Journal of the Operations Research
Society of Japan
2005, Vol. 48, No. 2, 97-110

A SIMPLICIAL BRANCH-AND-BOUND ALGORITHM FOR PRODUCTION-TRANSPORTATION PROBLEMS WITH INSEPARABLE CONCAVE PRODUCTION COST

Hidetoshi Nagai Takahito Kuno*
University of Tsukuba

(Received May 6, 2004; Revised November 10, 2004)

Abstract In this paper, we develop a branch-and-bound algorithm to solve 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.

Keywords: Nonlinear programming, global optimization, concave minimization, production-transportation problem, minimum cost flow problem, branch-and-bound algorithm

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 an affine function, the problem is reduced to a Hitchcock problem and can be solved in polynomial time [1, 2]. 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 [3, 4].

Compared with an ordinary multiextremal optimization problem, the production-transportation problem has some favorable characteristics. First, the variables 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 [7–9, 13–15]. 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

*The author was partially supported by the Grand-in-Aid for Scientific Research (C)(2) 155600487 from the Japan Society for the Promotion of Science.

pseudo-polynomial in n. However, they are all exponential in m and serve no practical use if m exceeds around five. Another promising class of algorithms is the branch-and-bound method [5, 10, 11], 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, each associated with a subproblem. 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 paper, 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 must 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 2, we describe the basic workings of this simplicial branch-and-bound algorithm. Although it is possible to define a convex envelope of the production cost on each simplex, the network structure needed in efficient solution to the subproblem is damaged by this subdivision scheme. In Section 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 4 is devoted to a report on computational results of comparing those procedures. In Section 5, we discuss some concluding remarks.

2. Problem Settings and the Simplicial Algorithm

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, N and arc set E. For each $(i, j) \in E$, 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

$$\sum_{j \in N} b_j \leq \sum_{i \in M} u_i. \tag{2.1}$$

Let us denote by $g(\mathbf{y})$ 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

$$\Delta^1 = \left\{ \mathbf{y} \in \mathbb{R}^m \;\middle|\; \sum_{i \in M} y_i = B, \;\; \mathbf{y} \geq \mathbf{0} \right\}, \tag{2.2}$$

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

$$\left| \begin{array}{lll} \text{minimize} & z = & \displaystyle\sum_{(i,j) \in E} c_{ij} x_{ij} + g(\mathbf{y}) \\ \text{subject to} & & \displaystyle\sum_{j \in N} x_{ij} = y_i, \quad i \in M \\ & & \displaystyle\sum_{i \in M} x_{ij} = b_j, \quad j \in N \\ & & \mathbf{x} \geq \mathbf{0}, \quad \mathbf{0} \leq \mathbf{y} \leq \mathbf{u}, \end{array} \right. \tag{2.3}$$

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 (2.3) is a special class of minimum concave-cost flow problem [4]. A major difference from the usual one is that the objective function is not assumed completely separable into univariate functions.

Let S denote the feasible set of (2.3). Since the objective function is continuous and S is a bounded polyhedron, problem (2.3) has an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ as long as (2.1) 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 [1]. These facts imply the following:

Lemma 2.1 *Under condition (2.1), problem (2.3) 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 [6, 12]. In the rest of this section, we will review its basic workings.

If (2.1) holds, any feasible production $\mathbf{y}$ of (2.3) belongs to the $(m-1)$-simplex Δ^1 defined in (2.2). Therefore, no feasible solution to (2.3) is lost even if we add $\mathbf{y} \in \Delta^1$ as a constraint. 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

$$\Delta = \Delta' \cup \Delta'', \quad \mathrm{int}(\Delta') \cap \mathrm{int}(\Delta'') = \emptyset, \tag{2.4}$$

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 $\mathbf{v}^i$, $i \in M$, e.g., we have $\mathbf{v}^i = B\mathbf{e}^i$ for the initial simplex Δ^1, where $\mathbf{e}^i$ denotes the ith unit vector. We therefore have

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

where $\mathbf{e}$ denotes a vector of ones and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)^\mathsf{T}$. If we select an edge, say $\mathbf{v}^p$–$\mathbf{v}^q$, of Δ and divide it at a midpoint, we can immediately have

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

where $\mathbf{v} = (1-\mu)\mathbf{v}^p + \mu \mathbf{v}^q$ for a fixed ratio $\mu \in (0, 1/2]$. We refer to this division rule as *bisection of ratio μ on edge $\mathbf{v}^p$–$\mathbf{v}^q$*. The outline of the algorithm is summarized as follows:

Let $\mathcal{D} := \{\Delta^1\}$ and repeat Steps 1–3 while $\mathcal{D} \neq \emptyset$.

Step 1. Take an appropriate Δ out of $\mathcal{D}$. Define a subproblem $\mathrm{P}(\Delta)$.

Step 2 (bounding operation). Compute a lower bound $\overline{z}(\Delta)$ on the optimal value $z(\Delta)$ of $\mathrm{P}(\Delta)$, where $z(\Delta)$ is set to $+\infty$ if $\mathrm{P}(\Delta)$ is infeasible. If $\overline{z}(\Delta)$ is larger than or equal to the value of the best feasible solution $(\mathbf{x}^\circ, \mathbf{y}^\circ)$ obtained so far, discard Δ and return to Step 1.

Step 3 (branching operation). Otherwise, divide Δ into two simplices Δ' and Δ'' and add them to $\mathcal{D}$.

In general, this class of algorithms is not guaranteed to terminate in finite iterations, and it generates an infinite sequence of nested simplices $\{\Delta^r \mid r = 1, \ldots\}$ such that $\Delta^1 \supset \Delta^2 \supset \cdots$. However, if we apply the bisection rule on the longest edge of Δ at Step 3, it is known [6, 12] that Δ^r shrinks to a singleton. This *exhaustiveness* enables us to converge the incumbent $(\mathbf{x}^\circ, \mathbf{y}^\circ)$ to an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ of (2.3), when we adopt the *best bound rule* selecting Δ of the least $\overline{z}(\Delta)$ at Step 1.

The simplicial subdivision (2.4) 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 [6, 12]. On the other hand, there is an obvious difficulty in the implementation against our problem (2.3). 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.

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. To compute a lower bound $\overline{z}(\Delta)$, we solve a relaxed 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 relaxed problem by substituting $\mathbf{y} = \sum_{i \in M} \lambda_i \mathbf{v}^i$ into the constraints as well (see [6, 12] 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.

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 (2.3). Each $\mathbf{v}^i$ and the feasible production $\mathbf{y}$ belong to the initial simplex defined by (2.2). Therefore, $\mathbf{e}^{\mathsf{T}}\mathbf{v}^i = B$ and $\mathbf{e}^{\mathsf{T}}\mathbf{y} = B$ hold; and besides 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 3.1 *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 (2.3). For any $(\mathbf{x}, \mathbf{y}) \in S \cap \Delta$, we have

$$\sum_{(i,j) \in E} c_{ij}x_{ij} + \mathbf{w}\mathbf{V}^{-1}\mathbf{y} \leq \sum_{(i,j) \in E} c_{ij}x_{ij} + g(\mathbf{y}), \tag{3.1}$$

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 $\overline{z}(\Delta)$ at Step 2 of the simplicial branch-and-bound algorithm. 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 $(\mathbf{x}^1, \mathbf{y}^1)$ if necessary. 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.

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 $\mathbf{y}$ using the first set of constraints and have a Hitchcock problem:

$$\left| \begin{array}{lll} \text{minimize} & z = & \displaystyle\sum_{(i,j)\in E} \overline{c}_{ij}x_{ij} \\ \text{subject to} & & \displaystyle\sum_{j\in N} x_{ij} \leq u_i, \quad i \in M \\ & & \displaystyle\sum_{i\in M} x_{ij} = b_j, \quad j \in N \\ & & \mathbf{x} \geq \mathbf{0}, \end{array} \right. \tag{3.2}$$

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 (3.2) always has an optimal solution $\mathbf{x}^2$ under condition (2.1). 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 [1, 2]. 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 $\overline{z}(\Delta)$ at Step 2. 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 (2.3), though it might be infeasible to $\mathrm{RP}_1(\Delta)$. Thereby, we can update the incumbent $(\mathbf{x}^\circ, \mathbf{y}^\circ)$ if necessary.

The relaxed 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:

$$s_i = \lceil \min\{y_i \mid \mathbf{y} \in \Delta\} \rceil, \quad t_i = \min\{\lfloor \max\{y_i \mid \mathbf{y} \in \Delta\} \rfloor, u_i\}, \quad i \in M, \tag{3.3}$$

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 2.1 that at least one optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ is integral, and satisfies $\mathbf{s} \leq \mathbf{y}^* \leq \mathbf{t}$ if $\mathbf{y}^* \in \Delta$. In other words, even if we replace $\mathbf{y} \in \Delta$ in $\mathrm{RP}_1(\Delta)$ by $\mathbf{s} \leq \mathbf{y} \leq \mathbf{t}$, no integral optimal solution to $\mathrm{P}(\Delta)$ is lost. Let us denote the resulting problem by

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

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

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

We should remark 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 (2.3) 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 (2.3) and its subproblem $\mathrm{P}(\Delta)$.

Lemma 3.2 *If $z^3(\Delta) = +\infty$, then subproblem $P(\Delta)$ has no integral feasible solution. Otherwise, let $\tilde{z}(\Delta)$ 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 (3.1) and the inclusive relation between the feasible sets of three problems. ■

Another remark on $\mathrm{RP}_3(\Delta)$ is that it is a minimum cost flow problem similar to the target problem (2.3). We can 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 and obtain $z^3(\Delta)$, naturally in polynomial time of m and n [1]. Again, $(\mathbf{x}^3, \mathbf{y}^3)$ might be infeasible to $\mathrm{P}(\Delta)$, but is feasible to (2.3) and can be used for the incumbent update.

The network flow relaxed problem $\mathrm{RP}_3(\Delta)$ has been drawn by exploiting the integrality of an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ to the target problem (2.3). 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 (3.3), we can see that Δ contains just one integral feasible production $\tilde{\mathbf{y}}$ of (2.3) if and only if

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

and Δ contains no integral feasible production of (2.3) if and only if

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

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 relaxed 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 (2.3) except $\tilde{\mathbf{y}}$. Therefore, we can stop the branching operation and can save computational time.

As seen in Section 2, 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 at Step 3. 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, i.e., Δ^r must satisfy one of the above 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 at Step 1, 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 $\mathbf{v}^i := B\mathbf{e}^i$;
  $\mathbf{V} := [\mathbf{v}^1, \ldots, \mathbf{v}^m]$; $\Delta := \mathrm{conv}(\mathbf{V})$; $\mathcal{D} := \{\Delta\}$; $z^\circ := +\infty$;
  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 $\mathrm{P}(\Delta)$;
    /* 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 $\mathrm{P}(\Delta)$ 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 $\mathbf{s} \leq \mathbf{t}$ and $\mathbf{e}^\mathsf{T}\mathbf{s} \leq B \leq \mathbf{e}^\mathsf{T}\mathbf{t}$ then begin
      $\mathbf{w} := [g(\mathbf{v}^1), \ldots, g(\mathbf{v}^m)]$; $\overline{g}(\mathbf{y}) := \mathbf{w}\mathbf{V}^{-1}\mathbf{y}$;
      define a relaxed problem $\mathrm{RP}_k(\Delta)$ of $\mathrm{P}(\Delta)$ using $\overline{g}$;
      solve $\mathrm{RP}_k(\Delta)$ and obtain a lower bound $\overline{z}(\Delta) := z^k(\Delta)$;
      let $(\mathbf{x}^k, \mathbf{y}^k)$ denote an optimal solution to $\mathrm{RP}_k(\Delta)$;
      if $\sum_{(i,j)\in E} c_{ij}x^k_{ij} + g(\mathbf{y}^k) < z^\circ$ then begin
        $(\mathbf{x}^\circ, \mathbf{y}^\circ) := (\mathbf{x}^k, \mathbf{y}^k)$; $z^\circ := \sum_{(i,j)\in E} c_{ij}x^k_{ij} + g(\mathbf{y}^k)$
      end;
      if $\overline{z}(\Delta) < z^\circ$ then begin
        /* Step 3. (branching operation; $\mu \in (0, 1/2]$) */
        select the longest edge $\mathbf{v}^p$–$\mathbf{v}^q$ of Δ and let $\mathbf{v} := (1-\mu)\mathbf{v}^p + \mu\mathbf{v}^q$;
        $\mathbf{V}' := [\mathbf{v}^1, \ldots, \mathbf{v}^{p-1}, \mathbf{v}, \mathbf{v}^{p+1}, \ldots, \mathbf{v}^m]$;
        $\mathbf{V}'' := [\mathbf{v}^1, \ldots, \mathbf{v}^{q-1}, \mathbf{v}, \mathbf{v}^{q+1}, \ldots, \mathbf{v}^m]$;
        $\Delta' := \mathrm{conv}(\mathbf{V}')$; $\Delta'' := \mathrm{conv}(\mathbf{V}'')$; $\mathcal{D} := \mathcal{D} \cup \{\Delta', \Delta''\}$
      end
    end
  end;
  $(\mathbf{x}^*, \mathbf{y}^*) := (\mathbf{x}^\circ, \mathbf{y}^\circ)$
end;

Theorem 3.3 *The algorithm* SIMPLICIAL-BB *terminates after finitely many iterations, and yields a globally optimal solution* $(\mathbf{x}^*, \mathbf{y}^*)$ *to (2.3).*

Proof: It is obvious that the algorithm yields an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ to (2.3) if it terminates. 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 (2.3), 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 (2.3), we have a nested sequence $\{\Delta^1, \ldots, \Delta^r\}$, where Δ^1 denotes the initial simplex given by (2.2). 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 (2.3). ■

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 (2.3). We refer to the algorithms by SBB1, SBB2 and SBB3, respectively.

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

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

where γ was selected from $\{0.1, 1.0, 10.0\}$, α_{ki} and β_k were 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 ranged from 4 to 7, and n was set to each of $\{10m, 20m, 40m, 80m\}$.

The algorithms were coded mainly using GNU Octave (version 2.1.34) [16], a Matlab-like computational tool, according to the description in Section 3. We also coded the revised simplex algorithm for solving the relaxed problem $\mathrm{RP}_1(\Delta)$, and the successive shortest path algorithm for solving $\mathrm{RP}_2(\Delta)$ and $\mathrm{RP}_3(\Delta)$ [1, 2]. Neither algorithm is polynomial, but we improved the efficiency by exploiting an optimal solution to the preceding relaxed 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 1–3 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

Table 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 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 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)

three tables. The algorithm SIMPLICIAL-BB is therefore expected to solve still more highly nonlinear problems as long as m is less than seven.

5. Concluding Remarks

The production-transportation problem (2.3) 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 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.

References

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin: *Network Flows* (Prentice-Hall, NJ, 1993).

[2] M.S. Bazaraa, J.J. Jarvis and H.D. Sherali: *Linear Programming and Network Flows* (2nd ed.) (John Wiley & Sons, NY, 1990).

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

[4] 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.

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

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

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

[8] 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.

[9] 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.

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

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

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

[13] 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.

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

[15] 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.

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

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