European Journal of Operational Research 145 (2003) 509–529

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Discrete Optimization

Multiobjective solution of the uncapacitated plant location problem

Elena Fernández [a], Justo Puerto [b,*]

[a] Departament d'Estadística i Investigació Operativa, Universitat Politècnica de Catalunya, Spain
[b] Departamento de Estadística e Investigación Operativa, Facultad de Matemáticas, Universidad de Sevilla, 41012 Sevilla, Spain

Received 10 September 2000; accepted 12 April 2001

Abstract

In this paper we consider the discrete multiobjective uncapacitated plant location problem. We present an exact and an approximate approach to obtain the set of non-dominated solutions. The two approaches resort to dynamic programming to generate in an efficient way the non-dominated solution sets. The solution methods that solve the problems associated with the generated states are based on the decomposition of the problem on two nested subproblems. We define lower and upper bound sets that lead to elimination tests that have shown to have a high performance. Computational experiments on a set of test problems show the good performance of the proposal.
© 2002 Published by Elsevier Science B.V.

Keywords: Multiobjective combinatorial optimization; Discrete location problems; Dynamic programming

1. Introduction

The uncapacitated plant location problem (UPLP) is a classical discrete location problem that has been widely studied and for which efficient techniques to obtain solutions are well known. This problem consists of opening a set of plants among a potential set of locations to allocate a given set of customers in order to minimize the set-up cost of opening the plants plus the cost of allocating the clients. The unfamiliar reader is addressed to the chapter by Cornuejols et al. (1990) in the book by Mirchandani and Francis (1990) for further references.

Although many references exist in the literature, we are not aware of any that addresses the scenario analysis for UPLP. Scenario analysis is a solution approach that looks for robust solutions with respect to different sets of parameters describing alternative settings likely to occur (Kouvelis and Yu, 1997). In the case of UPLP different scenarios are given by different sets of both set-up costs and allocation costs. This methodology is very useful in real applications to describe seasonal behavior, to gather different managerial

* Corresponding author. Tel.: +34-954-557940; fax: +34-954-622800.
E-mail addresses: e.fernandez@upc.es (E. Fernández), puerto@us.es (J. Puerto).

strategies, to take into account varying costs, to handle uncertainty in parameter estimation, etc. Our model can be interpreted under optic of uncertainty. In this framework uncertainty is driven by the different location scenarios that may occur. We will assume that several decision-makers interact. Each of them has to evaluate different scenarios. In this situation, the proposed solution has to be a compromise between the involved decision-makers. To fulfill this requisite we propose Pareto solutions with regards to the criteria controlled by the decision-makers.

One possible way to perform scenario analysis is to consider the problem from a multiobjective point of view. This can be naturally done by representing each possible setting by means of one different criterion. In this context the solution concept is the set of non-dominated or Pareto solutions with respect to the considered criteria. These solutions have the desirable property of being acceptable for all the settings since they cannot be improved componentwise.

Multicriteria analysis of location problems has received considerable attention within the scope of continuous and network models in the last years. Presently, there are several problems that are accepted as classical ones: the point-objective problem (see e.g. Wendell and Hurter, 1973; Hansen et al., 1980; Pelegrín and Fernández, 1988; Carrizosa et al., 1993), the continuous multicriteria min-sum facility location problem (Hamacher and Nickel, 1996; Puerto and Fernández, 1999), and the network multicriteria median location problem (Hamacher et al., 1998; Wendell et al., 1977), among others.

On the contrary, multicriteria analysis of discrete Location Problems has attracted less attention so far. However, several authors have dealt with problems and applications of multicriteria decision analysis in this field. For instance, Ross and Soland (1980) treated multiactivity–multifacility problems and proposed an interactive solution method to compute non-dominated solutions to compare them and choose from. In Lee et al. (1981) an application of integer goal programming to facility location with multiple competing objectives is studied. Solanki (1991) applies an approximation scheme to generate the set of non-dominated solutions to a bi-objective location problem. Recently, Ogryczak (1999) looks for symmetrically efficient location patterns in a multicriteria discrete location problem. In general, none of the above papers, focuses in the complete determination of the whole set of non-dominated solutions. The only exception is the paper by Ross and Soland (1980) that give a theoretical characterization but do not exploit its algorithmic possibilities.

Nowadays, multiobjective combinatorial optimization (MOCO) (see Ehrgott and Gandibleux, 2000; Ulungu and Teghem, 1994) provides an adequate framework to tackle various types of discrete multicriteria problems as, for instance, UPLP. Within this emergent research area several methods are known to handle different problems. Two of them are dynamic programming enumeration (see Villarreal and Karwan, 1981, for a methodological description and Klamroth and Wiecek, 2000, for a recent application to knapsack problems) and implicit enumeration (Zionts and Wallenius, 1980; Zionts, 1979; Klein and Hannan, 1982; Rasmussen, 1986; Ramesh et al., 1986). Another approach based in labeling algorithms can be seen in Captivo et al. (2000).

It is worth noting that most of MOCO problems are NP-hard and intractable (see Ehrgott and Gandibleux, 2000, for further details). Even in most of the cases where the single-objective problem is polynomially solvable the multiobjective version becomes NP-hard. This is the case of spanning tree problems and min-cost flow problems, among others. In the case of UPLP, the single-objective version is already NP-hard (see Krarup and Pruzan, 1983). This ensures that the multiobjective formulation is not solvable in polynomial time. In this context, when time and efficiency become a real issue, different alternatives can be used to approximate the Pareto optimal set. One of them is the use of general-purpose MOCO heuristics (Gandibleux et al., 2000). Another possibility is the design of "ad hoc" methods based on one of the following strategies: (1) computing the supported non-dominated solutions; and (2) performing a partial enumeration of the solutions space. Obviously, this last strategy does not guarantee the non-dominated character of all the generated solutions since we only consider the solutions obtained during the partial search. Nevertheless the reduction in computation time can be remarkable.

The aim of this work is to develop two different methods to obtain the Pareto set for the multiobjective UPLP. The first one is an exact method that determines the whole set of efficient solutions. The second method is an "ad hoc" approximate method that generates the set of supported non-dominated solutions.

Our approach to solve the multicriteria UPLP takes advantage of the structure of the problem where solving the problem requires addressing two nested decisions. First, finding the optimal set of plants, and second, finding the allocation of clients within the selected set of plants. This structure is adequate for using a dynamic programming approach where the states are associated with the plant-opening phase. The load of this scheme relies on the enumeration of the potential sets of open plants as well as on the resolution of the associated allocation subproblems. Therefore, the improvements on such a method are based on (1) obtaining tight bounds that allow the elimination of states, and (2) the development of efficient techniques to solve the allocation subproblems. We have found two different bounds that lead to three elimination tests that have shown to have a high performance. Additionally, we present a labeling method to solve exactly the allocation subproblem as a shortest path problem; and a scalarized approach that finds the supported efficient set. The efficiency of the proposed methods has been tested on a battery of test problems and the obtained results are reported.

This paper is organized as follows. In Section 2 we give the notation and the formulation of the problem. Section 3 deals with the solution of the allocation subproblems. Section 4 presents the lower and upper bound sets as well as the elimination tests. Section 5 describes the different components of the dynamic programming algorithm. The results of the computational experiments are presented and analyzed in Section 6. This paper ends with some concluding remarks.

2. Model and notation

Let $M = \{1, \dots, m\}$ and $N = \{1, \dots, n\}$, respectively, denote the sets of indices for plants and for clients, and $Q = \{1, \dots, q\}$ denote the set of indices for the considered criteria. Also, for the rth criterion, $r \in Q$, let $(f_i^r)_{i \in M}$ denote the set-up costs and $(c_{ij}^r)_{i \in M, j \in N}$ the allocation costs of clients to plants.

The multicriteria uncapacitated plant location problem is:

$$P \quad v\text{-min} \quad \left\{ \sum_{i \in M} f_i^1 y_i + \sum_{i \in M} \sum_{j \in N} c_{ij}^1 x_{ij}, \dots, \sum_{i \in M} f_i^q y_i + \sum_{i \in M} \sum_{j \in N} c_{ij}^q x_{ij} \right\} \tag{1}$$

$$\text{s.t.} \quad \sum_{i \in M} x_{ij} = 1 \quad \text{for all } j \in N, \tag{2}$$

$$x_{ij} \leqslant y_i \quad \text{for all } i \in M, \ j \in N, \tag{3}$$

$$x_{ij}, y_i \in \{0, 1\} \quad \text{for all } i \in M, \ j \in N. \tag{4}$$

As it is usual, v-min stands for vector minimum of the considered objective functions. y_i takes the value 1 if plant i is open and 0 otherwise. The binary variable x_{ij} is 1 if client j is assigned to plant i and 0 otherwise. Constraints (2), together with integrality conditions on the x variables, ensure that each client is assigned to exactly one plant, while constraints (3) guarantee that no client is assigned to a non-open plant.

Recall that in the single criterion case the integrality conditions on the x variables need not be explicitly stated. The reason is that when the x_{ij} represent the proportion of demand of client j satisfied by plant i (i.e. $0 \leqslant x_{ij} \leqslant 1$), there exists an optimal solution with $x_{ij} = 0, 1$. This property is not necessarily true when multiple criteria are considered because, in general, there might be non-dominated solutions with non-integer values and even non-supported non-dominated integer solutions.

In what follows, the set of open plants associated with a given solution to P will be represented alternatively in one of the following ways:

- A binary vector $(y_i)_{i\in M}$ such that $y_i = 1 \iff$ plant i is open.
- A set of indices $I \subseteq M$ such that $i \in I \iff$ plant i is open.

Similarly, we will represent feasible allocations within a given set of plants I, alternatively in one of the following two ways:

- A binary vector $(x_{ij})_{i\in M, j\in N}$ such that $x_{ij} = 1 \iff$ client j has been assigned to plant $i \in I$.
- A mapping $a : N \to I,\ a(j) = i \iff$ client j has been assigned to plant $i \in I$.

Thus, a solution s will be represented either by a pair of binary vectors (y, x) or by a pair (I, a).

The cost of a solution $s = (I, a)$ relative to each of the considered criteria, is the sum of the fixed costs of the open plants plus the allocation cost. It will be denoted by $C^r(s) = F^r(s) + G^r(s),\ r \in Q$, where $F^r(s) = \sum_{i\in I} f_i^r$ is the cost of opening the plants and $G^r(s) = \sum_{j\in N} c^r_{a(j),j}$ is the cost of the allocation of clients.

Two nested decisions need to be addressed in order to solve problem P. First, the set of plants to be opened has to be selected. Then the allocation of clients within this set of open plants has to be identified. This allows to tackle the problem using strategies that first select a set of open plants and then solve an allocation subproblem associated with the set of open plants. In our approach, we will exploit this structure of the problem by using dynamic programming techniques to solve P. In particular, we propose a recurrence that is based on decomposing P into the plant selection subproblem (PS) and the allocation subproblem (A). The state variable is the set of open plants (I). At a given state, the set of decision variables are the y's for the PS subproblem and the x's for the allocation subproblem. Thus, using the standard notation in multicriteria dynamic programming, P can be also expressed as

$$P \quad H_{y,x} = v\text{-}\min_I \{PS_y(I) \oplus A_x(I)\}, \tag{5}$$

where $A \oplus C = \{a + c : a \in A, c \in C\}$. $PS_y(I)$ is the plant selection subproblem associated with the state I,

$$\begin{aligned} PS_y(I) \quad v\text{-}\min \quad & \left\{ \sum_{i\in M} f_i^1 y_i, \ldots, \sum_{i\in M} f_i^q y_i \right\} \\ & y_i = 1, \quad i \in I, \\ & y_i \in \{0,1\}, \quad i \in M \setminus I. \end{aligned} \tag{4$'$}$$

The only solution to $PS_y(I)$ non-dominated from below is immediate to obtain and is given by $y_i = 1,\ i \in I,\ y_i = 0,\ i \in M \setminus I$.

Similarly, the allocation subproblem $A_x(I)$ can be written as

$$\begin{aligned} A_x(I) \quad v\text{-}\min \quad & \left\{ \sum_{i\in I}\sum_{j\in N} c^1_{ij} x_{ij}, \ldots, \sum_{i\in I}\sum_{j\in N} c^q_{ij} x_{ij} \right\} \\ \text{s.t.} \quad & \sum_{i\in I} x_{ij} = 1 \quad \text{for all } j \in N, && (2) \\ & x_{ij} \in \{0,1\} \quad \text{for all } i \in I,\ j \in N. && (4'') \end{aligned}$$

Thus, in what follows we will assume that any feasible state is represented by its set of open plants. Therefore, at a given state solutions differ one from another only in the allocation of clients to plants within the set of open plants.

In the next section we describe solution procedures to solve the allocation subproblem.

3. The allocation subproblem

In the previous section we have seen that obtaining the set of non-dominated solutions to the plant selection subproblem is straightforward. Now we will deal with obtaining the set of non-dominated solutions to the allocation subproblem. Besides, we will also characterize the set of supported non-dominated solutions that we use (1) to obtain valid upper bound sets and (2) to solve approximately problem P. Recall that supported non-dominated solutions are those that can be obtained solving scalarized linear subproblems. Note that the supported solutions can be obtained by performing parametric analysis of a series of scalar UPLP. Therefore, the computational load required to obtain the set of supported non-dominated solutions is much lower than the one required to identify the complete Pareto set. We first study the general procedure to determine the whole Pareto set and then we will address the characterization of the supported non-dominated solution set.

In the single objective case the exact solution of the allocation subproblem can be obtained easily. This is a decisive difference with the case when several objectives are considered. In this case obtaining the set of non-dominated solutions is not a simple task. It is important to recall now that, in general, for discrete problems, this set does not coincide with the set of non-dominated supported solutions. It is easy to find examples to show that this is also true for the allocation subproblem. Therefore we have to resort to more sophisticated techniques for obtaining such set.

3.1. The Pareto set for the allocation subproblem

In this subsection, we give a procedure to obtain the whole set of efficient solutions of the problem $A_x(I)$ for a given state I. To this end, we need a previous result. We denote any feasible allocation x by $x = (x_{\cdot j})_{j\in N}$, where $x_{\cdot j} = (x_{ij})_{i\in I}$ is a feasible allocation for client j. Moreover, we explicitly write the allocation subproblem for client j that obviously is

$$\begin{array}{lll} A_x^j(I) & v\text{-min} & \left\{ \sum_{i\in I} c_{ij}^1 x_{ij}, \dots, \sum_{i\in I} c_{ij}^q x_{ij} \right\} \\ & \text{s.t.} & \sum_{i\in I} x_{ij} = 1, \\ & & x_{ij} \in \{0,1\} \quad \text{for all } i \in I. \end{array}$$

Then, we have that any efficient allocation of clients to plants must be composed by efficient allocations for each individual client.

Proposition 1. *For any state I, x^* is an efficient solution for the problem $A_x(I)$ if for each client $j \in N$, $x^*_{\cdot j}$ corresponds to an efficient allocation in the subproblem $A_x^j(I)$ of client j.*

Proof. Let x^* be an efficient solution for $A_x(I)$ given by the mapping $a : N \to I$. Thus,

$$x^*_{ij} = \begin{cases} 1 & \text{if } i = a(j), \\ 0 & \text{otherwise.} \end{cases}$$

Assume that for client j^0, $x^*_{\cdot j^0}$ is dominated. Therefore, there must exist $x^\#_{\cdot j^0}$ such that

$$\left(\sum_{i\in I} c^1_{ij^0} x^\#_{ij^0}, \dots, \sum_{i\in I} c^q_{ij^0} x^\#_{ij^0} \right) \underset{\neq}{\leqslant} \left(\sum_{i\in I} c^1_{ij^0} x^*_{ij^0}, \dots, \sum_{i\in I} c^q_{ij^0} x^*_{ij^0} \right).$$

Then, the solution $x^0 = (x^0_{\cdot j})_{j \in N}$ given by

$$x^0_{\cdot j} = \begin{cases} x^*_{\cdot j} & \text{if } j \neq j^0, \\ x^{\#}_{\cdot j} & \text{if } j = j^0 \end{cases}$$

dominates x^* which contradicts that x^* is an efficient solution. □

The first consequence of this result is that we can obtain the set of efficient allocations for $A_x(I)$ by means of the efficient allocations of each client. It is worth noting that the set of efficient allocations for a client is straightforward to obtain. (a) Evaluate the costs of all the allocations to the open plants in I, and (b) compare the corresponding vectors to eliminate the dominated ones. It is also straightforward that the converse of this result does not hold in general. Let us denote by L_j, $j \in N$, the lists of efficient allocations for the clients (given by their corresponding mappings).

The second consequence of Proposition 1 is that we can calculate the set of efficient solutions for the whole allocation subproblem by searching for the non-dominated minimal length paths in a particular graph.

Consider the graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. The set of vertices V is given by two vertices, O and D, plus a vertex a for each $a \in L_j$, $\forall j \in N$. The edges of this graph are defined as follows. There are edges from $O \overset{(0,\ldots,0)}{\longrightarrow} a$ $\forall a \in L_1$. Besides, there are edges $a \overset{(c^1_{a(j),j},\ldots,c^q_{a(j),j})}{\longrightarrow} a'$ $\forall a \in L_j$, $\forall a' \in L_{j+1}, \forall j = 1, \ldots, n-1$. Finally, there are also edges from $a \overset{(c^1_{a(n),n},\ldots,c^q_{a(n),n})}{\longrightarrow} D$, $\forall a \in L_n$. It is now obvious that the non-dominated minimum length paths in G are associated with efficient solutions of $A_x(I)$. Indeed, these paths are non-dominated and their lengths are the sum of the costs of the allocations of each client in N. Besides, each edge on a path corresponds with an efficient allocation of a client j. Thus, using the above proposition the result follows.

3.2. The set of supported non-dominated solutions to the allocation subproblem

It is well known that the set of supported non-dominated solutions to a problem can be obtained by solving the scalarized problem for all possible values of the scalar weights. In this subsection we obtain such set for the allocation subproblem. First, we restrict to the case of two objectives and at the end of the subsection we show how to extend the results to the general case.

When two criteria are considered, the λ-scalarized version $SA_x(I, \lambda)$ of the allocation subproblem $A_x(I)$ can be expressed as

$$SA_x(I,\lambda) \quad \min \left\{ \sum_{i \in I} \sum_{j \in N} \left[\lambda c^1_{ij} + (1-\lambda) c^2_{ij} \right] x_{ij} \right\} = \min \left\{ \sum_{i \in I} \sum_{j \in N} \left[c^2_{ij} + \lambda (c^1_{ij} - c^2_{ij}) \right] x_{ij} \right\}$$

$$\text{s.t.} \quad \sum_{i \in I} x_{ij} = 1 \quad \text{for all } j \in N, \tag{2}$$

$$x_{ij} \in \{0,1\} \quad \text{for all } i \in I,\ j \in N \tag{4''}$$

for $0 \leqslant \lambda \leqslant 1$.

In general, for any λ the corresponding scalarized allocation subproblem can be solved as the sum of independent subproblems, i.e.

$$SA_x(I,\lambda) = \sum_{j \in N} SA^j_x(I,\lambda) = \sum_{j \in N} \min \left\{ \sum_{i \in I} \left[c^2_{ij} + \lambda \left(c^1_{ij} - c^2_{ij} \right) x_{ij} \right] \right\}$$

$$\text{s.t.} \quad \sum_{i \in I} x_{ij} = 1,$$

$$x_{ij} \in \{0,1\} \quad \text{for all } i \in I.$$

Fig. 1 depicts, the lines $c^2_{ij} + \lambda(c^1_{ij} - c^2_{ij})$ $\forall i \in I$ for a fixed $j \in N$. Thus, the solutions of $SA^j_x(I,\lambda)$ for $j \in N$ fixed and $\lambda \in [0,1]$ can be obtained by identifying the lower envelope of the set of lines $\{c^2_{ij} + \lambda(c^1_{ij} - c^2_{ij}), i \in I\}$.

Thus, for a given λ the solution to $SA^j_x(I,\lambda)$ is given by

$$x_{ij} = \begin{cases} 1, & i = i(j), \\ 0 & \text{otherwise}, \end{cases} \quad \text{where } i(j) = \arg\min_{i \in I}\left\{\lambda c^1_{ij} + (1-\lambda)c^2_{ij}\right\}.$$

Once we know the solution to $SA^j_x(I,\bar{\lambda})$ for a fixed value $\bar{\lambda}$ it is easy to obtain the interval of values of λ for which the optimal solution does not change. In particular, for a state I the optimal allocation for a given client $j \in N$ and λ fixed can be characterized as follows:

Proposition 2. *For a fixed value of λ, $i^* \in I$ is the optimal allocation for client $j \in N \iff$*

$$\operatorname*{Max}_{c^1_{ij}-c^1_{i^*j} < c^2_{ij}-c^2_{i^*j}} \frac{c^2_{ij} - c^2_{i^*j}}{\left(c^2_{ij} - c^2_{i^*j}\right) - \left(c^1_{ij} - c^1_{i^*j}\right)} \leqslant \lambda \leqslant \operatorname*{Min}_{c^2_{ij}-c^2_{i^*j} < c^1_{ij}-c^1_{i^*j}} \frac{c^2_{ij} - c^2_{i^*j}}{\left(c^2_{ij} - c^2_{i^*j}\right) - \left(c^1_{ij} - c^1_{i^*j}\right)}.$$

Proof. For a fixed value of λ, $i^* \in I$ is the optimal allocation for a given client $\iff c^2_{i^*j} + \lambda(c^1_{i^*j} - c^2_{i^*j}) \leqslant c^2_{ij} + \lambda(c^1_{ij} - c^2_{ij})$ $\forall i \in I$ which is equivalent to the stated condition. □

Proposition 3. *Let $i_1 \in I$ be the optimal allocation for a given client $j \in N$ and some $\lambda = \lambda_1 \geqslant 0$ fixed.*
(a) *If $i_2 \in I$ is the optimal allocation for client j for some $\lambda = \lambda_2$, $\lambda_2 > \lambda_1$ then $c^1_{i_2} - c^2_{i_2} \leqslant c^1_{i_1} - c^2_{i_1}$.*
(b) *If $i_2 \in I$ is the optimal allocation for client j for some $\lambda = \lambda_2$, $\lambda_2 < \lambda_1$ then $c^1_{i_2} - c^2_{i_2} \geqslant c^1_{i_1} - c^2_{i_1}$.*

Proof.
(a) If $c^1_{i_2} - c^2_{i_2} > c^1_{i_1} - c^2_{i_1}$ then

$$\begin{aligned} c^2_{i_2} + \lambda_2(c^1_{i_2} - c^2_{i_2}) &= c^2_{i_2} + \lambda_1(c^1_{i_2} - c^2_{i_2}) + (\lambda_2 - \lambda_1)(c^1_{i_2} - c^2_{i_2}) \\ &\geqslant c^2_{i_1} + \lambda_1(c^1_{i_1} - c^2_{i_1}) + (\lambda_2 - \lambda_1)(c^1_{i_2} - c^2_{i_2}) \\ &> c^2_{i_1} + \lambda_1(c^1_{i_1} - c^2_{i_1}) + (\lambda_2 - \lambda_1)(c^1_{i_1} - c^2_{i_1}) = c^2_{i_1} + \lambda_2(c^1_{i_1} - c^2_{i_1}). \end{aligned}$$

(b) It is similar to (a). □

Corollary 1. *Let $i^* \in I$ be the optimal allocation for a given client $j \in N$ and some $\lambda^* \geqslant 0$ fixed. Then $i^* \in I$ is the optimal allocation for client $j \in N$ for any $\lambda \in [\underline{\lambda}, \bar{\lambda}]$ where*

$$\underline{\lambda} = \operatorname{Max}\left\{ \operatorname*{Max}_{c^1_{ij}-c^2_{ij} > c^1_{i^*j}-c^2_{i^*j}} \left(\frac{c^2_{ij} - c^2_{i^*j}}{\left(c^1_{i^*j} - c^2_{i^*j}\right) - \left(c^1_{ij} - c^2_{ij}\right)}, 0 \right) \right\}$$

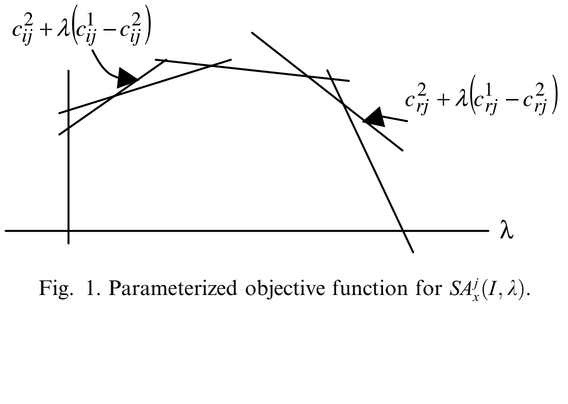
$c^2_{ij} + \lambda(c^1_{ij} - c^2_{ij})$
$c^2_{rj} + \lambda(c^1_{rj} - c^2_{rj})$
λ

Fig. 1. Parameterized objective function for $SA^j_x(I,\lambda)$.

and

$$\overline{\lambda} = \mathrm{Min}\left\{ \mathop{\mathrm{Min}}_{c^1_{ij}-c^2_{ij} < c^1_{i^*j}-c^2_{i^*j}} \left(\frac{c^2_{ij} - c^2_{i^*j}}{(c^1_{i^*j} - c^2_{i^*j}) - (c^1_{ij} - c^2_{ij})}, 1 \right) \right\}.$$

The above result allows establishing for each client $j \in N$, a partition of the λ space in intervals where all the elements of the same interval are associated with the same supported solution to $SA^j_x(I, \lambda)$.

By Proposition 1 the overall solution to $SA_x(I, \lambda)$ can be obtained by concatenation of the solutions to the problems $SA^j_x(I, \lambda)$ for all $j \in N$. Again, this produces the partition of the λ space in intervals where all the elements of the same interval are associated with the same supported solution to the overall allocation subproblem $SA_x(I, \lambda)$.

In passing, we note that the above approach generates specifically the whole set of extreme Pareto solutions for the allocation subproblem that results if integrality conditions on the x variables are not required. In that case the solutions of the corresponding allocation subproblems for the different clients are the same than when the x are binary variables.

The extension of the above procedure to the case of more than two objective functions is direct. The only change is that an alternative way to derive the partition on the λ-space is required to obtain the supported Pareto solutions of the problem. The difference is that now the partition of the λ-space is not given by intervals but it is defined by systems of inequalities. Indeed, for a parameter $\lambda = (\lambda_1, \ldots, \lambda_q)$, i^* is the optimal allocation for client j if and only if $\sum_{r=1}^q \lambda_r c^r_{i^*j} \leqslant \sum_{r=1}^q \lambda_r c^r_{ij}$ $\forall i \in I$. Therefore, the region of the λ-space for which i^* is the optimal allocation for client j is given by the set of inequalities:

$$\sum_{r=1}^{q} \lambda_r (c^r_{i^*j} - c^r_{ij}) \leqslant 0 \quad \forall i \in I.$$

These regions can be identified using parametric linear programming (see Gal, 1984).

Alternatively, one can find directly the non-dominated supported solutions of $A_x(I)$ using a general purpose algorithm. Each of the supported non-dominated solutions is associated with an extreme non-dominated solution of the multiobjective linear problem obtained from the continuous relaxation of $A_x(I)$. The algorithm by Isermann (1977) provides the complete set of solutions of these problems and the software package ADBASE by Steuer (1995) can be used in computer implementations to solve instances (small to medium size).

Example. Consider the following example with 5 potential plants, 3 clients and 2 objectives. C^1 and C^2 represent the allocation costs for each objective and the rows of F are the opening cost for each objective:

$$C^1 = \begin{pmatrix} 20 & 30 & 10 & 20 & 40 \\ 50 & 10 & 60 & 10 & 80 \\ 30 & 30 & 20 & 10 & 20 \end{pmatrix}, \quad C^2 = \begin{pmatrix} 40 & 20 & 20 & 10 & 40 \\ 10 & 50 & 20 & 20 & 30 \\ 30 & 10 & 10 & 30 & 20 \end{pmatrix}, \quad f = \begin{pmatrix} 7 & 3 & 5 & 8 & 2 \\ 2 & 6 & 1 & 3 & 4 \end{pmatrix}.$$

If $I = \{1, 2\}$ the supported non-dominated solutions to the allocation subproblem $SA^j_x(I, \lambda)$ are the following:

$$j = 1, \quad a(1) = \begin{cases} 2, & 0 \leqslant \lambda \leqslant 2/3, \\ 1, & 2/3 \leqslant \lambda \leqslant 1, \end{cases}$$

$$j = 2, \quad a(2) = \begin{cases} 1, & 0 \leqslant \lambda \leqslant 1/2, \\ 2, & 1/2 \leqslant \lambda \leqslant 1, \end{cases}$$

$$j = 3, \quad a(3) = 2, \quad \lambda \in [0, 1].$$

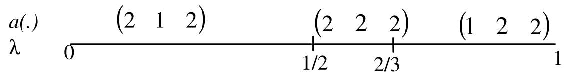

Fig. 2. Supported non-dominated solutions to the allocation subproblem.

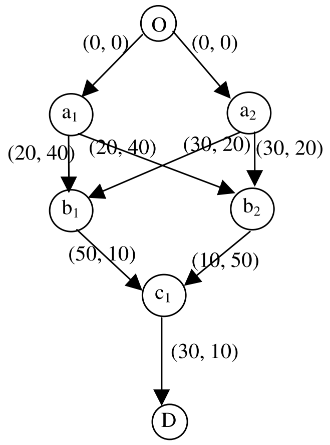

Fig. 3. Network for finding the non-dominated solutions to the allocation subproblem.

Thus, the supported non-dominated solutions to the allocation subproblem are depicted in Fig. 2

To obtain all the non-dominated solutions to the allocation subproblem we consider the network of Fig. 3 and we find the non-dominated minimum length paths in the network which are

O–a2–b1–c1–D with value $(110, 40)$.
O–a1–b1–c1–D with value $(100, 60)$.
O–a2–b2–c1–D with value $(70, 80)$.
O–a1–b2–c1–D with value $(60, 100)$.

4. Lower and upper bounds

The dynamic programming approach that we propose in (5) can be enhanced using bounds that allow the implicit enumeration of some of the states in the formulation. The bounds will be used in the usual way. That is, a state can be fathomed if all the elements in the lower bound set are dominated by at least one element of the upper bound set.

In this section we obtain lower bounds for the different states as well as upper bounds for the overall problem. Two different types of lower bounds will be considered. The first one is only valid for the state where it is generated while the second one is valid for some successors of the state where it is derived. The two of them can be used for comparison with the set of solutions non-dominated from below of a given state to eliminate some of its potential successors. Additionally, the second lower bound can be used for comparison with an upper bound set of the original problem P, to eliminate the current state as well as all its successors.

We first establish some simple relationship between different states that will be useful. Since the goal of any enumerative scheme is to explore as few states as possible, we focus on identifying those states for which ‘‘a priori’’ we know that any solution will be dominated from below by some solution of a different state.

Let I and I' be two states. If $I \supset I'$, I is called a *successor* of I' and I' is called a *predecessor* of I. When $I = I' \cup \{i^*\}$ for some $i^* \in M \setminus I'$, I is called an *immediate successor* of I', and I' is called an *immediate predecessor* of I. An immediate successor of I', I, is *worse* than I' if any solution of I is dominated from below by (or has the same value than) some solution of I'. Also, let $S(I)$ denote the set of all the efficient solutions of I, and $V(I) = PS_y(I) \oplus A_x(I) = \{(C^1(s), \dots, C^q(s)) : s \in S(I)\}$ denote the set of their corresponding non-dominated from below values. By definition I is worse than I' if and only if $V(I')$ is a lower bound set of $V(I)$.

A necessary and sufficient condition to eliminate a given state is stated in the next result.

Theorem 1. *Let I be an immediate successor of I'. I is worse than $I' \iff$ for all $N' \subseteq N$ there exists $a : N' \to I'$, such that $\sum_{j\in N'} c^r_{a(j),j} \leqslant f^r_{i^*} + \sum_{j\in N'} c^r_{i^*,j}$ $\forall r \in Q$.*

Proof. ($\Rightarrow$) Suppose there exist $N' \subseteq N$ and $r \in Q$ such that $\sum_{j\in N'} c^r_{a(j),j} > f^r_{i^*} + \sum_{j\in N'} c^r_{i^*,j}$ $\forall a : N' \to I'$. Let $s' = (I, a') \in S(I')$. The allocation $a^* : N \to I$, defined by

$$a^*(j) = \begin{cases} i^*, & j \in N', \\ a'(j), & j \notin N' \end{cases}$$

defines a feasible solution of I, $s = (I, a^*)$ that is not dominated from below by any solution of I'.

It is easy to check that $\forall r \in Q, C^r(s) < C^r(s')$. Indeed,

$$\begin{aligned} C^r(s) &= F^r(s) + G^r(s) = F^r(s') + f^r_{i^*} + \sum_{j\in N} c^r_{a^*(j),j} \\ &= F^r(s') + f^r_{i^*} + \sum_{j\in N'} c^r_{i^*,j} + \sum_{j\notin N'} c^r_{a'(j),j} < F^r(s') + \sum_{j\in N'} c^r_{a'(j),j} + \sum_{j\notin N'} c^r_{a'(j),j} = C^r(s'). \end{aligned}$$

($\Leftarrow$) Let $s = (I, a) \in S(I)$. Consider the set $N' = \{j \in N : a(j) = i^*\}$. By hypothesis, there exists $\hat{a} : N' \to I'$, such that

$$\sum_{j\in N'} c^r_{\hat{a}(j),j} \leqslant f^r_{i^*} + \sum_{j\in N'} c^r_{i^*,j} \quad \forall r \in Q.$$

The allocation $a' : N \to I'$, given by

$$a'(j) = \begin{cases} a(j), & j \in N', \\ \hat{a}(j), & j \notin N' \end{cases}$$

defines a feasible solution $s' = (I', a')$ of $S(I')$ that dominates s from below.

Now, $\forall r \in Q$ we can check that $C^r(s') < C^r(s)$. Indeed,

$$\begin{aligned} C^r(s') &= F^r(s') + G^r(s') = F^r(s') + \sum_{j\in N'} c^r_{a'(j),j} + \sum_{j\notin N'} c^r_{a'(j),j} \\ &\leqslant F^r(s') + f^r_{i^*} + \sum_{j\in N'} c^r_{a(j),j} + \sum_{j\notin N'} c^r_{a(j),j} \\ &= F^r(s) + \sum_{j\in N} c^r_{a(j),j} = C^r(s). \qquad \square \end{aligned}$$

Remark. The characterization of Theorem 1 says that for each objective, the cost of any solution that allocates any subset of clients to the new open plant i^*, can be improved by closing plant i^* and reallocating the clients within the set of plants I'.

Although the above result characterizes when a state is worse than its immediate predecessor, in practice it is very difficult to check because it requires the enumeration of all possible subsets $N' \subseteq N$. A sufficient condition which is easier to apply is now stated.

Proposition 4. *Let I be an immediate successor of I' $(I = I' \cup \{i^*\})$ and $I^\#$ a successor of I' such that $i^* \in I^\#$ (possibly I). If I is worse than I' then $\forall s \in S(I^\#)$, s is dominated from below by some solution of a state where plant i^* is not open.*

Proof. Since $I^\#$ is a successor of I' such that $i^* \in I^\#$, then either $I^\# = I$ or $I^\#$ is a successor of I. If $I^\# = I$ the proposition just states that I is worse than I'. Thus, suppose $I^\#$ is a successor of I, and let $s = (I^\#, a^\#) \in S(I)^\#$.

Define $N' = \{j \in N : a^\#(j) = i^*\}$. Since I is worse than I', there exists $a : N' \to I'$, such that $\forall r \in Q$

$$\sum_{j \in N'} c^r_{a(j),j} \leqslant f^r_{i^*} + \sum_{j \in N'} c^r_{i^*,j}.$$

Thus, the solution $s^* = (I^\# \setminus \{i^*\}, a^*)$ given by,

$$a^*(j) = \begin{cases} a(j), & j \in N', \\ a^\#(j), & j \notin N' \end{cases}$$

is a feasible solution to the immediate predecessor of $I^\#$ whose set of open plants is $I^\# \setminus \{i^*\}$ that dominates from below s. □

Remark. The above result states that if I is worse than I', then no successor of I' where plant i^* is open can provide solutions non-dominated from below to problem P. Therefore, such successors need not be explored (Elimination test 1).

Suppose that we know the set of solutions non-dominated from below of a given state I'. In general, if we want to know if one of its immediate successors is worse than I', we will have to: (a) solve the allocation subproblem of the successor, (b) find its solutions non-dominated from below, and (c) compare the two sets. We next present a lower bound of the set of solutions non-dominated from below of an immediate successor of I'. This lower bound set can be easily obtained and, in some cases, it will permit to establish that an immediate successor of I' is worse than I' without having to solve the associated allocation subproblem.

Proposition 5. *Let I be an immediate successor of I' $(I = I' \cup \{i^*\})$. For each $s = (I', a') \in S(I')$, define*

$$\Delta^r(s) = \sum_{j \in N} \min\left\{c^r_{a'(j),j} - c^r_{i^*,j}, 0\right\} - f^r_{i^*} \quad \forall r \in Q.$$

The set $L^1(I) = \{(C^1(s) - \Delta^1(s), \ldots, C^q(s) - \Delta^q(s)) : s \in S(I')\}$ is a lower bound for I.

Proof. The proof is immediate since $L^1(I)$ contains the values of the ideal allocations for each state $s \in S(I')$. □

Some elements can possibly be eliminated from $L^1(I)$. Those are the elements dominated from above by other elements of the set. We can assume that all such elements have been eliminated.

Note that the lower bound $L^1(I)$ is only valid for state I but not necessarily for any of its successors. We now propose another lower bound that is valid not only for the state where it is generated but also for some of its successors.

Proposition 6. *Let I be an immediate successor of a given state I' $(I = I' \cup \{i^*\})$, $\hat{I} \subseteq M \setminus I'$ be such that $i^* \in \hat{I}$, and $S(I' \cup \hat{I})$ be the set of efficient solutions for the allocation subproblem defined over the set of plants $I' \cup \hat{I}$. Then the set*

$$L^2(I) = \left\{ \left(\sum_{i \in I'} f_i^1 + f_{i*}^1 + G^1(s), \ldots, \sum_{i \in I'} f_i^q + f_{i*}^q + G^q(s) \right) : s \in S\left(I' \cup \hat{I}\right) \right\} = PS_y(I) \oplus A_x\left(I' \cup \hat{I}\right)$$

is a lower bound for I and for all the successors of I whose set of open plants is contained in $I' \cup \hat{I}$.

Proof. For any solution s' with set of open plants $I^\# \subseteq \hat{I} \cup I'$ there must exist $s^* \in S(I' \cup \hat{I})$ such that $G^r(s^*) \leqslant G^r(s')$ $\forall r \in Q$ because all the plants open in the solution s' can be used in the solution s^* as well.

On the other hand, $\sum_{i \in I'} f_i^r + f_{i^*}^r \leqslant \sum_{i \in \hat{I} \cup I'} f_i^r$ $\forall r \in Q$. Hence, by joining the two inequalities the result follows. □

Like in the case of $L^1(I)$, we will assume that the elements of $L^2(I)$ dominated from above by other elements of the set have been eliminated.

Example (*continued*). Consider $I' = \{3,4\}$ and $i^* = \{2\}$.

For any of the two objectives the potential savings of allocating any client to plant 2 are smaller than the cost of opening plant 2. Thus, the characterization of Theorem 1 applies and $I = \{3,4,2\}$ is worse than I'.

Consider, for instance, the solution $s = (I, a)$, $I = \{3,4,2\}$, $a = (3,4,2)$,

$C^1(s) = F^1(s) + G^1(s) = 16 + 50 = 66$,

$C^2(s) = F^2(s) + G^2(s) = 10 + 50 = 60$.

s is dominated by the solution $s' = (I', a')$, $I' = \{3,4\}$ $a' = (3,4,3)$ with value

$C^1(s') = F^1(s') + G^1(s') = 13 + 40 = 53$,

$C^2(s') = F^2(s') + G^2(s') = 4 + 50 = 54$.

To obtain $L^1(I)$ we first obtain $S(I') = \{s_1, s_2, s_3\}$, $s_1 = (I', a_1)$, $a_1 = (3,4,3)$ with value $(53,54)$; $s_2 = (I', a_2)$, $a_2 = (3,4,4)$ with value $(43,74)$; $s_3 = (I', a_3), a_3 = (4,4,3)$ with value $(63,44)$. Then,

$$\Delta^1(s_1) = 0 - 3 = -3; \quad \Delta^2(s_1) = 0 - 6 = -6,$$

$$\Delta^1(s_2) = 0 - 3 = -3; \quad \Delta^2(s_2) = 20 - 6 = 14,$$

$$\Delta^1(s_3) = 0 - 3 = -3; \quad \Delta^2(s_3) = 0 - 6 = -6,$$

$$L^1(I) = \{(53,54) - (-3,-6), (43,74) - (-3,14), (63,44) - (-3,-6)\} = \{(56,60), (46,60), (66,50)\}.$$

In order to apply Proposition 6 consider $\hat{I} = \{1,2\} \subseteq M \setminus I' = \{1,2,5\}$.

Now $I' \cup \hat{I} = \{1,2,3,4\}$ and $S(I' \cup \hat{I}) = \{s_1, s_2, s_3, s_4\}$ with $s_4 = (\{1,3,4\}, a_4)$, $a_4 = (4,1,3)$ and value $(90,30)$.

Then $L^2(I) = \{(13,4) + (3,6)\} \oplus \{\{90,30), (50,40), (40,50), (30,70)\}$ is a valid lower bound for the sets of plants $\{2,3,4\}$ and $\{1,2,3,4\}$.

Although the bound $L^2(I)$ may be obtained with any subset $\hat{I} \subseteq M \setminus I'$, in practice it is convenient to choose a generation policy for this kind of bound. The simplest one is to take $\hat{I} = \{i^*\}$. Then the bound is exact: it coincides with the set of non-dominated solutions of the state where it is calculated. But then the bound is only valid for the state where it is calculated. The larger the set $\hat{I}$, the larger is the number of states for which the bound is valid. But also, the larger the set $\hat{I}$, less accurate is the bound. Therefore, the strategy that maximizes the number of descendants of a state for which L^2 is valid consists of taking $\hat{I} = M \setminus I'$. However, this leads to no change in $A_x(I' \cup \hat{I})$ relative to node I' and only the opening costs could make the lower bound set improve. If we want to increase the possibility of improving the lower bound of state I', $M \setminus I'$ has to be reduced in at least one element. In that case, the obtained bound will be different from that of I' when some plant that is closed in I' is used in some non-dominated from below solution of $A_x(I' \cup \hat{I})$.

Corollary 2. *Let L(I) be the set that results from eliminating from $L^1(I) \cup L^2(I)$ all the elements dominated from above by other elements of the set. L(I) is a lower bound for state I.*

Corollary 3. *Let I be an immediate successor of a given state I'. If $\forall s \in L^1(I), \exists s' \in S(I')$ that dominates s from below, then I is worse than I' (Elimination test 2).*

Corollary 4. *Let U be an upper bound set of the original problem P. If $\forall s' \in L^2(I'), \exists s \in U$ that dominates s' from below, then I' (and all the successors for which $L^2(I')$ is valid) can be eliminated (Elimination test 3).*

We finish this section devoted to bounds describing a set of upper bounds for problem P. The first obvious upper bound for the problem is given by the non-dominated solutions of the allocation subproblem when the whole set of plants is considered open. Notice that this set of efficient solutions can be computed easily just evaluating the different allocations and eliminating those that are dominated. A more refined bound can be obtained by solving scalarized plant location problems with different scalar factors. These problems can be solved either exactly or approximately when the size makes the exact resolution not possible. The use of approximated solutions gives larger upper bounds but still valid for our elimination purposes. In any case, the set of upper bounds is enlarged dynamically at each state where efficient solutions are obtained. In our approach we have used the Erlenkotter heuristic (Erlenkotter, 1978) for solving the scalarized problems. First, the scalarizing factor is set to 0 and then it is sequentially increased by steps of 0.1.

5. Enumerative scheme

Two different strategies have been used in order to solve problem P. With the first strategy the problem is solved exactly and, hence, the whole set of Pareto solutions is obtained. As will be seen in the computational results section this strategy is costly. For this reason the second strategy only looks for the set of supported Pareto solutions. Thus it might be considered as an approximation to the actual solution set. Both strategies are based on the same enumerative scheme and state generation mechanism. In this section we describe both strategies as well as the policy used to carry out the search in the state space.

5.1. Solution strategies

Given that the search is based on enumerating the different sets of plants, and that the set of open plants is fixed at each state, the difference between solving a problem exactly or approximately reduces to the solution of the allocation subproblems. In order to solve the problem exactly, a labeling algorithm is used to solve the shortest path approach described in Section 3.1. When problem is solved approximately, the set of supported Pareto solutions of the allocation subproblem is obtained solving one scalarized problem for each set in the partition of the λ-space as described in Section 3.2.

5.2. State exploration

The search is performed by stages and each stage contains a collection of states. The states at stage k, correspond with all possible combinations of open plants with exactly k open plants. Thus, a state I at stage k has exactly k immediate predecessors at stage $k - 1$. These are associated with all the subsets of I with

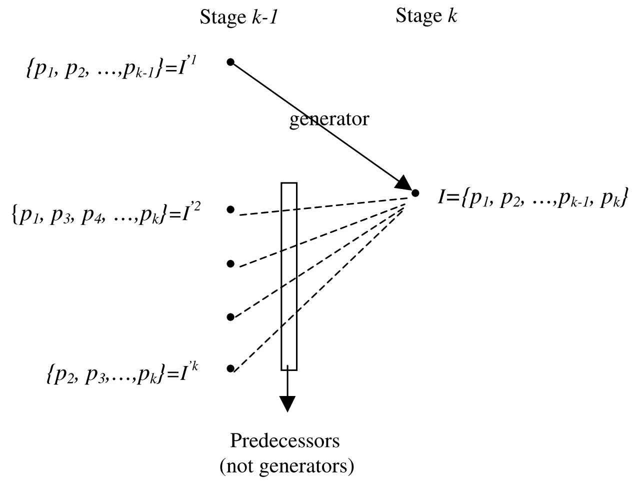

Fig. 4. State generation strategy.

cardinality $k-1$. The (unique) state of stage $k-1$ from which I is generated is called the *generator* of I (see Fig. 4). Similarly, state I at stage k is an immediate predecessor of exactly $m-k(m=|M|)$ states at stage $k+1$. They are the states whose set of open plants is $I\cup\{i\}$ for some $i\in M\setminus I$. Yet, I is the generator of only some of its immediate successors that will be called *descendants* of I.

The search consists basically of two types of activities that inter-relate one with the other. One type of action consists in exploring the different states while the other type of action consists in generating the successors of a given state. This second activity is the final step of the state exploration action. The search explores all the states of a stage before exploring any state of any subsequent stage.

Let U denote an upper bound set for P and let I' at stage $k-1$ denote the generator of state I at stage k. When state I is selected to be explored we proceed as follows:

1. Apply Elimination test 2: If $L(I')$ fulfills the conditions of Corollary 3, then eliminate I.
2. Apply Elimination test 3: If $L^2(I')$ fulfills the conditions of Corollary 4, then eliminate I'.
3. If I' has not been eliminated, solve the allocation subproblem associated with I to find the set of non-dominated solutions from below $S(I)$ and to obtain $V(I)$.
4. Apply Elimination test 1 (Proposition 4): If I is worse than I' then eliminate I.
5. Otherwise, generate the descendants of I.

5.3. State generation mechanism

Without loss of generality we suppose that the indices of plants have been relabeled so that the new indices correspond to the preference order for opening the plants.

For notational convenience, we assume that the indices of open plants at state I of stage k, are ordered by increasing values. That is, $I=\{p_1,p_2,\dots,p_{k-1},p_k\}$ where $p_1<p_2<\cdots<p_{k-1}<p_k$. The states for which state I' at stage $k-1$ is the generator are the immediate successors of I' whose additional open plant has an index greater than p_{k-1}. In this way, although state I at stage k with $I=\{p_1,p_2,\dots,p_{k-1},p_k\}$ has k immediate predecessors, I is always generated from its generator, namely, state I'^1 of stage $k-1$ with $I'^1=\{p_1,p_2,\dots,p_{k-1}\}$. However, I also has as immediate predecessors the states of stage $k-1$ with the following sets of open plants: $I'^2=\{p_1,p_3,p_4,\dots,p_k\}$, $I'^3=\{p_1,p_2,p_4,\dots,p_k\}$, $I'^4=\{p_1,p_2,p_3,p_5,\dots,p_k\},\dots,I'^{k-1}=\{p_1,p_2,\dots,p_{k-2},p_k\}$ and $I'^k=\{p_2,p_3,\dots,p_k\}$. In what follows, the immediate predeces-

sors of I will be denoted by $\{I'^r\}_{r=1,\dots,k}$. Without loss of generality we will assume that the generator of I is I'^1.

The descendants of state $I' = \{p_1, p_2, \dots, p_{k-1}, p_k\}$ at stage k with are its $m - p_k$ immediate successors $I^r = \{p_1, p_2, \dots, p_{k-1}, p_k, p^r_{k+1}\}$, where $p^r_{k+1} > p_k, r = 1, \dots, m - p_k$.

Let $I' = \{p_1, p_2, \dots, p_{k-1}, p_k\}$ denote a state at stage k whose descendants we want to generate and let $I = \{p_1, p_2, \dots, p_{k-1}, p_k, p_{k+1}\}$, $p_{k+1} > p_k$ denote one of its possible descendants. I is generated only if the following conditions hold:

1. None of the immediate predecessors of I, I'^r, has been eliminated (Elimination test 1 or Elimination test 3). If I'^r was eliminated because it was worse than some predecessor we can apply Elimination test 1. If I'^r was eliminated by Elimination test 3, Corollary 4 also applies to I since it is a successor of I'^r.
2. Elimination test 2 does not apply to I for any of its immediate predecessors. That is, Corollary 3 does not apply with I and $I' = I'^r$, for $r = 1, \dots, k + 1$.

5.4. Plant selection criterion

Progressing from a stage to the following one, requires the opening of a new plant. The efficiency of the proposed scheme relies on the criterion used to select the new open plant. We have used a selection criterion that is established beforehand and is fixed during all the exploration.

When calculating the upper bound set, the set of supported non-dominated solutions to problem P has been obtained. Then the frequency of each open plant within this set of solutions is recorded. The criterion to select the new open plant is by decreasing ordering of these frequencies. Let $\{o_1, o_2, \dots, o_m\}$ denote the indices of plants ordered according to these decreasing frequencies.

Recall that in order to obtain the lower bound $L^2(I)$, an allocation subproblem has to be solved. To reduce the number of such subproblems to be solved, it is desirable that $L^2(I)$ be valid for as many descendants as possible. On the other hand, we know that $L^2(I)$ is only valid for those descendants where certain plants are not open (see Section 4 for details on the computation of these bounds). For selecting the set $\hat{I}$ to obtain the lower bound $L^2(I)$, the criterion that chooses plants increasingly from $\{o_1, o_2, \dots, o_m\}$, permits one to know in advance which plants will always be closed in the descendants of a given state. Therefore, it favors applying efficiently Elimination test 3. In particular, if $i^*(I) = \max\{s \mid o_s \in I\}$ denotes the index of the only plant open in state I that was not open in its generator, then $\mathrm{cl}(I) = \{o_s \mid s > i^*(I)\}$ represents the set of plant indices that are currently closed but could be open in the descendants of I. Thus, taking $\hat{I} = i^*(I) \cup \mathrm{cl}(I)$ all the descendants of the current state will have a set of open plants contained in $I \cup \hat{I}$ and the generated bound is $L^2(I) = PS_y(I) \oplus A_x(I \cup \hat{I})$.

In practice, we even simplify the computation of these bounds reducing the number of allocation subproblems that need to be solved. Thus, instead of calculating $L^2(I)$ as explained, we use the bound set $PS_y(I) \oplus [\bigcap_{s:o_s \in \mathrm{cl}(I)} A_x(I^s)]$ where $I^s = M \setminus \{o_s\}$ for each s so that $o_s \in \mathrm{cl}(I)$. Note that on the whole there are m possible sets I^s, $s = 1, \dots, m$, that take part in all possible intersections. Therefore, for $s = 1, \dots, m$, $L^2(I^s)$ is calculated at the beginning of the process and then is used when needed during the exploration phase.

In passing, we note that $PS_y(I) \oplus [\bigcap_{s:o_s \in \mathrm{cl}(I)} A_x(I^s)] \supset L^2(I)$. Therefore, the lower bound set $PS_y(I) \oplus [\bigcap_{s:o_s \in \mathrm{cl}(I)} A_x(I^s)]$ is valid for the state I although it is larger than the original $L^2(I)$.

6. Computational experience

A series of computational experiments have been performed in order to evaluate the behavior of the proposed solution method. As has been shown in the previous sections, from the methodological point of view there is no difference dealing with two or more criteria. However, it is well-known that the computational load to solve n objective problems increases exponentially with n (Ehrgott and Gandibleux, 2000). For this reason in this section we have restricted to bicriteria problems.

Programs have been coded in FORTRAN-90 and executed in a PC with an Intel II processor at 233 MHz and 64 MB of RAM. Since this problem has not been previously addressed no available benchmark instances exist. Thus, our data generation strategy consists of combining pairs of single criterion UPLP. In the well-known Beasley's Library (htttp://mscmga.ms.ic.ac.uk/info.html) UPLP instances of the same size differ one from the other only in the opening costs and have the same allocation costs. In our context we consider that it is more convenient that both the opening and the allocation costs differ in the considered pairs of instances. For this reason we have used a different set of data which is also available in http://www-eio.upc.es/~elena/sscplp/index.html and has been used previously in the literature (see e.g., Delmaire et al., 1999; Hindi and Pienkosz, 1999).

Problems are divided into four groups of dimensions $10 \times 20, 15 \times 30, 20 \times 40$ and 20×50 (plants $\times$ clients). Each of these groups contains 15, 45, 28 and 28 problems, respectively. In fact, we have considered two different sets of problems, each of them composed of the mentioned groups and number of instances. A first battery that corresponds exactly to the referenced original data and a second battery where the opening costs of the first battery have been modified. In the original data, the opening costs are large as compared to the allocation costs (see Delmaire et al., 1999, for a description of the problem generator): the opening costs range in $[1000, 2000]$ while the allocation costs range in $[0, 100]$. In the second battery, the opening costs have been divided by 10 (and thus range in $[100, 200]$) while the allocation costs remain unchanged. In this section instances of the first battery will be referred to as "large-cost" and instances of the second battery will be referred to as "small-cost".

For each instance we have solved the problem twice: exactly and approximately. As has been mentioned, the approximate method gives the whole set of extreme solutions to the model that results when the x variables are allowed to take continuous values. Thus, our results also permit to compare the difficulty of the exact solution of the two models with our dynamic programming approach.

Tables 1 and 2 contain a summary of the results obtained for the two sets of problems. Columns *undom.* give the average number of non-dominated solutions of the problems; *states* give the average number of states generated in the exploration process; *stages* are the average of the maximum stage reached (deepest search level); *test*2, *test*3 and *test*1 show the average number of states eliminated by test 2, test 3 and test 1, respectively; finally, *solved* are the average number of states where the allocation problem had to be solved.

In general terms, the small-cost instances were harder to solve than the large-cost instances. Roughly speaking this is due to the fact that the contribution of the set-up costs to the overall cost of solutions is considerably smaller for small-cost than for large-cost instances. Thus, the accuracy of the lower bound sets is better for large-cost instances. The reason is that set-up cost of solutions are always evaluated exactly.

Table 1
Summary of computational experiments for small-cost instances

Small cost		Undom.	States	Stages	Test2	Test3	Test1	Solved
10×20	Exact	15.400	55.533	3.733	1.867	8.733	109.067	164.600
	Approx.	10.333	57.067	3.733	0.000	6.333	115.200	172.267
15×30	Exact	17.867	131.178	4.000	0.089	24.800	434.178	565.356
	Approx.	11.244	189.422	4.000	71.844	24.578	423.178	612.600
20×40	Exact	30.214	298.357	4.536	0.000	80.250	1251.786	1550.143
	Approx.	15.321	211.250	3.893	0.000	14.929	1128.321	1339.571
20×50	Exact	38.160	514.280	4.880	0.680	115.080	1249.760	1764.040
	Approx.	13.160	203.400	3.480	0.000	11.040	1085.920	1289.320

Table 2
Summary of computational experiments for large-cost instances

Large cost		Undom.	States	Stages	Test2	Test3	Test1	Solved
10×20	Exact	18.467	32.400	3.533	55.067	19.800	18.067	50.467
	Approx.	10.133	31.800	3.533	55.200	18.533	18.533	50.333
15×30	Exact	63.311	80.800	3.933	134.889	123.778	80.089	160.889
	Approx.	20.444	77.467	3.889	136.400	106.489	82.622	160.089
20×40	Exact	110.643	119.000	3.929	366.143	250.821	111.929	230.929
	Approx.	31.643	105.643	3.929	350.571	208.893	117.571	223.214
20×50	Exact	211.000	453.200	4.880	357.840	1330.240	383.080	836.280
	Approx.	51.760	282.720	4.800	326.280	693.760	437.800	720.520

It is only the part corresponding to the allocation costs which is approximated. Since the efficiency of the procedure relies on the performance of the tests which, in turn, depend on the accuracy of the lower bound sets, this explains the obtained results.

For equal size problems, the number of non-dominated solutions is much smaller for small-cost instances than for large-cost instances. Again, this can be explained by the fact that set-up costs have been divided by 10. Thus, the range of values for solutions is much smaller in the former case. As was expected, in the two cases the number of supported Pareto solutions (*Approx.*) is considerably smaller than the number of Pareto solutions (*Exact*). While in the case of small-cost instances the ratio between these two numbers increases moderately with size, for large-cost instances it increases notably with size. This relationship can be further appreciated in Fig. 5 where additionally these values are compared with the size of the initial upper bound set.

On the contrary, the average number of states and stages is larger for small-cost instances than for large-cost instances. These results are related with the efficiency of the elimination tests that, as will be seen, is higher for large-cost instances than for small-cost instances. Tables 3 and 4 show the distribution of the number of non-dominated solutions in the different stages of the search.

In general, the average number of stages reached does not differ significantly in the exact and the approximate executions neither for the small-cost nor for the large-cost instances. In all the cases that number seems to increase almost linearly with the number of plants. However, the difference in the average number of generated states between the exact and the approximate executions becomes important as size increases for small-cost instances whereas it remains moderate for the large-cost instances. For the two types of costs, with the exact executions the number of states seems to increase exponentially with the size of the problem. Yet, for the approximate executions there are important differences on the number of generated states

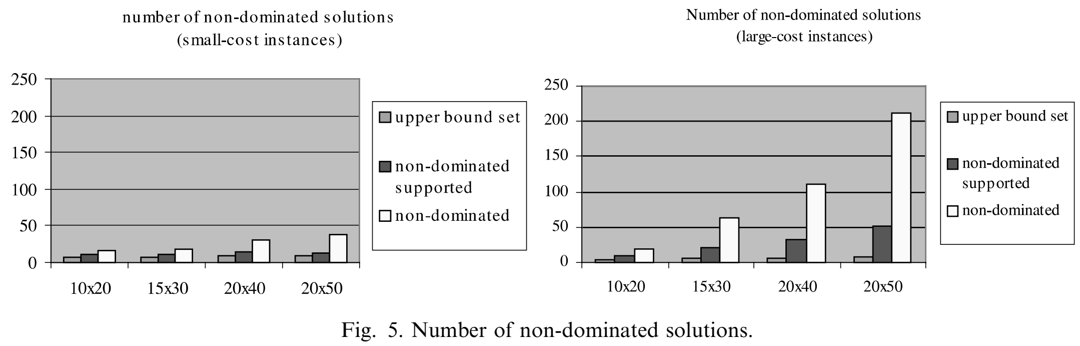

Fig. 5. Number of non-dominated solutions.

Table 3
Distribution of the number of non-dominated solutions for small-cost instances

Small cost		0	1	2	3	4	5	6	7	8	9
10 × 20	Exact	6.80	0.00	3.60	5.00	0.00					
	Approx.	6.87	0.00	2.07	1.40	0.00					
15 × 30	Exact	8.36	0.00	1.44	8.20	0.72	0.46	0.00	0.00		
	Approx.	8.42	0.00	1.49	1.33	0.00	0.00	0.00	0.00		
20 × 40	Exact	9.89	0.00	7.89	10.69	1.91	3.55	0.00	0.00	0.00	
	Approx.	9.93	0.00	4.11	1.29	0.00	0.00				
20 × 50	Exact	9.56	0.00	9.44	14.48	5.44	3.00	1.50	0.00	0.00	0.00
	Approx.	9.72	0.00	2.40	1.04	0.00					

Table 4
Distribution of the number of non-dominated solutions for large-cost instances

Large cost		0	1	2	3	4	5	6	7
10 × 20	Exact	2.27	1.73	14.47	0.00	0.00			
	Approx.	1.93	1.73	6.47	0.00	0.00			
15 × 30	Exact	3.18	0.96	56.60	2.90	0.00	0.00	0.00	
	Approx.	3.38	0.96	15.53	0.63	0.00	0.00		
20 × 40	Exact	2.68	1.71	98.29	8.42	1.11	0.00	0.00	
	Approx.	2.86	1.71	24.71	2.24	0.50	0.00	0.00	
20 × 50	Exact	3.20	0.92	179.72	1.86	0.00	0.00	0.00	
	Approx.	3.40	1.00	40.76	5.96	0.67	0.00	0.00	0.00

between small-cost and large-cost instances for equal size problems. Fig. 6 depicts a graphic with the average number of states for each of the two executions.

The efficiency of the different elimination tests can be seen in Fig. 7(a) and (b) where the proportion of the effectiveness is calculated over the average number of generated states. At the stages where they are applied, the elimination tests *test*2 and *test*3 use the exact value of the set-up costs at the stages and a lower bound set on the corresponding allocation subproblems. When the set-up costs are large compared to the allocation costs, the contribution of the lower bound set is not crucial to the overall lower bound set and, thus, the tests are applied very efficiently. However, when the contribution of the set-up costs decreases, as is the case with the small-cost instances, the role of the lower bound on the allocation subproblems increases and the efficiency of the tests reduces considerably. This occurs specially in the case of *test*2 that is

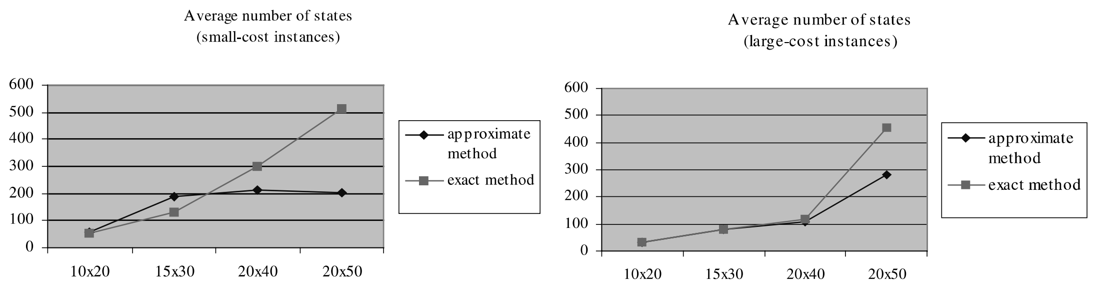

Fig. 6. Average number of generated states.

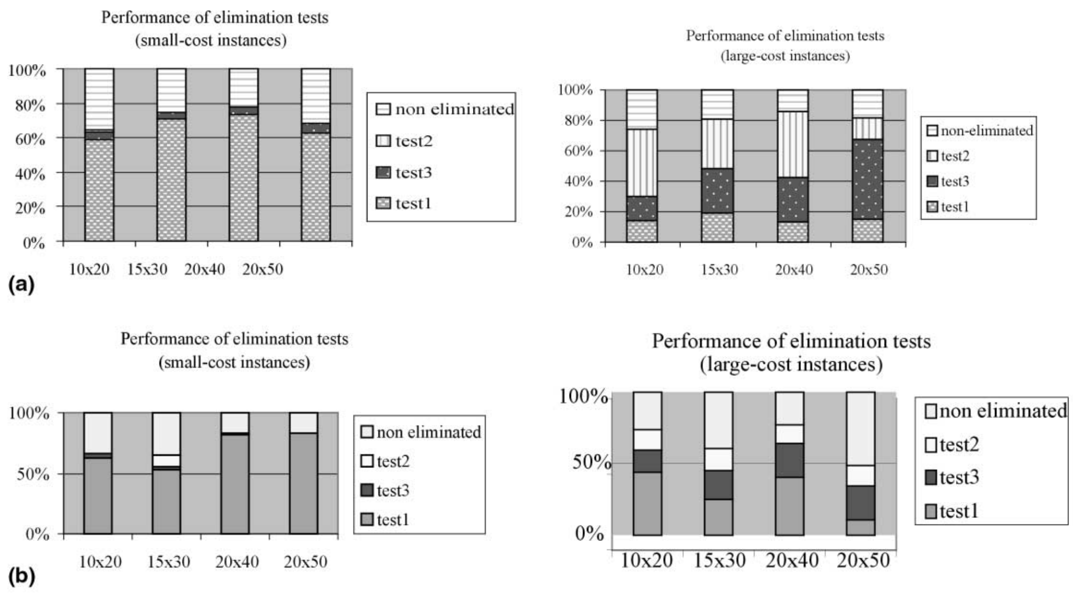

Fig. 7. Performance of elimination tests for: (a) exact executions, (b) approximate executions.

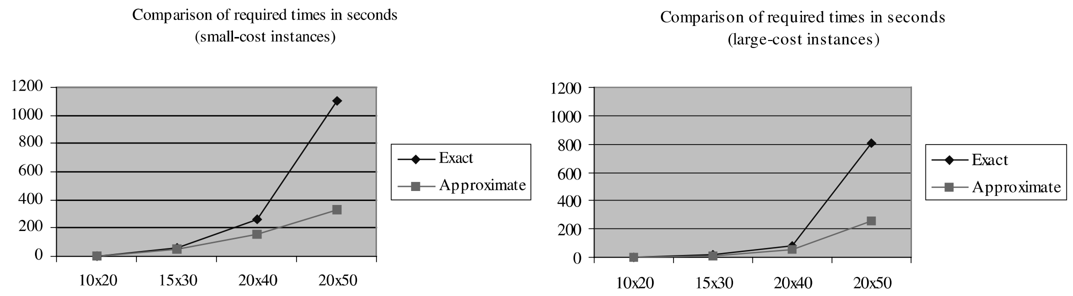

Fig. 8. Required CPU times in seconds.

hardly ever applied with small-cost instances of all sizes but whose efficiency is higher to test 1 for large-cost instances.

Finally, Fig. 8 shows the increase of times of the exact and approximate method. As was expected the exact resolution of the allocation subproblems results in a considerable increase on the overall execution time. In spite of the difficulty of the problem the execution times are reasonably small although the space requirements to store the search tree information are enormous (almost 0.5 GB for 20×50 problems).

7. Concluding remarks

Several heuristics can be developed to prune the exploration in depth of the search tree. It is straightforward to see that if we do not perform the whole exploration of the tree some of the solutions obtained might be actually dominated. Nevertheless, the computational experiments performed have shown that, even in moderately large problems, six levels of depth in the search are enough to find the whole

Pareto-optimal solution set (see Tables 3 and 4). Therefore, such a heuristic with a maximum depth level fixed to five or six would be almost exact and even with four levels the error committed would be less than 9% over the actual number of non-dominated solutions. Obviously, the loss of accuracy in this approach would be compensated with important reductions in CPU time and space requirements.

The second remark is on the application of this methodology to the general case of $q > 2$ criteria. In the paper we have presented the general approach to q criteria. The are only two differences between the bi-criteria and the general q-criteria cases: (1) one must solve q-criteria shortest path problems; and (2) one must obtain the supported non-dominated solutions using general purpose algorithms for multiobjective linear programming. Although these two questions are solved from the theoretical point of view (see Isermann, 1977; Gal, 1984; and Azevedo and Martins, 1991; Steuer, 1995) they introduce computational difficulties in the problem, due to the exponential behavior of the exact algorithms available to solve them.

Acknowledgements

The research of the first author has been partially supported by Spanish DGICYT grant number TIC2000-1750-C06-03. The research of the second author has been partially supported by Spanish DGICYT grant numbers PB97-0707 and BFM2001-2378.

References

Azevedo, J.A., Martins, E.Q.V., 1991. An algorithm for the multiobjective shortest path problem on acycil networks. Investigacao Operational 11, 52–69.

Captivo, M.E., Climaco, J., Figueira, J., Martins, E., Santos, J.L., Solving multiple criteria knapsack problems using labeling algorithms. Paper presented at IO2000, Setúbal. Available from //www.mat.uc.pt/~eqvm/cientificos/investigacao/Artigos/figueira.ps.gz.

Carrizosa, E., Conde, E., Fernández, F.R., Puerto, J., 1993. Efficiency in Euclidean constrained location problems. Operations Research Letters 14, 291–295.

Cornuejols, G., Nemhauser, G.L., Wolsey, L.A., 1990. The uncapacitated facility location problem. In: Mirchandani, P.B., Francis, R.L. (Eds.), Discrete Location Theory; Wiley-Interscience Series Discrete Mathematical Optimisation. Wiley, New York, pp. 55–117.

Delmaire, H., Díaz, J.A., Fernández, E., Ortega, M., 1999. Reactive GRASP and Tabu search based heuristics for the single source capacitated plant location problem. INFOR 37 (3), 194–225.

Ehrgott, M., Gandibleux, X., 2000. An Annotated Bibliography of Multicriteria Combinatorial Optimization. OR-Spektrum 22 (4), 425–460.

Erlenkotter, D., 1978. A dual-based procedure for uncapacitated facility location. Operations Research 26, 992–1009.

Gandibleux, X., Jaszkiewicz, A., Freville, A., Slowiñski, R. (Eds.), 2000. Multi-objective Metaheuristics Journal of Heuristics 6 (3), 291–431, special issue.

Gal, T., 1984. Linear parametric programming – A brief survey. Mathematical Programming Study 21, 43–68.

Hamacher, H.W., Labbé, M., Nickel, S., 1998. Multicriteria network location problems with sum objective. Networks 33, 79–92.

Hamacher, H.W., Nickel, S., 1996. Multicriteria planar location problems. European Journal of Operational Research 94, 66–86.

Hansen, P., Perreur, J., Thisse, J.F., 1980. Location theory, dominance and convexity: Some further results. Operations Research 28, 1241–1250.

Hindi, K.S., Pienkosz, K., 1999. Efficient solution of large scale, single source, capacitated plant location problem. Journal of the Operational Research Society 50, 268–274.

Isermann, H., 1977. The enumeration of the set of all efficient solutions of a linear multiple-objective programming. Operational Research Quarterly 28, 711–725.

Klamroth, K., Wiecek, M., 2000. Dynamic programming approach to the multiple criteria knapsack problem. Naval Research Logistics 47, 57–76.

Klein, D., Hannan, E., 1982. An algorithm for the multiple objective integer linear programming problem. European Journal of Operational Research 9, 378–385.

Kouvelis, P., Yu, G., 1997. Robust Discrete Optimization and Applications. Kluwer Academic Publishers, Dordrecht.

Krarup, J., Pruzan, P.M., 1983. The simple plant location problem: survey and synthesis. European Journal of Operational Research 12, 36–81.

Lee, S.M., Green, G.I., Kim, C., 1981. A multiple criteria model for the location-allocation problem. Computers and Operations Research 8, 1–8.

Ogryczak, W., 1999. On the distribution approach to location problems. Computers and Industrial Engineering 37, 595–612.

Pelegrín, B., Fernández, F.R., 1988. Determination of efficient points in multiple objective location problems. Naval Research Logistics Quarterly 35, 697–905.

Puerto, J., Fernández, F.R., 1999. Multicriteria mini-sum facility location problems. Journal of the Multicriteria Decision Analysis 8, 268–280.

Ramesh, R., Zionts, S., Karwan, M., 1986. A class of practical interactive branch and bounds algorithms for multicriteria integer programming. European Journal of Operational Research 26, 161–172.

Rasmussen, L.M., 1986. Zero-one programming with multiple criteria. European Journal of Operational Research 26, 83–95.

Ross, G.T., Soland, R.M., 1980. A multicriteria approach to location of public facilities. European Journal of Operational Research 4, 307–321.

Solanki, R., 1991. Generating the noninferior set in mixed biobjective linear programs: an application to a location problem. Computers and Operations Research 18, 1–15.

Steuer, R., 1995. Manual for the ADBASE. Multiple Objective Linear Programming Package, University of Georgia, Athens, Georgia.

Ulungu, E.L., Teghem, J., 1994. Multi-objective combinatorial optimization problems: A survey. Journal of Multi-criteria Decision Analysis 3, 83–104.

Villarreal, B., Karwan, M.H., 1981. Multicriteria Integer Programming: A (hybrid) dynamic programming recursive approach. Mathematical Programming 21, 204–233.

Wendell, R.E., Hurter, A.P., 1973. Location theory, dominance and convexity. Operations Research 21, 314–320.

Wendell, R.E., Hurter Jr., A.P., Lowe, T.J., 1977. Efficient points in location problems. AIIE-Trans. 9 (3), 238–246.

Zionts, S., 1979. A survey of multiple criteria integer programming methods. Annals of Discrete Mathematics 5, 389–398.

Zionts, S., Wallenius, J., 1980. Identifying efficient vectors: some theory and computational results. Operations Research 23, 785–793.