European Journal of Operational Research 43 (1989) 53–64
North-Holland

A solver for the multi-objective transshipment problem with facility location*

Włodzimierz OGRYCZAK, Krzysztof STUDZIŃSKI and Krystian ZORYCHTA
Institute of Informatics, Warsaw University, PKiN VIII p. 850, 00-901 Warsaw, Poland

Abstract: This paper describes the results of research, development and implementation of the Dynamic Interactive Network Analysis System (DINAS) which enables the solution of various multi-objective transshipment problems with facility location. DINAS utilizes an extension of the classical reference-point approach to handling multiple objectives. In this approach, the decision-maker (DM) forms his requirements in terms of aspiration and reservation levels, i.e., he specifies acceptable and required values for given objectives. For providing DINAS with solutions to single-objective problems, a special TRANSLOC solver was developed. It is based on the branch-and-bound scheme with a pioneering implementation of the simplex special ordered network (SON) algorithm with implicit representation of the VUB and SUB (variable and simple upper bound, respectively) constraints.

A pilot version of the system is implemented on an IBM PC/XT microcomputer. DINAS is prepared as a menu-driven and easy-to-use system armed with a special network editor.

Keywords: Multiple criteria programming, network programming, general transportation

1. Introduction

The distribution-location type problems belong to the class of most significant problems directly leading to real-life applications of mathematical programming methods. Steadily rising costs and inflation as well as legal and political considerations, competition, fuel scarcity and many other factors have led, in recent years, many organizations to examine their present and planned distribution patterns or facility locations more closely. For instance, the impact of the energy crisis caused real impetus for re-evaluation of existing and often outmoded distribution patterns and methods.

Suppose we have a number of facilities and a number of customers or customer zones. Finding the distribution pattern is a fairly straightforward mathematical programming problem (e.g., the transshipment problem). When we add the possibility of removing or adding a number of facilities with their associated fixed costs, we have a more complex facility location problem which is, in general, an integer programming problem. Many real-life problems in industry, business, government and nonprofit organizations include a variety of conflicting goals and objectives as functions of their distribution patterns and facility locations. Adding these functions as criteria of optimization we get a multi-criteria transshipment problem with facility location.

As an illustration, the problem of depots location in a sugar-beet distribution system may be considered. A similar single-objective problem was studied by Jasińska and Wojtych (1984). They were dealing with a real-life problem concerning a sugar enterprise in Lower Silesia, Poland.

Consider a agricultural region containing a number of farms that produce sugar-beet. Each farm is considered as a supply point and is characterized by its total supply during the sugar-beet

* This research was supported in part by the International Institute for Applied Systems Analysis under Contracted Study: "Theory, Software and Testing Examples for Decision Support Systems".

Received 16 January 1989

harvesting period. The sugar-beet is delivered to a number of sugar-mills. Each sugar-mill has some limited total production capacity during the production season.

Climate conditions, poor storage facilities or an underdeveloped transportation network may cause losses of sugar-beet volume, losses of sugar content in the sugar-beet, or extremely high transportation costs. To avoid these difficulties, a part of the sugar-beet supply has to be delivered to special depots and stored there temporarily. The depots are considered as purchasing centers. Some of them are already in use and an amount of the sugar-beet is shipped through them. For further improvement of the sugar-beet transportation system, some additional depots have to be opened or some existing ones should be modernized. Each potential depot is characterized by the upper bound on its throughput as well as by the corresponding investment cost. The investment cost is treated as the fixed cost associated with locating (or modernizing) of the potential depot. Moreover, unit shipping costs are connected with all the delivery routes: from farms to sugar-mills. Each of the routes is also characterized by a capacity which bounds the maximal flow of the sugar-beet along the route.

The problem is to determine the number, location and sizes of the depots in use. Moreover, the corresponding sugar-beet flow from farms to sugar-mills directly or through depots has to be found so as to minimize the total transportation cost or/and the depots investment cost (provided that the total amount of the sugar-beet is delivered from farms to sugar-mills).

The problem represents a class of transshipment problems with facility location. It is a single-objective optimization problem if only one of the objective functions, the total transportation cost or the total investment cost, is minimized. However, it should be treated as a double-objective optimization problem if both the objectives are considered simultaneously. Single- or double-objective optimization can be insufficient in real-life circumstances and some additional objectives should then be taken into consideration. For instance, the total amount of the sugar-beet flow through depots is sometimes considered to be minimized as direct flow from farms to sugar-mills is technologically more efficient. As another objective maximization of the total amount of sugar-beet delivered by railway or maximization of the sugar production volume can be considered. The objectives are, in general, not comparable and thereby our problem should be considered as a multi-objective optimization problem.

The problem described above can be formulated in terms of network analysis. A complete generalized network model for the discussed class of distribution-location problems is presented in the next section.

Due to the multiple-objective formulation and to the integrality of location variables, the multicriteria transshipment problem with facility location is complicated and computationally complex. Hence, the method designed for solving the problem should be stable and fast in order to produce correct results or an acceptable approximation of a correct result within reasonable time.

This paper describes the theoretical and methodological backgrounds of the Dynamic Interactive Network Analysis System (DINAS) which is being developed with the purpose of solving various multi-objective transshipment problems with facility location. Some initial experience with a pilot version of the system implemented on an IBM-PC XT microcomputer are also presented.

2. The generalized network model

In the previous section we have introduced a class of transshipment problems with facility location. In this section we define the mathematical model of such problems.

A network model of the problem consists of nodes that are connected by a set of direct flow arcs. The set of nodes is partitioned into two subsets: the set of fixed nodes and the set of potential nodes. The fixed nodes represent 'fixed points' of the transportation network, i.e , points which cannot be changed. Each fixed node is characterized by two quantities: supply and demand. The potential nodes are introduced to represent possible locations of new points in the network. Some groups of potential nodes represent different versions of the same facility to be located (e.g., different sizes of a warehouse). For this reason, potential nodes are organized in the so-called selections, i.e., sets of nodes with the multiple choice requirement. Each selection is defined by the list of included potential nodes as

well as by lower and upper numbers of nodes which have to be selected (located). Each potential node is characterized by a capacity which bounds maximal flow through the node. The capacities are also given for all arcs but not for the fixed nodes.

Several linear objective functions are considered in the problem. The objective functions are introduced into the model by given coefficients associated with several arcs and potential nodes. They will be called cost coefficients independently of their real character in the objective functions. The cost coefficients for potential nodes are, however, understood in a different way than for arcs. The cost coefficient connected to an arc is treated as the unit cost of the flow along the arc whereas the cost coefficient connected to a potential node is considered as the fixed cost associated with the use (location) of the node rather than as the unit cost.

For simplicity of the model and the solution procedure we transform the potential nodes into artificial arcs. The transformation is performed by duplication of all potential nodes. After the duplication is done, all the nodes can be considered as fixed and each potential node is replaced by an artificial arc which leads from the node to its copy. Due to the transformation we get a network with fixed structure as all the nodes are fixed. Potentiality of artificial arcs does not imply any complication because each arc in the network represents a potential flow. Moreover, all the bounds on flows (i.e., capacities) are connected to arcs after this transformation. Additional nonstandard discrete constraints on the flow are generated only by the multiple choice requirements associated with the selections. Cost coefficients are connected only to arcs, but the coefficients connected to artificial arcs represent fixed costs.

A mathematical statement of this transformed problem takes the form of the following generalized network model:

Minimize

$$\sum_{(i,\,j)\in A\setminus A_a} f_{ij}^p x_{ij} + \sum_{(i,\,j)\in A_a} f_{ij}^p y_{ij},$$

$$p = 1, 2, \ldots, n_o, \tag{1}$$

Subject to

$$\sum_{(i,\,j)\in A} x_{ij} - \sum_{(j,\,i)\in A} x_{ji} = b_i, \quad i \in N, \tag{2}$$

$$0 \leqslant x_{ij} \leqslant c_{ij}, \quad (i,\,j) \in A\setminus A_a, \tag{3}$$

$$0 \leqslant x_{ij} \leqslant c_{ij} y_{ij}, \quad (i,\,j) \in A_a, \tag{4}$$

$$g_k \leqslant \sum_{(i,\,j)\in S_k} y_{ij} \leqslant h_k, \quad k = 1, 2, \ldots, n_s, \tag{5}$$

$$y_{ij} = 0 \text{ or } 1, \quad (i,\,j) \in A_a, \tag{6}$$

where

n_o = number of objective functions,
N = set of nodes (including copies of potential nodes),
n_s = number of selections,
A = set of arcs (including artificial arcs),
A_a = set of artificial arcs,
f_{ij}^p = cost coefficient of the pth objective associated with arc (i, j),
b_i = supply-demand balance at node i,
c_{ij} = capacity of arc (i, j)
g_k, h_k = lower, resp. upper number of (artificial) arcs to be selected in the kth selection,
S_k = set of (artificial) arcs that belong to the kth selection,
x_{ij} = decision variable that represents flow along arc (i, j),
y_{ij} = decision variable, $= 1$ for selected arc and $= 0$ otherwise.

The generalized network model of this form includes typical network constraints (2) with simple upper bounds (3) as well as a special discrete structure (5)–(6) connected to the network structure by variable upper bounds (4). While analysing the model we have to take advantage of all these features.

3. Interactive procedure for handling multiple objectives

There are many different concepts for handling multiple objectives in mathematical programming. We decided to use the so-called reference point approach which was introduced by Wierzbicki (1982). This concept was further developed in many papers and was used as a basis for construction of the software package DIDAS (Dynamic Interactive Decision Analysis and Support system). The DIDAS package proved to be useful in analysing conflicts and assisting in decision-making situations (Grauer et al., 1984).

The basic concepts of the reference point approach is as follows:

(1) the decision-maker (DM) forms his requirements in terms of aspiration levels, i.e., he specifies acceptable values for given objectives;

(2) the DM works with the computer in an interactive way so that he can change his aspiration levels during sessions of the analysis.

In our system we extend the DIDAS approach. The extension relies on additional use of reservation levels which allow the DM to specify necessary values for given objectives (Wierzbicki, 1986).

Consider the multi-objective program associated with the generalized network model:

$$\begin{aligned} &\text{Minimize} && \boldsymbol{q} \\ &\text{Subject to} && \boldsymbol{q} = F(\boldsymbol{x}, \boldsymbol{y}), \\ & && (\boldsymbol{x}, \boldsymbol{y}) \in Q, \end{aligned}$$

where $\boldsymbol{q}$ represents the objective vector, F is the linear vector-function defined by (1), and Q denotes the feasible set of the generalized network model, i.e., the set defined by conditions (2)–(6).

The reference point technique works in two stages. In the first stage, the DM is provided with some initial information which gives him an overview of the problem. The initial information is generated by separate minimization of all the objectives. More precisely, the following single objective programs are solved:

$$\min\left\{ f^p(\boldsymbol{x}, \boldsymbol{y}) + \frac{r_o}{n_o} \sum_{i=1}^{n_o} f^i(\boldsymbol{x}, \boldsymbol{y}) : (\boldsymbol{x}, \boldsymbol{y}) \in Q \right\}, \quad p = 1, 2, \ldots, n_o \tag{7}$$

where f^p denotes the pth objective function and r_o is an arbitrarily small number.

The so-called pay-off matrix

$$\boldsymbol{D} = (q_{pj}), \quad p = 1, 2, \ldots, n_o, \quad j = 1, 2, \ldots, n_o,$$

which yields information on the range of numerical values of each objective is then constructed. The pth row of the matrix $\boldsymbol{D}$ corresponds to the vector $(\boldsymbol{x}^p, \boldsymbol{y}^p)$ which solves the pth program (7). Each quantity q_{pj} represents the value of the jth objective at this solution (i.e., $q_{pj} = f^j(x^p, y^p)$). The vector with elements q_{pp}, i.e., the diagonal of $\boldsymbol{D}$, defines the utopia (ideal) point. This point, further denoted by $\boldsymbol{q}^u$, is usually not attainable but it is presented to the DM as a lower limit to the numerical values of the objectives. Taking into consideration the jth column of matrix $\boldsymbol{D}$ we notice that the minimal value in that column is $q_{pp} = q_p^u$. Let q_j^n be the maximal value, i.e.,

$$q_j^n = \max_{1 \leq p \leq n_o} q_{pj}.$$

Point $\boldsymbol{q}^n$ is called the nadir point and may be presented to the DM as an upper guideline to the values of the objectives. Thus, for each objective f^p, a reasonable but not necessarily tight upper bound $\boldsymbol{q}^n$ and a lower bound $\boldsymbol{q}^u$ are known after the first stage of the analysis.

In the second stage, an interactive selection of efficient solutions is performed. The DM controls the selection by two vector parameters: his aspiration level $\boldsymbol{q}^a$ and his reservation level $\boldsymbol{q}^r$, where

$$\boldsymbol{q}^u \leqslant \boldsymbol{q}^a < \boldsymbol{q}^r \leqslant \boldsymbol{q}^n.$$

The support system searches for the satisfying solution while using an achievement scalarizing function as a criterion in the single-objective optimization. Namely, the support system computes the optimal solution to the following problem:

Minimize

$$\max_{1 \leq p \leq n_o} u_p(\boldsymbol{q}, \boldsymbol{q}^a, \boldsymbol{q}^r) + \frac{r_o}{n_o} \sum_{p=1}^{n_o} u_p(\boldsymbol{q}, \boldsymbol{q}^a, \boldsymbol{q}^r) \tag{8}$$

$$\begin{aligned} &\text{Subject to} && \boldsymbol{q} = F(\boldsymbol{x}, \boldsymbol{y}), \\ & && (\boldsymbol{x}, \boldsymbol{y}) \in Q, \end{aligned}$$

where r_o is an arbitrarily small number and u_p is a function which measures the deviation of results from the DM's expectations with respect to the pth objective, depending on a given aspiration level $\boldsymbol{q}^a$ and a reservation level $\boldsymbol{q}^r$.

The computed solution is an efficient (Pareto-optimal) solution to the original multi-objective model. It is presented to the DM as a current solution. The DM is asked whether he finds this solution satisfactory or not. If the DM does not accept the current solution, he has to enter new aspiration and/or reservation levels for some objectives. Depending on this new information supplied by the DM, a new efficient solution is computed and presented as a current solution. This process is repeated as many times as the DM needs.

The function $u_p(\boldsymbol{q}, \boldsymbol{q}^a, \boldsymbol{q}^r)$ is a strictly monotone function of the objective vector $\boldsymbol{q}$ with value $u_p = 0$ if $\boldsymbol{q} = \boldsymbol{q}^a$ and $u_p = 1$ if $\boldsymbol{q} = \boldsymbol{q}^r$. In our system

we use (similarly as in Wierzbicki, 1986) a piecewise linear function u_p defined as follows:

$$u_p(q, q^a, q^r) = \begin{cases} a_p(q_p - q_p^a)/(q_p^r - q_p^a) \\ \quad \text{if } q_p < q_p^a, \\ (q_p - q_p^a)/(q_p^r - q_p^a) \\ \quad \text{if } q_p^a \leqslant q_p \leqslant q_p^r, \\ b_p(q_p - q_p^r)/(q_p^r - q_p^a) + 1, \\ \quad \text{if } q_p^r < q_p, \end{cases}$$

where a_p and b_p ($p = 1, 2, \ldots, n_o$) are given positive parameters. If the parameters a_p and b_p satisfy the inequalities $a_p < 1$ and $b_p > 1$, then the achievement functions u_p are convex. Minimization of the function u_p is then equivalent to minimization of a variable u_p defined as follows:

$$u_p = v_p + b_p v_p^- - a_p v_p^+, \tag{9}$$

$$v_p - v_p^+ + v_p^- = (q_p - q_p^a)/(q_p^r - q_p^a), \tag{10}$$

$$0 \leqslant v_p \leqslant 1, \tag{11}$$

$$v_p^+ \geqslant 0, \qquad v_p^- \geqslant 0. \tag{12}$$

4. General concept of the TRANSLOC solver

The TRANSLOC solver has been prepared to provide the multi-objective analysis procedure with solutions to single-objective problems. According to the interactive procedure described in Section 3, the TRANSLOC solver has to be able to solve two kinds of single-objective problems: the first one associated with calculation of the decision support matrix (problems (7)) and the second one associated with minimization of the scalarizing achievement function (problem (8)). Both kinds of problems have, however, the same main constraints which represent the feasible set of the generalized network model. Moreover, the other constraints of both kinds of problems can be expressed in a very similar way. So, we can formulate a general single-objective problem for the TRANSLOC solver as follows:

Maximize s (13)

Subject to

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = b_i, \quad i \in N, \tag{14}$$

$$w_k + \sum_{(i,j)\in S_k} y_{ij} = h_k, \quad k = 1, 2, \ldots, n_s, \tag{15}$$

$$u_p - v_p + d_p^+ - d_p^- = 0, \quad p = 1, 2, \ldots, n_o, \tag{16}$$

$$v_p - \frac{1}{a_p} d_p^+ + \frac{1}{b_p} d_p^- + \sigma_p \left(\sum_{(i,j)\in A\setminus A_a} f_{ij}^p x_{ij} + \sum_{(i,j)\in A_a} f_{ij}^p y_{ij} \right) = \delta_p, \quad p = 1, 2, \ldots, n_o, \tag{17}$$

$$u_o - \sum_{p=1}^{n_o} u_p = 0, \tag{18}$$

$$s + z + (r_o/n_o) u_o = 0, \tag{19}$$

$$0 \leqslant x_{ij} \leqslant c_{ij}, \quad (i, j) \in A \setminus A_a, \tag{20}$$

$$0 \leqslant w_k \leqslant h_k - g_k, \quad k = 1, 2, \ldots, n_s, \tag{21}$$

$$x_{ij} \leqslant c_{ij} y_{ij}, \quad (i, j) \in A_a, \tag{22}$$

$$u_p \leqslant z, \quad p = 1, 2, \ldots, n_o, \tag{23}$$

$$y_{ij} = 0 \text{ or } 1, \quad (i, j) \in A_a, \tag{24}$$

and, depending on the kind of optimization,

$$d_p^+ = 0, \quad d_p^- = 0, \qquad p = 1, 2, \ldots, n_o, \tag{25}$$

for the utopia point calculation, or

$$d_p^+ \geqslant 0, \quad d_p^- \geqslant 0, \quad 0 \leqslant v_p \leqslant 1, \quad p = 1, 2, \ldots, n_o, \tag{26}$$

for the achievement scalarizing function optimization, where

$$\sigma_p = 1 \quad \text{and} \quad \delta_p = 0$$

during utopia point calculation, and

$$\sigma_p = \frac{1}{q_p^r - q_p^a} \quad \text{and} \quad \delta_p = \frac{-q_p^a}{q_p^r - q_p^a}$$

during minimization of the achievement scalarizing function, whereas all the other quantities are the same as in Sections 2 and 3.

The above single-objective problem is a typical mixed integer linear program, i.e., it is a typical linear program with integrality conditions for some variables (namely y_{ij}). Mixed integer linear programs are usually solved by branch-and-bound approach with utilization of the simplex method. The TRANSLOC solver also uses this approach. Fortunately, only a very small group of decision variables is required to be integer in our model.

Therefore, we can use a simple branch-and-bound scheme in the solver.

Even for a small transshipment problem with facility location, the corresponding linear program (13)–(26) has rather large size. For this reason it cannot be solved directly with the standard simplex algorithm. Therefore, it is necessary to take advantages of its special structure.

Note that the inequalities (20)–(21) and (25) or (26) are standard simple upper bounds (SUB) which are usually processed outside of the linear programming matrix (Orchard-Hays, 1968). Similarly, inequalities (22) and (23) can be considered as the so-called variable upper bounds (VUB) and processed outside of the matrix due to a special technique. Basic rules of the techniques for SUB and VUB processing are developed in Section 5.

The main group of equality constraints (14) represents typical network relations. Similarly, equalities (15) and (16) include only variables with unit coefficients. All the rows (14)–(16) can be handled in the simplex method as the so-called special ordered network (SON) structure. Basic rules of the SON technique used in the TRANSLOC solver are developed, in Section 6.

Thus, only a small number of inequalities (17)–(19) has to be considered as typical rows of linear program. While taking advantage of this fact, the TRANSLOC solver can process transshipment problems of quite large dimensions.

5. Implicit representation of VUB and SUB constraints

The single-objective program (13)–(26) includes many inequalities of special simple forms. They can be partitioned into two groups. The first one consists of the so-called simple upper bounds (SUB), i.e., inequalities of the form $0 \leqslant x_j \leqslant c_j$ for some variables x_j and constants c_j, such as conditions (20)–(21), (26) with respect to variables v_p, and continuous form of (24). The second one includes the so-called variable upper bounds (VUB), i.e., inequalities of the form $x_j \leqslant c_j x_k$ for some variables x_j, x_k and constants c_j, such as conditions (22).

SUB constraints are usually implicitly represented in commercial simplex codes (see, e.g., Orchard-Hays, 1968). Schrage (1975) proposed a technique for implicit representation of VUB constraints. The technique was further developed and led to effective implementations (see, e.g., Todd, 1982).

The techniques presented in the literature deal, however, only with a simple form of VUB constraints. Namely, it is assumed that $c_j = 1$ in all VUBs and there are no upper bounds on x_k variables. The restriction of consideration to only unit variable bounds usually does not imply any loss of generality since it can be attained by a proper scaling of the problem. Unfortunately, in our model such scaling techniques cannot be used without destroying the special SON structure (see Section 6). Therefore, we were forced to extend the VUB techniques in such a way that nonunit variable upper bounds as well as some simple upper bounds on x_k variables were acceptable.

With respect to the VUB and SUB structures, the linear program under consideration can be formulated as follows. The numerical data consist of an $m \times n$ matrix $\boldsymbol{A}$ of rank m, a column m-vector $\boldsymbol{b}$, a row n-vector $\boldsymbol{f}$ and a column n-vector $\boldsymbol{c}$. In addition, the index set $N = \{1, 2, \ldots, n\}$ is partitioned into $J \cup K$, where J represents the so-called sons, i.e., variables which appear on the left-hand size of variable upper bounds, and K represents the so-called fathers, i.e., variables which appear on the right-hand side of variable upper bounds. Any variable that is not involved in any variable upper bound is regarded as a childless father. Set J is further partitioned into sets $J(k)$, $k \in K$, where $J(k)$ is the set (possible empty) of sons of the father $k \in K$. It is assumed that a son has only one father and that no father has a father. The father connected to a son x_j will be denoted by $k(j)$. The problem is then

Maximize $\boldsymbol{fx}$

Subject to

$$\boldsymbol{Ax} = \boldsymbol{b},$$

$$x_j \leqslant c_j x_k \quad \text{for all } k \in K \text{ and } j \in J(k),$$

$$x_k \leqslant c_k \quad \text{for all } k \in K,$$

$$\boldsymbol{x} \geqslant 0.$$

Let s_j be a slack variable for the variable upper bound $x_j \leqslant c_j x_k$, so that

$$x_j + s_j = c_j x_k, \quad x_j \geqslant 0, \quad s_j \geqslant 0.$$

Consider a basic solution to the problem. The basis consists of the $m + v$ columns corresponding

to some sons x_j, some fathers x_k and some slacks s_j (where v denotes the number of VUBs). From each VUB either one slack s_j or one son x_j belongs to the basis. Calculation of the basic slacks is beyond our interest so they can be simply dropped from the basis, i.e., the corresponding rows and columns can be dropped. Furthermore, the basic sons which arrive in the other VUBs can be eliminated by submission $x_j = c_j x_k$. So, the whole basic solution can be computed from an $m \times m$ basis consisting of some linear combinations of columns from matrix A.

A basic solution to the problem is characterized as follows. The set of sons is partitioned into the three sets $J = J^L \cup J^U \cup J^B$, where J^L denotes the set of nonbasic sons fixed at their lower limits (i.e., $x_j = 0$), J^U denotes the set of nonbasic sons fixed at their upper limits (i.e., $x_j = c_j x_k$), and J^B denotes the set of basic sons. Similarly, the set of fathers is partitioned into three sets $K = K^L \cup K^U \cup K^B$, where K^L denotes the set of nonbasic fathers fixed at their lower limits (i.e., $x_k = 0$), K^U denotes the set of nonbasic fathers fixed at their upper limits (i.e., $x_k = c_k$), and K^B denotes the set of basic fathers. The basis B consists of the columns corresponding to basic sons $B_j = A_j$ and of the columns corresponding to basic fathers given by the formula

$$B_k = A_k + \sum_{j \in J(k) \cap J^U} c_j A_j.$$

Consider a basic solution given by a basis B and sets J^L, J^U, J^B, K^L, K^U, K^B. For the determination of a nonbasic variable to enter the basis in the simplex algorithm it is necessary to compute the so-called reduced costs. Let z_i denote an ordinary reduced cost connected to the column A_i, i.e.,

$$z_i = f_i - f^B B^{-1} A_i, \quad i \in J \cup K,$$

where f^B denotes the basic part of the cost vector f. Due to implicit representation of VUBs, the reduced costs associated with several nonbasic variables then take the form

$$d_j = z_j \quad \text{for } j \in J,$$

$$d_k = z_k + \sum_{j \in J(k) \cap J^U} c_j z_j \quad \text{for } k \in K.$$

Thus, in comparison with pricing in a standard simplex algorithm, pricing with implicit representation of VUBs needs calculation of linear combinations of ordinary reduced costs as the only one additional operation.

Due to handling of the SUB structure together with the VUB constraints, a nonbasic variable x_j or x_k is considered as potential incoming variable if one of the following conditions is fulfilled:

(A) $d_j < 0$ and $j \in J^L$,

(B) $d_j > 0$ and $j \in J^U$,

(C) $d_k < 0$ and $k \in K^L$,

(D) $d_k > 0$ and $k \in K^U$.

Implicit representation of VUBs makes some degenerated simplex iterations so simple that they can be performed on-line during pricing. Namely, if x_j is an incoming variable and $k(j) \in K^L$, then the corresponding simplex iteration depends only on change sets J^L and J^U, i.e., x_j is moved from the set J^L to the set J^U or vice versa. Such an operation can be performed while pricing before the computation of reduced costs for fathers.

Let x_s ($s \in J$ or $s \in K$) be a variable chosen to enter the basis. Considering changes in the basic solution while the value of x_s is either increased for $s \in J^L \cup K^L$ or decreased for $s \in J^U \cup K^U$ by a nonnegative parameter θ, we get six formulae for upper bounds on the parameter θ and six corresponding formulae for the determination of the outgoing variable (for details, see Ogryczak et all., 1987). Crossing these formulae with four types of incoming variables we get 19 types (5 crisscrossings are not allowed) of the simplex transformations performed in the algorithm with implicit representation of VUB and SUB structures. The simplest transformation depends only on moving some variable from one set to another without any change of the basis. Most of the transformations depend on performing one of the following operations:

(a) some basic column multiplied by a scalar is added to another basic column;

(b) some basic column is replaced by a nonbasic column or a linear combination of nonbasic columns.

More complex transformations use both the above operations and the most complex one needs two operations of type (a) and one operation of type (b).

6. The simplex SON algorithm

The simplex SON procedure was developed by Glover and Klingman (1981, 1985). It is a partitioning method for solving LP problems with embedded network structure. Every problem of this type is characterized by a full row rank matrix A partitioned as follows:

$$A = \begin{pmatrix} \mathbf{ANN} & \mathbf{ANL} \\ \mathbf{ALN} & \mathbf{ALL} \end{pmatrix},$$

where **ANN** $(m \times n)$ denotes the matrix corresponding to a pure network problem, and the other submatrices **ANL** $(m \times p)$, **ALN** $(q \times n)$, and **ALL** $(q \times p)$ consist of any real elements.

The matrix A of the auxiliary LP problem discussed in Section 3 has obviously this form. The matrix **ANN** is an incidence matrix corresponding to the transportation network studied in Section 2. Therefore, each constraint represented by a row of **ANN** corresponds to a node of the network and will be referred to as node constraint. Moreover, each variable represented by a column of **ANN** corresponds to an arc of the network and will be referred to as arc variable. There are two classes of **ANN** columns: columns containing exactly two nonzero entries in **ANN** (one $+1$ and one -1), called ordinary arcs, and columns containing exactly one nonzero entry in **ANN** ($+1$ or -1), called slack arcs. The -1 entry in a column indicates the node where the arc begins and the $+1$ entry in a column indicates the node where the arc ends. If a column has exactly one nonzero element pointing one of the arc endpoints, then an artificial node outside of the network can be meant as the second arc endpoint.

The SUB and VUB simplex algorithms use a basis B which is composed of $m+q$ linearly independent columns selected from matrix A. Any basis B may be partitioned as follows:

$$B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix},$$

where B_{11} is a nonsingular submatrix of **ANN**. It appears to be better for the effectiveness of the algorithm if the rank of B_{11} is as large as possible.

Let $x_B = (x_{B1}, x_{B2})$ denote the basic part of the decision variable vector x, where x_{B1} and x_{B2} correspond to the B_{11} and B_{12} submatrices, respectively. Thus, the basic variables x_{B1} are exclusively arc variable. The basic variables x_{B2} may also contain arc variables. Similarly, the rows of B_{11} are exclusively node rows but the matrix (B_{21}, B_{22}) may also contain node rows.

The basis inverse B^{-1} may be written as follows:

$$B^{-1} = \begin{pmatrix} B_{11}^{-1} + B_{11}^{-1}B_{12}V^{-1}B_{21}B_{11}^{-1} & -B_{11}^{-1}B_{12}V^{-1} \\ -V^{-1}B_{21}B_{11}^{-1} & V^{-1} \end{pmatrix},$$

where $V = B_{22} - B_{21}B_{11}^{-1}B_{12}$.

Define the so-called master basis tree (MBT) associated with a given basis. The set of nodes of the tree contains all nodes of our LP embedded network problem plus an external node called the master root. Thus, MBT always contains $m+1$ nodes $N = \{0, 1, \ldots, m\}$, where 0 is the master root, and m the number of arcs. The nodes of MBT that correspond to rows of B_{21} are called externalized roots (ERs). Each ER is connected to the master root by an externalized arc (EA).

All of the ordinary arcs in B_{11} belong to MBT. There may be two types of slack arcs associated with B_{11}. If a slack arc in B_{11} is a slack arc of ANN, then the arc is replaced by an arc between the master root and its unique node. If a slack arc in B_{11} is an ordinary arc in ANN, then it is replaced by an arc between its nodes in ANN (one of these endpoints is an ER node).

The arcs in the master basis tree have a natural orientation defined as follows: if an edge (u, v) belongs to MBT and node u is nearer the master root than v, then u is called the predecessor of v, and v is called the immediate successor of u.

The master basis tree is represented by the following node functions.

(1) PRED: the values of the function are defined as follows:

PRED$[i]$ = the predecessor of node i in MBT.

(2) THREAD: the function defines as connecting link (thread) which passes through each node exactly once. If i is a node on the thread, then THREAD $[i]$ is the next one. The alternation of the nodes on the thread is defined by using the preorder method of tree passage.

Let $P = \binom{P_1}{P_2}$ denote the column vector selected to enter the basis matrix (P_1 specifies the part of P associated with B_{11} and P_2 the part associated with B_{21}). Similarly, $\alpha = \binom{\alpha_{B1}}{\alpha_{B2}}$ denotes the representation of P in terms of B.

We have $\alpha = B^{-1}p$ and hence using the partitioning formula for B^{-1} we obtain the following system of equations

$$\alpha_{B2} = V^{-1}(-B_{21}B_{11}^{-1}P_1 + P_2),$$

$$\alpha_{B1} = B_{11}^{-1}(P_1 - B_{12}\alpha_{B2}).$$

Suppose that the matrix $D = V^{-1}$ (i.e., the right down corner part of the matrix B^{-1}) is attained in the explicit form. Thus, the multiplication by the matrix in the former formula may be simply performed. Both the formulae include a multiplication $x = B^{-1}G$ with some vector G. This multiplication is equivalent to solving an upper triangular system $\tilde{B}_{11}\tilde{x} = \tilde{G}$, where the matrix $\tilde{B}_{11}$ consists of the rows and columns of B_{11} ordered according to the corresponding nodes and arcs on the THREAD line of MBT.

Each column of $\tilde{B}_{11}$ has at most two nonzero elements. One of them is located at the diagonal and corresponds to a node v while the second one (if it exists) is located above the diagonal and corresponds to the predecessor of v. Hence, if the THREAD line is passed backward and node v has come across, then the value of the variable represented by v is computed and simultaneously the value of the variable represented by the predecessor of v is modified. Thus, a single pass through the master basis tree along the reverse of the THREAD line is sufficient for computing the x solution. The cost of such a procedure is proportional to the number of nodes in MBT.

Let $c_B = (c_{B1}, c_{b2})$ denote the vector of basis cost coefficients. The dual vector $w = (w_1, w_2) = c_B B^{-1}$ is needed at the pricing step of the simplex method and may be computed as follows:

$$w_2 = (c_{B2} - c_{B1}B_{11}^{-1}B_{12})V^{-1},$$

$$w_1 = (c_{B1} - w_2 B_{21})B_{11}^{-1}.$$

Multiplication by matrix V^{-1} may be directly computed since the matrix is assumed to be kept in explicit form. Further, both the last formulae include multiplications of the form $w = HB_{11}$ which can be effectively using the master basis tree structure for B_{11}, similarly as while computing the primal solution x.

Consider a single step of the simplex method. When the incoming and outgoing variables are chosen then the whole basis representation has to be changed and adjusted to the new situation. Thus, the problem arises how to change in a single simplex iteration the matrix D and the functions describing the master basis tree.

Let x_s and x_r denote the incoming and outgoing variables, respectively, and let x_t be the so-called transfer variable that belongs to x_{B2} and replaces x_r in x_{B1}, if it is possible.

At each iteration, the variables can alter by the transitions:

- Incoming variables x_s: $x_N \rightarrow x_{B1}$ or x_{B2}.
- Outgoing variable x_r: x_{B1} or $x_{B1} \rightarrow x_N$.
- Transfer variable x_t: $x_{B2} \rightarrow x_{B1}$, or no change.
- Transfer ER nodes: $x_{B1} \rightarrow x_{B2}$ (one ER more), or $x_{B2} \rightarrow x_{B1}$ (one ER less), or no change.

If an arc is added to the master basis tree, then a loop is closed. In order to have a tree in the next iteration also, the loop must be cut and exactly one arc from the loop must be deleted. It is the fundamental exchange rule for the master basis tree.

At each iteration the matrix D is transformed by elimination using a given pivot row. The following cases appear when the elimination is performed:

- the pivot row is within the rows of D;
- the pivot row is outside of D;
- a row and a column of D are dropped;
- a new row and a new column are added to D;
- a column (row) of D is replaced by another column (row) from outside of D.

Combining the elimination cases with the transition rules for the incoming, outgoing and transfer variables we get seven types of basis exchange steps. When x_{B1} is maximal relative to x_{B2}, exactly one of the seven types of basis exchange steps will occur and their updating prescriptions will maintain x_{B1} maximal.

The main features of the discussed approach are cheap multiplication algorithms with basis inverse, accelerated labelling algorithms for modifying the master basis tree in an efficient manner and a compact form of the basis inverse occupying a small memory space only.

7. PC implementation and numerical example

A pilot version of the DINAS system is implemented on an IBM-PC XT/AT or an IBM-com-

patibles. The system consists of three programs prepared in the C programming language:

- the interactive procedure for efficient solutions generation,
- the TRANSLOC solver for single-objective problems,
- the network editor for input data and results examination.

The basic concept of the interactive scheme for efficient solutions generation is as follows:

- the DM works with the system in an interactive way so that he can change his aspiration and reservation levels in any direction;
- after editing the aspiration and reservation levels, the system computes a new efficient solution by solving a corresponding single-objective problem;
- each computed efficient solution is put into a special solution base and presented to the DM as the current solution in the form of tables and bars which allow him to analyse performances of the current solution in comparison with the previous solutions.

Operations available in the DINAS interactive procedure are partitioned into three groups and corresponding three branches of the main menu: PROCESS, SOLUTION and ANALYSIS. The PROCESS branch contains basic operations connected with processing the multi-objective problem and generation of several efficient solutions. Included are operations such as editing and converting the problem, computation of the pay-off matrix and, finally, generating a sequence of efficient solutions depending on the edited aspiration and reservation levels.

The SOLUTION branch contains additional operations connected with the current solution. The DM can examine in details the current solution using the network editor or analyse only short characteristics such as objective values and selected locations. Values of the objective functions are presented in three ways: as a standard table, as bars in the aspiration/reservation scale and as bars in the utopia/nadir scale. The bars show percentage level of each objective value with respect to the corresponding scale. The DM may also print the current solution or save it, for using in next runs of the system with the same problem. There is also available a special command to delete the current solution from the solution base if the DM finds it as quite useless.

The ANALYSIS branch collects commands connected with operations on the solution base. The main command COMPARE allows the DM to perform comparison of all the efficient solutions from the solution base or of some subset of them. In the comparison, only the short characteristics of the solutions are used, i.e., objective values in the form of tables and bars as well as tables of selected locations. Moreover, some commands which allow the DM to select various efficient solutions from solution base as the current solution are included in this branch. There also exists the opportunity to restore some (saved earlier) efficient solution to the solution base.

DINAS is armed with the built-in network editor EDINET. EDINET is a full-screen editor specifically designed for input and edit data of the generalized network model defined in Section 2. The essence of the EDINET concept is a dynamic movement from some current node to its neighbouring nodes and vice versa, according to the network structure. The input data are inserted by a special mechanism of windows while visiting several nodes. Independently, a list of the nodes in alphabetic order and a graphic scheme of the network is available at any time. Some special windows are also used for defining objective functions.

DINAS can process problems consisted of:

- up to seven objective functions,
- a transportation network with up to one hundred of nodes and a few hundreds of arcs,
- up to fifteen potential locations.

However, rather smaller test problems have been solved up to now. They were usually prepared as parts or simplifications of real-life problems.

As an example we consider a small artificial part of the real-life sugar-beet transshipment system mentioned in Section 1. Eight fixed nodes (seven farms: Udry, Dubiel, Smrock, Gondek, Pogaj, Runo, Cuple; one sugar-mill Klew) and three potential depots (Tyn4, Tyn7, Jurga) are considered. A network scheme of this example is presented in Figure 1. The numbers inside the node circles denote supply amounts in case of farms, a (negative) demand on the sugar-beet for the sugar-mill, and capacities of the corresponding potential depots. Note that the sum of the farms supplies is equal to the demand of the sugar-mill. In fact, the potential depots Tyn4 and Tyn7 repre-

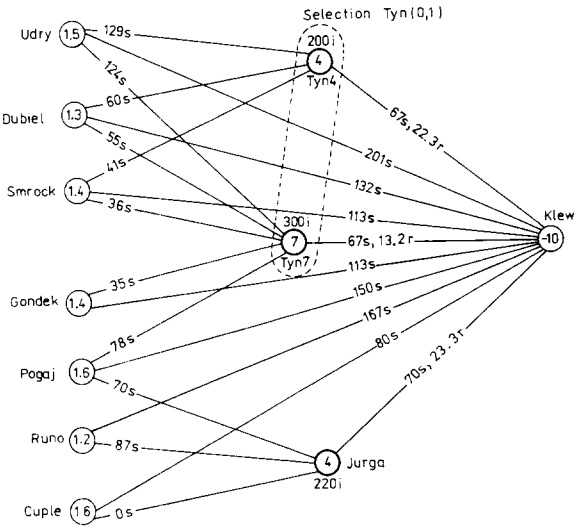

Figure 1

sent two versions of the same depot which has to be located. These versions differ only in their capacities and hence they are considered as one selection Tyn with its lower and upper bounds equal to 0 and 1, respectively. The arcs in the network represent possible flows of the sugar-beet and are directed from farms to the sugar-mill or to depots, and from the depots to the sugar-mill. All the arcs have unlimited capacities.

Three objectives are considered in the problem: minimization of the INVEST function, minimization of the SHIP function and maximization of the RAIL function. INVEST represents an investment cost associated with location of the potential depots. SHIP expresses a total transportation cost in the network. The RAIL function is used by the railway administration to evaluate shipping of the sugar-beet from the depots to the sugar-mill (these routes are serviced by the railway). The numbers in Figure 1 completed by the letters i, s or r, connected to arcs and potential nodes denote the corresponding objective coefficients (a coefficient is assumed to be equal to 0 if there is no number with the corresponding sign).

While performing the multi-objective analysis with the DINAS system, six efficient solutions have been identified. The objective values for all these efficient solutions are given in Table 1. Additionally, the corresponding locations of the potential depots are presented in Table 2. What is important, the whole multi-objective analysis including calculation of the pay-off matrix and six efficient solutions took only several minutes of interactive work with the IBM-PC XT microcomputer.

The PC version of the DINAS system was also used for solving an artificial part of the real-life four-objective problem connected with the health service districts reorganization (Ogryczak et al., 1988). Moreover, a real-life two-products transportation problem was modelled as a single-product network problem (about 50 nodes and 150 arcs) and successfully solved with DINAS.

8. Concluding remarks

Initial experiences with the DINAS system on small testing examples confirm appropriateness of the used methodology for solving multi-objective transshipment problems with facility location. The

Table 1
Objective values for the efficient solutions

Objective	Solution 1	Solution 2	Solution 3	Solution 4	Solution 5	Solution 6
INVEST	0	200	220	300	420	520
SHIP	1357.9	1337.9	1317.9	1294.5	1297.9	1258.5
RAIL	0	89.2	93.2	92.4	182.4	172.4

Table 2
Potential depots activity for the efficient solutions

Depot	Solution 1	Solution 2	Solution 3	Solution 4	Solution 5	Solution 6
Tyn4	No	Yes	No	No	Yes	No
Tyn7	No	No	No	Yes	No	Yes
Jurga	No	No	Yes	No	Yes	Yes

interactive scheme is very easy to understand and it provides the DM with the most important characteristics of generated efficient solutions. Moreover, the DM controls the system with unsophisticated parameters: aspiration and reservation levels. In effect, one easily reaches a satisfactory solution in a few interactive steps. Due to sufficiently good effectiveness of the TRANSLOC solver, one interactive step is performed within a reasonable time even if an IBM-PC XT is used.

On the other hand, we have noticed that introducing multiple objectives into the transshipment problem with facility location transformed this easy discrete problem into a complex one. Namely, it has been proved that the single-objective problem connected with minimization of the achievement function is far more complex than the original single-objective problems connected with minimization of several objective functions. The original single-objective problem is solved with the branch-and-bound method after examination of only a few subproblems while minimization of the achievement function usually requires to analyse many branches of the tree. Thus, for solving large real-life problems, rather a more advanced hardware than the standard IBM-PC XT should be used.

References

Glover, F. and Klingman, D. (1981), "The simplex SON method for LP/embedded network problems", *Mathematical Programming Study* 15, 148–176.

Glover, F. and Klingman, D. (1985), "Basis exchange characterization for the simplex SON algorithm for LP/embedded networks", *Mathematical Programming Study* 24, 141–157.

Grauer, M., Lewandowski, A. and Wierzbicki, A.P. (1984), "DIDASS—theory, implementation and experiences", in: M. Grauer and A.P. Wierzbicki (eds.), *Interactive Decision Analysis*, Springer, Berlin, 1984, 22–30.

Jasińska, E. and Wojtych, E. (1984), "Location of depots in a sugar-beet distribution system", *European Journal of Operational Research* 18, 396–402.

Ogryczak, W., Studziński, K. and Zorychta, K. (1987), "A solver for the transshipment problem with facility location", in: A. Lewandowski and A. Wierzbicki (eds.), *Theory, Software and Testing Examples for Decision Support Systems*, IIASA, Laxenburg, 125–145.

Ogryczak, W., Studziński, K. and Zorychta, K. (1988), "Solving multiobjective distribution-location problems with the DINAS system", in: A. Lewandowski and A.P. Wierzbicki (eds.), Aspiration based Decision Support Systems, Springer, Berlin, 1989 (*Lecture Notes in Economics and Mathematical Systems* 331), 230–250.

Orchard-Hays, W. (1968), *Advanced Linear-Programming Techniques*, McGraw-Hill, New York.

Schrage, L. (1975), "Implicit representation of variable upper bounds in linear programming", *Mathematical Programming Study* 4, 118–132.

Todd, M.J. (1982), "An implementation of the simplex method for linear programming problems with variable upper bounds", *Mathematical Programming* 23, 34–49.

Wierzbicki, A.P. (1982), "A mathematical basis for satisficing decision making", *Mathematical Modelling* 3, 391–405.

Wierzbicki, A.P. (1986), "On the completeness and constructiveness of parametric characterizations to vector optimization problems", *OR Spectrum* 8, 73–87.