given by an optimal solution $\widetilde{\boldsymbol{\lambda}}$ to the dual problem of $\widetilde{\mathrm{Q}}_g(\Delta)$ or $\widetilde{\mathrm{Q}}_h(\Delta)$. Our answer to the question is a simple one of fixing $\boldsymbol{\lambda} = \widetilde{\boldsymbol{\lambda}}$ even in $\mathrm{L}(\Delta; \boldsymbol{\lambda})$.

Note that the dual optimal solution $\widetilde{\boldsymbol{\lambda}}$ is yielded as a byproduct in solving the primal problem (3.14). Moreover, we see from the constraints of (4.4) that $\mathbf{y}(\widetilde{\boldsymbol{\lambda}}) = \mathbf{0}$ because the coefficient of $\mathbf{y}$ in the objective function of $\mathrm{L}(\Delta; \widetilde{\boldsymbol{\lambda}})$ must be a nonnegative vector. Therefore, deleting $\mathbf{y}$ from $\mathrm{L}(\Delta; \widetilde{\boldsymbol{\lambda}})$, we may solve the following to obtain $w(\widetilde{\boldsymbol{\lambda}})$:

$$\left| \begin{array}{lll} \text{minimize} & w = & f(\mathbf{x}) - \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}(\mathbf{A}\mathbf{x} - \mathbf{b}) \\ \text{subject to} & & \mathbf{x} \in \Delta, \quad \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} \geq \widetilde{w} - \theta_{r+1}. \end{array} \right. \tag{4.5}$$

Proposition 4.1 *Among the values $w(\widetilde{\boldsymbol{\lambda}})$, $\widetilde{w}$ and $z(\Delta)$ exist relationships:*

$$\widetilde{w} \leq w(\widetilde{\boldsymbol{\lambda}}) \leq z(\Delta), \tag{4.6}$$

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: Let w_j denote the objective function value of (4.5) at each vertex $\mathbf{v}_j$ of Δ. Since g is a convex envelope of f on Δ and $\widetilde{\boldsymbol{\lambda}}$ satisfies the constraints of (4.4), we have

$$\begin{aligned} w_j = f(\mathbf{v}_j) - \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}(\mathbf{A}\mathbf{v}_j - \mathbf{b}) &= g(\mathbf{v}_j) - \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}(\mathbf{A}\mathbf{v}_j - \mathbf{b}) \\ &\geq \boldsymbol{\theta}^{\mathsf{T}}\mathbf{v}_j + \theta_{r+1} - \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}(\mathbf{A}\mathbf{v}_j - \mathbf{b}) \\ &= (\boldsymbol{\theta}^{\mathsf{T}} - \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}\mathbf{A})\mathbf{v}_j + \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}\mathbf{b} + \theta_{r+1} \\ &\geq \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}\mathbf{b} + \theta_{r+1} = \widetilde{w}. \end{aligned}$$

This, together with the concavity of f, implies that the objective function value of (4.5) at any point in Δ is bounded from below by $\widetilde{w}$. Therefore, for $\mathbf{x}(\widetilde{\boldsymbol{\lambda}}) \in \Delta$, we have

$$\widetilde{w} \leq f(\mathbf{x}(\widetilde{\boldsymbol{\lambda}})) - \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}(\mathbf{A}\mathbf{x}(\widetilde{\boldsymbol{\lambda}}) - \mathbf{b}) = w(\widetilde{\boldsymbol{\lambda}}),$$

by noting $\mathbf{y}(\widetilde{\boldsymbol{\lambda}}) = \mathbf{0}$. Suppose that $\mathbf{x}(\widetilde{\boldsymbol{\lambda}}) \notin \{\mathbf{v}_1, \dots, \mathbf{v}_{r+1}\}$. Since $\mathbf{x}(\widetilde{\boldsymbol{\lambda}})$ lies on some vertex of $\Delta \cap \partial G$, there are some vertices $\mathbf{v}_p$, $\mathbf{v}_q$ of Δ and $\mu \in (0,1)$ such that $\mathbf{x}(\widetilde{\boldsymbol{\lambda}}) = (1-\mu)\mathbf{v}_p + \mu\mathbf{v}_q$. If f is strictly concave, we have $w(\widetilde{\boldsymbol{\lambda}}) > (1-\mu)w_p + \mu w_q \geq \widetilde{w}$. ■

Remark that $w(\widetilde{\boldsymbol{\lambda}})$ can be superior even to w° yielded by the usual relaxation $\mathrm{Q}(\Delta)$ because $\mathrm{Q}(\Delta)$ shares the objective function with $\widetilde{\mathrm{Q}}_g(\Delta)$ and hence w° might coincide with $\widetilde{w}$ when $\tilde{\mathbf{x}} \in \Delta$ and $\phi(\Delta) \geq \delta$. We should also remark that $w(\widetilde{\boldsymbol{\lambda}})$ can be computed in time polynomial in r if the value of f is given by oracle, though $\mathrm{L}(\Delta; \widetilde{\boldsymbol{\lambda}})$ is a concave minimization problem. Since the objective function of (4.5) is concave, $\mathbf{x}(\widetilde{\boldsymbol{\lambda}})$ is assumed to be 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 $\mathbf{v}_p$–$\mathbf{v}_q$ of Δ such that $\mathbf{v}_p \in \mathrm{int}(G)$ and $\mathbf{v}_q \notin G$, as well as at each $\mathbf{v}_j \in G$.

Example 4.4 Let us continue Examples 3.1 and 3.2. If we solve (3.17) as the relaxed problem $\widetilde{\mathrm{Q}}_g(\Delta)$ or $\widetilde{\mathrm{Q}}_h(\Delta)$ of (3.9), we cannot directly obtain the value $\widetilde{\lambda}_0$ of the dual variable corresponding to the first constraint of (3.9). However, it is an easy exercise in linear programming to show that $\widetilde{\lambda}_0 = \theta_0$ holds. Thus, $\mathrm{L}(\Delta; \widetilde{\boldsymbol{\lambda}})$ for (3.9) is as follows:

$$\left|\begin{array}{lll} \text{minimize} & w = & (1-\theta_0)\xi + f(\mathbf{x}) - \sum_{i \in M} \widetilde{\lambda}_i x_i + \sum_{j \in N} \widetilde{\lambda}_j b_j \\ \text{subject to} & & (\xi, \mathbf{x}) \in \Delta, \quad \theta_0\xi + \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} \geq \widetilde{w} - \theta_{r+1}. \end{array}\right.$$

Similarly, we can obtain $\mathrm{L}(\Delta; \tilde{\boldsymbol{\lambda}})$ for (3.19) from (3.20) in Example 3.3 without any difficulty. ■

4.2 Algorithm description and convergence properties

Let us summarize the discussion so far. Recall the three basic steps of the simplicial branch-and-bound algorithm given in Section 2. Step 2 of the bounding operation we propose is implemented for given $\epsilon \geq 0$ and $\delta > 0$ in two stages:

Step 2.1. If $\phi(\Delta) \geq \delta$, then solve $\widetilde{\mathrm{Q}}_g(\Delta)$. Otherwise, solve $\widetilde{\mathrm{Q}}_h(\Delta)$. Let $z^k := \widetilde{w}$. If $f(\mathbf{x}^*) - z^k \leq \epsilon$ for the incumbent $\mathbf{x}^*$, then discard Δ.

Step 2.2. If $f(\mathbf{x}^*) - z^k > \epsilon$, then solve $\mathrm{L}(\Delta; \widetilde{\boldsymbol{\lambda}})$ and let $z^k := w(\widetilde{\boldsymbol{\lambda}})$. If $f(\mathbf{x}^*) - z^k \leq \epsilon$, then discard Δ.

We may of course replace the backtracking criterion by $f(\mathbf{x}^*) - z^k \leq \epsilon|f(\mathbf{x}^*)|$, as in (2.3). The following is the detailed description of our simplicial algorithm for solving the master problem P:

```
algorithm REVISED_SBB
begin
    compute $v := \max\{\mathbf{e}^\mathsf{T}\mathbf{x} \mid \mathbf{x} \in X\}$ and let $\Delta^1 := \mathrm{conv}(\{\mathbf{0}, v\mathbf{e}_1, \dots, v\mathbf{e}_r\})$;
    $\mathcal{L} := \{1\}$; $z^* := +\infty$; $k := 1$;
    while $\mathcal{L} \neq \emptyset$ do begin
        select $i_k \in \mathcal{L}$ and let $\mathcal{L} := \mathcal{L} \setminus \{i_k\}$; $\Delta := \Delta^{i_k}$;            /* Step 1 */
        let $\mathbf{v}_1, \dots, \mathbf{v}_{r+1}$ denote the vertices of $\Delta$;
        $\phi(\Delta) := \min\{\|\mathbf{v}_p - \mathbf{v}_q\| \mid p = 1, \dots, r;\ q = p+1, \dots, r+1\}$;      /* Step 2.1 */
        if $\phi(\Delta) \geq \delta$ then begin
            $\mathbf{V} := [\mathbf{v}_1, \dots, \mathbf{v}_{r+1}]$; $\mathbf{W} := [V^\mathsf{T}, \mathbf{e}]^\mathsf{T}$;            /* $\widetilde{\mathrm{Q}}_g(\Delta)$ */
            $[\boldsymbol{\theta}^\mathsf{T}, \theta_{r+1}] := [f(\mathbf{v}_1), \dots, f(\mathbf{v}_{r+1})]\mathbf{W}^{-1}$
        end
        else begin
            $\mathbf{u} := \sum_{j=1}^{r+1} \mathbf{v}_j/(r+1)$; $\mathbf{d} := \nabla f(\mathbf{u})$;            /* $\widetilde{\mathrm{Q}}_h(\Delta)$ */
            $d_{r+1} := \min\{f(\mathbf{v}_j) - \mathbf{d}^\mathsf{T}\mathbf{v}_j \mid j = 1, \dots, r+1\}$; $[\boldsymbol{\theta}^\mathsf{T}, \theta_{r+1}] := [d^\mathsf{T}, d_{r+1}]$
        end;
        solve (3.14) of minimizing $\boldsymbol{\theta}^\mathsf{T}\mathbf{x} + \theta_{r+1}$ on $X$ to compute $\mathbf{x}^k := \tilde{\mathbf{x}}$ and $z^k := \widetilde{w}$;
        if $f(\mathbf{x}^k) < z^*$ then update $z^* := f(\mathbf{x}^k)$ and $\mathbf{x}^* := \mathbf{x}^k$;
        if $z^* - z^k > \epsilon$ then begin            /* Step 2.2 */
            if $\boldsymbol{\theta}^\mathsf{T}\mathbf{v}_j \geq \widetilde{w} - \theta_{r+1}$ for some $j \in \{1, \dots, r+1\}$ then begin
                define (4.5) for a dual optimal solution $\widetilde{\boldsymbol{\lambda}}$ to (3.14);      /* $\mathrm{L}(\Delta; \widetilde{\boldsymbol{\lambda}})$ */
                solve (4.5) and update $z^k := w(\widetilde{\boldsymbol{\lambda}})$;
                if $z^* - z^k > \epsilon$ then begin            /* Step 3 */
                    select the longest edge $\mathbf{v}_p$–$\mathbf{v}_q$ of $\Delta$;
                    let $\mathbf{v} := (1-\alpha)\mathbf{v}_p + \alpha\mathbf{v}_q$ for a fixed $\alpha \in (0, 1/2]$;
                    $\Delta^{2k} := \mathrm{conv}(\{\mathbf{v}_j \mid j \neq p\} \cup \{\mathbf{v}\})$; $\Delta^{2k+1} := \mathrm{conv}(\{\mathbf{v}_j \mid j \neq q\} \cup \{\mathbf{v}\})$;
                    $\mathcal{L} := \mathcal{L} \cup \{2k, 2k+1\}$
                end
            end
        end;
        $k := k + 1$
    end
end;
```

To analyze the convergence properties, we will first see how algorithm REVISED_SBB behaves if it does not terminate. In that case, an infinite sequence of nested simplices is generated as in (2.2); and it shrinks to a single point because Δ is subdivided according to the bisection rule. Moreover, we can show the following:

Lemma 4.2 *Suppose that algorithm REVISED_SBB generates an infinite sequence of simplices* $\{\Delta^{k_\ell} \mid \ell = 1, 2, \dots\}$ *such that*

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

Then we have

$$\lim_{\ell\to\infty}(f(\mathbf{x}^{k_\ell}) - z^{k_\ell}) = 0. \tag{4.7}$$

Proof: For each $\ell = 1, 2, \ldots$, we can assume without loss of generality that $\phi(\Delta^{k_\ell}) < \delta$ and

$$f(\mathbf{x}^{k_\ell}) > z^{k_\ell} \geq h^{k_\ell}(\mathbf{x}^{k_\ell}) = (\mathbf{d}^{k_\ell})^\mathsf{T}\mathbf{x}^{k_\ell} + d_{r+1}^{k_\ell}, \tag{4.8}$$

where h^{k_ℓ} represents the objective function of $\widetilde{\mathrm{Q}}_h(\Delta^{k_\ell})$. Let $\mathbf{u}^{k_\ell}$ denote the centroid of Δ^{k_ℓ} and $\mathbf{v}^{k_\ell}$ the vertex defining $d_{r+1}^{k_\ell}$ via (3.12). Then we have

$$\mathbf{d}^{k_\ell} = \nabla f(\mathbf{u}^{k_\ell}), \quad d_{r+1}^{k_\ell} = f(\mathbf{v}^{k_\ell}) - (\nabla f(\mathbf{u}^{k_\ell}))^\mathsf{T}\mathbf{v}^{k_\ell}.$$

Also let $\{\mathbf{v}'\} = \bigcap_{\ell=1}^{\infty}\Delta^{k_\ell}$. Then $\mathbf{u}^{k_\ell} \to \mathbf{v}'$ and $\mathbf{v}^{k_\ell} \to \mathbf{v}'$ as $\ell \to \infty$, because $\mathbf{u}^{k_\ell}$ and $\mathbf{v}^{k_\ell}$ are points of Δ^{k_ℓ}. Since f is assumed to be continuously differentiable, as $\ell \to \infty$ we have

$$\mathbf{d}^{k_\ell} \to \nabla f(\mathbf{v}'), \quad \mathbf{d}_{r+1}^{k_\ell} \to f(\mathbf{v}') - (\nabla f(\mathbf{v}'))^\mathsf{T}\mathbf{v}'.$$

Also, by taking a subsequence if necessary, we have $\mathbf{x}^{k_\ell} \to \mathbf{x}'$ for some $\mathbf{x}' \in X$ as $\ell \to \infty$, because $\mathbf{x}^{k_\ell}$'s are generated in the compact set X. Therefore, as $\ell \to \infty$ we have

$$h^{k_\ell}(\mathbf{x}^{k_\ell}) \to (\nabla f(\mathbf{v}'))^\mathsf{T}(\mathbf{x}' - \mathbf{v}') + f(\mathbf{v}') \geq f(\mathbf{x}'),$$

by noting the concavity of f. This, together with (4.8), implies (4.7). ■

The convergence of algorithm REVISED_SBB follows from this lemma.

Theorem 4.3 *Suppose $\epsilon = 0$. If algorithm REVISED_SBB terminates in finite time, $\mathbf{x}^*$ is a globally optimal solution to the master problem P. Even if not, every accumulation point of the sequence $\{\mathbf{x}^k \mid k = 1, 2, \ldots\}$ generated with the best-bound selection rule is a globally optimal solution to P.*

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

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

at the k_ℓth iteration. Note that $z(\Delta^1)$ is the optimal value of the master problem P and besides equals $\min\{z(\Delta^i) \mid i \in \mathcal{L}\}$. Therefore, we have

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

However, we see from Lemma 4.2 that $f(\mathbf{x}^{k_\ell}) - z^{k_\ell} \to 0$ as $\ell \to \infty$. This implies that $f(\mathbf{x}^{k_\ell}) \to z(\Delta^1)$ as $\ell \to \infty$. ■

Corollary 4.4 *When $\epsilon > 0$, algorithm REVISED_SBB with either of the selection rules, depth first or best bound, terminates after a finite number of iterations and yields a feasible solution $\mathbf{x}^*$ to the master problem P such that*

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) + \epsilon, \quad \forall \mathbf{x} \in X. \tag{4.9}$$

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

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

However, $f(\mathbf{x}^{k_\ell}) - z^{k_\ell} \to 0$ as $\ell \to \infty$, which contradicts the backtracking criterion. ■

If we adopt the other backtracking criterion $f(\mathbf{x}^*) - z^k \leq \epsilon |f(\mathbf{x}^*)|$, then (4.9) is replaced by

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) + \epsilon |f(\mathbf{x}^*)|, \quad \forall \mathbf{x} \in X,$$

but the corollary can be proved in the same way.

5 Numerical experiment

In this section, we present numerical results of having compared computer codes of REVISED_SBB and two existing algorithms, the one proposed in [11] and the standard simplicial branch-and-bound algorithm (see e.g., [7, 6, 19]). We refer to those codes here, as revsbb, sbb_1 and sbb_2, respectively. The test problem we solved is a concave quadratic minimization problem of the form:

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

where $\mathbf{A}' \in \mathbb{R}^{m' \times r'}$, $\mathbf{B}' \in \mathbb{R}^{m' \times (n'-r')}$, $\mathbf{b}' \in \mathbb{R}^{m'}$, $\mathbf{d} \in \mathbb{R}^{n'-r'}$, $\mathbf{C} \in \mathbb{R}^{r' \times r'}$ and σ is a positive weight. Along the lines of the experiment in [11], we generated $\mathbf{C} = [c_{ij}]$ so as to have two nonzero entries in each row, i.e., $c_{r'1}$, $c_{r'r'}$, c_{ii} and $c_{i,i+1}$ for $i = 1, \dots, r'-1$, where $c_{ii} = c_{r'r'} = 1, 0$ and the rest were drawn randomly from the uniform distribution on $[0.0, 1.0]$. Hence, $\mathbf{C}^\mathsf{T}\mathbf{C}$ has three nonzero entries at most in each row. Also, each component of $\mathbf{d}$ was a uniformly random number in the interval $[0.0, 1.0]$. 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', σ), we solved ten instances of (5.1) for each set using revsbb, sbb_1 and sbb_2 on a Linux workstation (Linux 2.4.21, Itanium2 processor 1.3GHz).

5.1 Computer codes

Each of the codes revsbb, sbb_1 and sbb_2 was written using GNU Octave (version 2.1.50) [16], a MATLAB-like computational tool, according to the depth-first rule. To adjust the form of (5.1) to (2.1), we introduced additional variables $\xi \in \mathbb{R}$, $\boldsymbol{\eta} \in \mathbb{R}^{m'}$ and applied the codes revsbb and sbb_1 to

$$\left|\begin{array}{lll} \text{minimize} & z = & -\sigma\xi - (1/2)\mathbf{x}^\mathsf{T}\mathbf{C}^\mathsf{T}\mathbf{C}^\mathsf{T}\mathbf{x} \\ \text{subject to} & & \mathbf{A}'\mathbf{x} + \mathbf{B}'\mathbf{y} + \boldsymbol{\eta} = \mathbf{b}', \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0} \\ & & \xi - \mathbf{d}^\mathsf{T}\mathbf{y} = \mathbf{0}, \qquad\quad (\xi, \boldsymbol{\eta}) \geq \mathbf{0}, \end{array}\right. \tag{5.2}$$

where we should note $\xi \geq 0$ because $\mathbf{d} \geq \mathbf{0}$. The size (m, n, r) of (5.2) is therefore equal to $(m' + 1, m' + n' + 1, r' + 1)$. As for sbb_2, we applied it directly to (5.1) because it uses only the relaxed problem $\mathrm{Q}(\Delta)$, which can be written with the slack variable $\boldsymbol{\eta}$ as follows [7, 6, 19]:

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

where $\mathbf{V}' = [\mathbf{v}_1, \dots, \mathbf{v}_{r'+1}]$ and $\mathbf{f}' = [f(\mathbf{v}_1), \dots, f(\mathbf{v}_{r'+1})]^{\mathsf{T}}$ for $r' + 1$ vertices $\mathbf{v}_j$'s of $\Delta \subset \mathbb{R}^{r'}$.

The backtracking criterion was $f(\mathbf{x}^*) - z^k \leq \epsilon|f(\mathbf{x}^*)|$ with $\epsilon = 10^{-5}$ in each code. As the subdivision rule of Δ, we adopted bisection of ratio $\alpha = 1/2$ in revsbb and sbb_1, but did not in sbb_2, because we found in our preliminary experiment that the convergence of sbb_2 with the bisection rule is too slow to compare with the other two codes. Instead, we took the way to bisect the longest edge of the minimal face of Δ which contains an optimal $\mathbf{x}^\circ = \mathbf{V}'\boldsymbol{\zeta}^\circ$ of $\mathrm{Q}(\Delta)$. Although this subdivision rule does not guarantee the convergence, sbb_2 using it terminated for every tested instance of (5.1) and generated the same output as revsbb and sbb_1 with the usual bisection rule.

5.2 Numerical results

In Figures 5.1–5.4, line plots are given for comparing the behavior of revsbb (solid lines with circle markers), sbb_1 (dotted lines with cross markers) and sbb_2 (dashed lines with triangle markers) when the size of constraint matrix $[\mathbf{A}', \mathbf{B}']$ was fixed at $(m', n') = (40, 80)$.

Figure 5.1 shows the variation in the average number of branching operations required by each code when σ was fixed at 5.0 and r' was changed in $\{16, 20, 24, 28, 30, 32, 34\}$. First, it is noteworthy that revsbb and sbb_1 took the same number of branching operations for each r'. Both codes incorporate a similar kind of bound tightening procedures based on Lagrangian relaxation. However, taking account of the relationship between the bounds $\widetilde{w}$ and $\overline{w}$ shown in Proposition 3.1, we can conclude that it is more effective in the code revsbb of REVISED_SBB. We also see that the tightening procedures work better for larger r', and in fact the dominance of the standard sbb_2 over revsbb and sbb_1 is reversed around $r' = 25$. The variations in the average CPU seconds are plotted in Figure 5.2. For every r', the code revsbb surpasses the other two codes. In particular, compared with sbb_2, it requires only fortieth part of the CPU seconds. This proves that problem (3.14) associated with $\widetilde{\mathrm{Q}}(\Delta)$ is easy enough to cancel out the inferiority of revsbb to sbb_2 in the number of branching operations for $r' < 25$.

Figures 5.3 and 5.4 show that the variations in the average number of branching operations and CPU seconds, respectively, required by each code when r' was fixed at 20 and σ was changed in $\{3.0, 3.5, 4.0, 5.0, 7.0, 10.0, 20.0\}$. Unfortunately, each code is rather sensitive to changes in σ, especially when $\sigma < 5$. Nevertheless, revsbb and sbb_1 need considerably less branching operations than the standard sbb_2 when $\sigma < 4$, which is totally due to the tight lower bound yielded by the Lagrangian relaxation. This,

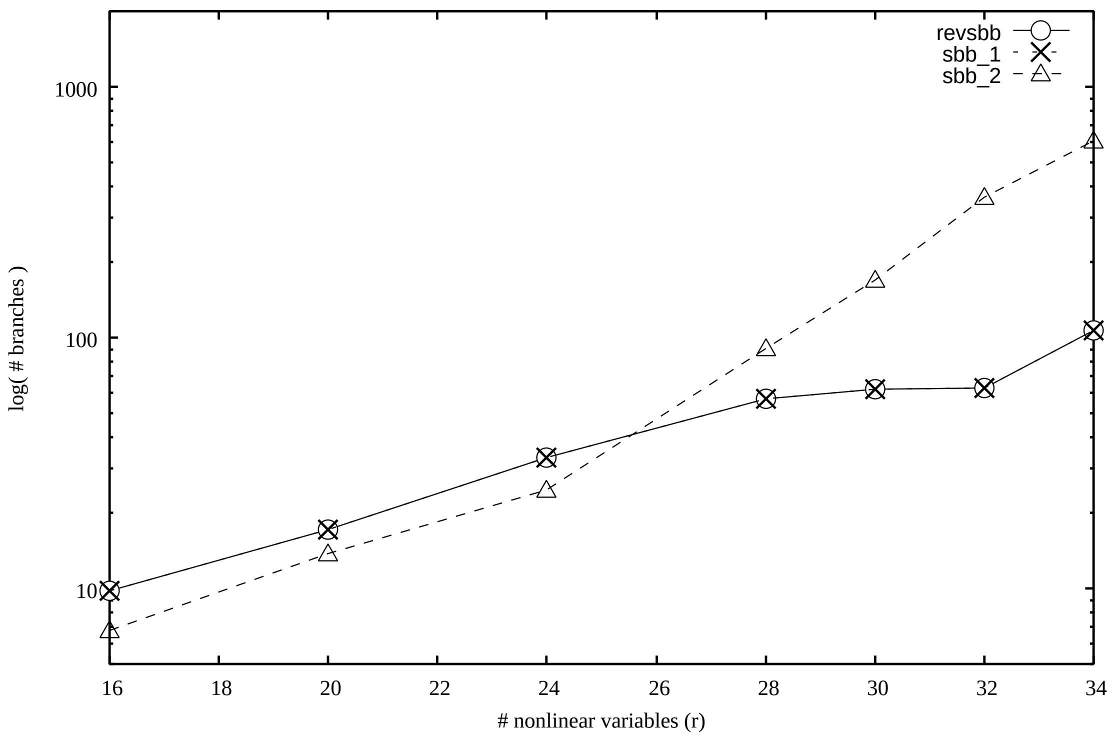

Figure 5.1: Numbers of branching operations when $(m', n', \sigma) = (40, 80, 5.0)$.

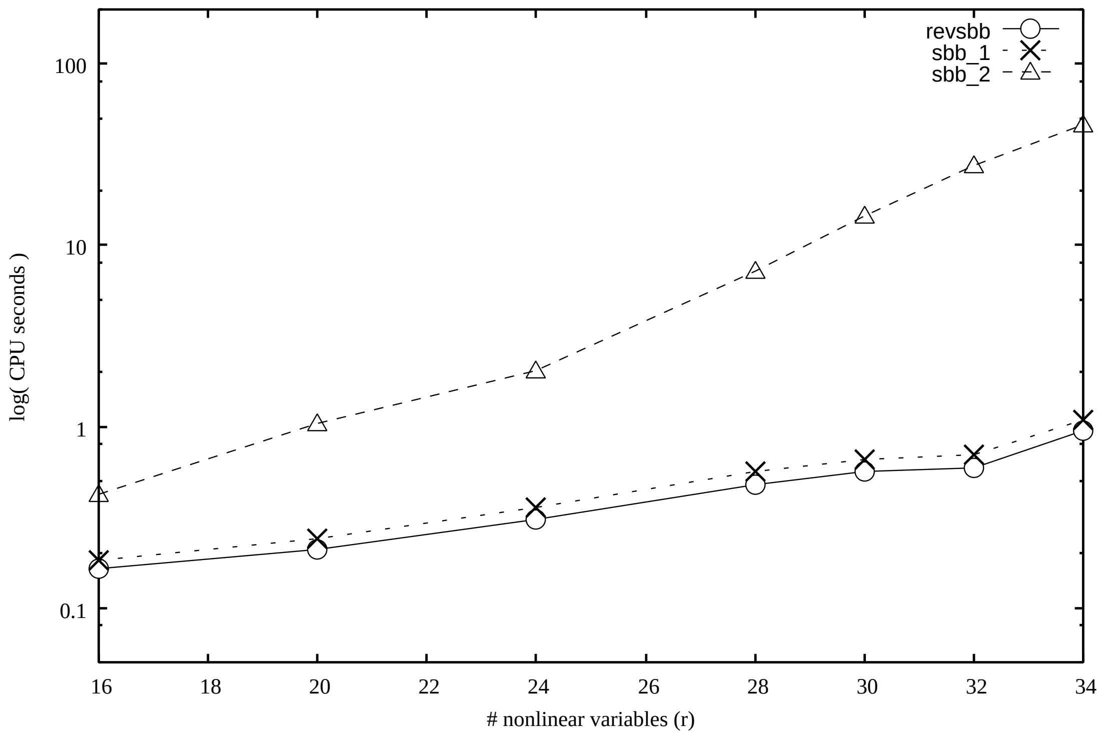

Figure 5.2: CPU seconds when $(m', n', \sigma) = (40, 80, 5.0)$.

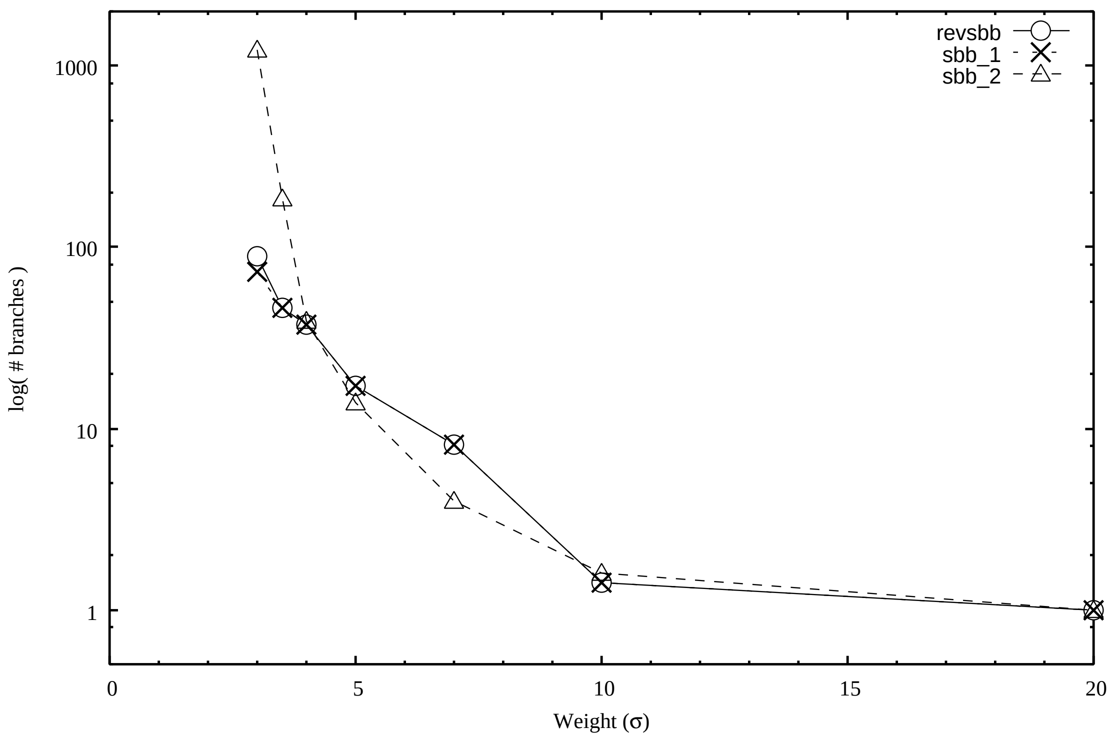

Figure 5.3: Numbers of branching operations when $(m', n', r') = (40, 80, 20)$.

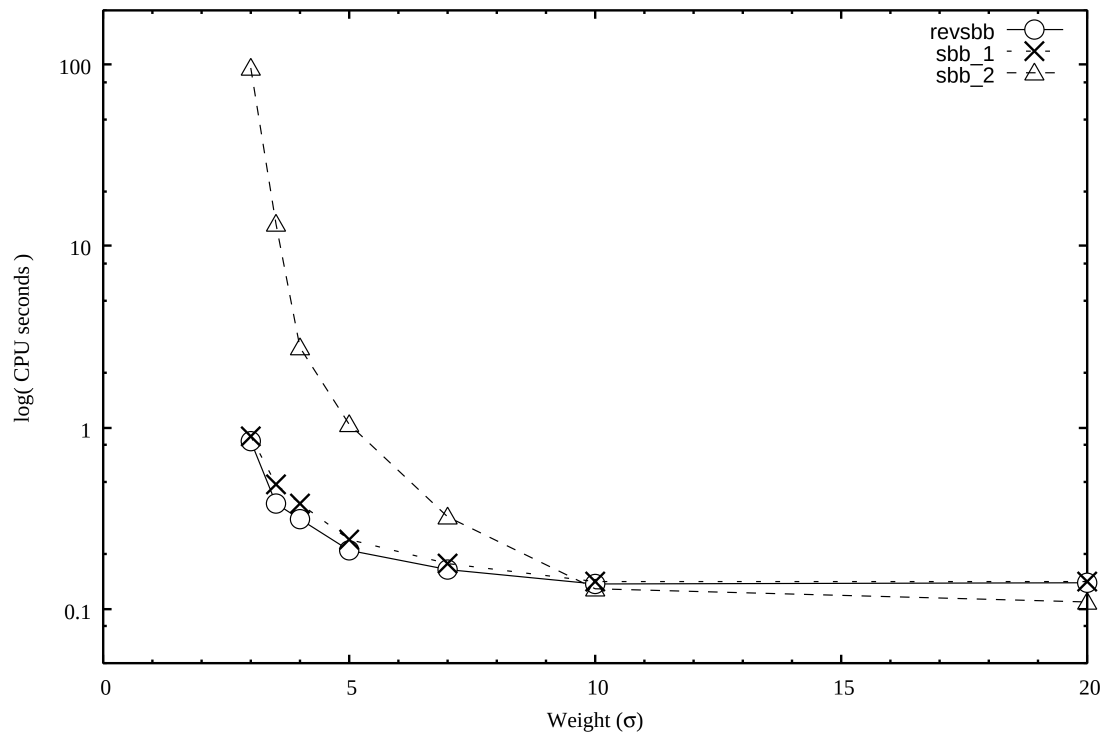

Figure 5.4: CPU seconds when $(m', n', r') = (40, 80, 20)$.

Table 5.1: Computational results of revsbb and sbb_1 when $\sigma = 5.0$.

		$r' = 0.2n'$		$r' = 0.3n'$		$r' = 0.4n'$		$r' = 0.5n'$	
$m' \times n'$		#	*time*	#	*time*	#	*time*	#	*time*
60×120	revsbb	18.2	0.445	79.9	1.082	103.6	2.430	230.9	9.081
	sbb_1	18.2	0.480	79.9	1.234	103.6	2.706	230.9	9.747
180×120	revsbb	21.6	2.418	58.8	3.486	111.3	6.034	199.2	13.06
	sbb_1	21.6	2.589	58.8	3.979	111.3	7.958	199.2	14.82
80×160	revsbb	37.6	1.102	97.0	2.865	128.2	7.850	256.6	27.83
	sbb_1	37.6	1.203	97.0	3.238	128.2	8.699	256.6	29.43
240×160	revsbb	12.6	7.026	38.6	8.892	83.2	14.83	151.2	32.01
	sbb_1	12.6	7.225	38.6	9.833	83.2	17.56	151.2	38.12
100×200	revsbb	45.6	2.180	88.4	5.245	115.4	15.77	227.6	63.60
	sbb_1	45.6	2.752	88.4	5.753	115.4	16.99	227.6	66.57
300×200	revsbb	9.6	19.06	47.4	24.16	110.8	39.09	236.4	105.2
	sbb_1	9.6	18.88	47.4	24.55	110.8	45.00	236.4	117.3

together with the ease of solution to (3.14), yields the significant advantage of revsbb against sbb_2 in computational time when $\sigma < 10$.

It would be clear from the above observation that revsbb and sbb_1 are of more promise than the standard sbb_2. To compare revsbb and sbb_1 in more detail, we next solved (5.1) of larger scale using those two codes. The size (m', n') ranged from $(60, 120)$ to $(300, 200)$ and σ was 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 the columns # and *time* show the average number of branching operations and CPU seconds, respectively, required by revsbb and sbb_1 for each (m', n', r'). Again, we notice that both codes took the same number of branching operations. Therefore, the difference between the CPU seconds of sbb_1 and revsbb directly reflects the difficulty of (3.6) and (3.14), associated with $\overline{\mathrm{Q}}(\Delta)$ and $\widetilde{\mathrm{Q}}(\Delta)$, repectively. Although the test problem (5.1) has no special structure, the computational time is improved by ten percent from sbb_1 to revsbb for each (m', n', r'). If we solve favorable structured problems, we can expect even more significant improvement. The number of branching operations and CPU seconds increase rather mildly as m' and n' increase, in contrast to the case of r'. We could solve still larger scale problems by elaborating the computer code of algorithm REVISED_SBB, as long as the number r' of nonlinear variables is less than half of n'.

References

[1] Ahuja, R.K., T.L. Magnanti and J.B. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice Hall (N.J,, 93).

[2] Avriel, M., W.E. Diewett, S. Schaible and I. Zang, *Generalized Concavity*, Plenum Press (N.Y., 88).

[3] Chvátal, V., *Linear Programming*, Freeman (N.Y., 1983).

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

[5] Horst, R., "An algorithm for nonconvex programming problems", *Mathematical Programming* **10** (1976), 312 – 321.

[6] Horst, R., P.M. Pardalos and N.Y. Thoai, *Introduction to Global Optimization*, 2nd ed., Kluwer Academic Publishers (Dordrecht, 200).

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

[8] Konno, H. and T. Kuno, "Linear multiplicative programming", *Mathematical Programming* **56** (1992), 51 – 64.

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

[10] Kuno, T., "A finite branch-and-bound algorithm for linear multiplicative programming", *Computational Optimization and Applications* **20** (2001), 119 – 135.

[11] Kuno, T. and H. Nagai, "A simplicial algorithm with two-phase bounding operation for a class of concave minimization problems", *Pacific Journal of Optimization* **1** (2005), 277–296.

[12] Kuno, T. and T. Utsunomiya, "A Lagrangian based branch-and-bound algorithm for production-transportation problems", *Journal of Global Optimization* **18** (2000), 59 – 73.

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

[14] Nagai, H. and T. Kuno, "A simplicial branch-and-bound algorithm for production-transportation problems with inseparable concave production cost", *Journal of the Operations Research Society of Japan* **48** (2005), 97–110.

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

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

[17] Rockafellar, R.T., *Convex Analysis*, Princeton (N.J., 1970).

[18] Ryoo, H.S. and N.V. Sahinidis, "Global optimization of multiplicative programs", *Journal of Global Optimization* **26** (2003), 387 – 418.

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

[20] Tuy, H., N.D. Dan and S. Ghannadan, "Strongly polynomial time algorithms for certain concave minimization problems on networks", *Operations Research Letters* **14** (1993), 99 – 109.

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