Math. Program. 85: 157–179 (1999) / DOI 10.1007/s10107980034a

Kaj Holmberg · Hoang Tuy

A production-transportation problem with stochastic demand and concave production costs

Received October 11, 1993 / Revised version received July 31, 1995
Published online November 24, 1998

Abstract. Well known extensions of the classical transportation problem are obtained by including fixed costs for the production of goods at the supply points (facility location) and/or by introducing stochastic demand, modeled by convex nonlinear costs, at the demand points (the stochastic transportation problem, [STP]). However, the simultaneous use of concave and convex costs is not very well treated in the literature. Economies of scale often yield concave cost functions other than fixed charges, so in this paper we consider a problem with general concave costs at the supply points, as well as convex costs at the demand points. The objective function can then be represented as the difference of two convex functions, and is therefore called a d.c. function. We propose a solution method which reduces the problem to a d.c. optimization problem in a much smaller space, then solves the latter by a branch and bound procedure in which bounding is based on solving subproblems of the form of [STP]. We prove convergence of the method and report computational tests that indicate that quite large problems can be solved efficiently. Problems up to the size of 100 supply points and 500 demand points are solved.

Key words. transportation – d.c. functions – decomposition methods – branch-and-bound

1. Introduction

The problem of transporting goods from a set of supply points (factories) to a set of demand points (customers) so as to minimize linear transportation costs is well known and very efficient solution methods exist. In facility location problems one also includes the decisions of whether or not one should open the supply points (build the factories) for certain fixed costs. Such problems are also well known and several solution methods exist. However, location problems with general concave costs are quite difficult to solve, [6], and heuristic solution methods have been proposed, see for example [14].

On the other hand, in ordinary transportation problems, stochastic demand has been introduced, and methods developed [1, 12]. The resulting problem, called the stochastic transportation problem, [STP], is a transportation problem with the demand constraints replaced by nonlinear convex costs as functions of the total inflow into each demand point.

However, the simultaneous use of these generalizations has received little attention until now. A few suggestions for solution methods can be found [18, 3], but more research remains to be done.

K. Holmberg: Linköping Institute of Technology, Department of Mathematics, S-581 83 Linköping, Sweden

H. Tuy: Institute of Mathematics, P.O. Box 631, Bo Ho, Hanoi, Vietnam. The paper was completed during the stay of this author at Linköping Institute of Technology, S-581 83 Linköping, Sweden

Mathematics Subject Classification (1991): 90C26, 90B06

In this paper we consider a further generalization with general concave costs at the supply points, as well as convex costs at the demand points. Economies of scale very often yield different concave cost functions, not only fixed charges, when producing goods at a factory.

Certainly the introduction of concave costs makes the problem difficult. In fact, the objective function is a sum of three terms: a linear transportation cost, a convex shortage penalty and a concave production cost. It is neither convex nor concave, but a *d.c. function*, i.e. a function that can be represented as a difference of two convex functions, [15].

The problem of minimizing a d.c. function under linear constraints is a nonconvex global optimization problem, which may have multiple local minima with substantially different values. Such multiextremal problems cannot be solved by standard methods of nonlinear programming which can at best locate a local minimum. Outer approximation methods and branch and bound methods for finding a global minimum have been suggested in [22–26]. However, most of these methods consider the problem in its general form (general d.c. objective function and general linear constraints) and therefore, are able to solve only problem instances of limited size. This should not be surprising, since the problem is known to be NP-hard, see e.g. [19].

In the production-transportation problem with d.c. cost we are considering here, the constraints are of transportation type and the objective function is separable. Furthermore, if the concave term in the objective function is linearized, then the problem becomes a [STP]. We show that by exploiting all these additional structures it is possible to devise a solution method capable of solving fairly large problems. Specifically, using the fact that the nonconvex component of the objective function involves only some of the variables, we can reduce the problem to a d.c. optimization problem of much smaller dimension than the original one [24]. Next, this reduced problem can be solved by a branch and bound procedure in which branching is performed by partitioning the space into rectangles (to take account of the separability of the variables) while bounding is based on solving subproblems of the form [STP] (to take advantage of the availability of efficient algorithms for [STP]). Computational experiments indicate that the algorithm obtained that way can solve problems with up to 100 supply points and 500 demand points in reasonable time. It should be noted that at present, for most global optimization algorithms, problems of this size are considered difficult.

The general idea of solving nonlinear nonconvex problems with a branch-and-bound method dividing the feasible region into rectangles is of course not new, see section 5. However, without exploiting some particular structure to reduce the dimension of the space on which branching is performed, such a method cannot be expected to be practical for large-scale problems. It is also imperative to have an efficient way of solving the resulting subproblems for bounding. To our knowledge there has been to date no successful implementations of methods for the type of problem we consider. This will be further discussed in section 5, where also (piecewise) linear cases are considered. It should also be noted that our method is directly applicable even if the production cost is discontinuous at the origin, as for example a fixed charge. However, we will not elaborate on this for problems with (piecewise) linear costs, since solving such problems is not our main goal.

The paper consists of 7 sections. After the Introduction, we state the problem in section 2 and also discuss two special cases: the Stochastic Transportation-Location Problem and the Stochastic Transportation Problem. Section 3 is devoted to the description of the solution method, and Section 4 to the convergence proof. In section 5 we discuss the relations of our method to previously known ones. In section 6 we present an illustrative example and the computational results. Finally, we close the paper by some concluding remarks.

2. The problem

We consider a transportation problem with m supply points (factories) and n demand points (customers). Let z_i be the amount that is shipped out of supply point i, y_j the amount that is shipped into demand point j and x_{ij} the amount that is shipped from supply point i to demand point j.

At supply point i there is a cost of producing z_i units, denoted by $g_i(z_i)$. We assume that $g_i(0) = 0$, and that $g_i(z_i)$ is lower semicontinuous, non-decreasing and concave due to economies of scale. The maximal capacity is denoted by S_i.

At demand point j the demand, d_j, is assumed to be stochastic. If y_j units are actually received by the demand point j then a penalty cost $\zeta_j(y_j, d_j)$ is incurred. In most cases of interest, each expected penalty $f_j(y_j) = E[\zeta_j(y_j, d_j)]$ is a convex function, see [1].

One often assumes that there is a probability density function, $\phi_j(d_j)$, which gives an expected demand as $E[d_j] = \int_0^\infty v\phi_j(v)\,dv$, and a continuous distribution function as $F_j(d_j) = \int_0^{d_j} \phi_j(v)\,dv$. Costs occur if y_j is strictly less or strictly greater than d_j. There are unit holding costs, $\alpha_j > 0$, and unit shortage costs, $\xi_j > 0$, which gives a total holding/shortage cost as

$$f_j(y_j) = \xi_j \int_{y_j}^{\infty} (v - y_j)\phi_j(v)\,dv + \alpha_j \int_0^{y_j} (y_j - v)\phi_j(v)\,dv$$

This can also be expressed as

$$f_j(y_j) = \xi_j(E[d_j] - y_j) + (\xi_j + \alpha_j) \int_0^{y_j} F_j(v)\,dv$$

The derivative of f_j (to be used later) is

$$\frac{df_j(y_j)}{dy_j} = -\xi_j + (\xi_j + \alpha_j)F_j(y_j)$$

$f_j(y_j)$ can be shown to be a convex function. The results and methods in this paper, however, are not restricted to this form of expected penalty cost function, but are applicable to many kinds of differentiable convex penalty cost functions.

Furthermore we assume linear transportation costs, with unit cost c_{ij} from supply point i to demand point j. (If there are concave transportation costs on some arcs, these concave costs can be incorporated in the production costs. The practical efficiency of our method is partly based on the fact that the number of variables with concave costs is relatively small compared to the total number of variables. For a problem with concave

costs on all variables, as the problem discussed by [2], we do not expect this approach to be very efficient in practice. On the other hand we believe that in many practical cases linear approximations of the transportation costs as compared to linear approximations of the production costs give much smaller errors and are thus much more acceptable.)

The problem is now to minimize the total costs, consisting of production costs, shortage/holding costs and transportation costs.

$$
\begin{aligned}
v^* = \min\ & \textstyle\sum_{i=1}^m \sum_{j=1}^n c_{ij}x_{ij} + \sum_{i=1}^m g_i(z_i) + \sum_{j=1}^n f_j(y_j) \\
\text{s.t.}\ & \textstyle\sum_{i=1}^m x_{ij} = y_j \quad j = 1,\dots,n && (1) \\
& \textstyle\sum_{j=1}^n x_{ij} = z_i \quad i = 1,\dots,m && (2) \\
& 0 \le z_i \le S_i \quad i = 1,\dots,m && (3) \\
& x_{ij} \ge 0 \quad \forall\, i, j && (4)
\end{aligned}
\qquad \text{[P]}
$$

The objective function is a sum of three terms: a linear transportation cost, $cx = \sum_{i=1}^m \sum_{j=1}^n c_{ij}x_{ij}$, a convex shortage penalty, $f(y) = \sum_{j=1}^n f_j(y_j)$, and a concave production cost, $g(z) = \sum_{i=1}^m g_i(z_i)$, such that each $g_i(z_i)$ is monotone nondecreasing. It is thus a d.c. function.

The constraints are linear with a simple structure. By adding the following redundant constraints,

$$
\sum_{j=1}^n y_j \le S_{TOT},\ y_j \ge 0\ \forall j,\ x_{ij} \le S_i\ \forall i
$$

where $S_{TOT} = \sum_{i=1}^m S_i$, the feasible set can be considered to be bounded.

Let us now discuss two special cases of [P]. The first one is obtained by letting g_i consist of a fixed charge (and possibly a linear cost). We then get the Stochastic Transportation-Location Problem, [STLP], treated in [18,3]. The [STLP] can be formulated as

$$
\begin{aligned}
v^* = \min\ & \textstyle\sum_{i=1}^m \sum_{j=1}^n c_{ij}x_{ij} + \sum_{i=1}^m r_i\delta_i + \sum_{j=1}^n f_j(y_j) \\
\text{s.t.}\ & \textstyle\sum_{i=1}^m x_{ij} = y_j \quad j = 1,\dots,n && (1) \\
& \textstyle\sum_{j=1}^n x_{ij} \le S_i\delta_i \quad i = 1,\dots,m && (2) \\
& \delta_i \in \{0,1\}\ i = 1,\dots,m && (3) \\
& y_j \ge 0 \quad \forall\, j && (4) \\
& x_{ij} \ge 0 \quad \forall\, i, j && (5)
\end{aligned}
\qquad \text{[STLP]}
$$

Here r_i is the fixed cost for starting production at supply point i, and δ_i is a binary variable equal to 1 if something is produced at supply point i and 0 if not. This problem has been solved by a heuristic approach in [18] and by generalized Benders decomposition [5] in [3].

The other special case of [P] occurs if $g_i(z_i)$ is linear: $g(z) = \sum_{i=1}^m g_i z_i = \sum_{i=1}^m \sum_{j=1}^n g_i x_{ij}$. Then the production costs can be incorporated in the transportation costs, and we get the better known Stochastic Transportation Problem, [STP] [16].

$$
\begin{aligned}
v^* = \min \ & \textstyle\sum_{i=1}^m \sum_{j=1}^n \bar{c}_{ij} x_{ij} + \sum_{j=1}^n f_j(y_j) \\
\text{s.t.} \ & \textstyle\sum_{i=1}^m x_{ij} = y_j \ \ j = 1, \ldots, n \quad (1) \\
& \textstyle\sum_{j=1}^n x_{ij} \le S_i \ \ i = 1, \ldots, m \quad (2) \qquad \text{[STP]} \\
& x_{ij} \ge 0 \ \ \forall\, i, j \quad (3) \\
& y_j \ge 0 \ \ \forall\, j \quad (4)
\end{aligned}
$$

The objective function of [STP] is convex. The [STP] has been treated in many papers and can be solved quite efficiently, for example with the Frank-Wolfe method [4] in [1,17], by cross decomposition [27] in [12], by the classical approach of separable programming [7], by the forest iteration method [20] and by mean value cross decomposition [10], and combinations of separable programming, Lagrangean relaxation with subgradient optimization and mean value cross decomposition [11]. The last method is shown to be quite efficient compared to the others in [11].

3. Solution method

Problem [P] is a difficult global optimization problem with $mn + m + n$ variables. However, there are only m variables appearing in nonconvex functions: $z_1, \ldots, z_m$ and very often $m \ll n$, though n may be fairly large. Furthermore, the objective function is separable and when $g(z)$ is linear, the problem reduces to the [STP] as discussed above.

To take advantage of this specific structure, we propose a solution method which reduces [P] to a d.c. optimization problem in R^m, then solves the latter by a branch and bound procedure in which bounding is based on solving subproblems of the form of [STP].

Specifically, for a given z in the rectangle $\Omega = \{z : 0 \le z \le S := (S_1, ..., S_m)\}$, consider the problem

$$
\begin{aligned}
\varphi(z) = \min \ & \textstyle\sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} + \sum_{j=1}^n f_j(y_j) \\
\text{s.t.} \ & \textstyle\sum_{i=1}^m x_{ij} = y_j \ \ j = 1, \ldots, n \quad (1) \\
& \textstyle\sum_{j=1}^n x_{ij} = z_i \ \ i = 1, \ldots, m \quad (2) \qquad \text{[Q}(z)\text{]} \\
& y_j \ge 0 \ \ j = 1, \ldots, n \quad (3) \\
& x_{ij} \ge 0 \ \ \forall\, i, j \quad (4)
\end{aligned}
$$

It is not hard to see that

Proposition 1. *$\varphi(z)$ is a convex function and* [P] *is equivalent to*

$$\min\{\varphi(z) + g(z) : z \in \Omega\} \qquad \text{[P}^*\text{]}$$

in the sense that the two problems have equal optimal values and if (x^*, y^*, z^*) *solves* [P] *then* z^* *solves* [P*] *and conversely if* z^* *solves* [P*] *then* (x^*, y^*, z^*) *solves* [P], *where* (x^*, y^*) *is an optimal solution of* [Q(z^*)].

Proof. If (x^*, y^*, z^*) solves [P] then z^* is feasible to [P*] and (x^*, y^*) is feasible to [Q(z^*)], hence $cx^* + f(y^*) \geq \varphi(z^*)$, and so $v^* \geq \varphi(z^*) + g(z^*)$, i.e. the optimal value of [P] is no less than that of [P*]. Conversely, if z^* solves [P*], with $\varphi(z^*) = cx^* + f(y^*)$ then (x^*, y^*, z^*) is feasible to [P], hence $\varphi(z^*) + g(z^*) = cz^* + f(y^*) + g(z^*) \geq v^*$, i.e. the optimal value of [P*] is no less than that of [P].

□

[P*] is still a d.c. optimization problem, but of much smaller dimension than [P]. Below we present a branch and bound solution method for [P*] which tries to take advantage of two basic structures of the problem:

1. *Separable objective function.* This suggests that an efficient branching method is by rectangular subdivision of the feasible domain Ω (see Lemma 1 and Proposition 3 below).
2. *Transportation constraints.* As mentioned above, for fixed z the problem reduces to [Q(z)] which is essentially the same as [STP] and can be solved by efficient algorithms [11]. To preserve this nice structure while decomposing the problem, bounding will be based on solving subproblems essentially of the form [STP].

Specifically, the rectangular subdivision is defined as follows.

Definition 1. *Let* $M = [p, q]$ *be a rectangle contained in* Ω. *Any point* $w \in M$, *together with an index* $i \in \{1, \ldots, m\}$, *determines a subdivision of* M *into two subrectangles*

$$\{z : p_i \leq z_i \leq w_i,\ p_t \leq z_t \leq q_t\ (\forall t \neq i)\} \text{ and } \{z : w_i \leq z_i \leq q_i,\ p_t \leq z_t \leq q_t\ (\forall t \neq i)\}.$$

This subdivision is called a subdivision via (w, i).

At each iteration, a rectangular partition of the feasible domain is available as a result of previous subdivisions. This partition is then refined by choosing a "promising" rectangle in the partition and subdividing it via some (w, i). Both the choices of the candidate M for further subdivision and the parameters (w, i) of subdision are guided by the lower bounds assigned to every member of the partition. The next proposition gives the method for computing these lower bounds.

For any rectangle $M = [p, q]$ contained in the feasible domain Ω, let $L_{M,i}(.)$ be the affine function which agrees with $g_i(.)$ at the endpoints of the segment $[p_i, q_i]$ and

$$L_M(x) = \sum_{i=1}^{m} L_{M,i}(z_i)$$

i.e.

$$L_{M,i}(z_i) = g_i(p_i) - \left(\frac{p_i}{q_i - p_i}\right)(g_i(q_i) - g_i(p_i)) + z_i \left(\frac{g_i(q_i) - g_i(p_i)}{q_i - p_i}\right)$$

Define the convex program

$$\begin{aligned} \beta(M) = \min\ & \textstyle\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} f_j(y_j) + L_M(z) \\ \text{s.t.}\ & \textstyle\sum_{i=1}^{m} x_{ij} = y_j \quad j = 1,\ldots,n \qquad (1) \\ & \textstyle\sum_{j=1}^{n} x_{ij} = z_i \quad i = 1,\ldots,m \qquad (2) \\ & y_j \geq 0\ \forall j,\ x_{ij} \geq 0\ \forall i, j,\ z \in M \quad (3) \end{aligned} \qquad [\mathrm{CP}(M)]$$

Proposition 2.

$$\beta(M) \leq \min\{\varphi(z) + g(z) : z \in M\} \qquad (1)$$

If an optimal solution $(\bar{x}(M); \bar{y}(M), \bar{z}(M))$ *of [CP(M)] satisfies* $L_M(\bar{z}(M)) = g(\bar{z}(M))$, *then equality holds in 1.*

Proof. Using the concavity of $g(z)$ it is easy to check that $L_M(z)$ is an underestimator of $g(z)$ over M. Therefore, $\beta(M) \leq \min\{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + f(y) + g(z) : (x, y, z)$ feasible to [P], $z \in M\}$, and hence (1), by Proposition 1. The second assertion is obvious. □

Clearly each [CP(M)] is a convex problem which differs from the [STP] only by the additional constraint $z \in M$, i.e. $p \leq z \leq q$. Therefore, the lower bounding subproblem for each partition rectangle can be solved by adapted versions of the algorithms in [11].

To obtain a complete description of the branch and bound procedure for solving [P*], it remains to specify the subdivision rule, i.e. the choice of:

- the candidate M_k for further subdivision at iteration k;
- the parameters (w^k, i_k) determining the subdivision of M_k.

As usual in a branch-and-bound procedure, M_k (the subrectangle in which the approximation is to be improved), is chosen to be the subrectangle with smallest lower bound.

$$M_k \in \arg\min_{M \in \mathcal{M}'_k} \beta(M)$$

where $\mathcal{M}'_k$ denotes the current partition. Note that this implies

$$\beta(M_k) \leq \min\{\varphi(z) + g(z) | z \in \Omega\} \qquad (2)$$

From this and Proposition 2 it immediately follows that if $\bar{z}^k = \bar{z}(M_k)$ satisfies $L_{M_k}(\bar{z}^k) = g(\bar{z}^k)$, then the equality holds in 2, i.e. $\bar{z}^k$ solves [P*]. Otherwise, $g_i(\bar{z}^k_i) - L_{M_k i}(\bar{z}^k_i) > 0$ for at least one i and we subdivide M via $(\bar{z}^k, i_k)$, where i_k is the index i that achieves the maximal discrepancy $g_i(\bar{z}^k_i) - L_{M_k,i}(\bar{z}^k_i)$ between the actual cost and the linear approximation. It turns out that this subdivision rule ensures that eventually this maximal discrepancy will tend to zero. Then the lower bound $\beta(M_k)$ will tend to the exact minimum of the objective function on M_k and hence, to its minimum over the whole feasible domain, thereby ensuring convergence of the procedure.

We can thus state:

Algorithm 1. *Initialization*

Let ε be a small positive constant. Take a subrectangle M_1 of Ω which is known to contain a global optimal solution of [P*]. Let z^1 be the best feasible point available, INV (incumbent value) = $\varphi(z^1) + g(z^1)$. Set $\mathcal{M}_1 = \mathcal{P}_1 = \{M_1\}, k = 1$.

Iteration $k = 1, 2, \ldots$

(i) For each $M \in \mathcal{P}_k$ solve the convex program [CP(M)] obtaining its optimal value $\beta(M)$ and optimal solution $(\bar{x}(M); \bar{y}(M), \bar{z}(M))$. Update INV and z^k.

(ii) Delete all $M \in \mathcal{M}_k$ such that $\beta(M) \geq \text{INV} - \varepsilon$. Let $\mathcal{M}'_k$ be the collection of remaining members of $\mathcal{M}_k$. If $\mathcal{M}'_k = \emptyset$ then terminate: z^k is a global ε-optimal solution of [P*]. Otherwise choose $M_k \in \text{argmin}\{\beta(M) : M \in \mathcal{M}'_k\}$.

(iii) Let $\bar{z}^k = \bar{z}(M_k)$. Select

$$i_k \in \arg\max_i \{g_i(\bar{z}_i^k) - L_{M_k,i}(\bar{z}^k)\}. \tag{3}$$

If $g_{i_k}(\bar{z}_{i_k}^k) - L_{M_k,i_k}(\bar{z}^k) = 0$ then terminate: $\bar{z}^k$ is a global optimal solution. Otherwise, divide M_k via $(\bar{z}^k, i_k)$. Let $\mathcal{P}_{k+1}$ be the partition of M_k and $\mathcal{M}_{k+1} = (\mathcal{M}'_k \setminus \{M_k\}) \cup \mathcal{P}_{k+1}$. Set $k \to k+1$ and go back to (i).

4. Convergence

In this section we give a formal proof of the convergence of the above algorithm.

Recall that $\bar{z}^k = \bar{z}(M_k)$ and denote $\bar{x}^k = \bar{x}(M_k)$, $\bar{y}^k = \bar{y}(M_k)$ (so $(\bar{x}^k, \bar{y}^k, \bar{z}^k)$ is an optimal solution of [CP(M)]).

Suppose the algorithm is infinite and let $\bar{z}$ be any cluster point of $\{\bar{z}^k\}$, e.g. $\bar{z} = \lim_{q\to\infty} \bar{z}^{k_q}$. Without loss of generality we may assume $\bar{x}^{k_q} \to \bar{x}$, $\bar{y}^{k_q} \to \bar{y}$ $(q \to \infty)$, and also $i_{k_q} = 1\ \forall q$, so that

$$1 \in \arg\max_i \left\{ g_i(\bar{z}_i^{k_q}) - L_{M_{k_q},i}(\bar{z}_i^{k_q}) \right\} \tag{4}$$

Note that if for some q we have $\bar{z}_1^{k_q} = 0$ then $g_i(\bar{z}_1^{k_q}) = 0 = L_{M_{k_q},1}(\bar{z}_1^{k_q})$, and the algorithm terminates at step (iii). Therefore, since the algorithm is infinite, we must have $\bar{z}_1^{k_q} > 0\ \forall q$.

Lemma 1. *If* $r_1 = g_1(0_+) > 0$ *then*

$$\bar{z}_1 = \lim_{q\to\infty} \bar{z}_1^{k_q} > 0 \tag{5}$$

In other words, $\bar{z}_1$ *is always a continuity point of* $g_1(t)$.

Proof. Let $M_{k_q,1} = [a_1^q, b_1^q]$, so that $\bar{z}_1^{k_q} \in [a_1^q, b_1^q]$ and $[a_1^q, b_1^q], q = 1, 2, \ldots$ is a sequence of nested intervals. If $0 \notin [a_1^{q_0}, b_1^{q_0}]$ for some q_0 then $0 \notin [a_1^q, b_1^q]\ \forall q \geq q_0$, and $\bar{z}_1^{k_q} \geq a_1^{q_0} > 0\ \forall q \geq q_0$, hence $\bar{z}_1 \geq a_1^{q_0} > 0$, i.e. (5) holds. Thus, it suffices to consider the case

$$a_1^q = 0\ \forall q,\ \bar{z}_1^{k_q} > 0\ \forall q. \tag{6}$$

Denote

$$\pi = \max_j \xi_j - \min_j c_{1j}. \tag{7}$$

Since $\sum_{j=1}^n \bar{x}_{1j}^{k_q} = \bar{z}_1^{k_q} > 0$ there exists j_0 such that $\bar{x}_{1j_0}^{k_q} > 0$. Define $(\tilde{x}^{k_q}, \tilde{y}^{k_q}, \tilde{z}^{k_q})$ such that $\tilde{x}_{1j_0}^{k_q} = 0,\ \tilde{x}_{ij}^{k_q} = \bar{x}_{ij}^{k_q}$ for all $(i, j) \neq (1, j_0)$, $\tilde{y}_{j_0}^{k_q} = \bar{y}_{j_0}^{k_q} - \bar{x}_{1j_0}^{k_q},\ \tilde{y}_j^{k_q} = \bar{y}_j^{k_q}$ for all $j \neq j_0$, $\tilde{z}_1^{k_q} = \bar{z}_1^{k_q} - \bar{x}_{1j_0}^{k_q},\ \tilde{z}_i^{k_q} = \bar{z}_i^{k_q}$ for all $i \neq 1$.

Clearly $(\tilde{x}^{k_q}, \tilde{y}^{k_q}, \tilde{z}^{k_q})$ is still feasible to [CP(M)], but when we replace $(\bar{x}^{k_q}, \bar{y}^{k_q}, \bar{z}^{k_q})$ by $(\tilde{x}^{k_q}, \tilde{y}^{k_q}, \tilde{z}^{k_q})$ the production cost at supply point 1 decreases by

$$L_{M_{k_q},1}(\bar{z}_1^{k_q}) - L_{M_{k_q},1}(\tilde{z}_1^{k_q}) = \frac{g_1(b_1^{k_q})}{b_1^{k_q}} \bar{x}_{1j_0}^{k_q} > 0,$$

the transportation cost in the arc $(1, j_0)$ decreases by

$$c_{1j_0} \bar{x}_{1j_0}^{k_q},$$

while the penalty incurred at demand point j_0 increases by

$$f_{j_0}(\bar{y}_{j_0}^{k_q} - \bar{x}_{1j_0}^{k_q}) - f_{j_0}(\bar{y}_{j_0}^{k_q}) \leq \xi_{j_0} \bar{x}_{1j_0}^{k_q}.$$

Thus the total cost decreases by at least

$$\delta = \frac{g_1(b_1^{k_q})}{b_1^{k_q}} \bar{x}_{1j_0}^{k_q} + (c_{1j_0} - \xi_{j_0}) \bar{x}_{1j_0}^{k_q}. \tag{8}$$

If $\pi \leq 0$ then $c_{1j_0} - \xi_{j_0} \geq 0$, hence $\delta > 0$ and $(\tilde{x}^{k_q}, \tilde{y}^{k_q}, \tilde{z}^{k_q})$ would be a better feasible solution than $(\bar{x}^{k_q}, \bar{y}^{k_q}, \bar{z}^{k_q})$, contradicting the optimality of the latter for [CP(M)]. Therefore $\pi > 0$.

Now suppose $r_1 = g_1(0_+) > 0$. Since $g_1(t)/t \to +\infty$ as $t \to 0_+$, there exists $\tau_1 > 0$ satisfying (see figure 1)

$$\frac{g_1(\tau_1)}{\tau_1} > \pi.$$

Observe that since $M_{k_q,1} = [0, b_1^{k_q}]$ is subdivided via $\bar{z}_1^{k_q}$, we must have $[0, b_1^{k_{q'}}] \subset [0, \bar{z}_1^{k_q}]$ for all $q' > q$, while $[0, \bar{z}_1^{k_q}] \subset [0, b_1^{k_q}]$ for all q. With this in mind, we shall prove that

$$b_1^{k_q} \geq \tau_1\ \forall q. \tag{9}$$

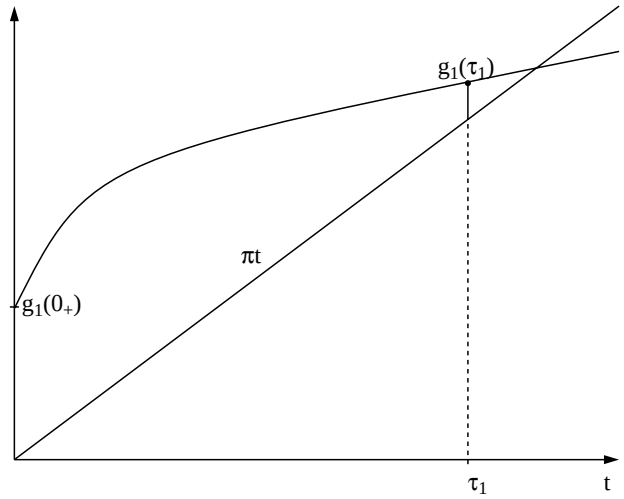


Fig. 1. A point τ_1 where $\frac{g_1(\tau_1)}{\tau_1} > \pi$

This will then imply, by the above observation, that

$$\bar{z}_1^{k_q} \geq \tau_1 \ \forall q,$$

and hence will complete the proof.

Suppose (9) does not hold, so that for some q we have $b_1^{k_q} < \tau_1$. Then $\bar{z}_1^{k_q} \leq b_1^{k_q} < \tau_1$, and since $g_1(t)$ is concave it is easily seen (see figure 2) that $g_1(b_1^{k_q}) \geq b_1^{k_q} g_1(\tau_1)/\tau_1$, i.e.

$$\frac{g_1(b_1^{k_q})}{b_1^{k_q}} \geq \frac{g_1(\tau_1)}{\tau_1}.$$

This, together with (8) implies that

$$\delta \geq \left(\frac{g_1(\tau_1)}{\tau_1} - \pi \right) \bar{x}_{1j_0}^{k_q} > 0$$

hence $(\tilde{x}^{k_q}, \tilde{y}^{k_q}, \tilde{z}^{k_q})$ is a better feasible solution than $(\bar{x}^{k_q}, \bar{y}^{k_q}, \bar{z}^{k_q})$. This contradiction shows that (9) must hold, thereby completing the proof of the lemma.

□

Theorem 1. *For $\varepsilon > 0$ Algorithm 1 is always finite. For $\varepsilon = 0$, either Algorithm 1 is finite or it generates an infinite sequence $\bar{z}^k = \bar{z}(M_k)$, $k = 1, 2, \ldots$. In the latter case, every cluster point of the sequence $\{\bar{z}^k\}$, or the sequence $\{z^k\}$, gives an optimal solution of* [P*].

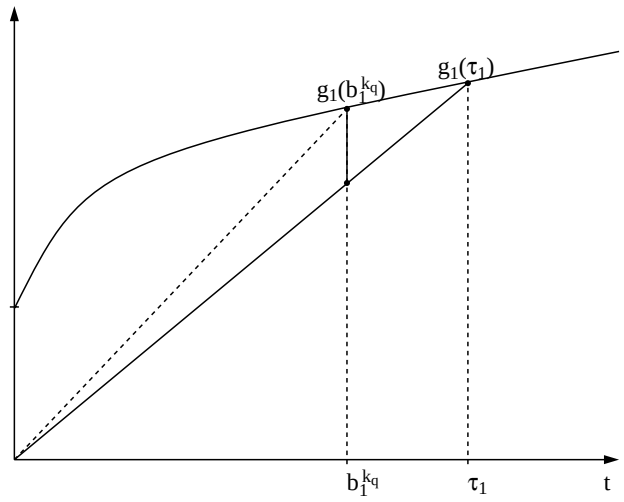
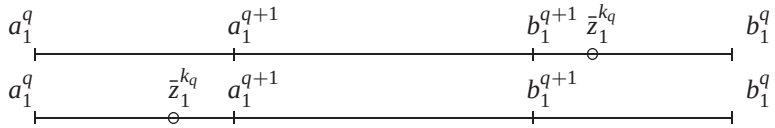


Fig. 2. The inequality following the concavity of $g_1(\tau_1)$

Proof. For $\varepsilon = 0$ let $\bar{z} = \lim_{q\to\infty} \bar{z}^{k_q}$ be a cluster point of $\{\bar{z}^k\}$, with, as previously, $i_{k_q} = 1\ \forall q$, and $M_{k_q,1} = [a_1^q, b_1^q]$. Since $\{a_1^q\}$ is nondecreasing, $\{b_1^q\}$ is nonincreasing, we have $a_1^q \to a_1^*$, $b_1^q \to b_1^*$. For each q, either of the following situations occurs.



If (A) occurs for infinitely many q, then

$$\lim_{q\to\infty} \bar{z}_1^{k_q} = \lim_{q\to\infty} b_1^q = b_1^*$$

If (B) occurs for infinitely many q, then

$$\lim_{q\to\infty} \bar{z}_1^{k_q} = \lim_{q\to\infty} a_1^q = a_1^*$$

Thus $\bar{z}_1 \in \{a_1^*, b_1^*\}$. By lemma 1, the concave function $g_1(t)$ is continuous at $\bar{z}_1$. Suppose for instance that $\bar{z}_1 = a_1^*$ (the other case is similar). Since $\bar{z}_1^{k_q} \to a_1^*$ and $a_1^q \to a_1^*$, it follows from the continuity of $g_1(t)$ at a_1^* that

$$g_1(\bar{z}_1^{k_q}) - L_{M_{k_q},1}(\bar{z}_1^{k_q}) \to 0$$

and hence, by (4),

$$g_i(\bar{z}_i^{k_q}) - L_{M_{k_q},i}(\bar{z}_i^{k_q}) \to 0 \;\; i = 1, \ldots, m$$

as $q \to \infty$. This implies that

$$g(\bar{z}^{k_q}) - L_{M_{k_q}}(\bar{z}^{k_q}) \to 0 \;\; q \to \infty$$

But from (2)

$$\varphi(\bar{z}^{k_q}) + L_{M_{k_q}}(\bar{z}^{k_q}) = \beta(M_{k_q}) \leq \min\{\varphi(z) + g(z) : z \in \Omega\}$$

Therefore

$$\varphi(\bar{z} + g(\bar{z}) = \min\{\varphi(z) + g(z) : z \in \Omega\}$$

i.e. $\bar{z}$ is an optimal solution of [P*], and hence $(\bar{x}, \bar{y}, \bar{z})$ is an optimal solution of [P].

□

Remark 1. Since the concave cost functions $g_i(t)$ may be discontinuous at $t = 0$, the above algorithm is directly applicable (without any preliminary modification of these functions) to problems with fixed charge costs. Furthermore, it is easily seen from the above proof that if $g_i(t)$ is discontinuous at $t = 0$ (so that $g_i(0_+) > 0 = g_i(0)$) then the problem does not change when $g_i(t)$ is replaced by a continuous function $\tilde{g}_i(t)$ linear for $0 \leq t \leq \tau_i$ and equal to $g_i(t)$ for $t \geq \tau_i$, where τ_i denotes a positive number satisfying $g_i(\tau_i) > \pi_i \tau_i$, and $\pi_i = \max_j \xi_j - \min_j c_{ij}$.

Remark 2. If the functions $g_i(t)$ are continuous then in Step (iii), instead of dividing M_{k,i_k} via $\bar{z}_{i_k}^k$, one can divide it via u_{k,i_k}, where u_{k,i_k} is any point in the segment $[0, \bar{z}_{i_k}^k]$ such that $g_{i_k}(u_{k,i_k}) - L_{M_k,i_k}(u_{k,i_k}) \geq g_{i_k}(\bar{z}_{i_k}^k) - L_{M_k,i_k}(\bar{z}_{i_k}^k)$. This does not affect adversely the convergence and may even speed it up in certain cases. Indeed, given any sequence $\bar{z}^{k_q} \to \bar{z}$, one can assume that $i_{k_q} = 1$ and $u_{k_q,1} \to \bar{u}$. Using then the same argument as in the above proof one can see that $\bar{u} \in \{a_1^*, b_1^*\}$, $g_1(u_{k_q,1}) - L_{M_{k_q},1}(u_{k_q,1}) \to 0$, hence $g_1(\bar{z}_1^{k_q}) - L_{M_{k_q},1}(\bar{z}_1^{k_q}) \to 0$, and, consequently, from (3) (where $i_{k_q} = 1$), $g(\bar{z}^{k_q}) - L_{M_{k_q}}(\bar{z}^{k_q}) \to 0$, implying, just as previously, the global optimality of $\bar{z}$.

5. Relations to previously known methods

Since our algorithm shares several common features with some previously published algorithms, it is useful to discuss the relations of our algorithm to these.

I. Falk and Soland's algorithm for separable nonconvex programming [2], was perhaps one of the earliest branch and bound algorithms to use rectangular partitions for branching and convex envelopes of the objective function for bounding over the partition sets. Nowadays these techniques have become quite common in global optimization (see e.g. [15]). Methods using these techniques differ mainly by the partitioning and bounding rules to take advantage of the particular structure of the problem under consideration.

Because the problem in [2] is too general, Falk and Soland's method cannot be expected to be efficient for large-scale instances, especially for problems with special

structure of the type we are considering here. In fact, to cope with the difficulty arising from the discontinuity of the objective function at certain points (as in the case of fixed charges), Falk and Soland's algorithm (Theorem 1 or 1' in [2]) uses the strong refining rule which could generate at each iteration up to 2^m new partition sets, and, furthermore, requires complete refinement, which implies generating an even larger number of partition sets. While clearly inefficient for large-scale problems, the strong refining rule is essential for the convergence of Falk-Soland's algorithm, as shown by the example given in [2]. To alleviate the computational burden of complete refinement, a procedure is suggested in [2] which in fact is itself a branch and bound procedure for determining the subproblem that corresponds to a member of the complete refinement; alternatively, a relaxed algorithm can be used which, however, guarantees convergence only in a weaker sense (Theorem 1').

By contrast, our algorithm uses a weak refining rule and does not require, even indirectly, complete refinement, which allows it to solve large-scale problems beyond the range of applicability of Falk and Soland's algorithm, while still guaranteeing convergence in just the same strong sense as in Theorem 1 of [2]. This is achieved by exploiting the specific structure of problem (P), namely: the variables are of 3 groups : x, y, z and the objective function is the sum of a separable convex function in y, and a separable nondecreasing concave function in (x, z). However, this requires a convergence proof totally different from Falk and Soland's. The basic idea underlying this proof can, besides, be used for a larger class of problems in which the objective function has the mentioned structure, but the constraint set may be an arbitrary polytope.

It should also be noted that although several algorithms for minimizing separable concave functions under linear constraints exist in the literature (see e.g. [15]), none, to our knowledge, is directly applicable to the case when these functions are discontinuous at the origin. The algorithm given in this paper is the first one able to compute directly a global optimal solution, without having to replace the objective function by a sequence of approximate continuous ones.

II. Other previously known algorithms which are in a sense or other related to ours are those by [18, 3], for the case when the concave costs are simply fixed charges and linear costs, [STLP]. Also for a transportation problem with piecewise linear cost functions, possibly discontinuous, [21] present a solution method with the same kind of branching as in our method.

Since the problem [P] considered in this paper has general nonlinear concave costs for production and nonlinear convex costs for the demand, it cannot be solved with the methods of [18, 3, 21].

On the other hand, as mentioned in Remark 1 at the end of the previous section, our method can as well solve the [STLP] and other problems with fixed charges and concave production costs, although it is not specifically designed for these problems. Rather, our goal is to develop an efficient method for problem [P], which is more general than [STLP].

Let us now discuss some practical considerations when using our method for the [STLP] and other problems with piecewise linear cost functions.

If the $g_i(t), i = 1, \ldots, m$ are concave piecewise linear nondecreasing functions, then, as mentioned in Remark 1, we can always assume that they are continuous at $t = 0$, hence continuous at every point (by replacing, if necessary, each $g_i(t)$ by a continuous

function linear in some interval $[0, \tau]$ and equal to $g_i(t)$ for all $t \geq \tau$). Furthermore, in Step (iii) of the Algorithm, instead of dividing M_{k,i_k} via $\bar{z}^k_{i_k}$ we can divide it via the breakpoint u_{k,i_k} of $g_{i_k}(t)$ nearest to $\bar{z}^k_{i_k}$ that satisfies $g_{i_k}(u_{k,i_k}) - L_{M_k,i_k}(u_{k,i_k}) \geq g_{i_k}(\bar{z}^k_{i_k}) - L_{M_k,i_k}(\bar{z}^k_{i_k})$. Since the number of breakpoints of each function $g_i(t)$ is finite, the Algorithm must terminate after finitely many steps.

So for piecewise linear cost functions, under the additional assumption of concavity, the method of [21] can be obtained as a special case of our method, if we restrict the branching to breakpoints of the cost function.

When a function $g_i(t)$ is of fixed charge type, e.g. $g_i(0) = 0$, $g_i(t) = r_i + \rho_i t > 0$ for $t > 0$ ($r_i > 0$, $\rho_i > 0$), it can be replaced by a continuous concave piecewise linear nondecreasing function with just one breakpoint u_i very near to 0. Computationally, there is no need to specify this breakpoint. It just suffices to assume that $u_i = \tau_i$, where $\tau_i > 0$ is arbitrarily small, so that replacing $g_i(t)$ with $\tilde{g}_i(t)$ as specified in Remark 1 does not change the problem. Then $\tilde{g}_i(t)$ is linear for $t \in [0, u_i]$ and $\tilde{g}_i(t) = g_i(t) = r_i + \rho_i t$ for $t \in [u_i, S_i]$. The subdivision of the interval $[0, S_i]$ via u_i then amounts to branching over the boolean variable δ_i, with $[0, u_i]$ corresponding to $\delta_i = 0$, and $[u_i, S_i]$ corresponding to $\delta_i = 1$ (each of the intervals $[0, u_i]$, $[u_i, S_i]$ will never be further divided because the function $g_i(t)$ is linear in these intervals).

A different method for the [STLP] is generalized Benders decomposition [5], as shown in [3]. In [8] we discuss the application of generalized cross decomposition, [9], but no computational results are given. A quick heuristic for the [STLP], using the Kuhn-Tucker conditions of the [STP] as a heuristic guide for changes in z, is presented in [18].

6. Illustrative example and computational results

Let us first discuss some implementation issues. In the branch-and-bound tree the following data are saved for each node: the lower bound, $\beta(M_k)$, the branching index, i_k, the value of the corresponding coordinate, $w^k(i_k)$ and the father index of the node. When a node has been investigated, either cut off or branched at, the sign of the father index is reversed. The correct rectangle for a certain node in the tree is found by tracing up through the tree the new bounds, while bearing in mind that the branching is made so that the upper rectangles have even node numbers, while the lower rectangles have odd numbers. We have not used information from solutions at previous nodes in the branch-and-bound tree, but has started from scratch at each node. The branch-and-bound procedure is thus quite straightforward.

All computationally efficient methods for solving the subproblem, [CP(M)], have asymptotic convergence. We get upper and lower bounds on the optimal objective function value, both converging to the optimal value. However, in practice, since we stop after a finite number of iterations, only upper and lower bounds are available when the decision about branching or cutting is made. Obviously the decision to cut a branch must be based on the lower bound. On the other hand, the branching must be based on the feasible solution, corresponding to the upper bound.

The usage of the lower bound when cutting branches might lead to unnecessary branching, when a branch could have been cut off. However, if the subproblem is solved

sufficiently well after the branching (or further down the tree), the final accuracy of the method is not affected.

The usage of an ϵ-optimal solution of the subproblem when deciding how to branch has in most cases no effect on the overall performance of the algorithm. However, if the approximation error is zero at the obtained solution point, it is not possible to branch at that point. This can happen even if the lower bound is not high enough to motivate cutting off the branch. In this case, one could either cut the branch, or search for another ϵ-optimal solution with a non-zero approximation error, where branching is possible. We have chosen the first option, even if it means that the overall accuracy is the sum of ε (used when cutting branches) and ϵ (used in the stopping criterion for the subproblem).

There are several choices of how to solve the subproblem, [CP(M)], and we have made some tests in order to find the one that makes the whole branch-and-bound method as efficient as possible. In [11] we find that separable programming combined with mean value cross decomposition, here denoted by SM, yields the most efficient solution method, while a modified version of the Frank-Wolfe method (used in [1] and modified as suggested in [17]), here denoted by FW, is not quite as efficient, when using $\epsilon = 1\%$. However, the solutions obtained by the two different methods might be of slightly different nature. While the line searches of the Frank-Wolfe method are likely to yield interior points, the separable programming approach yields solutions that are extreme in the obtained linear minimal cost network flow problem. This difference may affect the performance of our branch-and-bound method, since the branching is based on the actual values obtained in the subproblem solutions.

In the standard code used for solving the linear minimal cost network flow problems arising as the primal subproblem in mean value cross decomposition, both the costs and the capacities have to be integers. When solving an ordinary [STP], the non-integral capacities are scaled (typically multiplied by 10000) and truncated, which produces acceptable results. However, the linearization, $L_M(z)$, of the concave costs produces non-integral cost coefficients, so both the non-integral costs and the non-integral capacities have to be scaled and truncated. Furthermore, sometimes the linearization of the concave costs produce very steep slopes, i.e. very large cost coefficients. These circumstances together often result in overflow, i.e. integer values used in intermediate calculations in the network code become too large. Decreasing the scaling factor reduces the danger of overflow, but also the accuracy of the solution.

On the other hand, if after branching, the interval $[p_i, q_i]$ is small, the rounding of the coefficients mentioned above may have large effects. While the errors in cost and capacity coefficients probably do not change the optimal objective function value much, the dual solution may be quite different from what is should be. This is unfortunate if the dual solution is used as input to subsequent subproblems, as it is in a mean value cross decomposition method. A network code accepting real numbers for cost and capacity coefficients would eliminate this obstacle.

We have tested two branch-and-bound methods, BB-SM where the subproblem is solved by the SM method, and BB-FW, where the subproblem is solved by the FW method. BB-FW is found to be rather stable, while BB-SM fluctuates; it often outperforms BB-FW but is sometimes much worse. BB-SM is mostly faster than BB-FW when solving one single subproblem. However, sometimes the branch-and-bound

tree becomes much larger for BB-SM than for BB-FW. This is mainly an effect of the practical difficulties with rounding coefficients discussed above.

We have solved a number of randomly generated problems in various sizes, from 10 origins and 50 destinations to 100 origins and 500 destinations. The largest problem has 50.000 x-variables, 500 y-variables and 100 z-variables. However, the difficulty of the problems is found to depend not only on the size of the problem, but also on the relation between the concave costs and the linear and convex costs. This is also affected by the relation between m and n.

The supplies are drawn from a uniform distribution between 125 and 175. For each destination the shortage costs are drawn from a uniform distribution between 20 and 60 and the holding costs between 3 and 6. The probability density functions used are exponential distribution functions, which yields

$$\phi_j(d_j) = \lambda_j \exp(-\lambda_j d_j),\ F_j(d_j) = 1 - \exp(-\lambda_j d_j),\ E_j[d_j] = \frac{1}{\lambda_j}$$

and

$$f_j(y_j) = \alpha_j \left(y_j - \frac{1}{\lambda_j} \right) + \left(\frac{\xi_j + \alpha_j}{\lambda_j} \right) \exp(-\lambda_j y_j)$$

The parameters λ_j are drawn from a uniform distribution between 0.005 and 0.025, which yields expected demands in the interval between 40 and 200.

The coordinates for each point are generated randomly uniformly between 0 and 100, and the transportation cost coefficients are calculated as the Euclidean distance between the points multiplied by a constant. This constant is set to 0.5, which for the [STP] in average makes the nonlinear costs to be about 0.96 of the total costs at the optimum (following [1]).

The concave cost functions are chosen to be of the form

$$g_i(z_i) = \begin{cases} b_i z_i^{a_i} & \text{if } z_i > 0 \\ 0 & \text{if } z_i = 0 \end{cases}$$

where b_i is obtained by multiplying a random number, uniformly distributed between 1000 and 5000, with a chosen weight, c_g, (in the order of magnitude of 1.0). The exponents a_i are randomly generated in an interval $[\underline{a}, \bar{a}]$, where $\underline{a} \geq 0$ and $\bar{a} < 1$. The parameters c_g, $\underline{a}$ and $\bar{a}$ affect the difficulty of the problem significantly, so different values have been tested. For applications in chemical industry, we have encountered values of the exponents a_i in the interval [0.6, 0.7]. One can note that $g_i(z_i)$ is continuous if $a_i > 0$ and a step function, i.e. a fixed charge, if $a_i = 0$. We have not made any computational tests with concave functions of other forms, such as piecewise linear functions, although this would be perfectly feasible.

When a_i is positive and small, we note that after branching at some not very small value, the approximation in the upper part is much better than the approximation in the lower part. This depends on the form of concave function chosen in our tests (see figure 3). The effect is that in a majority of the cases, the upper branch is immediately cut off, while the error in the lower branch is not improved much. A heuristic improvement is to branch at a somewhat lower point than w_i^k. Some improvement was obtained by

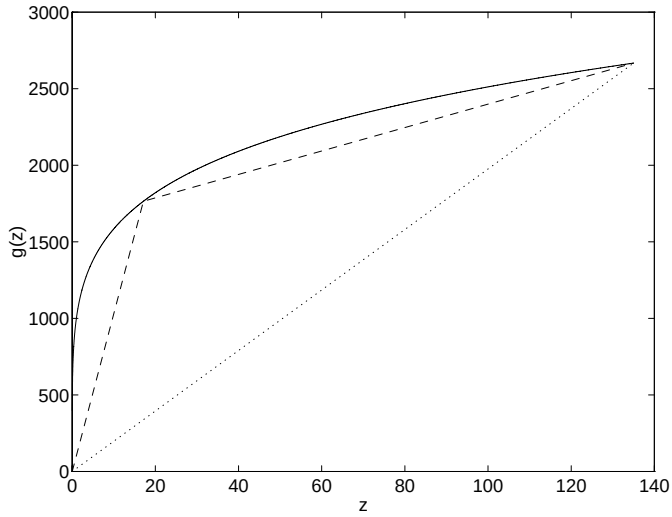


Fig. 3. Plot of function $g_1(z_1)$ (solid line). The dotted line is the first linearization, while the dashed line is the linearization after the branching

branching at $0.25w_i^k$ when $a_i < 0.1$, at $0.5w_i^k$ when $0.1 \leq a_i < 0.2$ and at $0.75w_i^k$ when $0.2 \leq a_i < 0.3$. If $a_i = 0\ \forall i$, the special case [STLP] is obtained. In that case we do not branch at w_i^k but close to zero.

Now we give a small illustrative example, with $m = 2$ and $n = 3$. The problem data are the following:

$c = \begin{pmatrix} 10 & 9 & 46 \\ 14 & 27 & 44 \end{pmatrix}$, $S = \begin{pmatrix} 135 \\ 147 \end{pmatrix}$, $h = \begin{pmatrix} 3.288 \\ 4.708 \\ 4.303 \end{pmatrix}$, $p = \begin{pmatrix} 52.46 \\ 47.17 \\ 20.74 \end{pmatrix}$, $\lambda = \begin{pmatrix} 0.00637 \\ 0.0153 \\ 0.00629 \end{pmatrix}$, $b = \begin{pmatrix} 1000 \\ 200 \end{pmatrix}$ and $a = \begin{pmatrix} 0.2 \\ 0.3 \end{pmatrix}$.

First the whole intervals $0 \leq z_1 \leq 135$ and $0 \leq z_2 \leq 147$ are linearized. The solution of the subproblem yields the lower bound 12455.9 and the upper bound 13972.0, which is used as the incumbent value. The largest error occurs for $i = 1$ and the solution value of this coordinate is $w_1^1 = 22.83$. However, since $a_1 = 0.2$ the branching point is moved to 17.12. In the lower branch the linearization is made for the interval $0 \leq z_1 \leq 17.12$ and in the upper branch for the interval $17.12 \leq z_1 \leq 135$, see figure 3. When solving the subproblem of the lower branch, we get a lower bound equal to 12577.5 and an upper bound equal to 12645.0 (which is an improved incumbent value) and the branch is cut off, since the relative error is approximately 0.5%. In the upper branch, the lower bound is increased to 12689.9, and the branch is cut off. Since all the branches are now cut off, the method is terminated. The ε-optimal solution is $z = \begin{pmatrix} 0.0 \\ 137.93 \end{pmatrix}$. Only 3 nodes had to be investigated, and a total of 21 Frank-Wolfe iterations were made.

Characteristics of the test problems are given in the tables 1 and 2. We give the number of supply points, m, and the number of demand points, n. (In table 1 we also give the number of supply points with zero cost (already open), m_o, and the number of supply points with $a_i = 0$ (i.e. fixed charges), m_z.) Furthermore we give the scaling factor for the concave cost function, c_g, the maximal and the minimal of the coefficients b_i and the maximal and the minimal of the coefficients a_i.

In table 1 a couple of smaller test problems are given: first two instances, **pa***, of [STLP] of the same size as in [18], m = 15 and n = 100 (9 of the 15 sources have zero cost, so there are only 6 z-variables), then two instances, **pb***, of [STLP] of the same size as in [3], m = 10 and n = 50, and finally 12 instances, **pc***, of the general problem [P] of various sizes with $0 \leq a_i \leq 0.4$.

In table 2 we give the main test problems: first 15 problems, **pd*** and **pe***, of various sizes and with various values of the parameters, then 13 problems, **pg***, of the size $m = 50$ and $n = 300$ with $a_i \in [0.1,\ 0.6]$ for increasing values of c_g, 5 problems, **ph***, of the same size with $a_i \in [0.6,\ 0.7]$ and 5 problems, **pi***, of the largest size, $m = 100$ and $n = 500$ with $a_i \in [0.6,\ 0.7]$.

The results of the tests are presented in tables 3 and 4. We give the number of nodes in the branch-and-bound tree, the total number of iterations in the subproblem and the CPU-time in seconds on a Sun Ultra workstation. The subproblems have been solved to the accuracy of 0.5% and the accuracy used to cut branches is 1%.

In the tables 3 and 4 we give computational results for both the two methods, BB-FW and BB-SM. In table 3 we find that **pa*** and **pb*** are difficult to solve for BB-FW, but easier for BB-SM. The same is true for **pc4** and **pc9**. On the other hand, for the problems **pc1**, **pc10** and **pc11**, BB-SM exhibits an extremely bad performance, while BB-FW does much better. It is obvious that there can be a big difference in difficulty for problems of similar or identical sizes. The problems **pc5, pc6, pc7, pc8** and **pc12**

Table 1. Problem characteristics for the first sets of test problems

Problem	m (m_o, m_z)	n	c_g	b_{MAX}/b_{MIN}	a_{MAX}/a_{MIN}
pa1	15 (9,6)	100	1.0	0/5770	0/0
pa2	15 (9,6)	100	0.8	0/5072	0/0
pb1	10 (0,10)	50	1.0	3428/6836	0/0
pb2	10 (0,10)	50	0.8	1882/5114	0/0
pc1	10 (0,0)	50	1.0	2900/6906	0.004/0.29
pc2	10 (0,0)	50	0.8	1863/4877	0.044/0.39
pc3	15 (9,0)	100	1.0	0/6686	0/0.37
pc4	15 (5,0)	100	0.8	0/5467	0/0.39
pc5	10 (0,0)	100	1.0	2514/6910	0.027/0.39
pc6	10 (0,0)	100	1.0	2954/6311	0.085/0.35
pc7	10 (0,0)	100	0.9	2002/6290	0.040/0.38
pc8	10 (0,0)	100	0.8	2088/4992	0.166/0.39
pc9	25 (0,0)	100	1.0	2067/6951	0.026/0.39
pc10	25 (0,0)	200	1.0	2230/6862	0.028/0.39
pc11	25 (0,0)	200	0.8	1765/5378	0.008/0.38
pc12	20 (0,0)	100	1.0	2076/6839	0.040/0.38

Table 2. Problem characteristics for the main test problems. ($m_o = 0, m_z = 0$)

Problem	m	n	c_g	b_{MAX}/b_{MIN}	a_{MAX}/a_{MIN}
pd1	25	200	1.0	1026/4813	0.20/0.69
pd2	20	100	1.0	1281/4927	0.20/0.69
pd3	25	100	1.0	1368/4688	0.21/0.69
pd4	25	200	1.0	1104/4908	0.21/0.64
pd5	25	200	0.5	688/2812	0.21/0.69
pd6	25	200	0.3	450/1452	0.22/0.69
pe1	25	200	0.4	466/1911	0.30/0.49
pe2	25	200	0.5	509/2406	0.30/0.48
pe3	25	200	0.6	711/2952	0.30/0.50
pe4	25	200	0.3	337/1469	0.42/0.58
pe5	25	200	0.4	500/1975	0.41/0.59
pe6	25	200	0.5	513/2407	0.40/0.60
pe7	25	200	0.05	59/244	0.51/0.70
pe8	25	200	0.1	110/491	0.50/0.68
pe9	25	200	0.2	229/037	0.50/0.69
pg1	50	300	0.0	0/0	0.11/0.60
pg2	50	300	0.2	209/958	0.11/0.59
pg3	50	300	0.4	428/1862	0.11/0.60
pg4	50	300	0.6	665/2874	0.10/0.58
pg5	50	300	0.8	832/3993	0.10/0.60
pg6	50	300	1.0	1080/4938	0.11/0.60
pg7	50	300	1.2	1297/5923	0.12/0.60
pg8	50	300	1.4	1434/6762	0.12/0.60
pg9	50	300	1.6	1677/7923	0.11/0.59
pg10	50	300	1.8	1811/8761	0.12/0.59
pg11	50	300	2.0	2075/9968	0.10/0.59
pg12	50	300	3.0	3075/14491	0.14/0.60
pg13	50	300	5.0	5287/24471	0.11/0.59
ph1	50	300	0.0	0/0	0.60/0.70
ph2	50	300	0.1	110/497	0.60/0.70
ph3	50	300	0.2	226/999	0.60/0.70
ph4	50	300	0.3	313/1496	0.60/0.70
ph5	50	300	0.4	459/1988	0.60/0.70
pi1	100	500	0.0	0/0	0.60/0.70
pi2	100	500	0.1	102/500	0.60/0.70
pi3	100	500	0.2	200/992	0.60/0.70
pi4	100	500	0.3	307/1486	0.60/0.70
pi5	100	500	0.4	431/1987	0.60/0.70

are all solved in a very short time, while other problems that are not larger, for example **pc1** and **pc2** for BB-SM, or **pb1** and **pb2** for BB-FW, take much longer.

Comparing BB-SM and BB-FW, we find that in many cases BB-SM is quickest, but in some cases it produces a much larger branch-and-bound tree than BB-FW. If one wish to keep the method as stable as possible, it might be best to sacrifice a little of the speed and use BB-FW. (It is possible that further modifications and the use of a better

Table 3. Computational results for the first sets of problems

Problem	BB-SM			BB-FW		
	Nodes	Iterations	CPU-time	Nodes	Iterations	CPU-time
pa1	4	15	0.628	28	940	1.838
pa2	6	30	0.279	10	414	0.794
pb1	31	136	0.335	115	5171	3.892
pb2	35	142	0.397	62	2151	1.665
pc1	167	618	1.991	12	121	0.097
pc2	139	299	1.044	36	1089	0.772
pc3	4	28	0.342	2	87	0.159
pc4	1	1	0.019	6	195	0.351
pc5	4	6	0.035	2	13	0.018
pc6	1	31	0.012	1	1	0.002
pc7	2	4	0.022	2	14	0.020
pc8	2	2	0.012	4	8	0.014
pc9	7	25	0.386	13	358	0.941
pc10	126	1681	160.872	14	132	0.694
pc11	48	278	20.261	34	578	3.096
pc12	1	1	0.008	2	10	0.022

accuracy in the network flow code may eliminate the "failures" of BB-SM, in which case it may be the best choice.)

In table 4 we give the results for our main test examples. It is further demonstrated that problems of the same size can behave very differently. Considering the values of $\underline{a}$ and $\bar{a}$, we find that **ph*** are easier than **pg***, which is natural, since the error in the linearization is larger the smaller a_i is. There is, however, also a very strong effect of the value of c_g on the difficulty of solving the problems. For small values of c_g the concave part of the cost is dominated by the convex part, and the problems are easy. (For $c_g = 0$, we get the [STP].) For large values of c_g, the concave part dominates, and these problems are also easy (solvable in fractions of a second). For some of these problems, **pg13**, **ph5** and **pi5**, the optimal solution is actually $z = 0$. However, somewhere between these extremes, we find a sharp increase in difficulty, especially for BB-FW. These effects are illustrated for **pg*** and **ph*** in figure 4. (Note that **pg*** have $a_i \in [0.1, 0.6]$ and **ph*** have $a_i \in [0.6, 0.7]$, but they have the same size.) The difficulty of the harder problems in each group naturally increases with problem size; the hardest of **pi*** are much harder than the hardest of **ph***.

Except for the "failures" (notably **pg5**, **pg7** and **pg9**), BB-SM seems to have the same basic behaviour. One can however note that the most difficult problem for BB-FW, **pi2**, is not that difficult for BB-SM.

In [13] we also present the result for two sets of instances of [STLP]. One of the size $m = 20$ and $n = 100$ and one of the size $m = 25$ and $n = 200$. Our method is not specifically developed for the [STLP], so the performance on those problems is not as impressive as for the others. We have solved larger problems than in [18, 3], but we do not claim that our method is superior. Again we note that the solution time is extremely dependent of c_g (up to 100 times longer for the same problem size), an effect not mentioned in [18, 3].

Table 4. Computational results for the main test problems

Problem	c_g	BB-SM			BB-FW		
		Nodes	Iterations	CPU-time	Nodes	Iterations	CPU-time
pd1	1.0	1	2	0.072	2	15	0.077
pd2	1.0	10	10	0.121	4	26	0.058
pd3	1.0	1	1	0.010	1	4	0.011
pd4	1.0	1	31	0.038	1	1	0.006
pd5	0.5	4	10	0.350	9	105	0.554
pd6	0.3	1	1	0.033	2	16	0.067
pe1	0.4	20	37	1.288	28	512	2.727
pe2	0.5	28	42	1.761	42	354	1.911
pe3	0.6	2	2	0.036	7	34	0.179
pe4	0.3	1	2	0.051	2	16	0.081
pe5	0.4	5	8	0.285	2	5	0.026
pe6	0.5	1	2	0.050	2	16	0.082
pe7	0.05	1	5	0.349	3	76	0.401
pe8	0.1	1	1	0.025	4	30	0.154
pe9	0.2	12	24	1.298	25	584	3.092
pg1	0.0	1	31	3.696	1	73	0.954
pg2	0.2	7	48	15.501	14	545	7.399
pg3	0.4	4	46	14.853	22	595	8.274
pg4	0.6	1	3	0.544	19	273	3.857
pg5	0.8	129	946	209.293	56	720	10.189
pg6	1.0	4	8	0.565	6	44	0.618
pg7	1.2	4	37	26.438	6	38	0.532
pg8	1.4	1	2	0.129	2	9	0.126
pg9	1.6	1	1	0.041	2	2	0.033
pg10	1.8	201	1377	851.432	4	12	0.175
pg11	2.0	1	1	0.083	2	9	0.126
pg12	3.0	1	1	0.037	1	1	0.016
pg13	5.0	1	31	0.116	1	1	0.017
ph1	0.0	1	31	2.981	1	72	0.942
ph2	0.1	3	45	4.728	8	129	1.813
ph3	0.2	1	1	0.049	2	8	0.110
ph4	0.3	1	1	0.037	1	1	0.016
ph5	0.4	1	31	0.104	1	1	0.015
pi1	0.0	1	31	13.067	1	105	4.210
pi2	0.1	1	3	2.494	115	2536	114.001
pi3	0.2	1	1	0.421	9	57	2.598
pi4	0.3	1	1	0.122	1	1	0.049
pi5	0.4	1	31	0.292	1	1	0.049

Summing up the computational tests, we find that the branch-and-bound trees are fairly limited in size, the average number of nodes in the branch-and-bound tree being less than 14 for BB-FW. This is encouraging, and indicates that the bounding subproblems are quite strong.

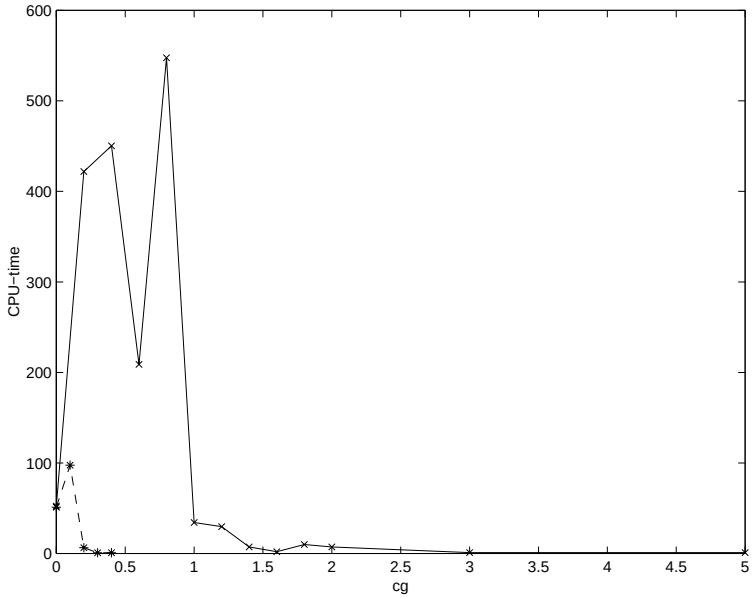


Fig. 4. Solution time as a function of c_g, for **pg*** (solid line) and **ph*** (dashed line), for the method BB-FW

7. Conclusions

In this paper we have introduced a new method for a production-transportation problem with stochastic demand (yielding convex costs at the destination points) and concave production costs. We prove convergence and report computational tests that indicate that large problems can be solved quite efficiently. We solve problems up to the size of 100 production units and 500 destinations (i.e. 100 variables appearing in concave functions).

We also study problems with different relations between concave and convex costs and find that this relation has a large impact on the solution time. If either the convex or the concave costs dominate, the problems (even larger instances) are easy to solve, while for a certain balance between the convex and concave costs, the problems are significantly harder. Increasing the problem size increases the solution time for the harder problems in each group much more than for the easier ones.

Acknowledgements. This work has been financed by the Swedish Research Council for Engineering Sciences, dnr. 92-824.

References

1. Cooper, L., LeBlanc, L.J. (1977): Stochastic transportation problems and other network related convex problems. Nav. Res. Logist. Quarterly **24**, 327–336
2. Falk, J.E., Soland, R.M. (1969): An algorithm for separable nonconvex programming problems. Manage. Sci. **15**, 550–569

3. Franca, P.M., Luna, H.P. (1982): Solving stochastic transportation-location problems by generalized Benders decomposition. Transportation Sci. **16**, 113–126
4. Frank, M., Wolfe, P. (1956): An algorithm for quadratic programming. Nav. Res. Logist. Quarterly **3**, 95–110
5. Geoffrion, A.M. (1972): Generalized Benders decomposition. J. Optim. Theory Appl. **10**, 237–260
6. Guisewite, G., Pardalos, P. (1990): Minimum concave cost network flow problems: Applications, complexity and algorithms. Ann. Oper. Res. **25**, 75–100
7. Holmberg, K. (1984): Separable programming applied to the stochastic transportation problem. Research report LiTH-MAT-R-1984-15. Department of Mathematics, Linköping Institute of Technology, Sweden
8. Holmberg, K., (1988): Generalized cross decomposition applied to the stochastic transportation-location problem. Research report LiTH-MAT-R-1988-15. Department of Mathematics, Linköping Institute of Technology Sweden
9. Holmberg, K. (1990): On the convergence of cross decomposition. Math. Program. **47**, 269–296
10. Holmberg, K. (1992): Linear mean value cross decomposition: A generalization of the Kornai-Liptak method. Eur. J. Oper. Res. **62**, 55–73
11. Holmberg, K. (1995): Efficient decomposition and linearization methods for the stochastic transportation problem. Comput. Optim. Appl. **4**, 293–316
12. Holmberg, K., Jörnsten, K. (1984): Cross decomposition applied to the stochastic transportation problem. Eur. J. Oper. Res. **17**, 361–368
13. Holmberg, K., Tuy, H. (1993): A production-transportation problem with stochastic demand and concave production costs. Research report LiTH-MAT-R-1993-30. Department of Mathematics, Linköping Institute of Technology Sweden. Accepted for publication in Math. Program.
14. Holmqvist, K., Migdalas, A., Pardalos, P. (1997): Greedy randomized adaptive search for a location problem with economies of scale. In: Bomze, I., Csendes, T., Horst, R., Pardalos, P., eds., Developments in Global Optimization, Nonconvex Optimization and Its Applications 18, pp. 301–313. Kluwer Academic Publishers
15. Horst, R., Tuy, H. (1993): Global Optimization, second edition. Springer, Berlin
16. LeBlanc, L.J., Cooper, L. (1974): The transportation-production problem. Transportation Sci. **8**, 344–354
17. LeBlanc, L.J., Helgason, R.V., Boyce, D.E. (1985): Improved efficiency of the Frank-Wolfe algorithm for convex network programs. Transportation Sci. **19**, 445–462
18. LeBlanc, L.J. (1977): A heuristic approach for large scale discrete stochastic transportation-location problems. Comp. Math. Appl. **3**, 87–94
19. Pardalos, P.M., Rosen, J.B. (1984): Constrained Global Optimization: Algorithms and Applications. Lect. Notes Comp. Sci 268, Springer, Berlin
20. Qi, L. (1985): Forest iteration method for stochastic transportation problem. Math. Program. Study **25**, 142–163
21. Rech, P., Barton, L.G. (1970): A non-convex transportation algorithm. IN: Beale, E.M., ed., Applications of Mathematical Programming Techniques, pp. 250–260
22. Tuy, H., Horst, R. (1988): Convergence and restart in branch and bound algorithms for global optimization algorithms. Math. Program. **41**, 161–184
23. Tuy, H. (1987): Global optimization of a difference of two convex functions. Math. Program. Study **30**, 150–182
24. Tuy, H. (1992): The complementary convex structure in global optimization. J. Glob. Optim. **2**, 21–40
25. Tuy, H. (1992): On nonconvex optimization problems with separated nonconvex variables. J. Glob. Optim. **2**, 133–144
26. Tuy, H. (1995): D.C. optimization: Theory, methods and algorithms. In: Horst, R., Pardalos, P., eds., Handbook of Global Optimization, pp. 149–216. Kluwer Academic Publishers
27. Van Roy, T.J. (1983): Cross decomposition for mixed integer programming. Math. Program. **25**, 46–63