Deterministic Algorithms for Some Global Optimization Problems

Hidetoshi Nagai

March, 2006

Deterministic Algorithms for Some Global Optimization Problems

Hidetoshi Nagai

(Doctoral Program in Computer Science)

Adviser

Associate Professor Takahito Kuno

Submitted to the Graduate School of
Systems and Information Engineering
the University of Tsukuba
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in Engineering

March, 2006

Abstract

We study two major classes of problems in global optimization, concave minimization and reverse convex programming. The former problem is of minimizing a concave function under linear constraints and the latter problem is of minimizing a linear function over the intersection of a convex and the complement of a convex set. In general, both problems have enormous number of locally optimal solutions, among which the usual algorithms can fail to find a globally optimal one. To locate a globally optimal solution without fail in a practical amount of time, we propose branch-and-bound algorithms incorporating some procedures for accelerating convergence. The main idea behind the algorithms is to exploit special structures potentially possessed by the real-world problems, e.g., network structures and low-rank nonconvexity. We assume that each target problem has some favorable structures of these kinds and define relaxation problems needed to solve in the bounding process in such a way that they inherit the structures from the target problem. While this approach enables us to solve each relaxation problem efficiently, the resulting lower bounds on the optimal value become worse than those obtained by existing approaches and cause a rapid growth of the branching tree. To overcome this drawback, we introduce some bound-tightening procedures based on Lagrangian relaxation and surrogate relaxation. Although these procedures require us to solve a concave minimization problem, we show that the computational time is bounded by a lower order polynomial in the problem size. We also report numerical results for the proposed algorithms, which indicate that those are far more promising compared with textbook algorithms.

Contents

List of Figures

List of Tables

Notations

Throughout this thesis, we use the following notations.

$\text{int}(S)$	interior of the set S
$\text{conv}(S)$	convex hull of the set S
$\text{cone}(S)$	convex cone of the set S
∂S	boundary of the set S
$\nabla f(\mathbf{x})$	gradient vector of the function f at $\mathbf{x}$
$\lceil x \rceil$, $\lfloor x \rfloor$	integers obtained by rounding x up and down, respectively
$\mathbf{I}$	identity matrix of appropriate size
$\mathbf{e}$	all-ones vector of appropriate dimension
$\mathbf{e}_i$	the ith unit vector of appropriate dimenstion
$\mathbf{0}$	all-zeros vector of appropriate dimension

Chapter 1

Introduction

Since G.B. Dantzig developed the simplex method for linear programming problems in 1947, optimization algorithms have been widely used in engineering, economics and other sciences. At the same time, we have encountered an increasing number of problems which we cannot solve successfully using standard techniques for linear and nonlinear programming. These are nonconvex global optimization problems, whose distinguishing feature is multiextremality, i.e., the presence of multiple locally optimal solutions, many of which fail to be globally optimal. Until the mid 1980's, most researchers and practitioners believed that the best approach to these inherently difficult classes of problems is heuristic or stochastic local search (see e.g., [41]). However, the emergence of inexpensive personal computers and powerful workstations enabled us to solve small- to medium-scale global optimization problems in a practical amount of time, using general purpose deterministic algorithms. Those include, among others, outer approximation, cutting plane, branch-and-bound, inner approximation, or combinations of these different concepts (see [19] for details). They have been applied to some important classes of global optimization problems such as

1. concave minimization: minimizing a concave function under linear or convex constraints;

2. reverse convex programming; minimizing a linear function over the intersec-

tion of a convex set and the complement of a convex set; and

3. d.c. optimization: minimizing or maximizing a d.c. function (difference of two convex functions) under d.c. constraints.

Unfortunately, we often observe a rapid increase of computational time taken by the above-mentioned general purpose algorithms as the size of the instance increases. Therefore, it is still practically beyond our scope to solve even a concave minimization problem with over one hundred variables, if it has no special structures.

In this thesis, we specifically study concave minimization and reverse convex programming problems, both assumed to be encountered in such areas as chemical engineering, financial engineering, network optimization, production and inventory control, and so on. Most of these problems, though highly nonconvex, can be characterized by special structures such as network flow and/or low-rank nonconvexity, i.e., the property of becoming convex when a relatively few variables are fixed. By exploiting these structures, we develop practically efficient algorithms based on the branch-and-bound algorithm, originally proposed by H. Tuy [52], in order to generate a globally optimal solution to each target problem of larger-scale. To be precise, we define relaxation problems needed to solve in the bounding process in such a way that they inherit the structures from the target problem. While this approach enables us to solve relaxation problems efficiently, the resulting lower bounds on the optimal value become worse than those obtained by general-purpose algorithms and cause a rapid growth of the branching tree. To overcome this drawback, we introduce some bound-tightening procedures based on Lagrangian relaxation and surrogate relaxation. Although these procedures require us to solve a concave minimization problem, we show that the computational time is bounded by a lower order polynomial in the problem size. We also report numerical comparisons our proposed algorithms with general-purpose algorithms.

In Chapter 2, we will first overview the basic workings of the standard branch-and-bound algorithms described in [19, 55].

In Chapter 3, we will 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 the usual linear programming relaxation 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 the Lagrangian relaxation. Computational results indicate that the proposed algorithm is promising, compared with a standard simplicial branch-and-bound algorithm.

In Chapter 4, we will further develop a simplicial branch-and-bound algorithm for concave minimization problems with low-rank nonconvex structures from the proposed algorithm in Chapter 3. The preceding algorithm unfortunately have two difficulties. First, the lower bound obtained by linear programming relaxation problem is numerically unstable when the subsimplex becomes smaller. To prevent this issue, we introduce a new underestimator, which is simply induced by a gradient vector of the concave function f. Using this underestimator, we can solve the problems more precisely with a less numerical trouble than the convex envelope of the standard underestimator. Second, although our proposed relaxation in Chapter 3 inherits special structures of the target problem to a great extent, it destroys the original low-rank nonconvex problem behind the target problem. We propose to remove all additional constraints imposed on the usual linear programming relaxation problem. Therefore, in the bounding operation, we solve a linear programming problem whose constraints are exactly the same as the target problem. Although the lower bound worsens by this enlargement of the feasible set, 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.

In Chapter 5, we will consider a concave production-transportation problem for an actual example of applying our algorithm to a concave minimization problem with special structures. This problem is a network flow problem of optimizing

production and transportation simultaneously. The production cost is assumed to be a concave function in light of scale economy. The proposed algorithm generates a globally optimal solution to this nonconvex minimization problem in finite time, without assuming the separability of the production-cost function unlike existing algorithms. We also report some computational results, which indicate that the algorithm is fairly promising for practical use.

On the other hand, the purpose of Chapter 6 is to develop a conical branch-and-bound algorithm for solving reverse convex programming problems. We also assume that the problems have low-rank nonconvexity. We propose an inexpensive bound-tightening procedure, which is based on the surrogate relaxation. This is almost the same as the procedure mentioned in Chapter 3 and 4, which is using Lagrangian relaxation. We show that this procedure considerably tightens the lower bounds yielded by the usual linear programming relaxation. We also report numerical results, which indicate that the proposed algorithm is much promising, compared with existing algorithms.

The proposed algorithms from Chapter 3 to 6 have been published respectively in [28], [29], [40], [39]. Finally, concluding remarks of the thesis will be discussed in Chapter 7.

Chapter 2

Overview of the Branch-and-Bound Algorithm

In this chapter, we will review the basic works of the standard simplicial branch-and-bound algorithm and the standard conical branch-and-bound algorithm described in [19, 55] on concave minimization problems and reverse convex programming problems, respectively. The algorithms will be adapted for applying the low-rank nonconvexity.

2.1 Simplicial Algorithm for Concave Minimization

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

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

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 feasible set and its projection onto the subspace $\mathbb{R}^r$, respectively, by

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

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

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

We assume that W is nonempty and bounded. The same is then true for the projection X; and so we have

$$v := \max\{\, \mathbf{e}^{\mathsf{T}}\mathbf{x} \mid \mathbf{x} \in X \,\} \tag{2.2}$$

is finite, where $\mathbf{e} \in \mathbb{R}^r$ is the all-ones vector. We also assume that the domain of f is large enough to include the 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 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^k, $k \in \mathcal{H}$, such that

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

where $\mathrm{int}(\,\cdot\,)$ represents the set of interior points. We solve subproblems of the master problem P one after another; the feasible set of each subproblem is restricted by Δ^k, and we need to solve the following with $\Delta = \Delta^k$ for every $k \in \mathcal{H}$:

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

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

procedure Standard_SBB

Let $\mathcal{H} := \{1\}$. Set the incumbent $\mathbf{x}^*$ by a vertex of X.

Repeat Steps 1–3 until $\mathcal{H} = \emptyset$.

Step 1 (subproblem selection). Take an appropriate index k out of $\mathcal{H}$, set $\mathcal{H} := \mathcal{H} \setminus \{k\}$, and let $\Delta := \Delta^k$.

Step 2 (bounding operation). Compute a lower bound z^k on the optimal value $z(\Delta)$ of $\mathrm{P}(\Delta)$. If we obtain a feasible solution $\mathbf{x}^k$ to P such that $f(\mathbf{x}^k) < f(\mathbf{x}^*)$, then update $\mathbf{x}^* := \mathbf{x}^k$. If $f(\mathbf{x}^*) \leq z^k$, then go back to Step 1.

Step 3 (branching operation). Otherwise, divide the simplex Δ into two subsimplices Δ^{2k} and Δ^{2k+1}, and set $\mathcal{H} := \mathcal{H} \cup \{2k, 2k+1\}$.

If $X \cap \Delta = \emptyset$, then $z(\Delta)$ is $+\infty$. When $\mathcal{H}$ 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 algorithm can generate an infinite sequence of simplices $\{\, \Delta^{k_\ell} \mid \ell = 1, 2, \dots \}$ such that

$$\Delta^{k_1} \supset \Delta^{k_2} \supset \cdots, \quad X \cap \left(\bigcap_{\ell=1}^{\infty} \Delta^{k_\ell} \right) \neq \emptyset. \tag{2.3}$$

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

$$f(\mathbf{x}^*) - \epsilon \leq z^k \quad \text{or} \quad f(\mathbf{x}^*) - \epsilon |f(\mathbf{x}^*)| \leq z^k, \tag{2.4}$$

and besides to subdivide Δ^1 in an exhaustive manner that makes $\bigcap_{\ell=1}^{\infty} \Delta^{k_\ell}$ a singleton. The simplest exhaustive subdivision rule is *bisection*: we 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 $\Delta = \mathrm{conv}(\{\mathbf{v}_1, \dots, \mathbf{v}_{r+1}\})$ 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

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

Note that the initial simplex Δ^1 has vertices $\mathbf{0}$ and $v\mathbf{e}_1, \ldots, v\mathbf{e}_r$ for v in (2.2), where $\mathbf{e}_i \in \mathbb{R}^r$ is the ith unit vector. Thus, starting from $\Delta^1 = \mathrm{conv}(\{\mathbf{0}, v\mathbf{e}_1, \ldots, v\mathbf{e}_r\})$, we can generate Δ^k for all $k \in \mathcal{H}$. If we adopt the bisection rule and (2.4) as the branching criterion with a tolerance $\epsilon > 0$, we can obtain a globally ϵ-optimal solution to P after a finite number of steps, using either of the following usual selection rules at Step 1:

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

Best bound. The set $\mathcal{H}$ is maintained as a list of *priority queue*. Let w^k be the key value for $k \in \mathcal{H}$, where w^1 is given by an appropriate value, and w^{2k}, w^{2k+1} are set by z^k at Step 3. An index k of least w^k is taken out of $\mathcal{H}$; and $2k$, $2k+1$ are put back to $\mathcal{H}$ at Step 3.

The most time-consuming step in the simplicial branch-and-bound algorithm is the bounding operation of Step 2. In Chapters 3 and 4, we will discuss some issues with Step 2 faced by existing algorithms and their resolution in treating our target problem (2.1).

2.2 Conical Algorithm for Reverse Convex Programming

The reverse convex programming problem we consider in this thesis is the following:

$$\left|\begin{array}{lll} \text{minimize} & z = \mathbf{c}^{\mathsf{T}}\mathbf{x} & \\ \text{subject to} & \mathbf{A}\mathbf{x} + \mathbf{D}\mathbf{y} = \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0} & \\ & g(\mathbf{x}) \geq 0, & \end{array}\right. \tag{2.5}$$

where $\mathbf{A} \in \mathbb{R}^{m\times r}$, $\mathbf{D} \in \mathbb{R}^{m\times(n-r)}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^r$, and $g : \mathbb{R}^r \to \mathbb{R}$ is a convex function. Let

$$F = \{\, \mathbf{x} \in \mathbb{R}^r \mid \exists \mathbf{y} \geq \mathbf{0},\, \mathbf{A}\mathbf{x} + \mathbf{D}\mathbf{y} = \mathbf{b},\, \mathbf{x} \geq \mathbf{0} \,\}$$
$$G = \{\, \mathbf{x} \in \mathbb{R}^r \mid g(\mathbf{x}) < 0 \,\},$$

and assume that both F and G are bounded and have interior points. Then (2.5) is embedded in the $\mathbf{x}$-space as:

$$\mathrm{P} \left| \begin{array}{ll} \text{minimize} & z = \mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & \mathbf{x} \in F \setminus G. \end{array} \right.$$

We assume that G contains at least one optimal solution $\mathbf{x}^\circ$ to the associated linear program $\min\{\, \mathbf{c}^{\mathsf{T}}\mathbf{x} \mid \mathbf{x} \in F \,\}$. This condition makes (2.5) nontrivial. For simplicity, we assume $\mathbf{x}^\circ = \mathbf{0}$ in the sequel.

The standard conical branch-and-bound algorithm on (2.5), or equivalently of P, is much the same as the simplicial algorithm described in the preceding section. It is obtained as a consequence of just replacing the simplices with cones.

Let $\Delta^1 = \{\, \mathbf{x} \in \mathbb{R}^r \mid \mathbf{x} \geq \mathbf{0} \,\}$. Then Δ^1 is a cone vertexed at $\mathbf{x}^\circ = \mathbf{0}$ and includes the polytope F. Starting from this cone Δ^1, we recursively divide it into subcones Δ^k, each vertexed at $\mathbf{x}^\circ$, satisfying

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

where $\mathcal{H}$ denotes the set of indices of subdivided cones. This procedure generates an infinite nested sequence of cones $\{\, \Delta^{k_\ell} \mid \Delta^{k_\ell} \supset \Delta^{k_{\ell+1}},\, \ell = 1, 2, \dots \,\}$. To guarantee the convergence of the algorithm, we need to subdivide Δ^1 in such an exhaustive manner that $\bigcap_{\ell=1}^{\infty} \Delta^{k_\ell}$ becomes a half line emanating from $\mathbf{x}^\circ$. Suppose that Δ^k is spanned by r linearly independent vectors $\mathbf{w}_i \in \mathbb{R}^r$, $i = 1, \dots, r$, and denote $\Delta^k = \mathrm{cone}(\{\mathbf{w}_1, \dots, \mathbf{w}_r\})$. The easiest exhaustive subdivision rule is *bisection*, i.e., we divide the longest edge of the associated base simplex $\mathrm{conv}(\{\mathbf{w}_1, \dots, \mathbf{w}_r\})$ of Δ^k, say $\mathbf{w}_p$–$\mathbf{w}_q$, at a fixed ratio of $\alpha \in (0, 1/2]$. Letting $\mathbf{w} = (1-\alpha)\mathbf{w}_p + \alpha\mathbf{w}_q$,

then we have

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

For each subcone $\Delta = \Delta^k$, $k \in \mathcal{H}$, we have a subproblem of P:

$$\mathrm{P}(\Delta) \left| \begin{array}{ll} \text{minimize} & z = \mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & \mathbf{x} \in (F \setminus G) \cap \Delta. \end{array} \right.$$

This problem is essentially the same as (2.5) and cannot be solved directly. Therefore, subproblems are recursively processed in almost the same three steps as the previous section:

procedure Standard_CBB

Let $\mathcal{H} := \{1\}$. Set the incumbent value z^* by $+\infty$.

Repeat Steps 1–3 until $\mathcal{H} = \emptyset$.

Step 1 (subproblem selection). Take an appropriate index k out of $\mathcal{H}$, set $\mathcal{H} := \mathcal{H} \setminus \{k\}$, and let $\Delta := \Delta^k$.

Step 2 (bounding operation). Compute a lower bound z^k on the optimal value $z(\Delta)$ of $\mathrm{P}(\Delta)$. If we obtain a feasible solution $\mathbf{x}^k$ to P such that $\mathbf{c}^{\mathsf{T}}\mathbf{x}^k < z^*$, then update $z^* := \mathbf{c}^{\mathsf{T}}\mathbf{x}^k$ and $\mathbf{x}^* := \mathbf{x}^k$. If $z^* \leq z^k$, or $g(\mathbf{x}^k) + \epsilon \geq 0$ for a solution $\mathbf{x}^k \in F$ satisfying $\mathbf{c}^{\mathsf{T}}\mathbf{x}^k = z^k$, then go back to Step 1.

Step 3 (branching operation). Otherwise, divide the cone Δ into two subcones Δ^{2k} and Δ^{2k+1}, and set $\mathcal{H} := \mathcal{H} \cup \{2k, 2k+1\}$.

Chapter 3

Simplicial Algorithm for Concave Minimization Problems

3.1 Introduction

In this chapter, we develop a branch-and-bound algorithm for solving a class of concave minimization problems to which many of problems with low-rank nonconvexity [24] reduce. The major feature of this class is that the variables involved in the objective function form a small fraction of the all variables. A typical example would be the linear multiplicative program [23, 27, 33, 46]:

$$\left| \begin{array}{ll} \text{minimize} & \displaystyle\prod_{j=1}^{r} (\mathbf{c}_j^{\mathsf{T}}\mathbf{y} + c_{j0}) \\ \text{subject to} & \mathbf{B}\mathbf{y} \leq \mathbf{b}, \quad \mathbf{y} \geq \mathbf{0}, \end{array} \right. \tag{3.1}$$

where $\mathbf{B} \in \mathbb{R}^{m' \times n'}$, $\mathbf{b} \in \mathbb{R}^{m'}$, $\mathbf{c}_j \in \mathbb{R}^{n'}$, $\mathbf{c}_{j0} \in \mathbb{R}$. We assume $\mathbf{c}_j^{\mathsf{T}}\mathbf{y} + c_{j0} > 0$ for any feasible solution $\mathbf{y}$. In general, the number r of affine functions of the objective function is assumed to be far less than the number of variables of (3.1). If we introduce a vector $\mathbf{x} = (x_1, \ldots, x_r)^{\mathsf{T}}$ of auxiliary variables, (3.1) reduces to

a concave minimization problem which has the low-rank nonconvexity:

$$\left|\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{r} \log(x_j) \\ \text{subject to} & -x_j + \mathbf{c}_j^{\mathsf{T}}\mathbf{y} \leq -c_{j0}, \quad j = 1, \ldots, r \\ & \mathbf{B}\mathbf{y} \leq \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}. \end{array}\right. \tag{3.2}$$

Another example is the production-transportation problem (see Chapter 5 and [32, 40, 56, 58]). This is a problem of minimizing the sum of concave production and linear transportation costs under the network flow constraints. If we move the transportation cost function to the set of constraints by means of an auxiliary variable, only the concave production cost and auxiliary variable are left in the objective function. For various other examples, the readers are referred to the textbook on low-rank nonconvex structures [24].

If the objective function is separated into a sum of univariate functions like the objective function of (3.2), problems of this class can be solved rather efficiently using the rectangular branch-and-bound algorithm [7, 27, 32]. To deal with a wider range of problems, we do not assume the separability of the objective function in this chapter. Thus, we tailor the simplicial branch-and-bound algorithm [15] to this class and apply some procedures for improving the efficiency. In Section 3.2, after giving the problem settings, we will bring up difficulties in implementation of the simplicial branch-and-bound algorithm. In Section 3.3, to overcome those difficulties, we will modify the linear programming relaxation problem to be solved in the bounding operation, by enlarging the feasible set. This modification does not affect the convergence property of the algorithm. However, it deteriorates the quality of the lower bound on the optimal value, which causes the rapid growth of the branching tree. To prevent it, we will propose the second bounding operation based on a Lagrangian relaxation in Section 3.4, and give a detailed description of the algorithm incorporating two bounding operations. Computational results to compare with the standard simplicial branch-and-bound algorithm are reported in Section 3.5. Lastly, we will discuss some remaining issues to be resolved in the future, in Section 3.6.

3.2 Problem Settings and the Simplicial Algorithm

3.2.1 Problem settings

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

$$\left| \begin{array}{ll} \text{minimize} & z = f(\mathbf{x}) \\ \text{subject to} & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array} \right. \tag{3.3}$$

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 $1 \leq r \leq n$. Letting X be the projection of the feasible set onto the space of $\mathbf{x}$, (3.3) can be embedded in $\mathbb{R}^r$:

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

We assume that X is bounded and has a nonempty interior. Then, we see that $v := \max\{ \sum_{j=1}^{r} x_j \mid \mathbf{x} \in X \}$ is finite. We also 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 [15, 19, 55], Standard_SBB described in Section 2.1.

3.2.2 Linear programming relaxation at Step 2 of Standard_SBB

The most time-consuming step in Standard_SBB 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 on the optimal value $z(\Delta)$ of

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

we usually replace the objective function f of $\mathrm{P}(\Delta)$ by its convex envelope g on Δ and solve a problem:

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

The convex envelope g is an affine function which agrees with f at $r+1$ vertices $\mathbf{v}_1, \ldots, \mathbf{v}_{r+1}$ 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 $\mathbf{v}_i$, $i = 1, \ldots, r+1$:

$$\mathbf{x} = \sum_{i=1}^{r+1} \mathbf{v}_i \xi_i, \quad \sum_{i=1}^{r+1} \xi_i = 1, \quad \boldsymbol{\xi} = (\xi_1, \ldots, \xi_{r+1})^{\mathsf{T}} \geq \mathbf{0}. \tag{3.4}$$

By the concavity of f, we immediately have

$$g(\mathbf{x}) = \sum_{i=1}^{r+1} f(\mathbf{v}_i)\xi_i \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in \Delta. \tag{3.5}$$

Substituting (3.4) into $\overline{\mathrm{P}}(\Delta)$, we see that $\overline{\mathrm{P}}(\Delta)$ is equivalent to a linear program of $n+1$ variables:

$$\left| \begin{array}{lll} \text{minimize} & z = \mathbf{f}^{\mathsf{T}}\boldsymbol{\xi} & \\ \text{subject to} & \mathbf{AV}\boldsymbol{\xi} + \mathbf{By} \leq \mathbf{b} & \\ & \mathbf{e}^{\mathsf{T}}\boldsymbol{\xi} = 1, \quad (\boldsymbol{\xi}, \mathbf{y}) \geq \mathbf{0}, & \end{array} \right. \tag{3.6}$$

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

$$\mathbf{f} = [f(\mathbf{v}_1), \ldots, f(\mathbf{v}_{r+1})]^{\mathsf{T}}, \quad \mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_{r+1}]. \tag{3.7}$$

Obviously, (3.6) has an optimal solution $(\overline{\boldsymbol{\xi}}, \overline{\mathbf{y}})$ if and only if $X \cap \Delta \neq \emptyset$. Then we get the following candidate $\overline{z}$ for the lower bound z^k:

$$\overline{z} = \begin{cases} \mathbf{f}^{\mathsf{T}}\overline{\boldsymbol{\xi}} & \text{if } X \cap \Delta \neq \emptyset \\ +\infty & \text{otherwise.} \end{cases} \tag{3.8}$$

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

Certainly, the linearized subproblem $\overline{\mathrm{P}}(\Delta)$ is by far easier to solve than $\mathrm{P}(\Delta)$. However, since the number of subproblems generated in the course of iterating Steps 1–3 can be 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 $\overline{\mathrm{P}}(\Delta)$ from scratch for each Δ^k. In the rectangular and combinatorial branch-and-bound algorithms, one can solve linearized subproblems successively using sensitivity analysis of the simplex method, or using algorithms specialized for some favorable structure of the original problem (3.3) if possible. Unfortunately, such a procedure does not work well on $\overline{\mathrm{P}}(\Delta)$ because

(a) when Δ changes, (3.6) associated with $\overline{\mathrm{P}}(\Delta)$ has a different constraint matrix,

(b) (3.6) does not inherit the structure of the original problem (3.3).

Due to (a), we cannot utilize the optimal solution to $\overline{\mathrm{P}}(\Delta^k)$ as the initial solution in solving $\overline{\mathrm{P}}(\Delta^{2k})$, (or $\overline{\mathrm{P}}(\Delta^{2k+1})$), through sensitivity analysis, because it might be neither primal feasible nor dual feasible for (3.6) associated with $\overline{\mathrm{P}}(\Delta^{2k})$. Moreover, due to (b), even if the original problem (3.3) has a nice structure, for instance, a network flow structure, we cannot apply any network flow algorithms to solve $\overline{\mathrm{P}}(\Delta)$. To overcome these difficulties, we need to add some new twists to the relaxation of subproblem $\mathrm{P}(\Delta)$ at Step 2.

3.3 Modified Linear Programming Relaxation

3.3.1 Enlargement of the feasible set

One way of removing 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 $\mathbf{s}$ and $\mathbf{t}$ be r-dimensional vectors whose jth components are given by

$$\begin{aligned} s_j &= \min\{\, v_{ij} \mid i = 1, \ldots, r+1 \,\}, \\ t_j &= \max\{\, v_{ij} \mid i = 1, \ldots, r+1 \,\}, \end{aligned} \tag{3.9}$$

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

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

Then we have $\Delta \subset \Gamma(\Delta)$. Instead of $\overline{\mathrm{P}}(\Delta)$, we propose to solve the following for a lower bound z^k at Step 2:

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

It should be noted that the objective function g is given as $g(\mathbf{x}) = \sum_{i=1}^{r+1} f(\mathbf{v}_i)\xi_i$, meaning that the constraint (3.4) should be contained in $\widetilde{P}(\Delta)$. This still offers 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 (3.7) 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 (3.7) 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 in variables $(\mathbf{x}, \mathbf{y})$ only:

$$\left| \begin{array}{ll} \text{minimize} & z = \mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \quad \mathbf{s} \leq \mathbf{x} \leq \mathbf{t}, \quad \mathbf{y} \geq \mathbf{0}. \end{array} \right. \tag{3.10}$$

If $X \cap \Gamma(\Delta) \neq \emptyset$, then (3.10) has an optimal solution $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})$. The following can be a candidate for the lower bound z^k:

$$\widetilde{z} = \begin{cases} \mathbf{c}^{\mathsf{T}}\widetilde{\mathbf{x}} + c_{r+1} & \text{if } X \cap \Gamma(\Delta) \neq \emptyset \\ +\infty & \text{otherwise.} \end{cases} \tag{3.11}$$

Lemma 3.1. *The optimal value $z(\Delta)$ of $P(\Delta)$, $\overline{z}$ in (3.8) and $\widetilde{z}$ in (3.11) satisfy the inequality*

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

Proof. Immediately follows from (3.5) and the inclusion relation of the feasible sets of $\widetilde{\mathrm{P}}(\Delta)$ and $\overline{\mathrm{P}}(\Delta)$. □

It should be emphasized that (3.10) has the same set of constraints as the original (3.3), 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 in almost the same way as the usual nonnegative constraint in the simplex and network flow algorithms [1, 5]. Therefore, when (3.3) has some favorable structure, we can exploit it in solving (3.10). In general, we can generate an optimal solution $(\widetilde{\mathbf{x}}^{2k}, \widetilde{\mathbf{y}}^{2k})$ to (3.10) associated with $\widetilde{\mathrm{P}}(\Delta^{2k})$ from the preceding $(\widetilde{\mathbf{x}}^k, \widetilde{\mathbf{y}}^k)$ in two steps: (i) restore the feasibility of $(\widetilde{\mathbf{x}}^k, \widetilde{\mathbf{y}}^k)$ for $\widetilde{\mathrm{P}}(\Delta^{2k})$ with dual pivoting operations, and (ii) reestablish the optimality of the resulting feasible basic solution with primal pivoting operations. Since $(\widetilde{\mathbf{x}}^k, \widetilde{\mathbf{y}}^k)$ violates only the bounding constraint, step (i) should require a very few pivoting operations. If step (i) fails, both $\widetilde{\mathrm{P}}(\Delta^{2k})$ and $\mathrm{P}(\Delta^{2k})$ are infeasible.

3.3.2 Convergence property when using $\widetilde{\mathrm{P}}(\Delta)$

The optimal solution $\widetilde{\mathbf{x}}$ to $\widetilde{\mathrm{P}}(\Delta)$ obtained by solving (3.10) is obviously feasible for the target problem P; and hence we can update the incumbent $\mathbf{x}^*$ with $\widetilde{\mathbf{x}}$ when $f(\widetilde{\mathbf{x}}) < f(\mathbf{x}^*)$. However, $\widetilde{\mathbf{x}}$ might be infeasible for $\mathrm{P}(\Delta)$ and can satisfy

$$f(\widetilde{\mathbf{x}}) < \widetilde{z}, \tag{3.12}$$

unlike the optimal solution $\overline{\mathbf{x}}$ of $\overline{\mathrm{P}}(\Delta)$. If (3.12) 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

$$f(\widetilde{\mathbf{x}}) < f(\mathbf{v}_i), \quad \forall i = 1, \ldots, r+1. \tag{3.13}$$

Recall that f is concave and achieves the minimum on Δ at some vertex. Therefore, the minimum of $f(\mathbf{v}_i)$'s is another lower bound on the value $z(\Delta)$ of $\mathrm{P}(\Delta)$. We propose to compute the lower bound z^k at Step 2 of Standard_SBB by

$$z^k = \max\{\tilde{z}, \min\{f(\mathbf{v}_1), \ldots, f(\mathbf{v}_{r+1})\}\}. \tag{3.14}$$

The essential reason for introducing (3.14) is merely to guarantee convergence of the algorithm. If neither (3.12) nor (3.13) holds, the simplicial branch-and-bound algorithm might generate an infinite sequence of nested simplices $\{ \Delta^{k_\ell} \mid \ell = 1, 2, \ldots \}$ when a tolerance ϵ in the branching criterion is zero. Even in that case, we see that z^{k_ℓ} defined in (3.14) 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 a positive tolerance ϵ is introduced, it follows from this property that $f(\mathbf{x}^*) - z^k \leq \epsilon$ holds for some $k = k_\ell$.

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*

$$\lim_{\ell \to \infty} \left(f(\widetilde{\mathbf{x}}^{k_\ell}) - z^{k_\ell} \right) = 0. \tag{3.15}$$

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

$$f(\mathbf{v}^{k_\ell}) \leq z^{k_\ell} < f(\widetilde{\mathbf{x}}^{k_\ell}), \tag{3.16}$$

where $\mathbf{v}^{k_\ell}$ is a vertex of Δ^{k_ℓ} minimizing f. Since the bisection makes an exhaustive sequence of simplices, we have $\{\mathbf{v}\} = \bigcap_{\ell=1}^{\infty} \Delta^{k_\ell}$. Then we see from the definition of $\Gamma(\Delta^{k_\ell})$ that $\bigcap_{\ell=1}^{\infty} \Gamma(\Delta^{k_\ell}) = \{\mathbf{v}\}$. 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 (3.16), implies (3.15). □

3.3.3 Some issues to resolve in relaxation $\widetilde{\mathrm{P}}(\Delta)$

Lemma 3.2 guarantees that Standard_SBB works if we use the relaxation $\widetilde{\mathrm{P}}(\Delta)$ instead of $\overline{\mathrm{P}}(\Delta)$ in the bounding operation of Step 2. There still remain two issues to resolve.

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 (3.10). 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 of Standard_SBB. Because most $\mathbf{U}$'s are different from their predecessors in only one column, we can obtain the inverse of the $2k$th matrix $\mathbf{U}^{2k}$ from that of $\mathbf{U}^k$ almost always in time $O(r)$ using the rank-one update [5]. Suppose that $\mathbf{U}^k$ and $\mathbf{U}^{2k}$ are the same except for the pth column. Let $\mathbf{u}_p^{2k}=[(\mathbf{v}_p^{2k})^{\mathsf{T}},1]^{\mathsf{T}}$ denote the pth column of $\mathbf{U}^{2k}$ and let

$$\mathbf{w}=(\mathbf{U}^k)^{-1}\mathbf{u}_p^{2k},\quad \mathbf{E}=\mathbf{I}+(\mathbf{e}_p-\mathbf{w})\mathbf{e}_p^{\mathsf{T}}/w_p,$$

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

$$(\mathbf{U}^{2k})^{-1}=\mathbf{E}(\mathbf{U}^k)^{-1}. \tag{3.17}$$

Note that $\mathbf{E}$ is an "eta matrix" with nonzero off-diagonal elements only in the pth column. Since $\mathbf{v}_p^{2k}$ is a convex combination $(1-\alpha)\mathbf{v}_p^k+\alpha\mathbf{v}_q^k$ of two vertices of Δ^k, we have $w_p=1-\alpha$, $w_q=\alpha$ and other components of $\mathbf{w}$ are zeros. As a result, the pth column of $\mathbf{E}$ has only two nonzero entries $1/(1-\alpha)$ and $-\alpha/(1-\alpha)$ in the pth and qth rows, respectively. We see from (3.17) that the inverse of $\mathbf{U}^{2k}$ is yielded if we replace only the pth and qth rows of $(\mathbf{U}^k)^{-1}$ by their affine combinations.

The second issue is much more serious. Although the lower bound $\widetilde{z}$ yielded by $\widetilde{\mathrm{P}}(\Delta)$ is somewhat tightened to z^k by $f(\mathbf{v}_i)$'s in (3.14), as shown in Lemma 3.1, $\widetilde{z}$ is not better than $\overline{z}$; and the difference is not expected to be small. 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 another procedure for tightening $\widetilde{z}$.

3.4 Algorithm Using Two-Phase Bounding Operation

3.4.1 Lagrangian relaxation and its solution

For $\widetilde{z}$ yielded by solving $\widetilde{\mathrm{P}}(\Delta)$, let $G = \{\, \mathbf{x} \in \mathbb{R}^r \mid g(\mathbf{x}) \geq \widetilde{z} \,\}$. Since $X \cap \Delta$ is a subset of this half space G, no feasible solution to $\mathrm{P}(\Delta)$ is overlooked even if we add the constraint $\mathbf{x} \in G$ to $\mathrm{P}(\Delta)$. The resulting problem is the following:

$$\left| \begin{array}{ll} \text{minimize} & z = f(\mathbf{x}) \\ \text{subject to} & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0} \\ & \mathbf{x} \in \Delta, \quad \mathbf{c}^{\mathsf{T}}\mathbf{x} \geq \widetilde{z} - c_{r+1}. \end{array} \right. \tag{3.18}$$

In the preceding section, we have relaxed the objective function f and the constraint $\mathbf{x} \in \Delta$. Instead, we relax $\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}$ here, by introducing a Lagrangian multiplier $\mathbf{0} \leq \boldsymbol{\lambda} \in \mathbb{R}^m$. Then we have

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

Note that $\mathbf{x} \geq \mathbf{0}$ 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; $\mathrm{P}(\Delta)$ is infeasible, then we can discard Δ.

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. [42]), we have

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

However, for $\mathrm{L}(\Delta; \boldsymbol{\lambda})$ to provide a lower bound better than $\widetilde{z}$, the multiplier $\boldsymbol{\lambda}$ should be appropriately chosen so that $z(\boldsymbol{\lambda}) > \widetilde{z}$ holds.

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 $\widetilde{\mathrm{P}}(\Delta)$. Replacing f and Δ by g and

$\Gamma(\Delta)$, respectively, in $\mathrm{L}(\Delta;\boldsymbol{\lambda})$, and further dropping the constraint $\mathbf{c}^{\mathsf{T}}\mathbf{x} \geq \widetilde{z} - c_{r+1}$, 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 (3.9). The right-hand side can also be viewed as a Lagrangian relaxation of (3.10), 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

$$\left|\begin{array}{lll} \text{maximize} & z = -\mathbf{b}^{\mathsf{T}}\boldsymbol{\lambda} + \mathbf{s}^{\mathsf{T}}\boldsymbol{\mu} - \mathbf{t}^{\mathsf{T}}\boldsymbol{\nu} & \\ \text{subject to} & \mathbf{A}^{\mathsf{T}}\boldsymbol{\lambda} - \boldsymbol{\mu} + \boldsymbol{\nu} = -\mathbf{c} & \\ & \mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda} \geq \mathbf{0}, \quad (\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}) \geq \mathbf{0}, & \end{array}\right. \tag{3.19}$$

which is the dual problem of (3.10). Let $(\widetilde{\boldsymbol{\lambda}}, \widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\nu}})$ be an optimal solution to (3.19).

Lemma 3.3. *For any $\boldsymbol{\lambda} \geq \mathbf{0}$, the inequality*

$$\phi(\boldsymbol{\lambda}) + c_{r+1} \leq \widetilde{z}$$

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

Since the dual optimal solution $(\widetilde{\boldsymbol{\lambda}}, \widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\nu}})$ is generated as a byproduct in solving the primal problem $\widetilde{\mathrm{P}}(\Delta)$, adopting $\widetilde{\boldsymbol{\lambda}}$ as the Lagrangian multiplier of $\mathrm{L}(\Delta;\widetilde{\boldsymbol{\lambda}})$ does not cost additional computation. As is easily seen, $\mathrm{L}(\Delta;\widetilde{\boldsymbol{\lambda}})$ can be decomposed into

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

$$\mathrm{L}_y(\Delta;\widetilde{\boldsymbol{\lambda}}) \left|\begin{array}{lll} \text{minimize} & z_y = & \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}\mathbf{B}\mathbf{y} \\ \text{subject to} & & \mathbf{y} \geq \mathbf{0}. \end{array}\right.$$

The latter problem has an obvious optimal solution $\mathbf{y}(\widetilde{\boldsymbol{\lambda}}) = \mathbf{0}$ because $\mathbf{B}^{\mathsf{T}}\widetilde{\boldsymbol{\lambda}} \geq \mathbf{0}$. 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

$$z(\widetilde{\boldsymbol{\lambda}}) = f(\mathbf{x}(\widetilde{\boldsymbol{\lambda}})) + \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}(\mathbf{A}\mathbf{x}(\widetilde{\boldsymbol{\lambda}}) - \mathbf{b}). \tag{3.20}$$

Theorem 3.4. *The optimal value $z(\Delta)$ of $P(\Delta)$, $\widetilde{z}$ of* (3.11)*, and $z(\widetilde{\boldsymbol{\lambda}})$ of* (3.20) *satisfy the inequality*

$$\widetilde{z} \leq z(\widetilde{\boldsymbol{\lambda}}) \leq z(\Delta). \tag{3.21}$$

Furthermore,

$$\widetilde{z} < z(\widetilde{\boldsymbol{\lambda}})$$

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 (3.19) 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 3.3, 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 $\mathbf{x}(\widetilde{\boldsymbol{\lambda}})$ is a vertex 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, from the strict concavity of f, we obtain

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

□

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 we assume an oracle telling the value of f. Since the objective function is concave, $\mathbf{x}(\widetilde{\boldsymbol{\lambda}})$ is a vertex of $\Delta \cap G$. Furthermore, the number of its vertices is $O(r^2)$ at most. We only need to check the objective function value at vertices $\mathbf{v}_i$ of Δ in G and at the intersection of ∂G with each edge $\mathbf{v}_i$–$\mathbf{v}_j$ of Δ connecting $\mathbf{v}_i \in \mathrm{int}(G)$ and $\mathbf{v}_j \notin G$.

3.4.2 Description of the modified algorithm

Now, recall the three basic steps of the simplicial branch-and-bound algorithm given in Section 2.1. We propose the bounding operation of Step 2 consisting of the following two phases:

Step 2.1. Solve $\widetilde{\mathrm{P}}(\Delta)$ and compute z^k defined in (3.14). If $f(\mathbf{x}^*) - \epsilon \leq z^k$ for the incumbent $\mathbf{x}^*$, discard Δ from further consideration.

Step 2.2. If $f(\mathbf{x}^*) - \epsilon > z^k$, solve $\mathrm{L}_x(\Delta; \widetilde{\boldsymbol{\lambda}})$ and compute $z(\widetilde{\boldsymbol{\lambda}})$. If $f(\mathbf{x}^*) - \epsilon \leq z(\widetilde{\boldsymbol{\lambda}})$, 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{ Σ_{j=1}^r x_j | x ∈ X } and let Δ^1 := conv({0, ve_1, ..., ve_r});
  H := {1}; z* := +∞; h := 1;
  while H ≠ ∅ do begin
    select k_h ∈ H and let H := H \ {k_h}; Δ := Δ^{k_h};          /* 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 $\widetilde{\mathbf{x}}^{k_h}$ be an optimal solution to $\widetilde{\mathrm{P}}(\Delta)$ and $\widetilde{z} := g(\widetilde{\mathbf{x}}^{k_h})$;
if $f(\widetilde{\mathbf{x}}^{k_h}) < z^*$ then update $z^* := f(\widetilde{\mathbf{x}}^{k_h})$ and $\mathbf{x}^* := \widetilde{\mathbf{x}}^{k_h}$;
$z^{k_h} := \max\{\widetilde{z}, \min\{f(\mathbf{v}_1), \ldots, f(\mathbf{v}_{r+1})\}\}$;
if $f(\mathbf{x}^*) - \epsilon > z^{k_h}$ then begin

/* *Step 2.2* */

define $\mathrm{L}_x(\Delta; \widetilde{\boldsymbol{\lambda}})$ for a dual optimal solution $(\widetilde{\boldsymbol{\lambda}}, \widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\nu}})$ to $\widetilde{\mathrm{P}}(\Delta)$;
compute an optimal solution $\mathbf{x}(\widetilde{\boldsymbol{\lambda}})$ to $\mathrm{L}_x(\Delta; \widetilde{\boldsymbol{\lambda}})$ and the value $z(\widetilde{\boldsymbol{\lambda}})$;
if $f(\mathbf{x}^*) - \epsilon > z(\widetilde{\boldsymbol{\lambda}})$ then begin

/* *Step 3* */

select the longest edge $\mathbf{v}_p$–$\mathbf{v}_q$ of Δ;
let $\mathbf{v} := (1-\alpha)\mathbf{v}_p + \alpha\mathbf{v}_q$ for a fixed $\alpha \in (0, 1/2]$;
$\Delta^{2k_h} := \mathrm{conv}(\{\, \mathbf{v}_i \mid i \neq p \,\} \cup \{\mathbf{v}\})$;
$\Delta^{2k_h+1} := \mathrm{conv}(\{\, \mathbf{v}_i \mid i \neq q \,\} \cup \{\mathbf{v}\})$;
$\mathcal{H} := \mathcal{H} \cup \{2k_h, 2k_h + 1\}$
end
end
end;
$h := h + 1$
end
end;

Theorem 3.5. *Suppose the tolerance ϵ is 0. If algorithm 2PHASE_BB terminates after finite iterations, $\mathbf{x}^*$ is a globally optimal solution to problem P. If not, the algorithm with the best-bound rule generates an infinite sequence $\{\, \widetilde{\mathbf{x}}^{k_h} \mid h = 1, 2, \ldots \}$. For each infinite subsequence $\mathcal{L}$ of iterations $h = 1, 2, \ldots$ such that the associated simplices are nested, any accumulation point of $\{\, \widetilde{\mathbf{x}}^{k_\ell} \mid \ell \in \mathcal{L} \,\}$ 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 simplices. Let $\{\, \Delta^{k_\ell} \mid \ell \in \mathcal{L} \,\}$ be a subsequence of nested simplices for some infinite subsequence

$\mathcal{L}$ of iterations $h = 1, 2, \ldots$. Since the best-bound rule is adopted, we have

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

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

$$z^{k_\ell} \leq z(\Delta^1) \leq f(\widetilde{\mathbf{x}}^{k_\ell}), \quad \forall \ell \in \mathcal{L}.$$

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 3.6. *When $\epsilon > 0$, algorithm 2PHASE_BB with either the depth-first rule or the best-bound rule terminates after a finite number of iterations. The incumbent $\mathbf{x}^*$ is a globally ϵ-optimal solution to problem P.*

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

3.5 Numerical Experiment

3.5.1 Instances

Let us report numerical results of having compared 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:

$$\left| \begin{array}{ll} \text{minimize} & -(1/2)\mathbf{x}^\mathsf{T}\mathbf{Q}^\mathsf{T}\mathbf{Q}\mathbf{x} - \omega\mathbf{d}^\mathsf{T}\mathbf{y} \\ \text{subject to} & \mathbf{A}'\mathbf{x} + \mathbf{B}'\mathbf{y} \leq \mathbf{b}', \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array} \right. \tag{3.22}$$

where $\mathbf{Q} \in \mathbb{R}^{r' \times r'}$, $\mathbf{A}' \in \mathbb{R}^{m' \times r'}$, $\mathbf{B}' \in \mathbb{R}^{m' \times (n'-r')}$, $\mathbf{b}' \in \mathbb{R}^{m'}$, $\mathbf{d} \in \mathbb{R}^{n'-r'}$ and ω is a positive weight. The matrix $\mathbf{Q} = [q_{ij}]$ is 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 $q_{ii} = 1.0$ for all $i = 1, \ldots, r'$ and the rest are drawn randomly from the uniform distribution on $[0.0, 1.0]$. Then $\mathbf{Q}^\mathsf{T}\mathbf{Q}$ has at most three nonzero entries in each row. Also, each component of $\mathbf{d}$ is a uniformly random number in $[0.0, 1.0]$. To make the feasible set bounded, $\mathbf{b}'$ is an all-ones vector and each component in the last row of $[\mathbf{A}', \mathbf{B}']$ is fixed to $1.0/n'$. Other components are all random numbers in $[-0.5, 1.0]$, where the percentages of zeros and negative numbers are about 20% and 10%, respectively. Selecting various sets of parameters (m', n', r', ω), we generated ten instances of (3.22) for each set, and solved them by 2phase and standard on a Linux workstation (Linux 2.4.21, Itanium2 processor 1.3GHz).

3.5.2 Computer codes

Both codes 2phase and standard are written in GNU Octave (version 2.1.50) [43], a Matlab-like computational tool. In both algorithms, the depth-first rule is adopted, and the tolerance ϵ is fixed to 10^{-4}. To adjust the form of (3.22) to (3.3), we introduce an additional variable ζ and apply the algorithm 2phase to

$$\left| \begin{array}{ll} \text{minimize} & -(1/2)\mathbf{x}^\mathsf{T}\mathbf{Q}^\mathsf{T}\mathbf{Q}\mathbf{x} - \omega\zeta \\ \text{subject to} & \mathbf{A}'\mathbf{x} + \mathbf{B}'\mathbf{y} \leq \mathbf{b}', \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \\ & \zeta - \mathbf{d}^\mathsf{T}\mathbf{y} \leq 0, \quad \zeta \geq 0. \end{array} \right. \tag{3.23}$$

where we should note $\zeta \geq 0$ because $\mathbf{d} \geq \mathbf{0}$. The size (m, n, r) of (3.23) is therefore equal to $(m'+1, n'+1, r'+1)$. As for standard, we apply it directly to (3.22) because it uses only the relaxation problem $\overline{\mathrm{P}}(\Delta)$, which can be written 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 $\alpha = 1/2$ is 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 (3.22) and generated the same output as 2phase with the usual bisection rule.

3.5.3 Numerical results

Figures 3.1–3.4 give the line plots for comparing the behavior of 2phase by solid lines with that of standard by broken lines when the size of constraint matrix $[\mathbf{A}', \mathbf{B}']$ is fixed to $(m', n') = (40, 80)$.

Figure 3.1 shows the variation in the average number of branching operations required by each algorithm when ω was fixed to 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 3.2. The algorithm 2phase surpasses standard in computational time for all value of r', which, we can understand, implies that the problem (3.10) 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 to make a simplified code 1phase, and tried to solve the same set of instances. It performed as well as 2phase did when $r' < 25$, but failed to terminate after 10^5 branching operations, for one instance with $r' = 26, 28$, 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 3.3 and 3.4 show the variations of the average number of branching operations and CPU seconds, respectively, required by each algorithm when r' was

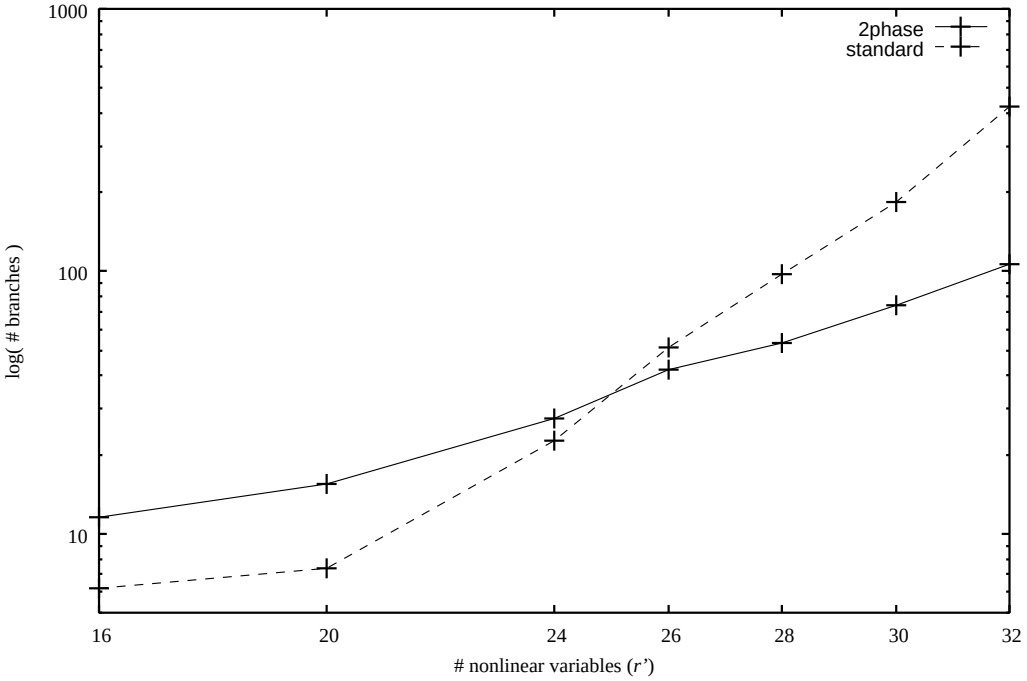

Figure 3.1: Numbers of branching operations when $(m', n', \omega) = (40, 80, 5.0)$.

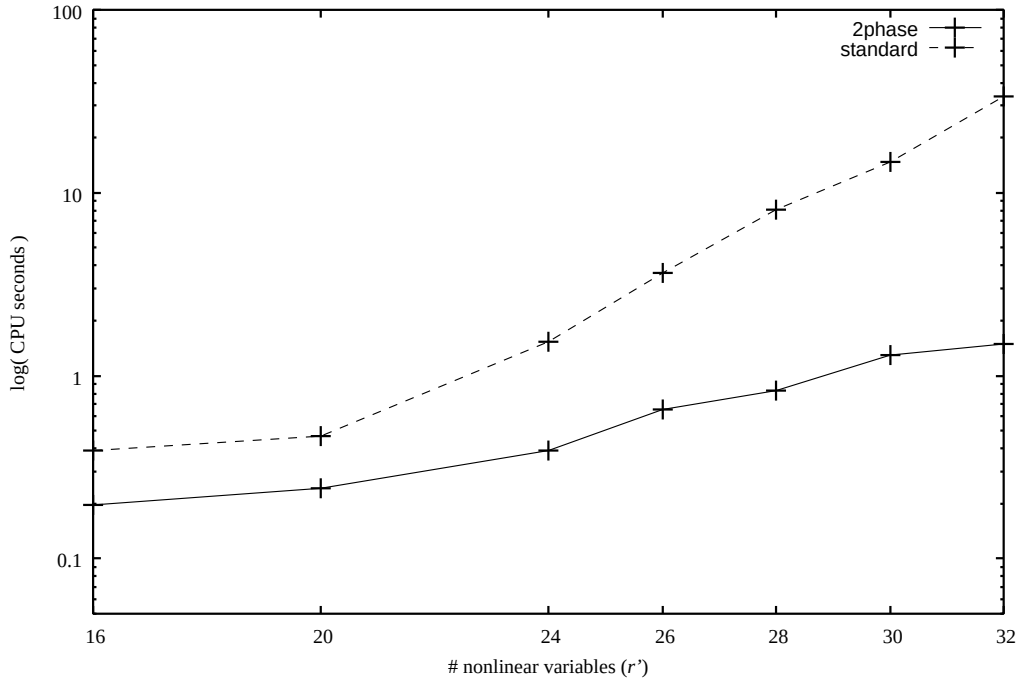

Figure 3.2: CPU seconds when $(m', n', \omega) = (40, 80, 5.0)$.

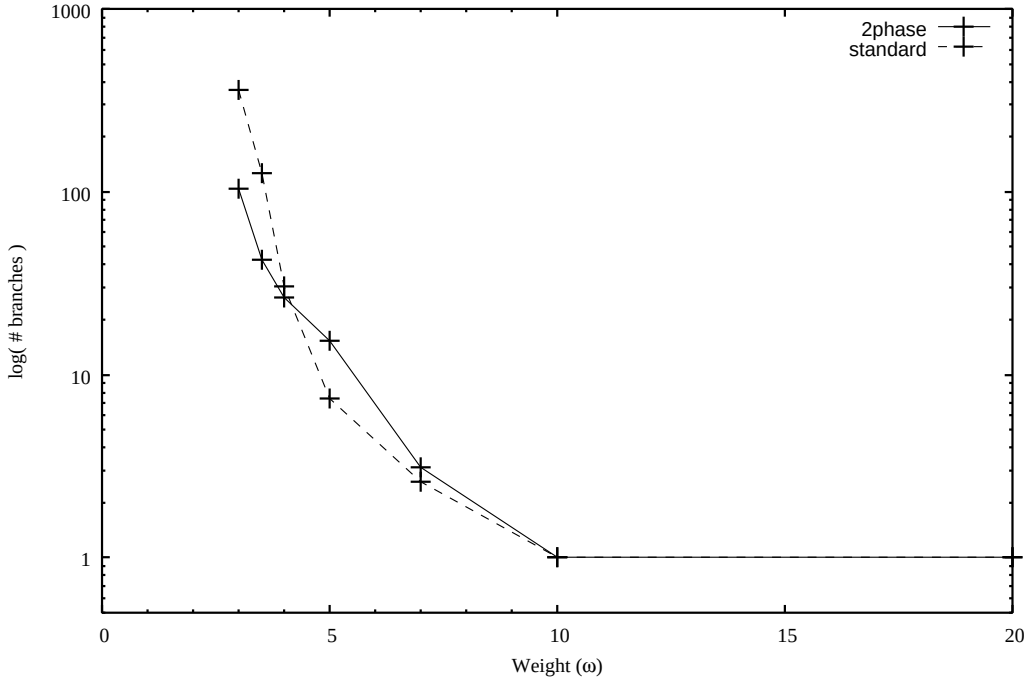

Figure 3.3: Numbers of branching operations when $(m', n', r') = (40, 80, 20)$.

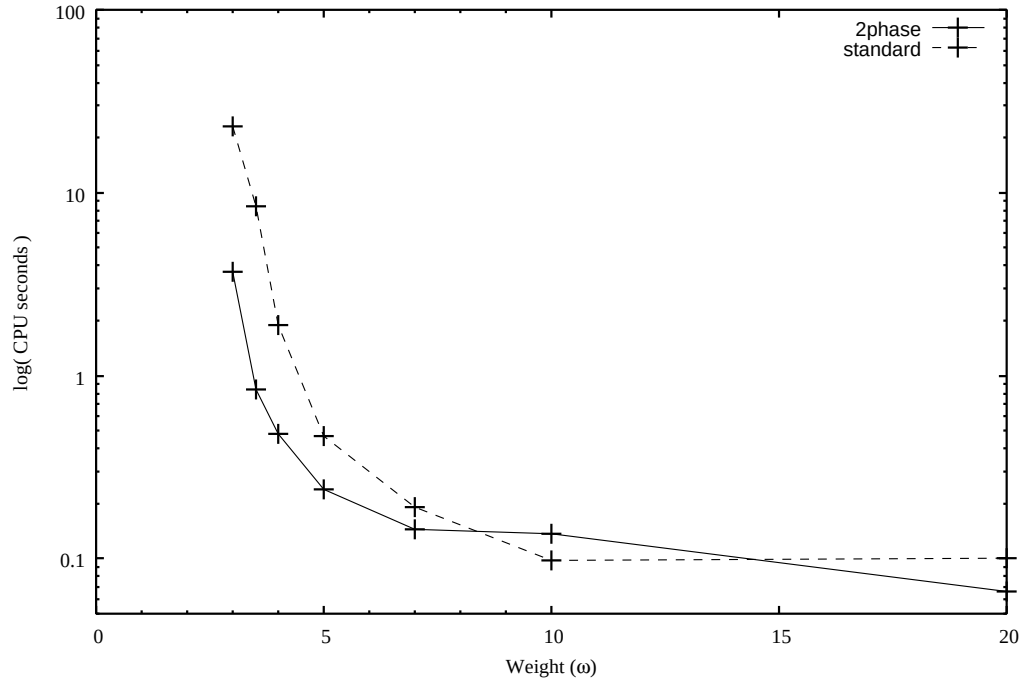

Figure 3.4: CPU seconds when $(m', n', r') = (40, 80, 20)$.

Table 3.1: 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

fixed to 20 and ω was changed in $\{3.0, 3.5, 4.0, 5.0, 7.0, 10.0, 20.0\}$. Unfortunately, both are very sensitive to the variation of ω, especially when $\omega < 5$. Nevertheless, 2phase needs considerably less branching operations than standard when $\omega \leq 4$, which is totally owing 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 (3.10), yields the significant advantage of 2phase against standard in computational time when $\omega < 10$. Incidentally, 1phase failed to terminate after 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 does, unless the concavity part of the objective function has a lot of weight. We therefore tested the code 2phase on (3.22) of size (m', n') from $(60, 120)$ to $(300, 200)$ with ω fixed to 5.0. The number of nonlinear variables r' varied from 20% to 50% of all the variables, i.e., the maximum size of (m', n', r') was $(300, 200, 100)$. The computational results are listed in Table 3.1, in which # and *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 kept less than 30% of all the variables.

3.6 Conclusion and Future Issues

We have developed a simplicial branch-and-bound algorithm for solving a low-rank concave minimization problem (3.3). The major feature of this problem is that only a small fraction of variables are involved in the objective function. In the bounding operation of the algorithm, we have proposed to enlarge the feasible set of linear programming relaxation 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 that both operations work very well and the algorithm has potential for solving much larger scale problems than the existing algorithm does.

To further expand the versatility of the algorithm, we need to resolve two issues in the future. The first issue is concerning the transformation of problems. Low-rank concave minimization problems can certainly be transformed into the form of (3.3). However, many of such transformations destroy the structure of the constraint, like the ones from (3.1) to (3.2) and from (3.22) to (3.23), and can take away from the devices in the first phase of our bounding operation. The second 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. Hence, 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.

Chapter 4

Revised Simplicial Algorithm for Concave Minimization Problems

4.1 Introduction

In this chapter, 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 its nonconvexity rank [24, 45] is $r << n - r$. It is known that when the value of each component of $\mathbf{x} \in \mathbb{R}^r$ is fixed, $F(\mathbf{D}_x\mathbf{x} + \mathbf{D}_y\mathbf{y})$ becomes affine for some orthogonal matrix $\mathbf{D} = [\mathbf{D}_x, \mathbf{D}_y] \in \mathbb{R}^{n\times n}$ with $\mathbf{D}_x \in \mathbb{R}^{n\times r}$. The matrix $\mathbf{D}$ is referred to as a *certificate* for the nonconvexity rank of F. If we try to minimize such concave functions, we can move the affine part of the objective function to the constraints by introducing an auxiliary variable. The resulting problem has a characteristic structure that the variables involved in the objective function are a small fraction of all 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 (see e.g., [32, 40, 56, 58]). 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 4.1 in Section 4.3).

Even if the objective function is not concave, some problems can be reduced to our intended class. An example would be the linear multiplicative programming problem (see e.g., [23, 27, 33, 46]):

$$\left| \begin{array}{ll} \text{minimize} & \displaystyle\prod_{i=1}^{r} (\mathbf{a}_i^{\mathsf{T}}\mathbf{y} + a_{i0}) \\ \text{subject to} & \mathbf{B}\mathbf{y} = \mathbf{b}, \quad \mathbf{y} \geq \mathbf{0}, \end{array} \right. \tag{4.1}$$

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 (4.1) is not concave (see [3]), 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:

$$\left| \begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{r} \log(x_i) \\ \text{subject to} & x_i - \mathbf{a}_i^{\mathsf{T}}\mathbf{y} = a_{i0}, \quad i = 1, \ldots, r \\ & \mathbf{B}\mathbf{y} = \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}. \end{array} \right. \tag{4.2}$$

In general, the number r of affine functions in the objective function of (4.1) is assumed to be far less than the number of variables $\mathbf{y}$. Therefore, the variables involved in the objective function of (4.2) are again a small portion of all the variables.

If the objective function is separated into a sum of univariate functions like (4.2), concave minimization problems can be solved rather efficiently using the rectangular branch-and-bound algorithm (see e.g., [7, 27, 32]). To deal with a wider range of problems, we do not assume the separability throughout this chapter. We then tailor the simplicial branch-and-bound algorithm in [15] to the class and to facilitate application of some procedures for improving the efficiency. After giving the problem settings in Section 4.2, we will bring up some difficulties in the implementation of existing bounding procedures and propose to simplify the linear programming relaxation to be solved at each iteration in Section 4.3. This simplification still guarantees the convergence but deteriorates the quality of the

lower bound on the optimal value. It can cause rapid growth of the branching tree. To prevent it, we will develop an additional bounding procedure based on the Lagrangian relaxation in Section 4.4. Lastly, we will close the chapter with a report of computational comparison of the proposed algorithm and two existing algorithms in Section 4.5.

4.2 Problem Settings

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 chapter is of minimizing the function f over a polyhedron in $\mathbb{R}^n$:

$$\left| \begin{array}{lll} \text{minimize} & z = & f(\mathbf{x}) \\ \text{subject to} & & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array} \right. \tag{4.3}$$

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, (4.3) can be embedded in $\mathbb{R}^r$:

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

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

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

has a finite value, where $\mathbf{e} \in \mathbb{R}^r$ is the all-ones vector. We also assume the domain of f large enough to include the 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 in [7, 28] reviewed in Section (2.1) as Standard_SBB is a standard method for locating a globally optimal solution of (4.3), or equivalently of P.

The bounding operation of Step 2 in Standard_SBB is the most time-consuming step. In the next section, we will discuss troublesome issues with Step 2 faced by existing algorithms and their resolution in treating our target problem (4.3).

4.3 Linear Programming Relaxations

4.3.1 Troublesome issues

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:

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

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 $\mathbf{v}_j$, $j = 1, \ldots, r+1$:

$$\mathbf{x} = \sum_{j=1}^{r+1} \zeta_j \mathbf{v}_j, \quad \sum_{j=1}^{r+1} \zeta_j = 1, \quad \boldsymbol{\zeta} = (\zeta_1, \ldots, \zeta_{r+1})^{\mathsf{T}} \geq \mathbf{0}. \tag{4.4}$$

By the concavity of f, we have

$$g(\mathbf{x}) = \sum_{j=1}^{r+1} \zeta_j f(\mathbf{v}_j) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in \Delta. \tag{4.5}$$

Substituting (4.4) into Q(Δ), we have an equivalent linear programming problem of $n+1$ variables:

$$\left| \begin{array}{lll} \text{minimize} & w = & \mathbf{f}^{\mathsf{T}}\boldsymbol{\zeta} \\ \text{subject to} & & \mathbf{AV}\boldsymbol{\zeta} + \mathbf{By} = \mathbf{b} \\ & & \mathbf{e}^{\mathsf{T}}\boldsymbol{\zeta} = 1, \quad (\boldsymbol{\zeta}, \mathbf{y}) \geq \mathbf{0}, \end{array} \right. \tag{4.6}$$

where

$$\mathbf{f} = [f(\mathbf{v}_1), \ldots, f(\mathbf{v}_{r+1})]^{\mathsf{T}}, \quad \mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_{r+1}]. \tag{4.7}$$

Obviously, (4.6) has an optimal solution $(\boldsymbol{\zeta}^\circ, \mathbf{y}^\circ)$ if and only if $X \cap \Delta \neq \emptyset$. Since the inequality in (4.5) holds, we can set the lower bound to

$$w^\circ = \begin{cases} \mathbf{f}^{\mathsf{T}}\boldsymbol{\zeta}^\circ & \text{if } X \cap \Delta \neq \emptyset \\ +\infty & \text{otherwise.} \end{cases} \tag{4.8}$$

When $X \cap \Delta \neq \emptyset$, we also have a feasible solution $\mathbf{x}^\circ$ to the subproblem $\mathrm{P}(\Delta)$, 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 $\mathbf{x}^\circ$.

The troublesome issues are

(a) when Δ changes, (4.6) associated with $\mathrm{Q}(\Delta)$ has a different set of constraints,

(b) (4.6) does not inherit the structure of the target problem (4.3).

Despite the vast number of subproblems (4.6) 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 primal feasible nor dual feasible, due to (a). Moreover, even if the target problem (4.3) has some favorable structure, like network flow, (b) prevents us from applying efficient algorithms to solve (4.6). These issues, however, have been resolved partly in Chapter 3, as will be seen below.

4.3.2 Modified relaxation proposed in Chapter 3

In Chapter 3, we have relaxed the constraint $\mathbf{x} \in \Delta$ of $\mathrm{Q}(\Delta)$ 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

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

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 our alternative to $\mathrm{Q}(\Delta)$ is as follows:

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

We have also proposed to abandon the variable transformation (4.4). 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:

$$\mathbf{c}^{\mathsf{T}}\mathbf{v}_j + c_{r+1} = f(\mathbf{v}_j), \quad j = 1, \ldots, r+1, \tag{4.9}$$

and $\overline{\mathrm{Q}}(\Delta)$ is formulated with variables $(\mathbf{x}, \mathbf{y})$:

$$\left| \begin{array}{lll} \text{minimize} & w = & \mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b}, \quad \mathbf{s} \leq \mathbf{x} \leq \mathbf{t}, \quad \mathbf{y} \geq \mathbf{0}, \end{array} \right. \tag{4.10}$$

where

$$[\mathbf{c}^{\mathsf{T}}, c_{r+1}] = \mathbf{f}^{\mathsf{T}} \begin{bmatrix} \mathbf{V} \\ \mathbf{e}^{\mathsf{T}} \end{bmatrix}^{-1}. \tag{4.11}$$

If $X \cap \Gamma(\Delta) \neq \emptyset$, then (4.10) has an optimal solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$. The lower bound can be set to

$$\overline{w} = \begin{cases} \mathbf{c}^{\mathsf{T}}\overline{\mathbf{x}} + c_{r+1} & \text{if } X \cap \Gamma(\Delta) \neq \emptyset \\ +\infty & \text{otherwise,} \end{cases} \tag{4.12}$$

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 $\mathbf{s} \leq \mathbf{x} \leq \mathbf{t}$, the subproblem (4.10) associated with $\overline{\mathrm{Q}}(\Delta)$ for each Δ shares constraints. We can solve the current subproblem (4.10), 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 by a very few pivoting operations of the dual and primal simplex algorithms. Also, (4.10) inherits favorable structures of (4.3) at any. Unfortunately, however, it is not that (4.10) inherits the structure of the original low-rank nonconvex problem behind the target (4.3).

Example 4.1. Let us consider the production-transportation problem mentioned in Section 4.1:

$$\left|\begin{array}{lll} \text{minimize} & z = & \displaystyle\sum_{i\in M}\sum_{j\in N} a_{ij}y_{ij} + f(\mathbf{x}) \\ \text{subject to} & & \displaystyle\sum_{j\in N} y_{ij} = x_i, \quad i \in M \\ & & \displaystyle\sum_{i\in M} y_{ij} = b_j, \quad j \in N \\ & & (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array}\right. \tag{4.13}$$

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 (4.13) 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 (4.3):

$$\left|\begin{array}{lll} \text{minimize} & z = & \xi + f(\mathbf{x}) \\ \text{subject to} & & \xi - \displaystyle\sum_{i\in M}\sum_{j\in N} a_{ij}y_{ij} = 0 \\ & & x_i - \displaystyle\sum_{j\in N} y_{ij} = 0, \quad i \in M \\ & & \displaystyle\sum_{i\in M} y_{ij} = b_j, \quad j \in N \\ & & (\xi, \mathbf{x}, \mathbf{y}) \geq \mathbf{0}. \end{array}\right. \tag{4.14}$$

The linear programming representation of $\overline{\mathrm{Q}}(\Delta)$ associated with (4.14) is then as follows:

$$\left|\begin{array}{lll} \text{minimize} & w = & \mathbf{c}_0\xi + \mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & & \xi - \displaystyle\sum_{i\in M}\sum_{j\in N} a_{ij}y_{ij} = 0 \\ & & x_i - \displaystyle\sum_{j\in N} y_{ij} = 0, \quad i \in M \\ & & \displaystyle\sum_{i\in M} y_{ij} = b_j, \quad j \in N \\ & & s_0 \leq \xi \leq t_0, \;\; \mathbf{s} \leq \mathbf{x} \leq \mathbf{t}, \;\; \mathbf{y} \geq \mathbf{0}. \end{array}\right. \tag{4.15}$$

This is neither a Hitchcock problem any longer, nor even a network flow problem. Therefore, the special purpose algorithm for network flow does not apply to (4.15). ■

In addition to this, we have another difficulty with this modification. To obtain the objective function of (4.10), we need to solve the linear system (4.9) 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 Chapter 3. However, its solution becomes numerically unstable as Δ becomes smaller, and might fail to be computed at worst, due to rounding errors.

4.3.3 New relaxation resolving all difficulties

Both difficulties involved in the above modification can be removed by combining two kinds of relaxation.

Let

$$\kappa(\Delta) = \min\{\, \|\mathbf{v}_p - \mathbf{v}_q\| \mid p = 1, \ldots, r;\ q = p+1, \ldots, r+1 \,\}, \tag{4.16}$$

and assume that (4.9) can be solved with sufficient precision if $\kappa(\Delta) \geq \delta$ for some number $\delta > 0$. While $\kappa(\Delta) \geq \delta$ is satisfied, we drop the bounding constraint $\mathbf{x} \in \Gamma(\Delta)$ from $\overline{\mathrm{Q}}(\Delta)$ and solve

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

If $\kappa(\Delta)$ 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

$$d_{r+1} = \min\{\, f(\mathbf{v}_j) - \mathbf{d}^{\mathsf{T}}\mathbf{v}_j \mid j = 1, \ldots, r+1 \,\}, \tag{4.17}$$

and let

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

From (4.17) and the concavity of f, we see that

$$h(\mathbf{x}) \leq f(\mathbf{x}), \quad \forall \mathbf{x} \in \Delta, \tag{4.18}$$

where the equality holds at some $\mathbf{v}_j$ that attains the minimum of (4.17). 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 $\kappa(\Delta)$; but in any case it is equivalent to a linear programming problem of the same form:

$$\left| \begin{array}{lll} \text{minimize} & w = & \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array} \right. \tag{4.19}$$

where

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

Since X is nonempty, (4.19) has an optimal solution $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})$ and we have the following lower bound on the value of $\mathrm{P}(\Delta)$:

$$\widetilde{w} = \boldsymbol{\theta}^{\mathsf{T}}\widetilde{\mathbf{x}} + \theta_{r+1}. \tag{4.20}$$

Proposition 4.2. *The optimal value $z(\Delta)$ of $P(\Delta)$, $\widetilde{w}$ of (4.20), $\overline{w}$ of (4.12), and w° of (4.8) satisfy the inequality:*

$$\widetilde{w} \leq \overline{w} \leq w^\circ \leq z(\Delta). \tag{4.21}$$

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 $\widetilde{w}$ can serve as z^k at Step 2 of Standard_SBB, though it is inferior to w° and $\overline{w}$. Problem (4.19) yielding $\widetilde{w}$, however, has the redeeming feature that the constraints are exactly the same as those of (4.3). 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 subproblem as the initial feasible basic solution and start the primal simplex algorithm immediately. The most important thing is that (4.19) inherits not only the structure of (4.3) but also that of the original problem.

Example 4.3. Again, consider the problem (4.14) which is reduced from the production-transportation problem (4.13). Associated with (4.14), we have the following linear programming representation of $\widetilde{\mathrm{Q}}_g(\Delta)$ and $\widetilde{\mathrm{Q}}_h(\Delta)$:

$$\left|\begin{array}{lll} \text{minimize} & w = & \theta_0\xi + \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & & \xi - \displaystyle\sum_{i\in M}\sum_{j\in N} a_{ij}y_{ij} = 0 \\ & & x_i - \displaystyle\sum_{j\in N} y_{ij} = 0, \quad i \in M \\ & & \displaystyle\sum_{i\in M} y_{ij} = b_j, \quad j \in N \\ & & (\xi, \mathbf{x}, \mathbf{y}) \geq \mathbf{0}. \end{array}\right. \tag{4.22}$$

If we substitute the first constraint into the objective function and eliminate ξ, then (4.22) reduces to

$$\left|\begin{array}{lll} \text{minimize} & w = & \theta_0 \displaystyle\sum_{i\in M}\sum_{j\in N} a_{ij}y_{ij} + \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & & \displaystyle\sum_{j\in N} y_{ij} = x_i, \quad i \in M \\ & & \displaystyle\sum_{i\in M} y_{ij} = b_j, \quad j \in N \\ & & (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array}\right. \tag{4.23}$$

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 4.4. Let us consider a more general example:

$$\left|\begin{array}{ll} \text{minimize} & z = F(\mathbf{w}) \\ \text{subject to} & \mathbf{Mw} = \mathbf{b}, \quad \mathbf{w} \geq \mathbf{0}, \end{array}\right. \tag{4.24}$$

where $\mathbf{M} \in \mathbb{R}^{m \times n}$, $F : \mathbb{R}^n \to \mathbb{R}$ is a concave function of nonconvexity rank r, which we assume is less than 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 $\mathbf{D}_x \in \mathbb{R}^{n \times r}$, problem (4.24) is equivalent to

$$\left|\begin{array}{lll} \text{minimize} & z = \xi + f(\mathbf{x}) & \\ \text{subject to} & \xi - \mathbf{a}^\mathsf{T}\mathbf{y} = a_0 & \\ & \mathbf{MD}_x\mathbf{x} + \mathbf{MD}_y\mathbf{y} = \mathbf{b} & \\ & \mathbf{D}_x\mathbf{x} + \mathbf{D}_y\mathbf{y} \geq \mathbf{0}. & \end{array}\right. \tag{4.25}$$

Hence, the subproblem of (4.25) has the simplex constraint $(\mathbf{x}, \xi) \in \Delta$. However, in our proposed linear programming representation of $\widetilde{\mathrm{Q}}_g(\Delta)$ and $\widetilde{\mathrm{Q}}_h(\Delta)$ associated with (4.25), the constraint $(\mathbf{x}, \xi) \in \Delta$ is dropped, and then it is formulated as follows:

$$\left|\begin{array}{lll} \text{minimize} & w = \theta_0\xi + \boldsymbol{\theta}^\mathsf{T}\mathbf{x} & \\ \text{subject to} & \xi - \mathbf{a}^\mathsf{T}\mathbf{y} = a_0 & \\ & \mathbf{MD}_x\mathbf{x} + \mathbf{MD}_y\mathbf{y} = \mathbf{b} & \\ & \mathbf{D}_x\mathbf{x} + \mathbf{D}_y\mathbf{y} \geq \mathbf{0}. & \end{array}\right. \tag{4.26}$$

Therefore, this problem can be transformed into the following with the same set of constraints as (4.24):

$$\left|\begin{array}{ll} \text{minimize} & w = (\boldsymbol{\theta}^\mathsf{T}, \theta_0\mathbf{a}^\mathsf{T})\mathbf{D}^{-1}\mathbf{w} \\ \text{subject to} & \mathbf{Mw} = \mathbf{b}, \quad \mathbf{w} \geq \mathbf{0}. \end{array}\right. \tag{4.27}$$

■

4.4 Revised Simplicial Algorithm

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

Besides the advantage we have seen in the preceding section, 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. In Chapter 3, we have proposed to adopt $\overline{\mathrm{Q}}(\Delta)$, which is $\widetilde{\mathrm{P}}(\Delta)$ in Chapter 3, and proved the convergence of the algorithm by the property of the constraint $\mathbf{x} \in \Gamma(\Delta)$. Although neither $\widetilde{\mathrm{Q}}_g(\Delta)$ nor $\widetilde{\mathrm{Q}}_h(\Delta)$ has such an additional constraint, the convergence behavior of the objective function of $\widetilde{\mathrm{Q}}_h(\Delta)$ 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 on the value of $\mathrm{P}(\Delta)$. Before proceeding to the convergence analysis, we discuss a procedure, based on the Lagrangian relaxation, for tightening $\widetilde{w}$. In Chapter 3, it has been reported that a similar procedure works well for tightening $\overline{w}$.

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 overlooked even if we add the constraint $\mathbf{x} \in G$. The resulting problem is then

$$\left|\begin{array}{lll} \text{minimize} & z = & f(\mathbf{x}) \\ \text{subject to} & & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0} \\ & & \mathbf{x} \in \Delta, \quad \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} \geq \widetilde{w} - \theta_{r+1}. \end{array}\right. \tag{4.28}$$

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:

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

If $\boldsymbol{\theta}^{\mathsf{T}}\mathbf{v}_j < \widetilde{w} - \theta_{r+1}$ for all vertices $\mathbf{v}_j$ 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 because Δ can never contain an optimal solution to the master problem P, we can discard it. In the rest of this subsection, we assume

$$\exists j \in \{1, \ldots, r+1\}, \quad \boldsymbol{\theta}^{\mathsf{T}}\mathbf{v}_j \geq \widetilde{w} - \theta_{r+1}. \tag{4.29}$$

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 $\boldsymbol{\lambda}$ (see e.g. [42]). The question is how to fix the value of $\boldsymbol{\lambda}$ in $\mathrm{L}(\Delta;\boldsymbol{\lambda})$ inexpensively so that $w(\boldsymbol{\lambda}) > \widetilde{w}$ holds. Suppose that $\boldsymbol{\lambda}$ is constant for a while, and consider a linear programming problem:

$$\left|\begin{array}{lll} \text{minimize} & w = & (\boldsymbol{\theta}^{\mathsf{T}} - \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{A})\mathbf{x} - \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{B}\mathbf{y} + \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{b} \\ \text{subject to} & & (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array}\right. \tag{4.30}$$

which is obtained from $\mathrm{L}(\Delta;\boldsymbol{\lambda})$ by dropping the constraint $\mathbf{x} \in \Delta$ and $\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} \geq \widetilde{w} - \theta_{r+1}$, 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 (4.30) has a trivial optimal value $\boldsymbol{\lambda}^{\mathsf{T}}\mathbf{b}$. These conditions simultaneously ensure the feasibility of the dual problem of (4.30):

$$\left|\begin{array}{lll} \text{maximize} & w = & \mathbf{b}^{\mathsf{T}}\boldsymbol{\lambda} \\ \text{subject to} & & \mathbf{A}^{\mathsf{T}}\boldsymbol{\lambda} \leq \boldsymbol{\theta} \\ & & \mathbf{B}^{\mathsf{T}}\boldsymbol{\lambda} \leq \mathbf{0}. \end{array}\right. \tag{4.31}$$

If we think of $\boldsymbol{\lambda}$ as a vector of variables, (4.31) is the dual problem of the linear programming problem (4.19) introduced in the previous section. Therefore, the value of $\boldsymbol{\lambda}$ maximizing the optimal value of (4.30) is given by an optimal solution $\widetilde{\boldsymbol{\lambda}}$ to the dual problem of $\widetilde{\mathrm{Q}}_g(\Delta)$ or $\widetilde{\mathrm{Q}}_h(\Delta)$. Hence, $\boldsymbol{\lambda} = \widetilde{\boldsymbol{\lambda}}$ is the desired multiplier.

Note that the dual optimal solution $\widetilde{\boldsymbol{\lambda}}$ is yielded as a byproduct in solving the primal problem (4.19). Moreover, we see from the constraints of (4.31) that $\mathbf{y}(\widetilde{\boldsymbol{\lambda}}) = \mathbf{0}$ because the coefficient of $\mathbf{y}$ in the objective function of $\mathrm{L}(\Delta;\widetilde{\boldsymbol{\lambda}})$ is a

nonnegative vector. Therefore, we could delete $\mathbf{y}$ from $\mathrm{L}(\Delta;\widetilde{\boldsymbol{\lambda}})$, and solve the following reduced problem to obtain $w(\widetilde{\boldsymbol{\lambda}})$:

$$\left|\begin{array}{lll} \text{minimize} & w = & f(\mathbf{x}) - \widetilde{\boldsymbol{\lambda}}^{\mathsf{T}}(\mathbf{A}\mathbf{x}-\mathbf{b}) \\ \text{subject to} & & \mathbf{x}\in\Delta,\ \ \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} \geq \widetilde{w} - \theta_{r+1}. \end{array}\right. \tag{4.32}$$

Proposition 4.5. *The optimal value $z(\Delta)$ of $P(\Delta)$, $\widetilde{w}$ of (4.20), and the optimal value $w(\widetilde{\boldsymbol{\lambda}})$ of $L(\Delta;\widetilde{\boldsymbol{\lambda}})$ satisfy the inequality*

$$\widetilde{w} \leq w(\widetilde{\boldsymbol{\lambda}}) \leq z(\Delta). \tag{4.33}$$

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 (4.32) 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 (4.31), 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 (4.32) 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 $\mathbf{x}(\widetilde{\boldsymbol{\lambda}}) \notin \{\mathbf{v}_1,\ldots,\mathbf{v}_{r+1}\}$. 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 $\mathbf{x}(\widetilde{\boldsymbol{\lambda}}) = (1-\mu)\mathbf{v}_p + \mu\mathbf{v}_q$. Since f is assumed to be strictly concave, we have $w(\widetilde{\boldsymbol{\lambda}}) > (1-\mu)w_p + \mu w_q \geq \widetilde{w}$. □

Note 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 $\widetilde{\mathbf{x}} \in \Delta$ and $\kappa(\Delta) \geq \delta$. We should also remark that $w(\widetilde{\boldsymbol{\lambda}})$ can be computed in time polynomial in r if we assume an oracle telling the value of f, though $\mathrm{L}(\Delta; \widetilde{\boldsymbol{\lambda}})$ is a concave minimization problem. Since the objective function of (4.32) is concave, $\mathbf{x}(\widetilde{\boldsymbol{\lambda}})$ is assumed to be a vertex of $\Delta \cap G$. Moreover, the number of its vertices is $O(r^2)$ at most. We need only to check the objective function value at each $\mathbf{v}_j \in G$ and at the intersection of ∂G with each edge $\mathbf{v}_p$–$\mathbf{v}_q$ of Δ connecting $\mathbf{v}_p \in \mathrm{int}(G)$ and $\mathbf{v}_q \notin G$.

Example 4.6. Let us continue Examples 4.1 and 4.3. If we solve (4.23) as the relaxation problem $\widetilde{\mathrm{Q}}_g(\Delta)$ or $\widetilde{\mathrm{Q}}_h(\Delta)$ of (4.14), we cannot directly obtain the value $\widetilde{\lambda}_0$ of the dual variable corresponding to the first constraint of (4.14). However, it is an easy exercise in linear programming to show that $\widetilde{\lambda}_0 = \theta_0$ holds. Thus, $\mathrm{L}(\Delta; \tilde{\boldsymbol{\lambda}})$ for (4.14) is as follows:

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

Similarly, we can obtain $\mathrm{L}(\Delta; \tilde{\boldsymbol{\lambda}})$ for (4.25) from (4.27) in Example 4.4 without any difficulty. ■

4.4.2 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 (2.1). 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 $\kappa(\Delta) \geq \delta$, then solve $\widetilde{\mathrm{Q}}_g(\Delta)$. Otherwise, solve $\widetilde{\mathrm{Q}}_h(\Delta)$. Let $z^k := \widetilde{w}$. If $f(\mathbf{x}^*) - \epsilon \leq z^k$ for the incumbent $\mathbf{x}^*$, then discard Δ.

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

We may of course replace the backtracking criterion by $f(\mathbf{x}^*) - \epsilon|f(\mathbf{x}^*)| \leq z^k$, as in (2.4). 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 $\Delta^1 := \mathrm{conv}(\{\mathbf{0}, v\mathbf{e}_1, \ldots, v\mathbf{e}_r\})$;

$\mathcal{L} := \{1\}$; $z^* := +\infty$; $k := 1$;

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 $\mathbf{v}_1, \ldots, \mathbf{v}_{r+1}$ denote the vertices of Δ;

$\kappa(\Delta) := \min\{\, \|\mathbf{v}_p - \mathbf{v}_q\| \mid p = 1, \ldots, r;\ q = p+1, \ldots, r+1 \,\}$; /* *Step 2.1* */

if $\kappa(\Delta) \geq \delta$ then begin

$\mathbf{V} := [\mathbf{v}_1, \ldots, \mathbf{v}_{r+1}]$; $\mathbf{W} := [\mathbf{V}^{\mathsf{T}}, \mathbf{e}]^{\mathsf{T}}$; /* $\widetilde{\mathrm{Q}}_g(\Delta)$ */

$[\boldsymbol{\theta}^{\mathsf{T}}, \theta_{r+1}] := [f(\mathbf{v}_1), \ldots, f(\mathbf{v}_{r+1})]\mathbf{W}^{-1}$

else

$\mathbf{u} := \sum_{j=1}^{r+1} \mathbf{v}_j/(r+1)$; $\mathbf{d} := \nabla f(\mathbf{u})$; /* $\widetilde{\mathrm{Q}}_h(\Delta)$ */

$d_{r+1} := \min\{\, f(\mathbf{v}_j) - \mathbf{d}^{\mathsf{T}}\mathbf{v}_j \mid j = 1, \ldots, r+1 \,\}$; $[\boldsymbol{\theta}^{\mathsf{T}}, \theta_{r+1}] := [d^{\mathsf{T}}, d_{r+1}]$

end;

solve (4.19) of minimizing $\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} + \theta_{r+1}$ on X to compute $\mathbf{x}^k := \widetilde{\mathbf{x}}$ 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 $j \in \{1, \ldots, r+1\}$ then begin

define (4.32) for a dual optimal solution $\widetilde{\boldsymbol{\lambda}}$ to (4.19); /* $\mathrm{L}(\Delta; \widetilde{\boldsymbol{\lambda}})$ */

solve (4.32) 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 Δ;

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 its convergence property, we will first show how algorithm REVISED_SBB behaves when it does not terminate. In that case, an infinite sequence of nested simplices is generated as in (2.3); and it shrinks to a single point because Δ is subdivided according to the bisection rule. Moreover, we can show the following:

Lemma 4.7. *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

$$\lim_{\ell \to \infty} (f(\mathbf{x}^{k_\ell}) - z^{k_\ell}) = 0. \tag{4.34}$$

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

$$f(\mathbf{x}^{k_\ell}) > z^{k_\ell} \geq h^{k_\ell}(\mathbf{x}^{k_\ell}) = (\mathbf{d}^{k_\ell})^{\mathsf{T}} \mathbf{x}^{k_\ell} + d_{r+1}^{k_\ell}, \tag{4.35}$$

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 $\mathbf{v}^{k_\ell}$ be the vertex defining $d_{r+1}^{k_\ell}$ via (4.17). 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, we have as $\ell \to \infty$

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

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 lie in the compact set X. Therefore, we have as $\ell \to \infty$

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

by the concavity of f. This, together with (4.35), implies (4.34). □

The convergence of algorithm REVISED_SBB follows from this lemma.

Theorem 4.8. *Suppose $\epsilon = 0$. If algorithm REVISED_SBB terminates in finite time, $\mathbf{x}^*$ is a globally optimal solution to the master problem P. If not, every accumulation point of the sequence $\{\, \mathbf{x}^k \mid k = 1, 2, \dots \}$ 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 $\{\, \Delta^{k_\ell} \mid \ell = 1, 2, \dots \}$. 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

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

However, we see from Lemma 4.7 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 4.9. *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*

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) + \epsilon, \quad \forall \mathbf{x} \in X. \tag{4.36}$$

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 (4.36) 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.

4.5 Numerical Experiment

4.5.1 Instances

In this section, we present numerical results of having compared REVISED_SBB and two existing algorithms, the algorithm proposed in Chapter 3 and the standard simplicial branch-and-bound algorithm (see e.g., [19, 17, 55]). We refer to those algorithms here, as revsbb, sbb_1 and sbb_2, respectively. The test problem we solved is a concave quadratic minimization problem of the form:

$$\left| \begin{array}{lll} \text{minimize} & z = & -(1/2)\mathbf{x}^\mathsf{T}\mathbf{C}^\mathsf{T}\mathbf{C}\mathbf{x} - \sigma \mathbf{d}^\mathsf{T}\mathbf{y} \\ \text{subject to} & & \mathbf{A}'\mathbf{x} + \mathbf{B}'\mathbf{y} \leq \mathbf{b}', \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \end{array} \right. \tag{4.37}$$

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 Chapter 3, we generate $\mathbf{C} = [c_{ij}]$ so as to have two nonzero entries in each row, i.e., $(c_{r'1}, c_{r'r'})$ and $(c_{ii}, c_{i,i+1})$ for $i = 1, \dots, r' - 1$, where $c_{ii} = 1.0$ for $i = 1, \dots, r'$ and the rests are drawn randomly from the uniform distribution on $[0.0, 1.0]$. Hence, $\mathbf{C}^\mathsf{T}\mathbf{C}$ has at most three nonzero entries in each row. Also, each component of $\mathbf{d}$ is a uniformly random number in $[0.0, 1.0]$. To make the feasible set bounded, $\mathbf{b}'$ is an all-ones

vector and each component in the last row of $[\mathbf{A}', \mathbf{B}']$ is fixed to $1.0/n'$. Other components are all random numbers in $[-0.5, 1.0]$, where the percentages of zeros and negative numbers are about 20% and 10%, respectively. Selecting various sets of parameters (m', n', r', σ), we generated ten instances of (4.37) for each set, and solve them by revsbb, sbb_1 and sbb_2 on a Linux workstation (Linux 2.4.21, Itanium2 processor 1.3GHz).

4.5.2 Computer codes

Each of the codes revsbb, sbb_1 and sbb_2 was written in GNU Octave (version 2.1.50) [43], a MATLAB-like computational tool, according to the depth-first rule. To adjust the form of (4.37) to (4.3), we introduced additional variables $\xi \in \mathbb{R}$, $\boldsymbol{\eta} \in \mathbb{R}^{m'}$ and applied the algorithms revsbb and sbb_1 to

$$\left|\begin{array}{lll} \text{minimize} \quad z = & -\sigma\xi - (1/2)\mathbf{x}^{\mathsf{T}}\mathbf{C}^{\mathsf{T}}\mathbf{C}^{\mathsf{T}}\mathbf{x} & \\ \text{subject to} & \mathbf{A}'\mathbf{x} + \mathbf{B}'\mathbf{y} + \boldsymbol{\eta} = \mathbf{b}', & (\mathbf{x}, \mathbf{y}) \geq \mathbf{0} \\ & \xi - \mathbf{d}^{\mathsf{T}}\mathbf{y} = \mathbf{0}, & (\xi, \boldsymbol{\eta}) \geq \mathbf{0}, \end{array}\right. \tag{4.38}$$

Note that $\xi \geq 0$ because $\mathbf{d} \geq \mathbf{0}$. The size (m, n, r) of (4.38) is therefore equal to $(m' + 1, m' + n' + 1, r' + 1)$. As for sbb_2, we applied it directly to (4.37) because it uses only the relaxation problem $\mathrm{Q}(\Delta)$, which can be written with the slack variable $\boldsymbol{\eta}$ as follows (see e.g., [19, 17, 55]):

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

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 $\Delta \subset \mathbb{R}^{r'}$.

The branching criterion is $f(\mathbf{x}^*) - z^k \leq \epsilon |f(\mathbf{x}^*)|$ with $\epsilon = 10^{-5}$ in each code. As the subdivision rule of Δ, we adopt bisection of ratio $\alpha = 1/2$ in revsbb and sbb_1, but does 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 with it terminated for every tested instance of (4.37) and generated the same output as revsbb and sbb_1 with the usual bisection rule.

4.5.3 Numerical results

In Figures 4.1–4.4, 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 $[\mathbf{A}', \mathbf{B}']$ is fixed to $(m', n') = (40, 80)$.

Figure 4.1 shows the variation in the average number of branching operations required by each algorithm when σ was fixed to 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 value of 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 4.2, we can conclude that it is more effective in the algorithm revsbb of REVISED_SBB. We also see that the tightening procedures work better for larger value of 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 4.2. For every value of r', the algorithm revsbb surpasses the other two algorithms. In particular, compared with sbb_2, it requires only fortieth part of the CPU seconds. This proves that problem (4.19) 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 4.3 and 4.4 show that the variations of the average number of branching operations and CPU seconds, respectively, required by each code when r' was fixed to 20 and σ was changed in $\{3.0, 3.5, 4.0, 5.0, 7.0, 10.0, 20.0\}$. Unfortunately,

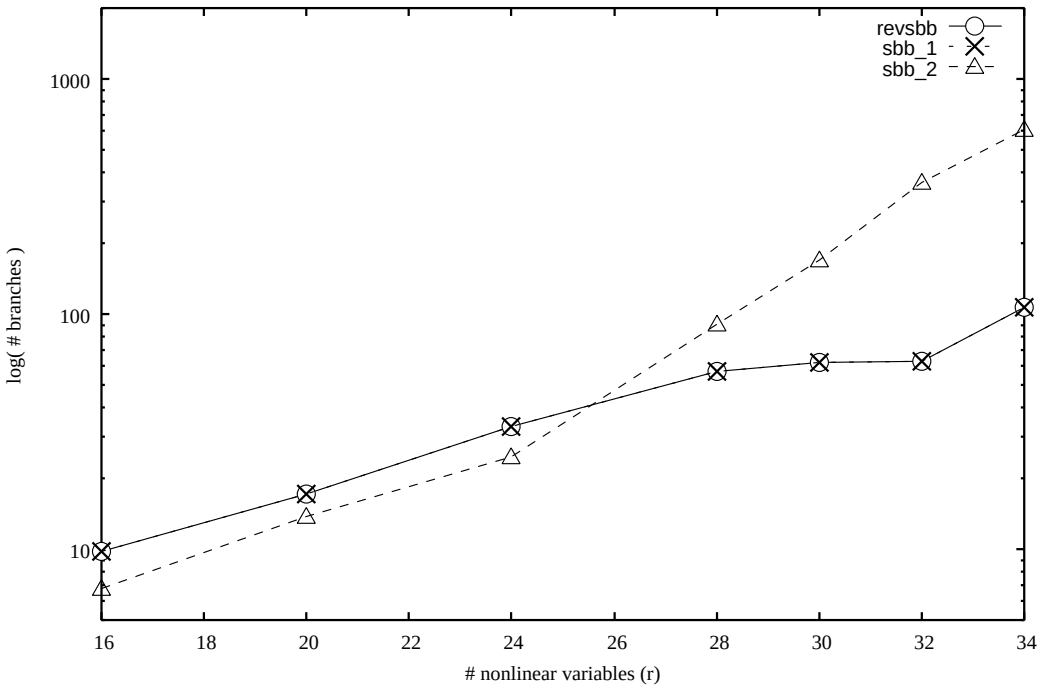

Figure 4.1: Numbers of branching operations when $(m', n', \sigma) = (40, 80, 5.0)$.

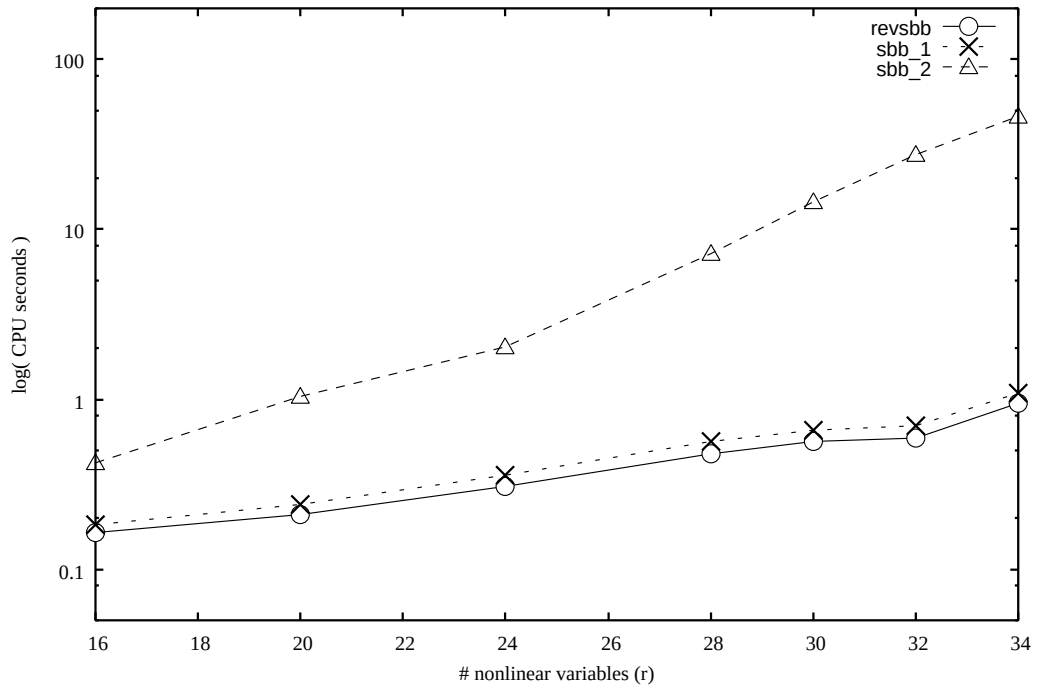

Figure 4.2: CPU seconds when $(m', n', \sigma) = (40, 80, 5.0)$.

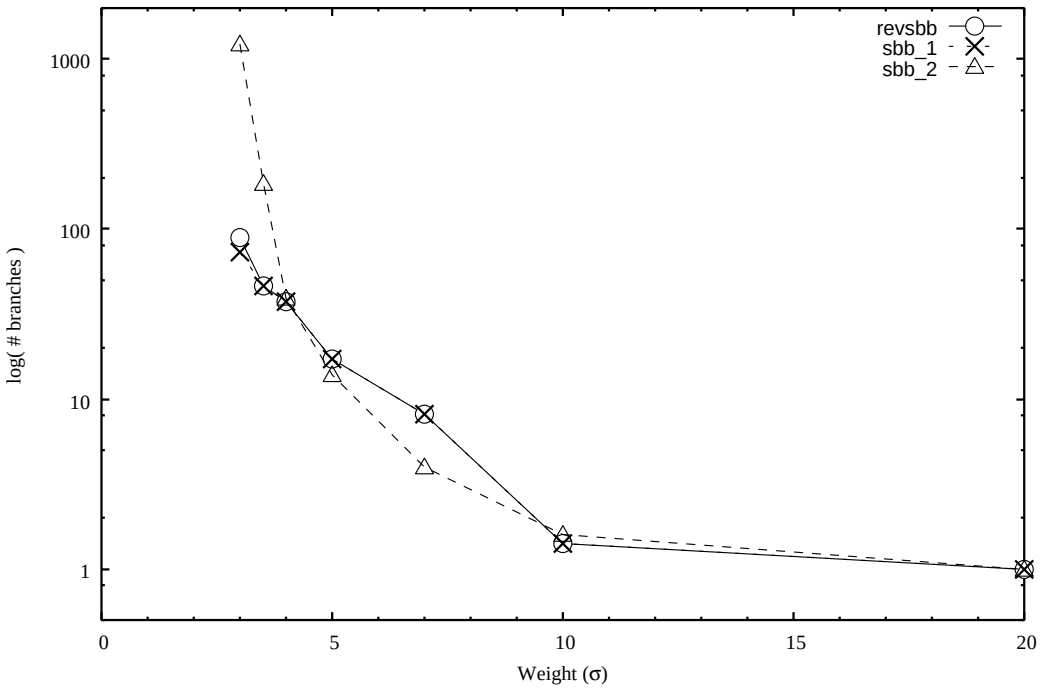

Figure 4.3: Numbers of branching operations when $(m', n', r') = (40, 80, 20)$.

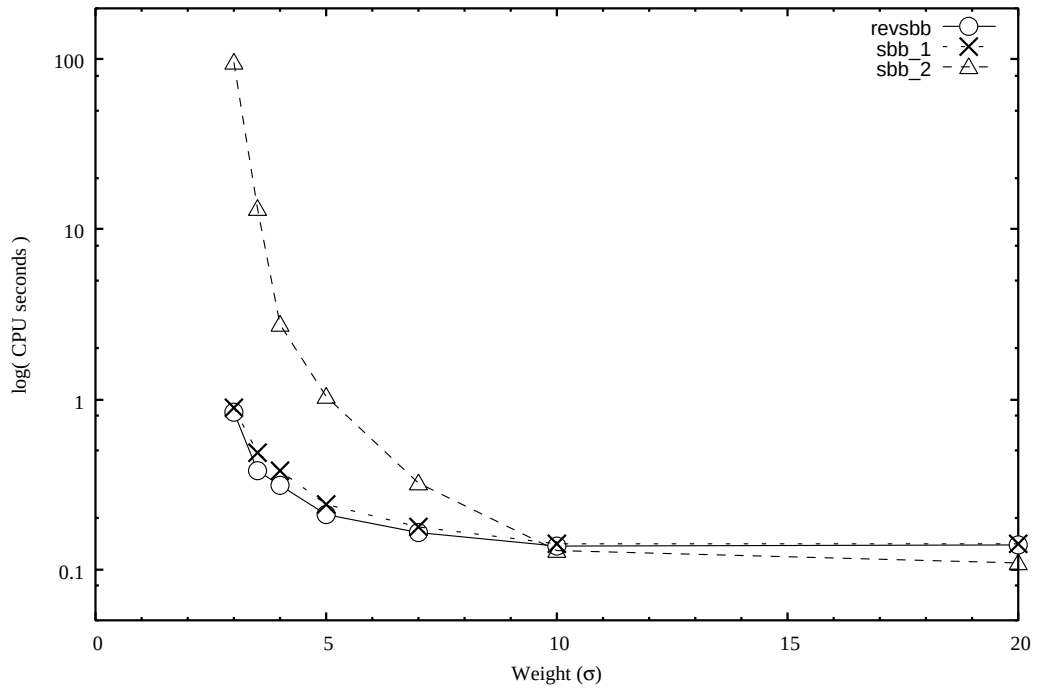

Figure 4.4: CPU seconds when $(m', n', r') = (40, 80, 20)$.

Table 4.1: 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

each algorithm is rather sensitive to variations of 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 owing to the tight lower bound yielded by the Lagrangian relaxation. This, together with the ease of solution to (4.19), 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 (4.37) of larger scale using those two algorithms. The size (m', n') ranged from $(60, 120)$ to $(300, 200)$ and σ was fixed to 5.0. The number of nonlinear variables r' varied from 20% to 50% of all the variables, i.e., the maximum size of (m', n', r') was $(300, 200, 100)$. The computational results are listed in Table 4.1, 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 (4.10) and (4.19), associated with $\overline{\mathrm{Q}}(\Delta)$ and $\widetilde{\mathrm{Q}}(\Delta)$, repectively. Although the test problem (4.37) 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'.

Chapter 5

Simplicial Algorithm for Concave Production-Transportation Problems

5.1 Introduction

The production-transportation problem is a kind of network flow problem and arises when we try to simultaneously optimize production at factories manufacturing a common product, and transportation of finished goods to warehouses with given demands. If the production cost is given by an affine function, the problem is reduced to a Hitchcock problem and can be solved in polynomial time (see [1, 4]). However, due to scale of economy, the production cost is assumed to be a nondecreasing and concave function of production. As a result, the problem can have multiple locally optimal solutions, many of which fail to be globally optimal. From the viewpoint of computational complexity, the production-transportation problem is equivalent to the capacitated minimum concave-cost flow problem, which is known as a typical NP-hard problem (see [9, 10]).

Compared with ordinary multiextremal optimization problems, the production-transportation problem has some favorable characteristics. First, the vari-

ables functioning nonlinearly are much fewer than other variables. If the numbers of factories and warehouses are m and n respectively, the concave production cost is a function of only m variables among a total of $m + mn$ variables. Second, the associated Hitchcock problem is easy to solve and provides a good approximate solution. Each of the algorithms proposed so far exploits at least one of these characteristics. The extreme use of the first characteristic can be seen in a series of parametric algorithms in [26, 30, 31, 56, 57, 58]. The number of factories m is assumed to be a constant in single digit, and locally optimal solutions are enumerated as changing the quantity of production. Algorithms of this class are low-order polynomial or pseudo-polynomial in n. However, they are all exponential in m and serve no practical use when m exceeds around five. Another promising class of algorithms is the branch-and-bound method proposed in [49, 14, 32], where the second characteristic is fully exploited for the bounding operation. In the existing branch-and-bound algorithms, the production cost is further assumed to be a separable function, i.e., a sum of m univariate functions. The feasible production set is subdivided into a set of m-dimensional rectangles to generate subproblems. Under the separability assumption, it is easy to compute a convex envelope, i.e., a maximal affine function underestimating the production cost on the rectangle. Using this convex envelope, the subproblem is linearized into a Hitchcock problem, whose value is a tight lower bound on the optimal value.

In this chapter, we develop a branch-and-bound algorithm to solve this concave cost network flow problem. Unlike the existing algorithms, we do not impose the separability assumption on the production cost. Since the factories manufacture a common product, they would accommodate one another with raw materials. Therefore, the production cost of each factory usually depends upon the production of other factories as well. If the production cost is inseparable, the rectangle subdivision of the feasible production set has no advantage any more. Instead, we propose a simplicial subdivision, which subdivides the feasible production set into a set of simplices. In Section 5.2, we describe the problem settings. Although the simplicial branch-and-bound algorithm is possible to define a convex envelope of

the production cost on each simplex, the network structure needed in efficiently solving the subproblem is damaged by this subdivision scheme. In Section 5.3, we devise some procedures for restoring the network structure of each subproblem and linearize it into a network flow problem giving a lower bound on the optimal value. Section 5.4 is devoted to a report on computational results of comparing those procedures. In Section 5.5, we discuss some concluding remarks.

5.2 Problem Settings

Let M denote the set of m factories and N the set of n warehouses. Also let $E = M \times N$. Then $G = (M, N, E)$ constitutes a bipartite graph of node sets M and N, and arc set E. For each $i \in M$ and $j \in N$, the production capacity of factory i and the demand of warehouse j are u_i and b_j units, respectively, and the cost of shipping a unit from factory i to warehouse j is c_{ij}, where u_i and b_j are both positive integers and c_{ij} is a real number. Note that for the problem to make sense, it is necessary that

$$\sum_{j \in N} b_j \leq \sum_{i \in M} u_i. \tag{5.1}$$

Let $g(\mathbf{y})$ be the total cost of producing y_i units at each factory $i \in M$, where $\mathbf{y} = (y_1, \ldots, y_m)^{\mathsf{T}}$. We assume that g is a nonlinear, nondecreasing and concave function defined on some open convex set including

$$\Delta^1 = \left\{ \mathbf{y} \in \mathbb{R}^m \;\middle|\; \sum_{i \in M} y_i = B, \;\; \mathbf{y} \geq \mathbf{0} \right\}, \tag{5.2}$$

where $B = \sum_{j \in N} b_j$. Letting $\mathbf{x} = (\, x_{ij} \mid (i,j) \in E \,)^{\mathsf{T}}$ denote the flow variables of finished products from factories to warehouses, then our problem is formulated as

follows:

$$\left|\begin{array}{ll} \text{minimize} & z = \displaystyle\sum_{(i,j)\in E} c_{ij}x_{ij} + g(\mathbf{y}) \\ \text{subject to} & \displaystyle\sum_{j\in N} x_{ij} = y_i, \quad i \in M \\ & \displaystyle\sum_{i\in M} x_{ij} = b_j, \quad j \in N \\ & \mathbf{x} \geq \mathbf{0}, \quad \mathbf{0} \leq \mathbf{y} \leq \mathbf{u}, \end{array}\right. \tag{5.3}$$

where $\mathbf{u} = (u_1, \ldots, u_m)^\mathsf{T}$. If we add a source 0 of supply B to the graph G and connect it to each node $i \in M$ through a directed arc $(0, i)$ of capacity u_i, we see that (5.3) is a special class of minimum concave-cost flow problem (see [10]). A major difference from the usual one is that the objective function is not assumed completely separated into univariate functions.

Let S denote the feasible set of (5.3). Since the objective function is continuous and S is a bounded polyhedron, problem (5.3) has an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ as long as (5.1) holds. Moreover, $(\mathbf{x}^*, \mathbf{y}^*)$ is assumed to be a vertex of S because the objective function is concave. As seen above, the set of all constraints is essentially the same as minimum cost flow problems; and hence, the constraint matrix possesses the total unimodularity (see [1] for details). These facts imply the following:

Lemma 5.1. *Under condition (5.1), problem (5.3) has a globally optimal solution* $(\mathbf{x}^*, \mathbf{y}^*)$*, each component of which is an integer.*

When the objective function is inseparable and concave, one of the popular solution methods is the simplicial branch-and-bound algorithm in [19, 55]. In the rest of this section, we will review its basic workings.

If (5.1) holds, any feasible production $\mathbf{y}$ of (5.3) belongs to the $(m-1)$-simplex Δ^1 defined in (5.2). Therefore, no feasible solution to (5.3) is overlooked if we add

the constraint $\mathbf{y} \in \Delta^1$. The resulting problem is then given below for $\Delta = \Delta^1$:

$$\mathrm{P}(\Delta) \left| \begin{array}{lll} \text{minimize} & z = & \displaystyle\sum_{(i,j)\in E} c_{ij}x_{ij} + g(\mathbf{y}) \\ \text{subject to} & & \displaystyle\sum_{j\in N} x_{ij} = y_i, \quad i \in M \\ & & \displaystyle\sum_{i\in M} x_{ij} = b_j, \quad j \in N \\ & & \mathbf{x} \geq \mathbf{0}, \quad \mathbf{y} \leq \mathbf{u}, \quad \mathbf{y} \in \Delta. \end{array} \right.$$

To locate $(\mathbf{x}^*, \mathbf{y}^*)$, the simplicial branch-and-bound algorithm solves this problem recursively, as replacing Δ by simplices Δ' and Δ'' such that

$$\Delta = \Delta' \cup \Delta'', \quad \mathrm{int}(\Delta') \cap \mathrm{int}(\Delta'') = \emptyset, \tag{5.4}$$

where $\mathrm{int}(\,\cdot\,)$ represents the set of relative interior points. The simplex Δ is usually maintained as a convex hull of its m vertices $\mathbf{v}^i$, $i \in M$:

$$\Delta = \left\{ \mathbf{y} \in \mathbb{R}^m \;\middle|\; \mathbf{y} = \sum_{i\in M} \lambda_i \mathbf{v}^i, \;\; \mathbf{e}^{\mathsf{T}}\boldsymbol{\lambda} = 1, \;\; \boldsymbol{\lambda} \geq \mathbf{0} \right\},$$

where $\mathbf{e}$ denotes a vector of ones and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)^{\mathsf{T}}$. For the initial simplex Δ^1 we have $\mathbf{v}^i = B\mathbf{e}^i$, where $\mathbf{e}^i$ denotes the ith unit vector. If we select an edge of Δ, say $\mathbf{v}^p$–$\mathbf{v}^q$, and divide it at a midpoint $\mathbf{v} := (1-\mu)\mathbf{v}^p + \mu\mathbf{v}^q$ for a fixed ratio $\mu \in (0, 1/2]$, we can immediately generate Δ' and Δ'' as

$$\Delta' = \left\{ \mathbf{y} \in \mathbb{R}^m \;\middle|\; \mathbf{y} = \lambda_p \mathbf{v} + \sum_{i\in M\setminus\{p\}} \lambda_i \mathbf{v}^i, \;\; \mathbf{e}^{\mathsf{T}}\boldsymbol{\lambda} = 1, \;\; \boldsymbol{\lambda} \geq \mathbf{0} \right\}$$

$$\Delta'' = \left\{ \mathbf{y} \in \mathbb{R}^m \;\middle|\; \mathbf{y} = \lambda_q \mathbf{v} + \sum_{i\in M\setminus\{q\}} \lambda_i \mathbf{v}^i, \;\; \mathbf{e}^{\mathsf{T}}\boldsymbol{\lambda} = 1, \;\; \boldsymbol{\lambda} \geq \mathbf{0} \right\}.$$

We refer to this division rule as *bisection of ratio μ on edge $\mathbf{v}^p$–$\mathbf{v}^q$*.

The simplicial subdivision (5.4) has an advantage over other subdivision schemes in computing a tight lower bound of g. Since each point $\mathbf{y} \in \Delta$ can be represented as $\mathbf{y} = \sum_{i\in M} \lambda_i \mathbf{v}^i$ for some $\boldsymbol{\lambda} \geq \mathbf{0}$ such that $\mathbf{e}^{\mathsf{T}}\boldsymbol{\lambda} = 1$, a lower bound of g at $\mathbf{y}$ is given simply by $\overline{g}(\mathbf{y}) = \sum_{i\in M} \lambda_i g(\mathbf{v}^i)$. This function $\overline{g}$ is an affine function of

$\mathbf{y}$, which agrees with g at m vertices of Δ, and known as a convex envelope of g, i.e., a maximal convex function underestimating g on Δ. On the other hand, there is a difficulty in the implementation against our problem (5.3). Except for the initial one, the additional constraint $\mathbf{y} \in \Delta$ damages the network structure of $\mathrm{P}(\Delta)$, which is essential to efficient computation of $\overline{z}(\Delta)$. In the subsequent section, we develop some procedures for overcoming this difficulty.

5.3 Finite Simplicial Algorithm

The key to efficiency of the simplicial branch-and-bound algorithm is mainly held by the bounding operation of Step 2 of Standard_SBB in Section 2.1. To compute a lower bound $\overline{z}(\Delta)$, we solve a relaxation problem of $\mathrm{P}(\Delta)$, where the concave function g is replaced by its convex envelope $\overline{g}$ on Δ. Usually, instead of representing $\overline{g}$ explicitly as a function of $\mathbf{y}$, we eliminate $\mathbf{y}$ altogether from the relaxation problem by substituting $\mathbf{y} = \sum_{i \in M} \lambda_i \mathbf{v}^i$ into the constraints as well (see [19, 55] for details). This approach is handy but destroys the network structure completely. Here, we take an alternative approach which keeps the damage as small as possible.

5.3.1 Linear programming relaxation

For m vertices $\mathbf{v}^i$, $i \in M$, of Δ, let

$$\mathbf{V} = [\mathbf{v}^1, \ldots, \mathbf{v}^m], \quad \mathbf{w} = [g(\mathbf{v}^1), \ldots, g(\mathbf{v}^m)].$$

Since $\mathbf{v}^i$'s are linearly independent if they are generated according to the bisection rule, we can uniquely identify $\boldsymbol{\lambda} = \mathbf{V}^{-1}\mathbf{y}$ for any $\mathbf{y} \in \Delta$. Substituting it to $\overline{g}(\mathbf{y}) = \sum_{i \in M} \lambda_i g(\mathbf{v}^i)$, we have

$$\overline{g}(\mathbf{y}) = \mathbf{w}\mathbf{V}^{-1}\mathbf{y}, \quad \forall \mathbf{y} \in \Delta.$$

Similarly, we can rewrite the simplex Δ as follows:

$$\Delta = \{\, \mathbf{y} \in \mathbb{R}^m \mid \mathbf{e}^{\mathsf{T}}\mathbf{V}^{-1}\mathbf{y} = 1, \;\; \mathbf{V}^{-1}\mathbf{y} \geq \mathbf{0} \,\}.$$

Note that $\mathbf{e}^\mathsf{T}\mathbf{V}^{-1}\mathbf{y} = 1$ is satisfied by any feasible production $\mathbf{y}$ of (5.3). Since each $\mathbf{v}^i$ and the feasible production $\mathbf{y}$ belong to the initial simplex defined by (5.2), the equalities $\mathbf{e}^\mathsf{T}\mathbf{v}^i = B$ and $\mathbf{e}^\mathsf{T}\mathbf{y} = B$ hold. Then we have

$$\mathbf{e}^\mathsf{T}\mathbf{V}^{-1}\mathbf{y} = (1/B)\mathbf{e}^\mathsf{T}\mathbf{y} = 1.$$

We can therefore replace the constraint $\mathbf{y} \in \Delta$ of $\mathrm{P}(\Delta)$ by $\mathbf{V}^{-1}\mathbf{y} \geq \mathbf{0}$. Furthermore, replacing g by $\overline{g}$, we have a linear programming problem:

$$\mathrm{RP}_1(\Delta) \left| \begin{array}{lll} \text{minimize} & z = & \displaystyle\sum_{(i,j)\in E} c_{ij}x_{ij} + \mathbf{w}\mathbf{V}^{-1}\mathbf{y} \\ \text{subject to} & & \displaystyle\sum_{j\in N} x_{ij} = y_i, \quad i \in M \\ & & \displaystyle\sum_{i\in M} x_{ij} = b_j, \quad j \in N \\ & & \mathbf{x} \geq \mathbf{0}, \quad \mathbf{y} \leq \mathbf{u}, \quad \mathbf{V}^{-1}\mathbf{y} \geq \mathbf{0}. \end{array} \right.$$

Let $(\mathbf{x}^1, \mathbf{y}^1)$ be an optimal solution when $\mathrm{RP}_1(\Delta)$ is feasible, and let

$$z^1(\Delta) = \begin{cases} \displaystyle\sum_{(i,j)\in E} c_{ij}x^1_{ij} + \mathbf{w}\mathbf{V}^{-1}\mathbf{y}^1, & \text{if } \mathrm{RP}_1(\Delta) \text{ is feasible} \\ +\infty, & \text{otherwise.} \end{cases}$$

Lemma 5.2. *If $z^1(\Delta) = +\infty$, then subproblem $P(\Delta)$ is infeasible. Otherwise, $P(\Delta)$ has an optimal solution of value $z(\Delta)$, and we have*

$$z^1(\Delta) \leq z(\Delta).$$

Proof. Both feasible sets of $\mathrm{P}(\Delta)$ and $\mathrm{RP}_1(\Delta)$ coincide with $S \cap \Delta$, where S denotes the feasible set of (5.3). For any $(\mathbf{x}, \mathbf{y}) \in S \cap \Delta$, we have

$$\sum_{(i,j)\in E} c_{ij}x_{ij} + \mathbf{w}\mathbf{V}^{-1}\mathbf{y} \leq \sum_{(i,j)\in E} c_{ij}x_{ij} + g(\mathbf{y}), \tag{5.5}$$

because $\overline{g}(\mathbf{y}) = \mathbf{w}\mathbf{V}^{-1}\mathbf{y}$ is a convex envelope of g on Δ. □

We see from this lemma that the optimal value $z^1(\Delta)$ of $\mathrm{RP}_1(\Delta)$ can serve as the lower bound $\overline{z}(\Delta)$ on the value of $\mathrm{P}(\Delta)$. Moreover, since any feasible solution of $\mathrm{RP}_1(\Delta)$ is feasible to $\mathrm{P}(\Delta)$, we can update the incumbent $(\mathbf{x}^\circ, \mathbf{y}^\circ)$ by $(\mathbf{x}^1, \mathbf{y}^1)$. The only drawback of $\mathrm{RP}_1(\Delta)$ is that the constraint $\mathbf{V}^{-1}\mathbf{y} \geq \mathbf{0}$ still spoils the network structure and prevents us from applying efficient network flow algorithms.

5.3.2 Network flow relaxation

The easiest way to restore the network structure of $\mathrm{RP}_1(\Delta)$ is to drop the constraint $\mathbf{V}^{-1}\mathbf{y} \geq \mathbf{0}$:

$$\mathrm{RP}_2(\Delta) \left| \begin{array}{lll} \text{minimize} & z = & \displaystyle\sum_{(i,j)\in E} c_{ij}x_{ij} + \mathbf{w}\mathbf{V}^{-1}\mathbf{y} \\ \text{subject to} & & \displaystyle\sum_{j\in N} x_{ij} = y_i, \quad i \in M \\ & & \displaystyle\sum_{i\in M} x_{ij} = b_j, \quad j \in N \\ & & \mathbf{x} \geq \mathbf{0}, \quad \mathbf{0} \leq \mathbf{y} \leq \mathbf{u}. \end{array} \right.$$

Then we can eliminate $\mathbf{y}$ and have a Hitchcock problem:

$$\left| \begin{array}{lll} \text{minimize} & z = & \displaystyle\sum_{(i,j)\in E} \overline{c}_{ij}x_{ij} \\ \text{subject to} & & \displaystyle\sum_{j\in N} x_{ij} \leq u_i, \quad i \in M \\ & & \displaystyle\sum_{i\in M} x_{ij} = b_j, \quad j \in N \\ & & \mathbf{x} \geq \mathbf{0}, \end{array} \right. \tag{5.6}$$

where $\overline{c}_{ij} = c_{ij} + \mathbf{w}\mathbf{V}^{-1}\mathbf{e}^i$ and $\mathbf{e}^i$ denotes the ith unit vector. Since the feasible set does not depend on Δ, problem (5.6) always has an optimal solution $\mathbf{x}^2$ under condition (5.1). It is well known that the number of arithmetic operations needed to compute $\mathbf{x}^2$ is a lower order polynomial in m and n (see [1, 4]). The optimal value $z^2(\Delta) := \sum_{(i,j)\in E} \overline{c}_{ij}x^2_{ij}$ obviously exceeds neither $z^1(\Delta)$ nor $z(\Delta)$, and hence can serve as the lower bound $\overline{z}(\Delta)$ on the value of $\mathrm{P}(\Delta)$. Let $y^2_i = \sum_{j\in N} x^2_{ij}$ for each $i \in M$. Then $(\mathbf{x}^2, \mathbf{y}^2)$ is a feasible solution to the target problem (5.3), though it might be infeasible to $\mathrm{P}(\Delta)$. Thereby, we can update the incumbent $(\mathbf{x}^\circ, \mathbf{y}^\circ)$.

The relaxation problem $\mathrm{RP}_2(\Delta)$ fairly meets our requirements. Unfortunately, however, the removal of $\mathbf{V}^{-1}\mathbf{y} \geq \mathbf{0}$ degrades the quality of the lower bound. To improve the lower bound, we need to retighten the constraints.

Let us introduce $2m$ numbers:

$$s_i = \lceil \min\{\, y_i \mid \mathbf{y} \in \Delta \,\} \rceil, \quad t_i = \min\{\, \lfloor \max\{\, y_i \mid \mathbf{y} \in \Delta \,\} \rfloor, u_i \,\}, \quad i \in M, \tag{5.7}$$

where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ represent the integers obtained by rounding up and down, respectively. Also let $\mathbf{s} = (s_1, \ldots, s_m)^\mathsf{T}$ and $\mathbf{t} = (t_1, \ldots, t_m)^\mathsf{T}$. Unless $\mathbf{y} \in \Delta$ is an integral vector, it might not satisfy $\mathbf{s} \leq \mathbf{y} \leq \mathbf{t}$. However, we see from Lemma 5.1 that at least one optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ is integral, and satisfies $\mathbf{s} \leq \mathbf{y}^* \leq \mathbf{t}$ if $\mathbf{y}^* \in \Delta$. Then, even if we replace the constraint $\mathbf{y} \in \Delta$ in $\mathrm{RP}_1(\Delta)$ by $\mathbf{s} \leq \mathbf{y} \leq \mathbf{t}$, no integral optimal solution to $\mathrm{P}(\Delta)$ is overlooked. Let us denote the resulting problem by

$$\mathrm{RP}_3(\Delta) \left| \begin{array}{lll} \text{minimize} & z = \displaystyle\sum_{(i,j)\in E} c_{ij}x_{ij} + \mathbf{w}\mathbf{V}^{-1}\mathbf{y} \\ \text{subject to} & \displaystyle\sum_{j\in N} x_{ij} = y_i, & i \in M \\ & \displaystyle\sum_{i\in M} x_{ij} = b_j, & j \in N \\ & \mathbf{x} \geq \mathbf{0}, \quad \mathbf{s} \leq \mathbf{y} \leq \mathbf{t}. \end{array} \right.$$

Let $(\mathbf{x}^3, \mathbf{y}^3)$ be an optimal solution when $\mathrm{RP}_3(\Delta)$ is feasible, and let

$$z^3(\Delta) = \begin{cases} \displaystyle\sum_{(i,j)\in E} c_{ij}x^3_{ij} + \mathbf{w}\mathbf{V}^{-1}\mathbf{y}^3, & \text{if } \mathrm{RP}_3(\Delta) \text{ is feasible} \\ +\infty, & \text{otherwise.} \end{cases}$$

Note that $z^3(\Delta)$ is not always a lower bound on $z^1(\Delta)$, nor even on $z(\Delta)$, because there is no inclusive relation between the sets Δ and $\{\, \mathbf{y} \mid \mathbf{s} \leq \mathbf{y} \leq \mathbf{t} \,\}$. However, since the target problem (5.3) has an integral optimal solution, the use of $z^3(\Delta)$ as the lower bound $\overline{z}(\Delta)$ in the branch-and-bound algorithm is justified if we understand that the integral constraint on $(\mathbf{x}, \mathbf{y})$ is hidden in (5.3) and its subproblem $\mathrm{P}(\Delta)$.

Lemma 5.3. *If $z^3(\Delta) = +\infty$, then subproblem $P(\Delta)$ has no integral feasible solution. Otherwise, let $\widetilde{z}(\Delta)$ denote the value of the best integral solution to $P(\Delta)$. Then we have*

$$z^2(\Delta) \leq z^3(\Delta) \leq \tilde{z}(\Delta).$$

Proof. Since all integral points in Δ satisfy $\mathbf{s} \leq \mathbf{y} \leq \mathbf{t}$, problem $\mathrm{P}(\Delta)$ has no integral feasible solution if $\mathrm{RP}_3(\Delta)$ is infeasible. The rest follows from (5.5) and the inclusive relation between the feasible sets of three problems. □

Furthermore, $\mathrm{RP}_3(\Delta)$ can be transformed into a minimum cost flow problem similar to the target problem (5.3). We construct the underlying network as follows. First, we introduce a source 0 of supply $\sum_{i\in M} t_i$ and a sink n' of demand $\sum_{i\in M} t_i - B$ to the graph G. Then, we connect these auxiliary nodes, 0 and n', respectively to each node $i \in M$ with directed arcs $(0, i)$ of capacity t_i and (i, n') of capacity $t_i - s_i$. In this network, it is easy to see that the flow on each arc $(0, i)$, $i \in M$, is equal to t_i for every feasible flow. Therefore, the difference between t_i and the flow on arc (i, n') gives the value of y_i in $\mathrm{RP}_3(\Delta)$. Since the objective function is linear, we can solve this network flow problem in polynomial time of m and n (see [1]). Again, $(\mathbf{x}^3, \mathbf{y}^3)$ might be infeasible to $\mathrm{P}(\Delta)$, but is feasible to (5.3) and can be used for the incumbent update.

The network flow relaxation problem $\mathrm{RP}_3(\Delta)$ has been drawn by exploiting the integrality of an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ to the target problem (5.3). We can use a similar idea to terminate the simplicial branch-and-bound algorithm within finite iterations. Using $\mathbf{s}$ and $\mathbf{t}$ defined in (5.7), we can see that Δ contains just one integral feasible production $\tilde{\mathbf{y}}$ of (5.3) if and only if

$$\text{(a) } s_i = t_i \text{ for all } i \in M \quad \text{and} \quad \sum_{i\in M} s_i = \sum_{i\in M} t_i = B;$$

and Δ contains no integral feasible production of (5.3) if and only if

$$\text{(b) } s_i > t_i \text{ for some } i \in M, \quad \text{and/or} \quad \text{(c) } \sum_{i\in M} s_i > B \text{ or } \sum_{i\in M} t_i < B.$$

In the case of (a), we can compute the value of the best integral solution to $\mathrm{P}(\Delta)$ by solving $\mathrm{P}(\Delta)$ with fixed $\tilde{\mathbf{y}}$, which is reduced to a Hitchcock problem. On the other hand, in the case of (b) and/or (c), we need not solve any relaxation problem $\mathrm{RP}_k(\Delta)$, $k = 1, 2, 3$. In either case, each nested simplex from Δ does not contain any integral feasible production of (5.3) except $\tilde{\mathbf{y}}$. Therefore, we can stop the branching operation and can save computational time.

As seen in Section 2.1, when the algorithm does not terminate, it generates an infinite sequence of nested simplices $\{ \Delta^r \mid r = 1, \dots \}$, which converges to a

singleton if we apply the bisection rule on the longest edge of Δ. Let

$$d(\Delta) = \max\{\, \|\mathbf{v}^i - \mathbf{v}^j\| \mid i < j,\ \ i, j \in M \,\}.$$

Then we have $d(\Delta^r) \geq d(\Delta^{r+1})$ for each r, and $d(\Delta^r) \to 0$ as $r \to \infty$. In this sequence, if $d(\Delta^r) < \sqrt{2}$ holds, then Δ^r contains at most one integral point, and Δ^r must satisfy one of the stopping criteria (a)–(c). This implies that if we apply these criteria (a)–(c), each nested sequence $\{\, \Delta^r \mid r = 1, \dots \}$ generated by the algorithm is finite. Hence, even by adopting the *depth first rule* to select Δ, we can guarantee the finite convergence of the algorithm. The depth first rule selects Δ most recently added to $\mathcal{D}$, and requires less memory than the best bound rule.

We are now ready to give a detailed description of the algorithm, where $\mathrm{conv}(\mathbf{V})$ denotes the convex hull of the columns of $\mathbf{V}$:

algorithm Simplicial_BB
begin
  for $i \in M$ do $\mathbf{v}^i := B\mathbf{e}^i$;
  $\mathbf{V} := [\mathbf{v}^1, \dots, \mathbf{v}^m]$; $\Delta := \mathrm{conv}(\mathbf{V})$; $\mathcal{D} := \{\Delta\}$; $z^\circ := +\infty$;
  while $\mathcal{D} \neq \emptyset$ do begin
    /* Step 1. (best bound or depth first) */
    select $\Delta \in \mathcal{D}$ and set $\mathcal{D} := \mathcal{D} \setminus \{\Delta\}$;
    define a subproblem P(Δ);
    /* Step 2. (bounding operation) */
    for $i \in M$ do begin
      $s_i := \lceil \min\{\, y_i \mid \mathbf{y} \in \Delta \,\} \rceil$; $t_i := \min\{\, \lfloor \max\{\, y_i \mid \mathbf{y} \in \Delta \,\} \rfloor, u_i \,\}$
    end;
    if $\mathbf{s} = \mathbf{t}$ and $\mathbf{e}^{\mathsf{T}}\mathbf{s} = B = \mathbf{e}^{\mathsf{T}}\mathbf{t}$ then begin
      $\tilde{\mathbf{y}} := \mathbf{s}$;
      solve P(Δ) with fixed $\tilde{\mathbf{y}}$ and obtain an optimal solution $\tilde{\mathbf{x}}$;
      if $\sum_{(i,j)\in E} c_{ij}\tilde{x}_{ij} + g(\tilde{\mathbf{y}}) < z^\circ$ then begin
        $(\mathbf{x}^\circ, \mathbf{y}^\circ) := (\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$; $z^\circ := \sum_{(i,j)\in E} c_{ij}\tilde{x}_{ij} + g(\tilde{\mathbf{y}})$

```
        end
    else if s ≤ t and e^T s ≤ B ≤ e^T t then
        w := [g(v^1), ..., g(v^m)]; ḡ(y) := wV^{-1}y;
        define a relaxed problem RP_k(Δ) of P(Δ) using ḡ;
        solve RP_k(Δ) and obtain a lower bound z̄(Δ) := z^k(Δ);
        let (x^k, y^k) denote an optimal solution to RP_k(Δ);
        if Σ_{(i,j)∈E} c_ij x^k_ij + g(y^k) < z° then begin
            (x°, y°) := (x^k, y^k); z° := Σ_{(i,j)∈E} c_ij x^k_ij + g(y^k)
        end;
        if z̄(Δ) < z° then begin
                                    /* Step 3. (branching operation; μ ∈ (0, 1/2]) */
            select the longest edge v^p–v^q of Δ and let v := (1 − μ)v^p + μv^q;
            V' := [v^1, ..., v^{p−1}, v, v^{p+1}, ..., v^m];
            V'' := [v^1, ..., v^{q−1}, v, v^{q+1}, ..., v^m];
            Δ' := conv(V'); Δ'' := conv(V''); D := D ∪ {Δ', Δ''}
        end
    end
end;
(x*, y*) := (x°, y°)
end;
```

Theorem 5.4. *The algorithm* Simplicial_BB *terminates after finitely many iterations, and yields a globally optimal solution* $(\mathbf{x}^*, \mathbf{y}^*)$ *to (5.3).*

Proof. If the algorithm terminates, it is obvious that it yields an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ to (5.3). Let us show that it terminates in finite time, assuming $\mu = 1/2$ for simplicity. We can prove other cases similarly.

If we apply the algorithm to (5.3), it generates a branching tree, each node of which corresponds to a subproblem $\mathrm{P}(\Delta)$. If we trace the tree from an arbitrary node $\mathrm{P}(\Delta^r)$ to the root (5.3), we have a nested sequence $\{\Delta^1, \ldots, \Delta^r\}$, where Δ^1 denotes the initial simplex given by (5.2). As we have seen, such a sequence

has a finite length because the algorithm backtracks along the branching tree if $d(\Delta^r) < \sqrt{2}$. More precisely, the length is bounded by $m \log B$ when $\mu = 1/2$, since Δ^1 has m edges of length $\sqrt{2}B$. Therefore, the branching tree contains a total of $O(2^{m \log B})$ nodes at most. This implies that the algorithm solves $O(2^{m \log B})$ linear programming problems $\mathrm{RP}_k(\Delta)$'s, even in the worst case, and yields $(\mathbf{x}^*, \mathbf{y}^*)$ optimal for (5.3). □

5.4 Computational Results

Let us report numerical results of comparing the algorithms of using $z^1(\Delta)$, $z^2(\Delta)$ and $z^3(\Delta)$ as the lower bound $\overline{z}(\Delta)$ on randomly generated instances of problem (5.3). We refer to the algorithms by SBB1, SBB2 and SBB3, respectively.

Each instance is generated in the following manner: c_{ij}'s are integers drawn from the uniform distribution on $[1, 10]$; u_i's and b_j's are fixed to 200 and $\lfloor (\sum_{i \in M} 0.75 u_i)/n \rfloor$, respectively; and the concave production cost is defined by

$$g(\mathbf{y}) = \gamma \sum_{k \in M} \beta_k \sqrt{\sum_{i \in M} \alpha_{ki} y_i},$$

where γ is selected from $\{0.1, 1.0, 10.0\}$, α_{ki} and β_k are random numbers such that $\alpha_{ki} \in (1.0, 2.0)$ if $k = i$, otherwise $\alpha_{ki} \in (0.0, 1.0)$, and $\beta_k \in [10.0, 20.0]$. The size of m range from 4 to 7, and n is set to each of $\{10m, 20m, 40m, 80m\}$.

The algorithms were coded mainly in GNU Octave (version 2.1.34) [43], a Matlab-like computational tool, according to the description in Section 5.3. We also coded the revised simplex algorithm for solving the relaxation problem $\mathrm{RP}_1(\Delta)$, and the successive shortest path algorithm for solving $\mathrm{RP}_2(\Delta)$ and $\mathrm{RP}_3(\Delta)$ (see [1, 4]). Neither algorithm is polynomial, but we improved the efficiency by exploiting an optimal solution to the preceding relaxation problem as the initial solution. While Matlab-like tools are powerful for matrix computation due to binary libraries for linear algebra, they are generally poor at other operations, especially at processing discrete structures. We therefore took the way to call a

shortest path procedure coded in C++ (GCC version 2.96) from the successive shortest path program of Octave. Each program code of SBB1, SBB2 and SBB3 adopted the depth first rule, $\mu = 1/2$, and solved ten instances for each (m, n, γ) on a Linux workstation (Linux 2.4.18, Itanium 2 processor, 1GHz).

Tables 5.1–5.3 show the results, each for $\gamma \in \{0.1, 1.0, 10.0\}$. The average CPU seconds (*time*) and the average number of branching operations (*branches*) taken by SBB1, SBB2 and SBB3 are listed in each row. The worst figures are also given in brackets. These figures are omitted to list if there were instances not solved within 10,000 seconds. We see from the tables that both SBB2 and SBB3 are superior to SBB1 in CPU seconds for all (m, n, γ) except $(7, 70, 1.0)$, even though they require many more branching operations than SBB1. This implies that the computational burden of solving each of $\mathrm{RP}_2(\Delta)$ and $\mathrm{RP}_3(\Delta)$ is low enough to cancel the dominance of $z^1(\Delta)$ over $z^2(\Delta)$ and $z^3(\Delta)$. Since the number of branching operations required by SBB3 is rather less than that by SBB2, we can conclude that $z^3(\Delta)$ is tightened sufficiently from $z^2(\Delta)$. Also, we should remark that the gap of performance among SBB1, SBB2 and SBB3 tends to widen as the size of n increases for each m. As to the effect of change in γ, we can see that it is fairly mild if we compare three tables. The algorithm Simplicial_BB is therefore expected to solve still more highly nonlinear problems when m is less than seven.

5.5 Concluding Remarks

The production-transportation problem (5.3) can be thought of as an example of the simplest supply chain models. However, if we assume the production cost to be an inseparable concave function of a total amount of production, it is not so easy to figure out a globally optimal solution even for small-scale problems. To solve this intractable problem, we proposed a simplicial branch-and-bound algorithm, Simplicial_BB. Unlike the usual simplicial algorithms, our algorithm always maintains the network structure possessed by the original problem in the course of computation. This causes rather rapid growth of branching trees but enables

Table 5.1: Computational results when $\gamma = 0.1$

	SBB1		SBB2		SBB3	
$m\times\ \ n$	*time*	*branches*	*time*	*branches*	*time*	*branches*
4× 40	0.277 (0.389)	14.2 (29)	0.076 (0.113)	27.0 (35)	0.075 (0.113)	26.8 (41)
4× 80	1.193 (1.660)	19.8 (31)	0.106 (0.191)	35.8 (65)	0.103 (0.186)	35.6 (65)
4×160	7.772 (9.799)	25.2 (31)	0.133 (0.177)	39.0 (51)	0.130 (0.174)	38.8 (51)
4×320	58.71 (70.31)	30.0 (35)	0.269 (0.369)	57.8 (79)	0.246 (0.274)	47.0 (51)
5× 50	1.002 (2.480)	48.4 (137)	0.538 (0.905)	183.6 (317)	0.474 (0.745)	163.8 (255)
5×100	4.565 (7.231)	48.8 (81)	0.571 (0.799)	175.6 (255)	0.534 (0.738)	166.2 (227)
5×200	31.28 (41.00)	56.2 (73)	0.750 (1.050)	192.4 (267)	0.707 (0.962)	178.8 (239)
5×400	272.1 (464.1)	85.2 (137)	1.943 (2.648)	335.8 (435)	1.645 (2.327)	257.2 (369)
6× 60	5.262 (9.480)	184.4 (381)	2.365 (3.978)	848.8 (1,447)	2.299 (3.851)	841.8 (1,427)
6×120	31.38 (56.58)	248.2 (433)	3.417 (4.829)	1,060 (1,407)	3.273 (4.474)	1,041 (1,365)
6×240	195.1 (475.7)	220.6 (485)	8.505 (16.62)	1,811 (3,383)	5.455 (10.57)	1,099 (2,035)
6×480	1,482 (2,439)	265.8 (445)	17.44 (26.77)	2,358 (3,565)	10.48 (16.79)	1,283 (2,027)
7× 70	28.42 (70.04)	677.4 (1,537)	19.66 (29.80)	6,512 (9,939)	17.95 (25.78)	5,981 (8,403)
7×140	169.8 (363.2)	801.4 (1,653)	34.24 (64.06)	9,157 (16,671)	29.75 (50.31)	8,088 (13,341)
7×280	1,435 (3,678)	993.0 (2,075)	46.57 (79.88)	8,947 (14,881)	42.44 (68.37)	8,100 (12,651)
7×560	− (−)	− (−)	259.1 (678.6)	26,617 (67,753)	104.5 (200.4)	9,500 (17,997)

Table 5.2: Computational results when $\gamma = 1.0$

	SBB1		SBB2		SBB3	
$m \times n$	*time*	*branches*	*time*	*branches*	*time*	*branches*
4× 40	0.430 (0.588)	34.2 (53)	0.177 (0.264)	67.4 (103)	0.158 (0.221)	58.0 (83)
4× 80	1.995 (4.180)	56.0 (145)	0.251 (0.561)	89.6 (201)	0.225 (0.473)	79.0 (167)
4×160	11.13 (16.41)	53.6 (83)	0.265 (0.454)	79.4 (133)	0.254 (0.366)	74.2 (105)
4×320	91.95 (113.8)	62.4 (73)	0.633 (0.791)	137.6 (171)	0.459 (0.636)	88.0 (127)
5× 50	2.334 (5.190)	144.4 (345)	1.933 (3.612)	697.0 (1,271)	1.337 (2.533)	463.4 (867)
5×100	9.596 (13.95)	150.2 (221)	1.820 (2.975)	591.0 (955)	1.411 (2.004)	440.4 (621)
5×200	71.10 (163.2)	184.2 (465)	2.651 (4.063)	700.2 (1,029)	1.948 (3.188)	490.8 (783)
5×400	753.8 (1,348)	297.0 (567)	7.277 (13.24)	1,270 (2,337)	4.228 (7.424)	661.0 (1,133)
6× 60	12.65 (25.45)	511.0 (889)	6.587 (16.50)	2,376 (5,761)	5.154 (8.041)	1,850 (2,841)
6×120	79.38 (139.0)	688.6 (1,119)	8.365 (12.49)	2,616 (3,729)	7.484 (10.60)	2,303 (3,099)
6×240	556.2 (1,301)	743.4 (1,649)	39.78 (65.01)	8,983 (14,933)	16.06 (24.20)	3,288 (4,809)
6×480	5,326 (8,455)	1,027 (1,559)	95.72 (178.7)	13,143 (26,907)	34.78 (51.55)	4,225 (6,255)
7× 70	75.69 (174.6)	1,971 (4,599)	102.3 (204.4)	33,421 (67,281)	60.78 (97.37)	19,453 (30,453)
7×140	574.7 (1,164)	2,955 (5,711)	214.2 (460.8)	57,208 (122,727)	96.92 (168.1)	25,222 (42,921)
7×280	4,272 (6,886)	3,285 (5,995)	306.4 (548.3)	57,941 (104,947)	142.6 (204.7)	25,957 (36,763)
7×560	− (−)	− (−)	− (−)	− (−)	614.0 (2,518)	55,686 (222,799)

Table 5.3: Computational results when $\gamma = 10.0$

	SBB1		SBB2		SBB3	
$m\times\ \ n$	*time*	*branches*	*time*	*branches*	*time*	*branches*
4× 40	0.875 (1.679)	74.8 (151)	0.313 (0.515)	117.0 (197)	0.266 (0.452)	99.2 (171)
4× 80	3.700 (9.218)	96.6 (275)	0.402 (1.085)	140.2 (377)	0.337 (0.962)	116.2 (341)
4×160	25.35 (41.04)	104.2 (163)	0.509 (0.969)	138.6 (263)	0.452 (0.832)	121.2 (227)
4×320	221.0 (386.9)	121.6 (201)	0.971 (2.006)	158.2 (299)	0.861 (1.476)	151.2 (267)
5× 50	4.374 (8.649)	265.6 (585)	2.812 (5.921)	977.6 (2,003)	1.640 (3.037)	572.0 (1,067)
5×100	16.26 (34.73)	223.2 (541)	2.511 (4.777)	803.8 (1,513)	1.713 (3.305)	537.0 (1,027)
5×200	151.0 (216.3)	330.4 (469)	3.439 (4.812)	821.0 (1,235)	2.475 (3.346)	580.4 (757)
5×400	1,331 (1,973)	301.4 (397)	4.510 (9.583)	549.6 (1,241)	3.440 (6.079)	417.4 (769)
6× 60	25.52 (45.64)	860.8 (1,473)	8.973 (13.70)	3,102 (4,635)	6.729 (9.543)	2,356 (3,299)
6×120	132.0 (214.9)	802.0 (1,135)	8.935 (12.56)	2,572 (3,513)	7.931 (10.86)	2,331 (3,157)
6×240	1,543 (3,954)	1,692 (4,781)	49.98 (135.1)	9,915 (24,789)	24.69 (69.10)	4,771 (13,239)
6×480	– (–)	– (–)	93.65 (309.7)	8,309 (26,343)	34.88 (84.34)	3,033 (7,155)
7× 70	194.4 (516.0)	4,794 (13,073)	193.0 (465.9)	60,740 (141,607)	96.56 (202.5)	30,429 (62,887)
7×140	1,614 (3,066)	5,958 (12,243)	282.6 (1,003)	69,003 (230,187)	131.1 (321.6)	32,353 (76,935)
7×280	– (–)	– (–)	270.1 (940.2)	40,475 (132,011)	181.5 (418.7)	27,447 (57,615)
7×560	– (–)	– (–)	2,704 (9,233)	208,187 (727,397)	358.1 (669.0)	26,316 (46,591)

us to use efficient network flow procedures, and results in the advantage over the algorithm ignoring the network structure, as seen in the previous section. Since we tested the algorithms on limited instances, we can not make a final conclusion. Nonetheless, the algorithm Simplicial_BB is fairly promising for practical use and will serve as a stepping stone to solve further complicated supply chain models.

Chapter 6

Conical Algorithm for Reverse Convex Programming Problems

6.1 Introduction

Let us consider a class of reverse convex programs, i.e., linear programs with an additional reverse convex constraint (LPARC). The feasible set of this class is a difference of a polyhedron and an open convex set. We need to optimize a linear function on such a nonconvex set, which might be disconnected. Therefore, LPARC can have multiple locally optimal solutions, many of which fail to be globally optimal. Although LPARC is just a subclass of the reverse convex program, it involves a wide variety of problems (see e.g., [19]). Among others, of importance is the linear complementarity problem: find $\mathbf{x} \in \mathbb{R}^n$ such that

$$\mathbf{x} \geq \mathbf{0}, \quad \mathbf{M}\mathbf{x} + \mathbf{q} \geq \mathbf{0}, \quad \mathbf{x}^{\mathsf{T}}(\mathbf{M}\mathbf{x} + \mathbf{q}) = 0,$$

where $\mathbf{M} \in \mathbb{R}^{n \times n}$ and $\mathbf{q} \in \mathbb{R}^n$. Even this well-known problem is an instance of LPARC:

$$\left|\begin{array}{ll} \text{minimize} & z \\ \text{subject to} & \mathbf{M}\mathbf{x} - \mathbf{y} = -\mathbf{q}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0} \\ & z - \displaystyle\sum_{j=1}^{n} \min\{x_j, y_j\} \geq 0. \end{array}\right.$$

To solve LPARC, various algorithms have been proposed since the pioneer work by Hillestad [11]. In this chapter, we focus on the conical branch-and-bound algorithm, which was originally proposed by Tuy [52] for concave minimization problems and applied to LPARC later in, e.g., [36, 38]. In the bounding process, we usually relax each subproblem into a linear program and solve it to obtain a lower bound on the optimal value. We will show that the lower bound yielded by this linear programming relaxation can be tighten considerably using a nonlinear surrogate relaxation. Recently, it was reported in [28, 29] that a similar procedure works well in simplicial branch-and-bound algorithms for concave minimization problems. After giving our problem settings of LPARC in Section 6.2, we explain basic workings of the standard conical branch-and-bound algorithm in Section 6.3. We then describe the nonlinear surrogate relaxation and incorporate it into the branch-and-bound algorithm in Section 6.4. Section 6.5 is devoted to a report of numerical results on the proposed algorithm.

6.2 Problem Settings

The problem we consider in this chapter is the following LPARC:

$$\left|\begin{array}{lll} \text{minimize} & z = \mathbf{c}^{\mathsf{T}}\mathbf{x} & \\ \text{subject to} & \mathbf{A}\mathbf{x} + \mathbf{D}\mathbf{y} = \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0} & \\ & g(\mathbf{x}) \geq 0, & \end{array}\right. \tag{6.1}$$

where $\mathbf{A} \in \mathbb{R}^{m \times r}$, $\mathbf{D} \in \mathbb{R}^{m \times (n-r)}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^r$, and $g : \mathbb{R}^r \to \mathbb{R}$ is a convex function. In many applications, we can assume that r is much smaller than n because of the low-rank nonconvexity (see [24]). Low-rank-nonconvex structured instances of LPARC are generally formulated into

$$\left|\begin{array}{lll} \text{minimize} & z = \mathbf{c}^{\mathsf{T}}\mathbf{x} + \mathbf{d}^{\mathsf{T}}\mathbf{y} & \\ \text{subject to} & \mathbf{A}\mathbf{x} + \mathbf{D}\mathbf{y} = \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0} & \\ & g(\mathbf{x}) + \mathbf{h}^{\mathsf{T}}\mathbf{y} \geq 0, & \end{array}\right. \tag{6.2}$$

where $\mathbf{d}$, $\mathbf{h} \in \mathbb{R}^{n-r}$ and $n >> r$. If we introduce auxiliary variables ζ_-, ζ_+, η_- and η_+, then (6.2) reduces to the form of (6.1):

$$\left|\begin{array}{lll} \text{minimize} & \mathbf{c}^\mathsf{T}\mathbf{x} + \zeta_+ - \zeta_- & \\ \text{subject to} & \mathbf{A}\mathbf{x} + \mathbf{D}\mathbf{y} = \mathbf{b}, & (\mathbf{x}, \mathbf{y}) \geq \mathbf{0} \\ & \zeta_+ - \zeta_- - \mathbf{d}^\mathsf{T}\mathbf{y} = 0, & (\zeta_-, \zeta_+) \geq \mathbf{0} \\ & \eta_+ - \eta_- - \mathbf{h}^\mathsf{T}\mathbf{y} = 0, & (\eta_-, \eta_+) \geq \mathbf{0} \\ & g(\mathbf{x}) + \eta_+ - \eta_- \geq 0. & \end{array}\right.$$

Let

$$\begin{aligned} F &= \{\, \mathbf{x} \in \mathbb{R}^r \mid \exists \mathbf{y} \geq \mathbf{0},\ \mathbf{A}\mathbf{x} + \mathbf{D}\mathbf{y} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0} \,\} \\ G &= \{\, \mathbf{x} \in \mathbb{R}^r \mid g(\mathbf{x}) < 0 \,\}, \end{aligned}$$

and assume that both F and G are bounded and have interior points. Then (6.1) is embedded in the $\mathbf{x}$-space as $\min\{\, \mathbf{c}^\mathsf{T}\mathbf{x} \mid \mathbf{x} \in F \setminus G \,\}$. We assume that at least one optimal solution $\mathbf{x}^\circ$ to the associated linear program $\min\{\, \mathbf{c}^\mathsf{T}\mathbf{x} \mid \mathbf{x} \in F \,\}$ is a point in G. This condition makes (6.1) nontrivial, but provides us with a valuable information about its optimality [19, 55]:

Proposition 6.1. *If $F \setminus G \neq \emptyset$, there exists a globally optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ to (6.1) such that $\mathbf{x}^*$ is located at the intersection of an edge of the polyhedron F with the boundary of the set G.*

For simplicity, we assume $\mathbf{x}^\circ = \mathbf{0}$ in the sequel.

6.3 Overview of the Conical Algorithm

In this section, we will overview the basic workings of the standard conical branch-and-bound algorithm on (6.1) [19, 55].

Let $\Delta^1 = \{\, \mathbf{x} \in \mathbb{R}^r \mid \mathbf{x} \geq \mathbf{0} \,\}$. Then Δ^1 is a cone vertexed at $\mathbf{x}^\circ = \mathbf{0}$ and includes the polytope F. Starting from this cone Δ^1, we recursively divide it into subcones, each vertexed at $\mathbf{x}^\circ$, satisfying

$$\Delta^k = \Delta^{2k} \cup \Delta^{2k+1}, \quad \mathrm{int}(\Delta^{2k}) \cap \mathrm{int}(\Delta^{2k+1}) = \emptyset, \quad k = 1, 2, \ldots,$$

where int(·) denotes the interior. This procedure generates an infinite sequence of cones $\{\, \Delta^{k_\ell} \mid \Delta^{k_\ell} \supset \Delta^{k_{\ell+1}},\ \ell = 1, 2, \dots \,\}$. To guarantee the convergence of the algorithm, we need to subdivide Δ^1 in such an exhaustive manner that $\bigcap_{\ell=1}^{\infty} \Delta^{k_\ell}$ becomes a half line emanating from $\mathbf{x}^\circ$. Suppose that Δ^k is spanned by r linearly independent vectors $\mathbf{w}_i \in \mathbb{R}^r$, $i = 1, \dots, r$, and denote $\Delta^k = \mathrm{cone}(\{\mathbf{w}_1, \dots, \mathbf{w}_r\})$. The easiest exhaustive subdivision rule is *bisection*, i.e., we may divide the longest edge of $\Delta^k = \mathrm{conv}(\{\mathbf{w}_1, \dots, \mathbf{w}_r\})$, say $\mathbf{w}_p$–$\mathbf{w}_q$, at a fixed ratio of $\alpha \in (0, 1/2]$, where conv(·) denotes the convex hull. Letting $\mathbf{w} = (1-\alpha)\mathbf{w}_p + \alpha\mathbf{w}_q$, then we have

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

For each subcone $\Delta = \Delta^k$, we have a subproblem of (6.1):

$$\mathrm{P}(\Delta) \left| \begin{array}{ll} \text{minimize} & \mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & \mathbf{x} \in (F \setminus G) \cap \Delta. \end{array} \right.$$

This problem is essentially the same as (6.1) and cannot be solved directly. We instead compute a lower bound on the optimal value of P(Δ). If the bound is greater than or equal to the value of the best feasible solution $\mathbf{x}^*$ to (6.1) obtained so far, we can discard P(Δ) from further consideration. For each $i = 1, \dots, r$, let β_i be a positive number such that $g(\beta_i \mathbf{w}_i) = 0$, and let

$$\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_r], \quad \mathbf{v}_i = \beta_i \mathbf{w}_i.$$

Then we have $\Delta = \{\, \mathbf{x} \in \mathbb{R}^r \mid \mathbf{x} = \mathbf{V}\boldsymbol{\lambda},\ \boldsymbol{\lambda} \geq \mathbf{0} \,\}$. We also see from the convexity of g that

$$\Delta \setminus G \subset \{\, \mathbf{x} \in \mathbb{R}^r \mid \mathbf{x} = \mathbf{V}\boldsymbol{\lambda},\ \mathbf{e}^{\mathsf{T}}\boldsymbol{\lambda} \geq 1,\ \boldsymbol{\lambda} \geq \mathbf{0} \,\},$$

where $\mathbf{e}$ is the all-ones vector. This implies that a lower bound of P(Δ) is given as the optimal value of a linear program:

$$\overline{\mathrm{P}}(\Delta) \left| \begin{array}{lll} \text{minimize} & \mathbf{c}^{\mathsf{T}}\mathbf{x} & \\ \text{subject to} & \mathbf{A}\mathbf{x} + \mathbf{D}\mathbf{y} = \mathbf{b}, & \mathbf{y} \geq \mathbf{0} \\ & \mathbf{x} - \mathbf{V}\boldsymbol{\lambda} = \mathbf{0}, & \boldsymbol{\lambda} \geq \mathbf{0} \\ & \mathbf{e}^{\mathsf{T}}\boldsymbol{\lambda} \geq 1, & \end{array} \right.$$

which is known as the *linear programming relaxation* of $\mathrm{P}(\Delta)$. Let $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ denote an optimal solution to $\overline{\mathrm{P}}(\Delta)$, and let $z(\overline{\mathrm{P}}) = \mathbf{c}^{\mathsf{T}}\overline{\mathbf{x}}$.

6.4 Surrogate Relaxation and the Proposed Algorithm

To tighten the lower bound $z(\overline{\mathrm{P}})$, we propose here a kind of surrogate relaxation of $\mathrm{P}(\Delta)$.

Let us consider the dual problem of $\overline{\mathrm{P}}(\Delta)$:

$$\left|\begin{array}{lll} \text{maximize} & \mathbf{b}^{\mathsf{T}}\boldsymbol{\pi} + \eta & \\ \text{subject to} & \mathbf{A}^{\mathsf{T}}\boldsymbol{\pi} + \boldsymbol{\rho} = \mathbf{c}, & \mathbf{D}^{\mathsf{T}}\boldsymbol{\pi} \leq \mathbf{0} \\ & \mathbf{e}\eta - \mathbf{V}^{\mathsf{T}}\boldsymbol{\rho} \leq \mathbf{0}, & \eta \geq 0. \end{array}\right. \tag{6.3}$$

We can obtain an optimal solution $(\overline{\boldsymbol{\pi}}, \overline{\eta}, \overline{\boldsymbol{\rho}})$ to (6.3) as a byproduct in solving $\overline{\mathrm{P}}(\Delta)$. For this $\overline{\boldsymbol{\pi}} \in \mathbb{R}^m$, let us define the following:

$$\mathrm{S}(\Delta)\left|\begin{array}{lll} \text{minimize} & \mathbf{c}^{\mathsf{T}}\mathbf{x} & \\ \text{subject to} & \overline{\boldsymbol{\pi}}^{\mathsf{T}}\mathbf{A}\mathbf{x} + \overline{\boldsymbol{\pi}}^{\mathsf{T}}\mathbf{D}\mathbf{y} = \overline{\boldsymbol{\pi}}^{\mathsf{T}}\mathbf{b}, & (\mathbf{x}, \mathbf{y}) \geq \mathbf{0} \\ & \mathbf{x} - \mathbf{V}\boldsymbol{\lambda} = \mathbf{0}, & \boldsymbol{\lambda} \geq \mathbf{0} \\ & g(\mathbf{x}) \geq 0, & \end{array}\right.$$

where $\mathbf{x} \geq \mathbf{0}$ is redundant and can be eliminated. Let us denote by $z(\mathrm{S})$ the optimal value of this problem.

Proposition 6.2. *Between $z(S)$ and $z(\overline{P})$, there exists a relationship:*

$$z(S) \geq z(\overline{P}).$$

Proof. Consider the linear programming relaxation of $\mathrm{S}(\Delta)$:

$$\left|\begin{array}{lll} \text{minimize} & \mathbf{c}^{\mathsf{T}}\mathbf{x} & \\ \text{subject to} & \overline{\boldsymbol{\pi}}^{\mathsf{T}}\mathbf{A}\mathbf{x} + \overline{\boldsymbol{\pi}}^{\mathsf{T}}\mathbf{D}\mathbf{y} = \overline{\boldsymbol{\pi}}^{\mathsf{T}}\mathbf{b}, & \mathbf{y} \geq \mathbf{0} \\ & \mathbf{x} - \mathbf{V}\boldsymbol{\lambda} = \mathbf{0}, & \boldsymbol{\lambda} \geq \mathbf{0} \\ & \mathbf{e}^{\mathsf{T}}\boldsymbol{\lambda} \geq 1. & \end{array}\right. \tag{6.4}$$

The dual of this problem is

$$\left|\begin{array}{ll} \text{maximize} & \mathbf{b}^{\mathsf{T}}\overline{\boldsymbol{\pi}}\zeta + \eta \\ \text{subject to} & \mathbf{A}^{\mathsf{T}}\overline{\boldsymbol{\pi}}\zeta + \boldsymbol{\rho} = \mathbf{c}, \quad \mathbf{D}^{\mathsf{T}}\overline{\boldsymbol{\pi}}\zeta \leq \mathbf{0} \\ & \mathbf{e}\eta - \mathbf{V}^{\mathsf{T}}\boldsymbol{\rho} \leq \mathbf{0}, \quad \eta \geq 0. \end{array}\right. \tag{6.5}$$

Then $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ and $(1, \overline{\eta}, \overline{\boldsymbol{\rho}})$ are feasible for (6.4) and (6.5), respectively. Moreover, we have $\mathbf{c}^{\mathsf{T}}\overline{\mathbf{x}} = \mathbf{b}^{\mathsf{T}}\overline{\boldsymbol{\pi}} + \overline{\eta}$, and see that $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ and $(1, \overline{\eta}, \overline{\boldsymbol{\rho}})$ are optimal for these problems. Thus, even the relaxed problem of $\mathrm{S}(\Delta)$ has the same optimal value $z(\overline{\mathrm{P}})$ as $\overline{\mathrm{P}}(\Delta)$. □

Problem $\mathrm{S}(\Delta)$ belongs to the same class of (6.1), but we can solve it in polynomial time if the value of g is given by oracle. Let $F' = \{\, \mathbf{x} \in \mathbb{R}^r \mid \overline{\boldsymbol{\pi}}^{\mathsf{T}}\mathbf{A}\mathbf{x} \geq \overline{\boldsymbol{\pi}}^{\mathsf{T}}\mathbf{b} \,\} \cap \Delta$. Then we have $\mathbf{x}^\circ \in \arg\min\{\, \mathbf{c}^{\mathsf{T}}\mathbf{x} \mid \mathbf{x} \in F' \,\}$, and further

$$F' = \{\, \mathbf{x} \in \mathbb{R}^r \mid \exists(\mathbf{y}, \boldsymbol{\lambda}) \geq \mathbf{0},\ \overline{\boldsymbol{\pi}}^{\mathsf{T}}\mathbf{A}\mathbf{x} + \overline{\boldsymbol{\pi}}^{\mathsf{T}}\mathbf{D}\mathbf{y} = \overline{\boldsymbol{\pi}}^{\mathsf{T}}\mathbf{b},\ \mathbf{x} - \mathbf{V}\boldsymbol{\lambda} = \mathbf{0},\ \mathbf{x} \geq \mathbf{0} \,\}$$

by noting $\mathbf{D}^{\mathsf{T}}\overline{\boldsymbol{\pi}} \leq \mathbf{0}$ and $\mathbf{y} \geq \mathbf{0}$. We see from Proposition 6.1 that $\mathrm{S}(\Delta)$ has an optimal solution $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})$ such that $\widetilde{\mathbf{x}}$ lies on some edge of F'. Since F' is an intersection of the cone Δ with r edges and a halfspace, the maximum number of its edges is $r(r+1)/2$. This implies that $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})$ can be found if we evaluate g at most $O(r^2)$ times.

We are now ready to give a detailed description of our proposed algorithm for solving (6.1). Here, $\epsilon \geq 0$ is a given tolerance.

algorithm Conical_BB
begin
  $\Delta^1 := \mathrm{cone}(\{\mathbf{e}_1, \ldots, \mathbf{e}_r\})$, where $\mathbf{e}_i$ is the ith unit vector;
  $\mathcal{H} := \{\Delta^1\}$; $z^* := +\infty$;
  while $\mathcal{H} \neq \emptyset$ do begin
    select a cone $\Delta^k \in \mathcal{H}$; $\mathcal{H} := \mathcal{H} \setminus \{\Delta^k\}$;
    $\Delta := \Delta^k = \mathrm{cone}(\{\mathbf{w}_1, \ldots, \mathbf{w}_r\})$;
    for $i = 1, \ldots, r$ do compute β_i such that $g(\beta_i \mathbf{w}_i) = 0$ and $\beta_i > 0$;

$\mathbf{V} := [\beta_1 \mathbf{w}_1, \ldots, \beta_r \mathbf{w}_r]$;
let Δ denote $\{\, \mathbf{x} \in \mathbb{R}^r \mid \mathbf{x} = \mathbf{V}\boldsymbol{\lambda},\, \boldsymbol{\lambda} \geq \mathbf{0} \,\}$;
/* *bounding operation* */
solve $\overline{P}(\Delta)$, and obtain a lower bound $z^{\overline{P}}$;
let $(\overline{\mathbf{x}}^k, \overline{\mathbf{y}}^k, \overline{\boldsymbol{\lambda}}^k)$ be an optimal solution to $\overline{P}(\Delta)$;
if $g(\overline{\mathbf{x}}^k) \geq -\epsilon$ then begin
if $z^{\overline{P}} < z^*$ then $z^* := z^{\overline{P}}$; $\mathbf{x}^* := \overline{\mathbf{x}}^k$; $\mathbf{y}^* := \overline{\mathbf{y}}^k$
else if $z^{\overline{P}} < z^*$ then
define $S(\Delta, \overline{\boldsymbol{\pi}})$ for the dual optimal solution $(\overline{\boldsymbol{\pi}}, \overline{\eta}, \overline{\boldsymbol{\rho}})$ to $\overline{P}(\Delta)$;
solve $S(\Delta, \overline{\boldsymbol{\pi}})$, and obtain a lower bound z^S;
search for a local optimal solution $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}, \widetilde{\boldsymbol{\lambda}})$ to $P(\Delta)$;
if $\mathbf{c}^{\mathsf{T}}\widetilde{\mathbf{x}} < z^*$ then $z^* := \mathbf{c}^{\mathsf{T}}\widetilde{\mathbf{x}}$; $\mathbf{x}^* := \widetilde{\mathbf{x}}$; $\mathbf{y}^* := \widetilde{\mathbf{y}}$;
if $z^S < z^*$ then begin
/* *branching operation* */
select the longest edge $\mathbf{w}_p$–$\mathbf{w}_q$ of $\mathrm{conv}(\{\mathbf{w}_1, \ldots, \mathbf{w}_r\})$;
let $\mathbf{w} := (1-\alpha)\mathbf{w}_p + \alpha \mathbf{w}_q$ for a fixed ratio $\alpha \in (0, 1/2]$;
$\Delta^{2k} := \mathrm{cone}(\{\, \mathbf{w}_i \mid i \neq p \,\} \cup \{\mathbf{w}\})$;
$\Delta^{2k+1} := \mathrm{cone}(\{\, \mathbf{w}_i \mid i \neq q \,\} \cup \{\mathbf{w}\})$;
$\mathcal{H} := \mathcal{H} \cup \{\Delta^{2k}, \Delta^{2k+1}\}$
end
end
end
end;

We refer to $(\mathbf{x}, \mathbf{y})$ satisfying the following as an *ϵ-feasible solution* to (6.1):

$$\mathbf{A}\mathbf{x} + \mathbf{D}\mathbf{y} = \mathbf{b}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0}, \quad g(\mathbf{x}) + \epsilon \geq \mathbf{0}.$$

Theorem 6.3. *If $\epsilon > 0$, then algorithm Conical_BB terminates after finitely many iterations and yields an ϵ-feasible solution $(\mathbf{x}^*, \mathbf{y}^*)$ to (6.1) such that $\mathbf{c}^{\mathsf{T}}\mathbf{x}^* \leq \mathbf{c}^{\mathsf{T}}\mathbf{x}$ for all $(\mathbf{x}, \mathbf{y})$ feasible to (6.1).*

Proof. When Conical_BB terminates in a finite number of iterations, the assertion is obvious. Suppose that the algorithm does not terminate and generates an infinite sequence of nested cones $\{ \Delta^{k_\ell} \mid \Delta^{k_\ell} \supset \Delta^{k_{\ell+1}}, \ell = 1, 2, \dots \}$ such that

$$g(\overline{\mathbf{x}}^{k_\ell}) < -\epsilon < 0, \quad \ell = 1, 2, \dots .$$

Let $\mathbf{v}_i^{k_\ell}$ be the ith column of $\mathbf{V}$ for each Δ^{k_ℓ}. Since $\bigcap_{\ell=1}^{\infty} \Delta^{k_\ell}$ is a half line, we have $\mathbf{v}_i^{k_\ell} \to \mathbf{v}$ as $\ell \to \infty$ for all $i = 1, \dots, r$, where $g(\mathbf{v}) = 0$. Here, we should notice that $\overline{\mathrm{P}}(\Delta^{k_\ell})$ is also an instance of LPARC because $g'(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{e}^{\mathsf{T}}\boldsymbol{\lambda} - 1$ is a convex function. Let

$$\begin{aligned} F' &= \{ (\mathbf{x}, \boldsymbol{\lambda}) \mid \exists \mathbf{y} \geq \mathbf{0},\ \mathbf{A}\mathbf{x} + \mathbf{D}\mathbf{y} = \mathbf{b},\ \mathbf{x} - \mathbf{V}\boldsymbol{\lambda} = \mathbf{0},\ (\mathbf{x}, \boldsymbol{\lambda}) \geq \mathbf{0} \} \\ G' &= \{ (\mathbf{x}, \boldsymbol{\lambda}) \mid \mathbf{e}^{\mathsf{T}}\boldsymbol{\lambda} - 1 < 0 \}. \end{aligned}$$

Then we have $(\mathbf{x}^\circ, \boldsymbol{\lambda}^\circ) = (\mathbf{0}, \mathbf{0}) \in \arg\min\{ \mathbf{c}^{\mathsf{T}}\mathbf{x} \mid (\mathbf{x}, \boldsymbol{\lambda}) \in F' \}$ by assumption, and besides $(\mathbf{x}^\circ, \boldsymbol{\lambda}^\circ) \in G'$. Therefore, for the optimal solution $(\overline{\mathbf{x}}^{k_\ell}, \overline{\mathbf{y}}^{k_\ell}, \overline{\boldsymbol{\lambda}}^{k_\ell})$ of $\overline{\mathrm{P}}(\Delta^{k_\ell})$ we have $(\overline{\mathbf{x}}^{k_\ell}, \overline{\boldsymbol{\lambda}}^{k_\ell}) \in \partial G'$ by Proposition 6.1, where $\partial \cdot$ denotes the boundary; and $\mathbf{e}^{\mathsf{T}}\overline{\boldsymbol{\lambda}}^{k_\ell} = 1$ holds for each $\ell = 1, 2, \dots$. This means that $\overline{\mathbf{x}}^{k_\ell}$ is given as convex combination of $\mathbf{v}_i^{k_\ell}$'s, and then $\overline{\mathbf{x}}^{k_\ell} \to \mathbf{v}$ as $\ell \to \infty$. Therefore, we have $g(\overline{\mathbf{x}}^{k_\ell}) \to 0$ as $\ell \to \infty$, which is a contradiction. □

6.5 Numerical Results

In this section, we present numerical results of having compared our algorithm incorporating the surrogate relaxation $\mathrm{S}(\Delta)$ with a standard algorithm only using the linear programming relaxation $\overline{\mathrm{P}}(\Delta)$. We refer to those algorithms as cbb_s and cbb_lp, respectively. Both adopted the depth first rule in selecting $\Delta^k \in \mathcal{H}$, $\alpha = 1/2$, $\epsilon = 10^{-4}$ and were written in GNU Octave (version 2.1.34) [43] and C++ (GCC version 2.96) in part.

The test problem we solved is as follows:

$$\left|\begin{array}{ll} \text{minimize} & \mathbf{c}^{\mathsf{T}}\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} + [\mathbf{D}', \mathbf{I}]\mathbf{y} = \mathbf{e}, \quad (\mathbf{x}, \mathbf{y}) \geq \mathbf{0} \\ & \displaystyle\sum_{j=1}^{r} \gamma_j x_j^2 \geq 1, \end{array}\right. \tag{6.6}$$

where $\mathbf{I} \in \mathbb{R}^{m \times m}$ is an identity matrix; $\mathbf{e} \in \mathbb{R}^m$ is an all-ones vector; each component in the last row of $\mathbf{A} \in \mathbb{R}^{m \times r}$ and $\mathbf{D}' \in \mathbb{R}^{m \times (n-r-m)}$ is fixed to $1.0/(n-m)$, and other componets of $[\mathbf{A}, \mathbf{D}']$ are all random numbers in $[-2.0, 8.0]$, and about 50% of them are zeros; each component of $\mathbf{c} \in \mathbb{R}^r$ is drawn randomly from the uniform distribution on $[10.0, 11.0]$; and each number γ_j is positive and selected so that (6.6) is feasible but not trivial. Selecting seven sets of parameters (m, n, r), we generated ten instances of (6.6), and solved them by cbb_s and cbb_lp on a Linux workstation (Linux 2.4.18, Itanium 2 processor 1GHz).

Table 6.1 shows the average number of branching operations and CPU seconds for each set of (m, n, r). Each figure in brackets represents the number of instances not solved in two hours. We see from this table that the surrogate relaxation $\mathrm{S}(\Delta)$ is of help to cut down the number of branching operations considerably, which also implies that the inequality in Proposition 6.2 held strictly in many iteration of cbb_s. As a consequence, cbb_s is much faster than cbb_lp in CPU seconds. Figure 6.1 compares the difference of the average CPU seconds taken by cbb_s and cbb_lp when $(m, n) = (10, 40)$. As for the instances not solved in two hours, the CPU seconds are plotted at $7,200$ seconds expediently.

Even though we tested the codes on rather limited instances, the performance of the proposed algorithm was promising, compared with the standard one. For some instances, however, both codes were numerically unstable due to rounding errors, and failed to terminate in two hours. In the next paper, we will discuss how to resolve this troublesome issue.

Table 6.1: Average numbers of branchig operations and CPU seconds

	cbb_lp		cbb_s	
$m,\ n,\ r$	branch	time(sec)	branch	time(sec)
10, 30, 10	4031	58	489	8
10, 40, 10	10959	129	1687	23
10, 50, 10	21998	334	5177	91
20, 50, 10	9891	190	3062	50
30, 50, 10	11392	237	3179	67
10, 40, 15		(3)		(3)
10, 40, 20		(7)		(3)

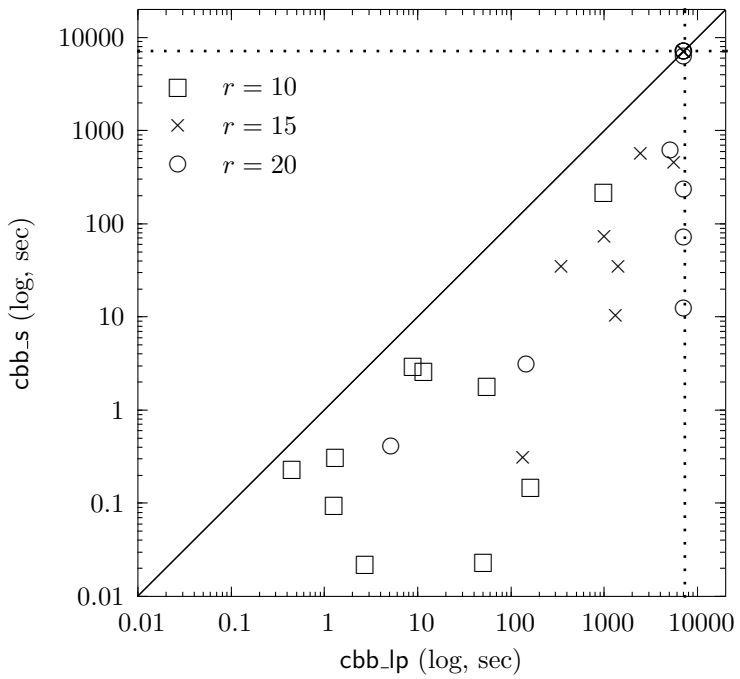

Figure 6.1: CPU seconds when $(m, n) = (10, 40)$

Chapter 7

Conclusion

Our main results in this thesis is to propose simplicial branch-and-bound algorithms for concave minimization problems, which can use special structures of the problems fully, unlike the standard ones. Also, part of the procedures proposed in them can be applied to a conical branch-and-bound algorithm for reverse convex programming problems. In spite of that our numerical experiments to verify the effectiveness of the proposed algorithms have been done on restricted instances, the results show that our algorithms are considerably promizing, compared with the standard ones.

The goal of our study is that we have subproblems hold the favorable structures of the original problem, in order to make full use of low-rank nonconvexity. In a series of studies from Chapter 3 to 5, we can say that we have achieved our aim. However, for the reverse convex programming problems considered in Chapter 6, we should refine our algorithm more and more. For example, even if the problem has a network structure except for the reverse convex constraint, our algorithm cannot use it at all. We will resolve this issue in future works.

In our algorithms, we compute a lower bound for the optimal value to each subproblem without relaxing a concave function or a reverse convex constraint which have been relaxed into linear in earlier studies. We expect that this procedure can be applied to some of other global optimization problems. Actually, we lately have attempted applying it to more general problems containing the reverse

convex programming problems. The results will be appeared in elsewhere.

Furthermore, a new underestimator of the concave function have been proposed in Chapter 4. This is a fairly simple but quite new and hopeful approach. This underestimator would greatly help reseachers who are troubled about rounding errors the same as us.

Acknowledgements

I would first like to express my sincere gratitude to warm encouragement and support of my adviser, Professor Takahito Kuno in pursuing this study. It was he who introduced me to the field of global optimization. This thesis would not have been possible without his help.

I would especially like to thank Professors Yoshitsugu Yamamoto and Akiko Yoshise for their valuable suggestions and thoughtful criticisms. I am also grateful to Professors Maiko Shigeno, Junya Gotoh, Masahiro Hachimori, Nobuo Ohbo, and Jiro Tanaka for their perceptive comments. My gratitude also goes to Professor Kazutoshi Ando of Shizuoka University for his support when he was in University of Tsukuba.

I wish to extend special thanks to my friends Daisuke Zenke, Ayami Suzuka, Hiroaki Sorimachi, Kiyoshi Yamamoto, Takafumi Nakanishi, Azusa Nozaki, and other colleagues of the laboratory with which I affiliate.

Finally, a special word of appreciation goes to my family.

Bibliography

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

[2] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete and Computational Geometry* **8** (1992), 295–313.

[3] M. Avriel, W. E. Diewett, S. Schaible and I. Zang. *Generalized Concavity*. Plenum Press, N.Y., 1988.

[4] M. S. Bazaraa, J. J. Jarvis and H. D. Sherali. *Linear Programming and Network Flows*, 2 ed. John Wiley & Sons, N.Y., 1990.

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

[6] G. B. Dantzig. *Linear programming and extensions*. Princeton Univ. Press, 1963.

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

[8] J. Fülöp. A finite cutting plane method for solving linear programs with an additional reverse convex constraint. *European Journal of Operational Research* **44** (1990), 395–409.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, CA, 1979.

[10] G. M. Guisewite and P. M. Pardalos. Minimum concave-cost network flow problems:applications, complexity, and algorithms. *Annals of Operations Research* **25** (1990), 75–100.

[11] R. J. Hillestad. Optimization problems subject to a budget constraint with economies of scale. *Operations Research* **23** (1975), 1091–1098.

[12] R. J. Hillestad and S. E. Jacobsen. Linear programs with an additional reverse convex constraint. *Applied Mathematics and Optimization* **6** (1980), 257–269.

[13] R. J. Hillestad and S. E. Jacobsen. Reverse convex programming. *Applied Mathematics and Optimization* **6** (1980), 63–78.

[14] K. Holmberg and H. Tuy. A production-transportation problem with stochastic demand and concave production costs. *Mathematical Programming* **85** (1999), 157–179.

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

[16] R. Horst and P. M. Pardalos. *Handbook of Global Optimization*. Kluwer Academic Publishers, 1995.

[17] R. Horst, P. M. Pardalos and N. V. Thoai. *Introduction to Global Optimization*. Kluwer Academic Publishers, Dordrecht, 1995.

[18] R. Horst and N. V. Thoai. Dc programming: Overview. *Journal of Optimization Theory and Applications* **103** (1999), 1–43.

[19] R. Horst and H. Tuy. *Global Optimization: Deterministic Approaches*, 3 ed. Springer-Verlag, Berlin, 1996.

[20] S. E. Jacobsen and K. Moshirvaziri. Computational experience using an edge search algorithm for linear reverse convex programs. *Journal of Global Optimization* **9** (1996), 153–167.

[21] B. Jaumard and C. Meyer. A simplified convergence proof for the cone partitioning algorithm. *Journal of Global Optimization* **13** (1998), 407–416.

[22] B. Jaumard and C. Meyer. On the convergence of cone splitting algorithms with ω-subdivisions. *Journal of Optimization Theory and Applications* **110** (2001), 119–144.

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

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

[25] B. Korte and J. Vygen. *Combinatorial Optimization*. Springer, 2000.

[26] T. Kuno. A pseudo-polynomial algorithm for solving rank three concave production-transportation problems. *Acta Mathematica Vietnamica* **22** (1997), 159–182.

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

[28] T. Kuno and H. Nagai. A simplicial algorithm with two-phase bounding operation for a class of concave minimization problems. *Pacific Journal of Optimization* **1** (2005), 297–313.

[29] T. Kuno and H. Nagai. A simplicial branch-and-bound algorithm conscious of special structures in concave minimization problems. Technical Report CS-TR-05-2, University of Tsukuba, Ibaraki, 2005. Submitted to *Computational Optimization and Applications*.

[30] T. Kuno and T. Utsunomiya. A decomposition algorithm for solving certain classes of production-transportation problems with concave production cost. *Journal of Global Optimization* **8** (1996), 67–80.

[31] T. Kuno and T. Utsunomiya. A pseudo-polynomial primal-dual algorithm for globally solving a production-transportation problem. *Journal of Global Optimization* **11** (1997), 163–180.

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

[33] T. Kuno, 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.

[34] M. Locatelli and N. V. Thoai. Finite exact branch-and-bound algorithms for concave minimization over polytopes. *Journal of Global Optimization* **18** (2000), 107–128.

[35] K. Moshirvaziri. Construction of test problems for concave minimization under linear and nonlinear constraints. *Journal of Optimization Theory and Applications* **98** (1998), 83–108.

[36] K. Moshirvaziri and M. A. Amouzegar. A subdivision scheme for linear programs with an additional reverse convex constraint. *Asia-Pacific Journal of Operations Research* **15** (1998), 179–192.

[37] K. Moshirvaziri and M. A. Amouzegar. A cutting plane algorithm for linear reverse convex programs. *Annals of Operations Research* **105** (2001), 201–212.

[38] L. D. Muu. A convergent algorithm for solving linear programs with an additional reverse convex constraint. *Kybernetika* **21** (1985), 428–435.

[39] H. Nagai and T. Kuno. A conical branch-and-bound algorithm for a class of reverse convex programs. Technical Report CS-TR-05-3, University of Tsukuba, Ibaraki, 2005. Submitted to *The 4th International Conference on*

Nonlinear Analysis and Convex Analysis (NACA), Okinawa (Japan), June, 2005.

[40] H. Nagai 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.

[41] G. L. Nemhauser, A. H. G. RinnooyKan and M. J. Todd, Eds. *Optimization*, vol. 1 of *Handbooks in Operations Research and Management Science*. Elsevier Science Publishers, Amsterdam, 1989.

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

[43] Octave home page. http://www.octave.org/.

[44] M. Porembski. How to extend the concept of convexity cuts to derive deeper cutting planes. *Journal of Global Optimization* **15** (1999), 371–404.

[45] R. T. Rockafellar. *Convex Analysis*. Princeton, N.J., 1970.

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

[47] S. Sen and H. D. Sherali. On the convergence of cutting plane algorithms for a class of nonconvex mathematical programs. *Mathematical Programming* **31** (1985), 42–56.

[48] S. Sen and H. D. Sherali. Nondifferentiable reverse convex programs and facetial convexity cuts via a disjunctive characterization. *Mathematical Programming* **37** (1987), 169–183.

[49] R. M. Soland. Optimal facility location with concave costs. *Operations Research* **22** (1974), 373–382.

[50] N. V. Thoai and H. Tuy. Convergent algorithms for minimizing a concave function. *Mathematics of Operations Research* **5** (1980), 556–566.

[51] N. V. Thuong and H. Tuy. A finite algorithm for solving linear programs with an additional reverse convex constraint. *Lecture Note in Economics and Mathematical Systems* **255** (1984), 291–302.

[52] H. Tuy. Concave programming under linear constraints. *Soviet Mathematics* **5** (1964), 1437–1440.

[53] H. Tuy. Convex programs with an additional reverse convex constraint. *Journal of Optimization Theory and Applications* **52** (1987), 463–486.

[54] H. Tuy. Canonical dc programming problem: Outer approximation methods revisited. *Operations research letters* **18** (1995), 99–106.

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

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

[57] H. Tuy, S. Ghannadan, A. Migdalas and P. Värbrand. Strongly polynomial algorithm for a production-transportation problem with concave production cost. *Optimization* **27** (1993), 205–227.

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