REACTIVE GRASP AND TABU SEARCH BASED HEURISTICS FOR THE SINGLE SOURCE CAPACITATED PLANT LOCATION PROBLEM[1]

HUGUES DELMAIRE, JUAN A. DÍAZ, ELENA FERNÁNDEZ

Departament d'Estadística i Investigació Operativa
Universitat Politècnica de Catalunya, Pau Gargallo, 5, 08028 Barcelona, Spain
{hugues, jadiaz, elena}@eio.upc.es

MARUJA ORTEGA

Departamento de Computación y T.I., Universidad Simón Bolívar, Apartado 89000
Caracas 1081-A, Venezuela
mof@ldc.usb.ve

ABSTRACT

This paper considers the Single Source Capacitated Plant Location Problem (SSCPLP). SSCPLP is a discrete location problem. It allows capacities on the plants to be opened and constrains each client to be served by a single open plant. The following algorithms are proposed: A Reactive GRASP heuristic; a Tabu Search heuristic; and two different hybrid approaches that combine elements of the GRASP and the Tabu Search methodologies. The elements of the proposed heuristics are presented. The Reactive GRASP algorithm is a self-tuning heuristic in which the calibration process is replaced by an automated criterion for selecting the parameter value. The Tabu Search heuristic provides the framework for the first of the hybrid approaches. It consists of two phases. The GRASP methodology is used for the first one, which can be viewed as a strong diversification mechanism. The second one consists of an intensification phase. The second hybrid algorithm follows the framework of the Reactive GRASP heuristic. It also consists of two phases and Tabu Search is used in the second phase as a mechanism to strengthen the Local Search. Computational experiments have been performed to evaluate the behavior of the proposed methods. The results on two different sets of test problems show that the proposed methods are very efficient. In particular, all of them outperform previous heuristic approaches in small computation times. Moreover, the outcome of different series of experiments carried out with CPLEX confirm the quality of all the heuristics and, in particular, of the two hybrid approaches.

RÉSUMÉ

Cet article traite du problème de localisation d'usine avec contraintes de capacités et source unique (SSCPLP). Le SSCPLP est un problème discret de localisation. Il alloue des capacités aux usines qui seront ouvertes et contraint chaque client à être servis par une seule usine. Les algorithmes suivants sont proposés: une heuristique réactive GRASP, une heuristique de recherche TABU, et deux approches hybrides différentes qui combinent des éléments des méthodologies GRASP et TABU. Les composants des heuristiques proposées sont présentés. L'algorithme réactif GRASP est une heuristique auto-régulatrice dans laquelle le processus de calibration est remplacé par un critère automatique pour sélectionner la valeur du paramètre. L'heuristique de recherche TABU fournit l'ossature de la première approche hybride. Elle est constituée de deux phases. Dans la première, la méthodologie GRASP est utilisée et peut être vue comme un puissant mécanisme de diversification. La seconde consiste en une phase d'intensification. Le second algorithme hybride suit l'ossature de l'heuristique réactive GRASP. Il consiste également de deux phases. La recherche TABU est utilisée dans la seconde phase comme mécanisme de renforcement de la recherche locale. Des tests ont été effectués pour évaluer le comportement des méthodes proposées.Les résultats sur deux ensembles de problèmes tests montrent que les méthodes proposées sont très performantes. En

[1]Recd. Aprl 1998; Acc. Nov. 1998

particulier, toutes battent les approches heuristiques antèrieures dans des temps de calcul très courts. De plus, les résultats de différentes séries de tests effectués avec CPLEX confirment la qualité des heuristiques et en particulier, des deux approches hybrides.

1. INTRODUCTION

There exist a variety of discrete location models which cover a wide range of problems and situations (see Mirchandani and Francis, 1990, for a comprehensive review). In all of them two finite sets are given. One corresponds to potential sites for plants that can be opened, and the other one to clients that have to be served. In all of the models there are two decisions. First, which of the sites should be selected for opening plants. Second, how should the open plants be allocated to satisfy the clients' demand.

Very often the models do not take into account the potential capacity on the facilities to be opened. However, in most applications this capacity is a real issue. In the studies where the capacities of the plants have been considered (generally referred to as the Capacitated Plant Location Problems (CPLP)), it is most common to allow splitting the clients' demands in such a way that each client can be served by more than one open plant. A recent review can be found in Sridaran (1995); heuristic approaches are presented in Jacobsen (1983) and an algorithm for large instances is given in Beasley (1988). Although there are many applications that can be successfully represented with such a model it is also easy to find other applications in which the demand of each client has to be satisfied from a single facility (see, for instance, Barceló and Casanovas, 1984; Barceló *et al.*, 1990; Barceló, Fernández and Jörnsten, 1991; Beasley, 1990; Delmaire *et al.*, 1997; Klincewicz and Luss, 1986; Kuehn and Hamburger, 1963; Neebe and Rao, 1983; Pirkul, 1987). These include situations where clients must be served by a single plant in order to maintain an acceptable level of client satisfaction. This occurs, for instance, when each delivery causes an operating cost to the client to maintain and update its inventory. Thus multiple deliveries would increase such costs. A different application is the choice of the site for platforms to be used for the drilling of oil wells (see Balas, 1982; Devine and Lesso, 1972) which also requires assigning each oil well to exactly one of the platforms.

This paper considers the Single Source Capacitated Plant Location Problem (SSCPLP), where splitting the clients' demands is not allowed and each client must be served by a single open plant.

Both CPLP and SSCPLP, are known to be NP-hard decision problems. However, in CPLP, for a given set of open plants, the corresponding allocation problem is a linear program (specifically, a transportation problem). On the contrary, in SSCPLP, for a given set of open plants, the associated allocation problem is a particular case of the Generalized Assignment Problem (GAP) which is NP- hard itself. A *priori* this would imply that SSCPLP is much harder to solve optimally than CPLP. While in CPLP all the assignment variables are continuous, in SSCPLP those variables are binary. Therefore, solving SSCPLP to optimality by enumeration could require exploring a considerable number of nodes of the search tree, even for small and medium-sized problems. This is probably why such an approach has not been explicitly considered for SSCPLP. Some enumeration schemes that partially explore the search tree can be found in the literature. In Neebe and Rao (1983) SSCPLP is modeled as a Set Partitioning problem and a branch and bound algorithm is presented. However, the maximum size of the tree is limited to 25 nodes. A different partial enumeration scheme in which the space of the locations is explored can be found in Barceló, Fernández and Jörnsten (1991).

Most of the previous work on SSCPLP is focused on obtaining good Lagrangean duals, whose solutions improve the lower bounds provided by the LP relaxation. These

bounds are then compared to the upper bounds obtained from different types of interchange heuristics. Several Lagrangean relaxations for SSCPLP can be found in Barceló and Casanovas (1984), Barceló *et al.* (1990), Barceló, Fernández and Jörnsten (1991), Beasley (1990), Guignard and Opaswongkarn (1990), Klincewicz and Luss (1986) and Pirkul (1987).

Generally the optimal solution to these problems is not known, although usually good lower bounds for them are at hand. This situation fully justifies considering efficient heuristic approaches to obtain good quality feasible solutions. Recently, Delmaire *et al.* (1997), proposed a variety of heuristics based on the following methodologies: Evolutive Algorithms, GRASP, Simulated Annealing and Tabu Search (TS). The proposed approaches were tested on the set of problems used in Barceló, Fernández and Jörnsten (1991) and the results considerably improved the best-known solutions in very small computation times. In particular, the proposed Tabu Search heuristic gave the best results in terms of the quality of the solutions, while the heuristic based on GRASP was one order of magnitude faster than all of the other heuristics and provided the second-best results in terms of quality. The present paper can be considered a sequel of Delmaire *et al.* (1997) in the sense that only the metaheuristics that proved to be useful for solving SSCPLP in that work have been considered here. Therefore, only the GRASP and the Tabu Search methodologies are now considered. Specifically, the algorithms that are proposed in the current work are the following: A Reactive GRASP heuristic, a Tabu Search heuristic and two different hybridization schemes that combine the GRASP and Tabu Search methodologies. The algorithms presented in this work share three characteristics with the procedures proposed in Delmaire *et al.* (1997). (1) They consider two separate stages: one basically focused on the plants selection problem and a second one mainly oriented to the Allocation Subproblem. Throughout the paper, the first stage will also be referred to as the constructive phase since it is at this level that different sets of open plants are selected and initial allocations within the open plants are obtained. The second phase will also be referred to as the improving phase. (2) They explore the same neighborhood structures. And, (3) they operate on the same relaxation of the original problem. A penalty term is added to the objective function of the relaxed problem to measure the violation of feasibility with respect to the original problem.

The characteristics that differentiate the algorithms proposed here with the ones presented in Delmaire *et al.* (1997) are next summarized. As opposed to the standard GRASP procedure of Delmaire *et al.* (1997), the GRASP method that is proposed here is a reactive one. The so-called reactive methods are, in essence, metaheuristic based self-tuning algorithms. These methods are designed to overcome the possible lack of robustness of standard metaheuristic based algorithms. To the best of our knowledge, reactive GRASP has been previously proposed only for the Matrix Decomposition Problem by Prais and Ribeiro (1997). In the case of SSCPLP, the standard GRASP of Delmaire *et al.* (1997) is extremely efficient in terms of computation time but it is very limited with respect to the number of different solutions (specially, different sets of open plants) that it generates. The Reactive GRASP that we propose overcomes this lack of variability and improves considerably the quality of the generated solutions. Although it involves a considerable increase in computation time over the standard GRASP of Delmaire et al. (1997), the times are still very small. The Tabu Search algorithm that is proposed here maintains the constructive phase of the algorithm of Delmaire *et al.* (1997), but considers a different approach for the improving phase. Specifically, the diversification phase for the Allocation Subproblem is no longer frequency based and consists of temporarily increasing the capacity of the plants. The intensification phase

for the Allocation Subproblem also presents a basic difference with the one of Delmaire *et al.* (1997) in that the strategy used to explore the list of candidate moves has changed. Finally, the tabu tenure of a forbidden solution is randomly selected in a given interval, instead of using a fixed parameter value as in Delmaire *et al.* (1997).

In our opinion, the proposed TS heuristic can already be considered to be a hybrid approach. Although TS provides mechanisms that can be used within a constructive phase, in our proposal the TS methodology is not applied at the constructive phase and a very simple procedure based on the GRASP methodology is used at this stage. However, the TS methodology provides the framework for the proposed heuristic and is also used to obtain improved solutions to the Allocation Subproblem. In this context, the constructive phase can be seen as a strong diversification mechanism within a Tabu Search scheme. Two different hybrid approaches are also proposed. They are designed to: (1) enhance intensification mechanisms and increase the number of generated elite solutions; and (2) study the trade-off between the constructive and the improving phases. The first approach follows the structure of the TS heuristic and incorporates additional features within a TS framework. It stresses the role of the improving phase and generates the elite solutions at this stage. In particular, it incorporates the Local Search (LS) component of the Reactive GRASP heuristic within the inner iterations of the TS procedure applied to the Allocation Subproblem. The second hybrid approach stresses the role of the constructive phase. The framework is provided by Reactive GRASP and TS is used in the improving phase although the role of such a phase is considerably reduced.

The proposed heuristics have been tested over the set of test problems used in Delmaire *et al.* (1997) and over a new set of test problems that have been generated to evaluate the performance of the heuristics for larger instances. For each approach, three different measures have been considered: a) the value of the best solution it provides, b) its robustness, measured in terms of the average deviation from the value of best known solution and, c) its efficiency in terms of the required computational times. To evaluate the quality of the methods, results have been compared to those reported in Delmaire *et al.* (1997) and to those produced by CPLEX within different time-limits. These confirm the effectiveness of the proposed methods, especially in the case of the two hybrid algorithms.

This work is structured as follows: Sections 1 and 2 present, respectively, the model that is considered for SSCPLP and a relaxation that will be used throughout the paper. Section 3 describes the neighborhoods that are explored by the different algorithms. In Section 4 a Reactive GRASP procedure for SSCPLP is proposed. Section 5 describes the Tabu Search algorithm that gives the framework for the two hybrid algorithms for SSCPLP. The proposed hybrid algorithms are presented in Section 6. The results comparing the performance of the different algorithms in the computational experiments are given and discussed in Section 7. Finally, Section 8 contains some conclusions and final remarks.

2. THE MODEL FOR (SSCPLP)

Let $I = \{1, \ldots, n\}$ and $J = \{1, \ldots, m\}$ denote, respectively, the sets of indices of clients and plants for SSCPLP. Also, let $\mathbf{f} = (f_j)_{j \in J}$ denote the vector of fixed costs of opening the plants, $\mathbf{c} = (c_{ij})_{i \in I, j \in J}$ the matrix of assignment costs of clients to plants, $\mathbf{b} = (b_j)_{j \in J}$ the vector of potential capacities of the plants and $\mathbf{d} = (d_i)_{i \in I}$ the vector of demands of the clients. Then, the Single Source Capacitated Plant Location Problem can be modeled as:

$$(SSCPLP) \quad \min \quad z = \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{j\in J} f_j y_j \tag{1}$$

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

$$\sum_{i\in I} d_i x_{ij} = 1 \quad \text{for all } j \in J \tag{3}$$

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

For each $j \in J$, the decision variable y_j takes the value 1 if the facility at site j is opened and 0 otherwise. Similarly, for each pair $i \in I$, $j \in J$ the decision variable x_{ij} takes the value 1 if client i is fully served from plant j and 0 otherwise. Constraints (2), together with the integrity conditions on the assignment variables, state that each client is served from one single plant. The role of the capacity constraints (3) is twofold. On the one hand, they insure that when a plant is opened, the overall demand assigned to it does not exceed its capacity. On the other hand, they also guarantee that no client is assigned to a site where no plant is opened. In SSCPLP, the capacity constraint (3) associated with each plant represents a 0-1 knapsack constraint which, combined with constraints (2), is in general much more difficult than its corresponding continuous version in CPLP. Note that in SSCPLP, for a given set of open plants, the associated Allocation Subproblem is a particular case of the Generalized Assignment Problem (GAP), which is a well-known NP-hard combinatorial optimization problem.

From a formal point of view, the "only" difference between CPLP and SSCPLP is that in CPLP the assignment variables x are continuous, $0 \leq x \leq 1$, while in SSCPLP they are binary variables, *i.e.* $x \in \{0,1\}$. For problems of the same size, *i.e.* the same number of sites (m) and the same number of clients (n), single sourcing implies a remarkable increase in the number of $\{0,1\}$ variables which goes from m in CPLP to $m + mn$ in SSCPLP.

For each client $i \in I$, the allocation costs have been expressed in terms of the cheapest assignment for the client. Let $cmin_i$ denote the cost of the best assignment for client i, *i.e.* $cmin_i = \min\{c_{ij} : j \in J\}$. Then, for each pair $i \in I$, $j \in J$, $c_{ij} = \Delta_{ij} + cmin_i$ where Δ_{ij} represents the increment from the best assignment for serving client i from plant j. The Δ_{ij}'s allow fair comparisons for deciding the most suitable set of clients to be allocated to an open plant, particularly when the assignment costs have different ranges for the various clients. Since all clients have to be assigned to some open plant, $K = \sum_{i\in I} cmin_i$ is a fixed assignment cost which has to be incurred. Therefore, objective (1) can be expressed as

$$\min z = K + \min\left\{ \sum_{i\in I}\sum_{j\in j} \Delta_{ij}x_{ij} + \sum_{j\in J} f_j y_j \right\}$$

and the following equivalent objective function can be considered for SSCPLP:

$$z' = \sum_{i\in I}\sum_{j\in j} \Delta_{ij}x_{ij} + \sum_{j\in j} f_j y_j \tag{1'}$$

3. THE RELAXED PROBLEM RSSCPLP

In SSCPLP feasibility can be difficult to achieve even when an appropriate set of open plants is at hand. This is due to the difficulty of the Allocation Subproblem that, as has already been pointed out, is a GAP for which even the feasibility question is NP-complete. In this context, where finding feasible solutions is not a simple task, if the

neighborhoods explored by heuristic methods are constrained to feasible solutions, very few solutions are usually generated and the procedures terminate very quickly with a (normally not very good) local optimum. On the other hand, when some violation of feasibility is allowed, the search is endowed with a higher level of flexibility and much larger areas of the search space can be explored. In those cases, the deviation from feasibility can be measured and controlled through a penalty term in the objective function. This strategy has been used in Evolutive Algorithms (see, for instance Michalewicz, 1992) and in Tabu Search based heuristics, where it can be interpreted in terms of strategic oscillation (see Glover, 1989, 1990). Yagiura *et al.* (1997) have also applied it to a variable depth search method for the GAP. To the best of our knowledge, it has only been used in Delmaire *et al.* (1997) in the context of GRASP algorithms.

In this work this strategy has been applied to the different approaches to be presented and the original SSCPLP problem has been replaced by the relaxation:

$$
\begin{aligned}
(RSSCPL) \quad \min \quad z'' &= \sum_{i\in I}\sum_{j\in J}\Delta_{ij}x_{ij} + \sum_{j\in J} f_j y_j + P(x) && (1'')\\
s.t. \quad \sum_{j\in J} x_{ij} &= 1 \quad \text{for all } i \in I && (2)\\
\sum_{j\in j} b_j y_j &\geq D && (3')\\
x_{ij} &\leq y_j \quad \text{for all } i \in I, j \in J && (3'')\\
x_{ij}, y_j &\in \{0,1\} \quad \text{for all } i \in I, j \in J && (4)
\end{aligned}
$$

The aggregated demand constraint (3′) is a surrogate constraint that is obtained by adding up constraints (3) and taking into account constraints (2). Its right hand side D takes the value $\sum_{i\in I} d_i$. D is usually referred to as the aggregated demand. Constraint (3′) is a single 0-1 knapsack type constraint that is much easier to handle than the set of constraints (3). It insures that the plants that are opened have enough joint capacity to satisfy the overall demand of the clients. However, it does not even guarantee the feasibility of the Allocation Subproblem associated with a set of open plants. Now, constraints (3″) are needed to prevent assigning clients to non-open plants. The penalty term $P(x)$ in the objective function measures the infeasibility, relative to constraints (3) of SSCPLP, of the solutions to RSSCPLP. It takes the form

$$P(x) = \sum_{j\in J} \rho_j s_j \tag{5}$$

where $s_j = \max\{\sum_{i\in I} d_i x_{ij} - b_j,\ 0\}$ denotes the amount of capacity of plant j violated by a solution. The parameters ρ_j vary in the different approaches to be proposed and will be described later on. Due to the definition of the s_j, $j \in J$, objective (1″) is no longer a linear function. However, RSSCPLP is a relaxation of SSCPLP since feasible solutions for SSCPLP are also feasible for RSSCPLP and for such solutions (1″) and (1′) take the same value. Since the only constraints of SSCPLP that can be violated by feasible solutions to RSSCPLP are the inequalities (3), the inclusion of $P(x)$ in (1″) guides the search processes towards the feasible domain.

In general, when referring to solutions both to SSCPLP or to RSSCPLP we will assume that the vector x satisfies the assignment constraints (2). It will also be assumed that no client is assigned to a non-open plant, *i.e.* $x_{ij} = 1 \Rightarrow y_j = 1$. Nevertheless, unless otherwise stated, feasibility relative to the capacity constraints (3) will not be implicitly assumed. Alternatively, solutions will be represented by pairs $\sigma = (O, A)$ where $O \subseteq J$ is the set of open plants and A is the assignment for the clients; *i.e.* $O = \{j \in J\colon\ y_j = 1\}$, and $A\colon I \to J$ such that $A(i) \in O$ and $A(i) = j \Leftrightarrow x_{ij} = 1$.

4. NEIGHBORHOODS

One of the characteristics shared by the different procedures presented here is that they explore the same neighborhoods. These neigborhoods were also used in Delmaire *et al.* (1997) and are described in this section. They can be classified in two groups. In the first group, the set of open plants is fixed and the considered moves only involve the assignments. The neighborhoods are:

Clients' Shift Neighborhood (see Figure 1)

$N_1(\sigma) = \{\sigma' = (O', A') : O' = O, \exists! i^* \in I \text{ s.t. } A'(i^*) \neq A(i^*)\}$

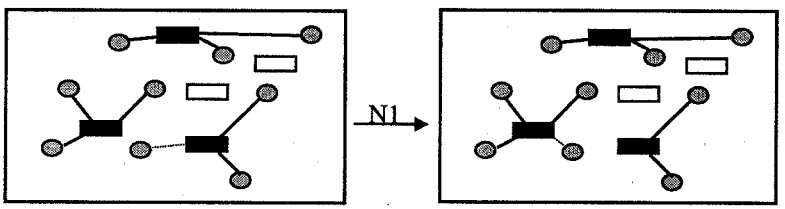

Figure 1: Clients' Shift Neighborhood

Clients' Swap Neighborhood (see Figure 2)

$N_2(\sigma) = \{\sigma' = (O', A') : O' = O, \exists i_1, i_2 \in I \text{ s.t. } A'(i_1) = A(i_2), A'(i_2) = A(i_1), \forall i \neq i_1, i_2\ A'(i) = A(i)\}$

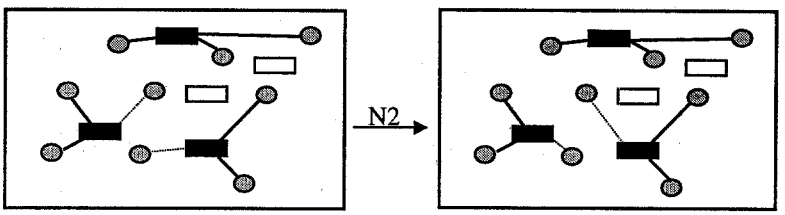

Figure 2: Clients' Swap Neighborhood

Open-Open Plants' Swap Neighborhood (see Figure 3)

$N_3(\sigma) = \{\sigma' = (O', A') : O' = O, \exists j_1, j_2 \in O \text{ s.t. } \forall i \text{ s.t. } A(i) = j_1\ A'(i) = j_2, \forall i \text{ s.t. } A(i) = j_2\ A'(i) = j_1, \forall i \text{ s.t. } A(i) \neq j_1, j_2\ A'(i) = A(i)\}$

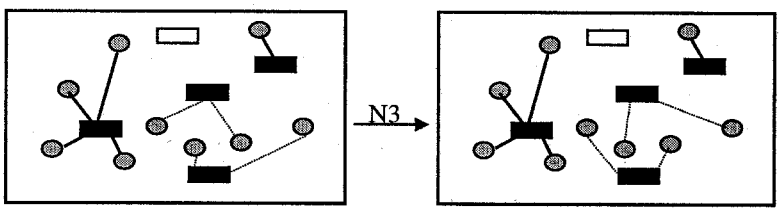

Figure 3: Open-Open Plants' Swap Neighborhood

The solutions in the Clients' Swap Neighborhood and in the Open-Open Plants' Swap Neighborhood can also be generated as compositions of "individual" reassignment moves

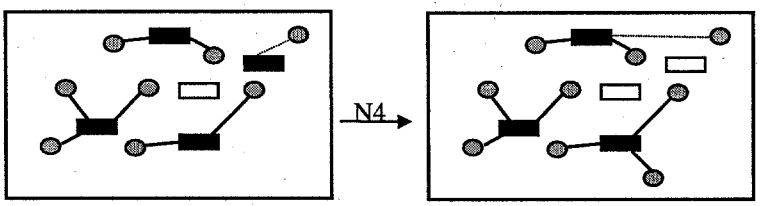

Figure 4: Closing-Plant Neighborhood

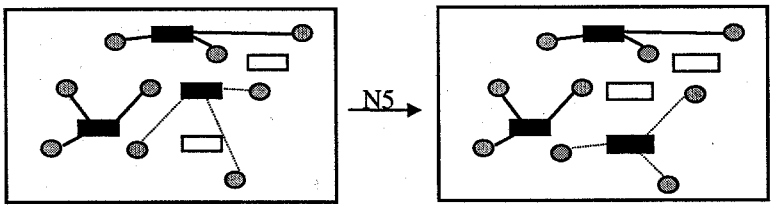

Figure 5: Open-Closed Plants' Swap Neighborhood

in the Clients' Shift Neighborhood. Often, the penalty term in the objective function of RSSCPLP makes some of these "individual" moves very unattractive, while the composition considered as a single move can be a desirable choice. For this reason these two neighborhoods have been considered independently.

The neighborhoods of the second group consider moves concerning the set of open facilities (as well as the corresponding reassignments). These are:

Closing-Plant Neighborhood (see Figure 4)

$N_4(\sigma) = \{\sigma' = (O', A') : O' = O \backslash \{j*\}, \forall i \text{ s.t. } A(i) \neq j^* \ A'(i) = A(i)\}$

Open-Closed Plants' Swap Neighborhood (see Figure 5)

$N_5(\sigma) = \{\sigma' = (O', A') : \exists j_1 \in O, j_2 \in J \backslash O \text{ s.t. } O' = O \backslash \{j_1\} \cup \{j_2\}, \forall i \text{ s.t. } A(i) = j_1 \ A'(i) = j_2, \forall i \text{ s.t. } A(i) \neq j_1 \ A'(i) = A(i)\}$

All the considered neighborhoods are "natural" to the problem structure in the sense that for a given solution it is relatively easy to obtain neighbor solutions feasible to RSSCPLP. However, some of these neighborhoods seem to explore areas of the solution space that are more distant from the original solution than others. Specifically, the neighborhoods of the second group, where not only some assignments are modified but also the open-plants set changes, seem to lead to more distant solutions. These neighborhoods are explored in the Local Search of the Reactive GRASP as a mechanism to "diversify" the search when no more moves in the "closer" neighborhoods are found. Our experience tells us that exploring "remote" neighborhoods after a reasonable search in the "closer" neighborhoods is a good decision for this algorithm. The same neighborhoods are also explored in the second hybrid algortihm which is embedded in a Reactive GRASP framework. However, the Tabu Search algorithm only explores some neighborhoods of the first group, namely: clients' shift and clients' swap. This is due to the structure of this algorithm where diversification is achieved with other techniques. Yet, the first hybrid approach, that is embedded in a TS framework, again incorporates the search in those neighborhoods within the TS procedure, both as an additional diversification mechanism and as a procedure to generate elite solutions on which intensification can be applied.

GRASP Algorithm (α)

begin
for a fixed number of iterations **do**
Construct a Greedy Randomized Solution (α)
Apply Local Search
end

Figure 6: Sketch of GRASP Algorithm

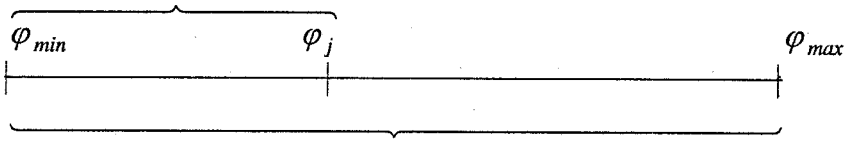

Figure 7:

5. REACTIVE GRASP FOR SSCPLP

One of the possible drawbacks of metaheuristic based algorithms is lack of robustness, in the sense that their effectiveness can be very dependent on the chosen values of the parameters. This usually implies a considerable task to "tune" such values in order to achieve higher performance. This has led to the study of the so called reactive methods which, in essence, are metaheuristic based self-tuning algorithms. In Battiti and Tecchiolli (1994, 1995) and Battiti (1996) pioneer reactive methods based on Tabu Search are developed. Recently, Prais and Ribeiro (1997) have proposed a Reactive GRASP algorithm for the matrix decomposition problem. This section describes a Reactive GRASP algorithm for (SSCPLP) based on the standard GRASP algorithm proposed in Delmaire *et al.* (1997).

GRASP is a metaheuristic originally proposed by Feo and Resende (1995). It is an iterative method that, within each iteration, contains two phases. The first phase builds a solution from scratch, while the second phase is a local search that tries to improve the solution obtained in the first stage. An outline of a standard GRASP algorithm is provided in Figure 6:

The construction phase seeks a compromise between quality and variety of the solutions, that is achieved by partially randomizing a greedy procedure. In particular, the solution is iteratively constructed one element at a time. The choice of the next element to be added is determined by randomly selecting one element from a Restricted Candidate List (RCL) that contains the best candidates, as measured by the greedy function (as opposed to standard greedy procedures in which the choice of the next element is determined by the top candidate).

Let φ denote the greedy function for a minimization problem. RCL can be defined to have a fixed number of elements or (a more flexible option) to contain all the elements within a given distance of the top candidate as a function of φ. Additionally, if in this second case, the closeness (as measured by φ) of a candidate to the top element is evaluated relative to the range of values taken by all the elements (as opposed to relative to the value of the top candidate), the threshold value can be expressed as $\alpha_\bullet(\varphi_{\max} - \varphi_{\min})$ where $\alpha \in [0, 1]$. Therefore, for a given value of α, $0 \leq \alpha \leq 1$, the Restricted Candidate List can be defined as (see Figure 7):

```
initialization:
        for each possible parameter value α do
         begin
                for a fixed number of iterations do
                 begin
                        Construct a Greedy Randomized Solution (α)
                        Apply Local Search
                 end
                 Evaluate its utility ut_α
         end
while (not stop_criterium) do
 begin
        Select α with highest utility
        Construct a Greedy Randomized Solution (α)
        Apply Local Search
        Update its utility ut_α
 end
```

Figure 8: Reactive GRASP Algorithm

$RCL = \{j: (\varphi_j - \varphi_{\min})/(\varphi_{\max} - \varphi_{\min}) \leq \alpha\}$.

That is, first each candidate is evaluated using the greedy function. Then, RCL contains all elements whose greedy function values are below some threshold which, in turn, depends on the selected value of the index α. In standard GRASP, α is a parameter of the procedure whose value is fixed *a priori*. Obviously, the performance of the procedure relies strongly on the adequacy of the selected value. Thus, a calibration process for selecting the "best" value of α, or repeating the process for different values of α is required. In Reactive GRASP, this is performed by measuring the goodness for each possible value of α and by defining an automated selection criterion for this parameter's value at the different iterations of the process. Before this step, the range of possible values for α needs to be discretized. In the proposed algorithm for SSCPLP, two conditions are required for a given value of α to be considered good. The first one is measured in terms of the quality of the solutions that it generates. In particular, it is desired to produce a small average deviation from the value of the best solution known so far. The second condition refers to its "variability": it is desirable to generate many different solutions. Specifically, for a given α, let $\overline{dev}_\alpha$ denote the average deviation of the solutions provided by α from the value of the best solution known so far. Its *variability*, v_α, is defined as the proportion of different solutions obtained with α divided by the total number of iterations in which it is selected. The *utility* of a given α, ut_α is then defined as follows:

$$ut_\alpha = v_\alpha \sqrt{dev_\alpha} \tag{6}$$

Each time a value of α is used both, its variability and utility, are updated. Therefore, at each iteration of the process the most profitable value of α to that point is known. Initially, for each α a fixed number of iterations are performed and its utility is evaluated. Then, at each iteration the most effective value of α (the one with the highest utility) is selected and used. The procedure stops when there is no improvement for a fixed number of iterations. An outline of the Reactive GRASP algorithm is provided in Figure 8.

The construction phase and the Local Search that are used in the Reactive GRASP for SSCPLP are the ones proposed in Delmaire *et al.* (1997). They are next summarized.

while (end=false) **do**
Explore N1 (Client's Shift Neighborhood)
Explore N2 (Client's Swap Neighborhood)
if (initial_solution has not been updated in N1 and N2) **then**
Explore N3 (Open-Open Plants' Swap Neighborhood)
if (initial_solution has not been updated in N1, N2, and N3) **then**
Explore N4 (Closing Plant Neighborhood)
if (initial_solution has not been updated in N1, N2, N3 and N4) **then**
Explore N5 (Open-Closed Plants Interchange Neighborhood)
if (initial_ solution has not been updated) end=true

Final Local Search Stage (Search for feasible solutions)
Explore N1 and N2 (Perform movements only if infeasibility is reduced)

Figure 9: Implementation of the Local Search Phase

Construction Phase

At each step of the construction phase one plant is opened and several clients are assigned to it. The assignment is always feasible to SSCPLP, in the sense that the capacity constraints on the opened plants are never violated by partial solutions. Once a client is assigned to a plant it is never reassigned during this phase. It is further assumed that, within each plant, clients are ordered by increasing values of the Δ_{ij}'s (ties are broken by decreasing values of the demands). Given a partial solution, for each non opened plant j, the greedy function is defined as

$$\varphi_j = \Big(f_j + \sum_{i \in C_j} \Delta_{ij}\Big)/|C_j| \tag{7}$$

where C_j is the set of all the unassigned clients that fit into location j, assuming that clients are assigned to j according to the ordering described above. The numerator of this expression is the cost associated with opening plant j and assigning to it all the clients indexed in C_j. Therefore φ_j can be interpreted as the overall cost per client assigned to plant j, if this plant were to be opened.

Local Search

For the Local Search, the capacity constraints on the plants are relaxed as explained before, and the problem RSSCPLP is considered. The penalty term in the objective function (1″) is obtained by considering the assignment costs $\Delta_{ij} + \rho_{ij}$ where ρ_{ij} is defined as:

$$\rho_{ij} = \begin{cases} (f_j/b_j)s_j\Big(d_i/\sum_{k:A(k)=j} d_k\Big) & \text{if client } i \text{ is assigned to plant } j \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

By definition these penalties are measured in units of cost and can be interpreted in the following way: $d_i/\sum_{k:A(k)=j} d_k$ is the ratio of the total demand assigned to plant j that can be "attributed" to client i. Since f_j/b_j represents the cost per unit of capacity associated with plant j, $(f_j/b_j)s_j$ would be the cost of having s_j extra units of capacity in plant j. Thus, ρ_{ij} can be seen as the cost of increasing the capacity of plant j, as to make the current set of assignments to the plant feasible, that should be charged to client i. The value of ρ_{ij} depends not only on client i being assigned to plant j, but also on which other clients are assigned to plant j. Thus, adding up all the assignment

costs associated with a violated plant we obtain:

$$\sum_{i \in C_j} (\Delta_{ij} + \rho_{ij}) = \sum_{i \in C_j} \Delta_{ij} + (f_j/b_j)s_j \tag{9}$$

which means that the extra cost of increasing the capacity of plant j in order to make its current set of assignments feasible, should be $(f_j/b_j)s_j$. This gives the value $\rho_j = f_j/b_j$ for the penalty term (5) in (1″). The algorithm of Figure 9 describes the implementation of the Local Search phase.

In the Local Search phase, the strategies that guide the search in the different neighborhoods, as well as the acceptance criteria are the following:

Clients' Shift Neighborhood (Explore N1):

For a given solution, clients with strictly positive assignment costs $\Delta_{ij} + \rho_{ij}$ are ordered by decreasing values in a list. The list is explored, starting with the first element (the client with highest assignment cost). Selected clients are reassigned to the first open plant for which reassignment would improve the cost function (1″) (if it exists). For reassignments, the plants are considered in the order they were opened in the construction phase. After reassigning a client, the list is reordered and another client is selected. When a client is not reassigned, it is removed from the list. The search continues until the list is empty.

Clients' Swap Neighborhood (Explore N2):

The above list is considered and managed as before. Now, when a client is chosen, it is swapped with the first subsequent element of the list that improves the cost function (1″) (if it exists).

Open-Open Plants' Swap Neighborhood (Explore N3):

For each pair of open plants the overall cost of swapping the assignments of their allocated clients is evaluated. The best interchange, among the ones that improve the cost value is performed.

Closing Plant Neighborhood (Explore N4):

If any open plant has one single client assigned to it, and there is a reassignment of the client that, together with the savings of closing the plant, improves the cost function (1″), the plant is closed provided that the remaining open plants satisfy the aggregated demand constraint.

Open-Closed Plants' Swap Neighborhood (Explore N5):

For each pair of plants (j_1, j_2), j_1 open and j_2 closed, the overall cost of closing j_1, opening j_2 and reassigning all the clients of j_1 to j_2 is evaluated. The best interchange, among those that improve the cost value (1″) and satisfy the aggregated demand constraint is performed.

Achieving feasibility (Final Local Search Stage):

Since the goal of the search is finding feasible solutions to SSCPLP and the Local Search operates on RSSCPLP, a final local search stage has been included. This stage is used when the solution that results after applying the above strategies to the search in the neighborhoods is not feasible to SSCPLP. Then, a series of moves in Clients' Shift and Clients' Swap Neighborhoods are performed. At this point, the goal that guides the search is attaining feasibility. Therefore, only moves that reduce infeasibility are considered and the "best" one is performed. When none of the considered moves improve the objective function value (1″) (which is usually the case) the "best" move is taken to be the one which deteriorates less the objective value.

6. TABU SEARCH ALGORITHM FOR SSCPLP

This section presents a Tabu Search algorithm for SSCPLP. Tabu Search, introduced by Glover (1989, 1990), is a metaheuristic method for intelligent problem solving. Its power is based on the use of adaptive memory to record historical information for guiding the search process. The simple form of Tabu Search uses only the short-term memory component. It is a form of aggressive exploration that seeks to make good moves, subject to the constraints implied by tabu restrictions. The primary goal of such constraints is to go beyond points of local optimality. Long-term memory is also used for intensification and diversification purposes which are two highly important components of Tabu Search. Intensification strategies exploit features historically found to be good, while diversification strategies encourage the search process to explore unvisited regions. The use of longer term memory for intensification and diversification purposes makes the Tabu Search methodology significantly stronger. The Tabu Search algorithm for SSCPLP proposed here has the same two-phase structure as the TS procedure of Delmaire *et al.* (1997). The constructive phase is also taken from Delmaire *et al.* (1997) and performs a diversification. Here the primary objective is to construct a set of solutions given by promising combinations of open plants. The improving phase is new. It considers the Allocation Subproblem and performs an intensification on each solution provided by the constructive phase. Both phases apply to RSSCPLP. However, the objective function considered in the constructive phase is (1′), while in the improving phase it is (1″).

Constructive phase (Diversification phase)

The diversification phase generates a set of k solutions using a variation of the greedy randomized algorithm. The greedy function (7) is used to iteratively select the plants to be opened and to assign clients. The difference from the construction phase of GRASP is that plants are opened only until the aggregated demand constraint (3′) is satisfied, instead of opening as many plants as needed to allocate all clients without violating the capacity constraints. Then, the remaining unassigned clients (if any) are processed in descending order of demand and assigned to the plant with greatest available capacity, even if that implies violating its capacity. The reason for this variation is that the goal of this phase is to identify promising combinations of plants, leaving to the next phase the Allocation Subproblem that will be considered independently. There is a second reason (related to the first one) which is that the selected combinations of plants should be as "small" as possible, given that the set of open plants is fixed in the second phase. During the intensification phase of the algorithm, the best r solutions associated with different plant sets are considered (for more details see Delmaire *et al.*, 1997).

Improving Phase (Intensification on solutions)

This phase does not consider the possibility of modifying the initial set of open plants. It deals with the Allocation Subproblem of clients to plants. Therefore, only shift and swap moves are considered. The neighbor generation mechanism differs from the one used in Delmaire *et al.* (1997). The following rules define the generation mechanism used in this Tabu Search implementation:

- Clients are processed in order of decreasing values of Δ_{ij}.
- For a given client i, shift and swap moves are evaluated and the best admissible move as measured by objective function (1″) is associated with the client.
- If the best admissible move for the client being processed improves the objective function value (1"), the candidate generation mechanism is halted and the move performed.

```
Constructive phase   (Strong Diversification Phase)
        Generate k initial solutions
        Select the best r sets of open plants among the k previous solutions
Improving phase      (Intensification on sets of plants to find solutions to the Allocation Subproblem)
        for (each solution) do
         begin
                for (l iterations) do
                 begin
                        Apply short-term memory component
                          (Intensification for the Allocation Subproblem)
                        Apply diversification using perturbation on the open plants capacities
                         (Diversification for the Allocation Subproblem)
                end
         end
```

Figure 10: Tabu Search Algorithm for SSCPLP

- Otherwise, if all clients have been processed but none of the associated moves improve objective function (1"), the admissible move with the lowest increment is selected and performed.

To identify admissible moves, candidate moves in N1 and N2 that are not tabu-active are considered and the following steps are performed. First, the two plants associated with the move are identified. Second, for each of the two plants, the increment on the violated capacity resulting from performing the move, is evaluated. When, for a given plant, the move does not increase the violated capacity, the above increment is recorded as zero. Finally, if the sum of the increments associated with the two involved plants is within a percentage (0% - 3%) of the total capacity of the open plants, the move is considered admissible. This candidate generation mechanism is very flexible, since for each client, all candidate moves in N1 and N2 are explored. Although this implies considerable effort, it is highly probable that an admissible move that improves the objective function value before all clients have been processed will be found. The reason is that clients are processed in descending order of Δ_{ij} and the moves associated with those clients are the ones that have a better chance to improve the objective function value.

The penalty term in the objective function (1") is defined by

$$P(x) = \rho \sum_{j \in O: s_j > 0} s_j$$

where ρ is a penalty parameter which increases or decreases depending on the number of feasible and infeasible solutions obtained during the last iterations (see Gendreau, Laporte and Séguin, 1996). This strategic oscillation behavior permits alternation between feasible and infeasible solutions.

Recency-based memory is managed by means of dynamic random tabu tenures. The attribute used for a move is an ordered pair (i, j) that means that, before performing the move, client i was assigned to plant j. The tabu tenure t of a move is randomly selected within a range $[t_{\min}, t_{\max}]$. Therefore, if client i is moved from plant j_1 to plant j_2, the reverse move will be forbidden during the next t iterations. The simple type of aspiration criterion is used.

The Tabu Search presented here also differs from the one presented in Delmaire *et al.* (1997) in the diversification strategy employed. Here diversification is performed without the use of memory. Instead, a perturbation of the open plants capacities is

```
for (k iterations) do
 begin
  Diversification phase: Use variation of GRASP to construct an initial solution
  Intensification phase:  (Intensification on initial solution)
        for (l iterations) do
         begin
              Apply short-term memory component
                (Use TS to intensify on solutions to Allocation Subproblem)
              Perturb the r.h.s's
                (Use TS to diversify on solutions to Allocation Subproblem)
         end
end
```

Figure 11: Hybrid algorithm within a TS framework for SSCPLP

used for diversification purposes (the capacity of each open plant is incremented in a 10% of its value). Since diversification strategies are designed to drive the search toward unvisited regions, perturbation of plants capacities helps to identify good assignments that otherwise would not be identified. Recall that the short-term memory phase will fail to identify good assignments, either when these assignments are associated with moves with high infeasibility measures (violated capacity of the open plants), or when they require a number of transition moves in which this infeasibility measure is high. When the capacity of the open plants is slightly incremented, these assignments have a chance to be selected and performed. This diversification strategy has the effect of perturbing the current solution, to an extent that the search is directed toward an unexplored region. It can also be interpreted as a strategic oscillation scheme, since it has the effect of decreasing the penalty term value during its application. The diversification scheme used has proven to be effective, since it increases the ability of the method to explore the solution space. As depicted in Figure 10, when the short-term memory component of the algorithm is completed, the capacity of the open plants is incremented and the process continues for a fixed number of iterations. Then the original open plants capacities are recovered and the short-term memory component reinitiated. This cycle, which alternates between intensification and diversification phases, is repeated for l iterations. An outline of the Tabu Search algorithm for the SSCPLP is depicted in Figure 10.

In the TS algorithm of Figure 10, occasionally the second phase will not be applied to some of the r solutions selected in the first phase. For each selected solution, a test is applied just before entering the second stage. The test compares the value of a lower bound associated with the solution and the current incumbent value. The lower bound is computed by adding up two terms. The first term is the sum of the fixed costs of the plants that are opened in the solution. For the second term, the cheapest assignment cost within the set of open plants is evaluated for each customer, and their sum is calculated, When the lower bound is geater than or equal to the incumbent value, the second stage is not applied to the selected solution. The reason is that the incumbent value cannot improve within the given set of open plants and the second phase only considers the Allocation Subproblem where the set of open plants is fixed.

7. THE HYBRID APPROACHES

The TS heuristic sketched in Figure 10 can already be considered to be a hybrid approach since it combines different methodologies at different stages of the procedure. Recall that TS provides mechanisms that could be used in the constructive phase like,

```
Constructuve Stage        (Strong Diversification phase)
  Use variation of GRASP to construct k solutions
  Select the r best solutions associated to different sets of open plants.

Improving Stage
for each solution selected in the first stage do
  for (l iterations) do
    begin
        Apply LS
                (Build elite solutions for the overall SSCPLP problem)
        Apply short-term memory component
                (Use TS to intensify on elite solutions to Allocation Subproblem)
        Perturb the r.h.s's
                (Use TS to diversify on Allocation Subproblem)
    end
end
```

Figure 12: Hybrid Algorithm 1

for instance, constructive heuristics based on surrogate constraints, some strategic oscillation approaches that iteratively alternate between constructive and deconstructive steps, or simply the use of a frequency matrix in a constructive phase (see Glover and Laguna, 1997). However, in our proposal, the TS methodology is not applied in the constructive phase and a very simple procedure based on the GRASP methodology is used at this stage. On the other hand, the TS methodology provides the framework for the SSCPLP algorithm and is also applied to obtain the solutions to the Allocation Subproblem. In fact, the TS heuristic of Figure 10 is a variation of the hybrid algorithm within a TS framework shown in Figure 11.

In the scheme of Figure 11 the intensification phase is applied to every solution generated in the diversification phase. Instead, in the proposed TS algorithm, the intensification phase is only applied to the r best solutions generated in the diversification phase. Since the intensification phase, in turn, consists of repeated applications of the intensification/diversification cycle to the current solution of the Allocation Subproblem, this should result in a considerable reduction in computation time.

Nevertheless, in the framework of Tabu Search the efficiency of any procedure relies strongly on the effectiveness of its diversification mechanisms and on its capacity to generate elite solutions. The TS heuristic depicted in Figure 10 contains two different diversification mechanisms. (1) The procedure used in the constructive phase to generate different sets of open plants, which can be seen as a strong diversification mechanism with respect to the plants selection problem. (2) The diversification phase of the improving stage which achieves diversification for the Allocation Subproblem. However, the capacity to generate elite solutions of the TS heuristic shown in Figure 10 is rather limited, since these are only obtained in the intensification phase of the improving stage where the set of open plants is fixed. This means that no elite solutions for the plants' selection problem are generated at any stage of the procedure. The first hybrid approach proposed in this section follows the structure of the TS heuristic outlined in Figure 10. It focuses on increasing the number of generated elite solutions (both to the plants' selection problem and to the Allocation Subproblem). In particular, it incorporates the LS of the Reactive GRASP within the inner iterations of the improving stage. An outline of the first hybrid algorithm is presented in Figure 12.

```
begin
    Construct a Greedy Randomized Solution (α)
    Apply L S
    for (l' iterations) do:
        Apply short-term memory component
            (Use TS to intensify on elite solutions to Allocation Subproblem)
        Perturb the r.h.s's
            (Use TS to diversify on solutions to Allocation Subproblem)
        Apply LS
    end
end
```

Figure 13: Hybrid algorithm within a Reactive GRASP framework for SSCPLP

```
Constructive Stage        (Strong Diversification phase)
        Use Reactive GRASP to construct k elite solutions to SSCPLP.
        Select the r best solutions.

Improving Stage           (Intensification on elite solutions to SSCPLP)
for each solution generated in the first stage do
  for (l iterations) do
    begin
        Apply short-term memory component
                (Use TS to intensify on elite solutions to Allocation Subproblem)
        Perturb the r.h.s's
                (Use TS to diversify on Allocation Subproblem)
        Apply LS
                (Build elite solutions for SSCPLP)
end
```

Figure 14: Hybrid Algorithm 2

The inclusion of the LS at the improving stage of the algorithm has the following characteristics:

1. It has an important effect in terms of the number of elite solutions that are generated. Since the obtained solutions are local optima in the considered neighborhoods, they can be seen as elite solutions to the overall SSCPLP problem.
2. It makes the improving stage of the overall algorithm a hybrid procedure in itself. In the improving stage, the TS methodology is combined with a standard Local Search.
3. The improving phase is more flexible than the one of the TS heuristic of Figure 10. It is no longer true that the second phase operates uniquely on the Allocation Subproblem. After incorporating the LS to the second phase, this phase operates jointly on the plants' selection problem and on the Allocation Subproblem.

In the second hybrid approach the framework is provided by the Reactive GRASP Methodology. As shown in Figure 13, this is done with a variation of the inner iterations of the Reactive GRASP Algorithm presented in Figure 8:

Once more, the efficiency of the overall procedure can be considerably improved by only applying the cycle (intensification/diversification + LS) to the most promising

solutions. As before this is due to the cost of the repeated applications of the mentioned cycle. Here the selection of the r best solutions is not restricted to those associated with different sets of open plants. This reduces to some extent the level of diversification attained for the plants' selection problem. However, this increases notably the capability of the constructive phase to produce elite solutions. The r best solutions obtained with Reactive GRASP can safely be considered to be elite for the overall SSCPLP problem. This confidence in the quality of the solutions resulting from the constructive stage, permits one to limit the role of the improving stage, by reducing the number of cycles of intensification/diversification, which now will be denoted by l'. An outline of the second hybrid algorithm is shown in Figure 14.

In the two hybrid algorithms, the same test that is used in the TS heuristic of Figure 10 is applied to the solutions selected at the first stage. It is now possible to change the initial set of open plants during the second stage of the two hybrid heuristics. Hence, the test does no longer insure that the incumbent value could not improve during the second phase. However, experimentation indicates that this occurs very seldom. Thus, we consider that the increase on computation produced when applying the second stage to such a solution is not worthwhile.

8. COMPUTATIONAL EXPERIENCE

Two different sets of problems have been used to test the heuristic approaches presented here. They are publicly available in http://www-eio.upc.es/~elena. The first set contains the 33 problems used to test the heuristic approaches proposed in Delmaire *et al.* (1997) as well the Lagrangean relaxation approaches presented in Barceló, Fernández and Jörnsten (1991). It is divided into four classes, C_1, C_2, C_3 and C_4, according to the dimension, $(m \times n)$, of the problems. C_1 contains six (10×20) problems, C_2 contains eleven (15×30) problems, C_3 contains eight (20×40) problems and C_4 contains eight (20×50) problems. The test problems of this set have the following characteristics (see Barceló, 1985, for details):

1. Demands d_i are integers generated from a uniform distribution in the interval centered in D_{prom} that has $D_{\min}$ as lower limit where $D_{\min} = 10$ and $D_{prom} = 20$.
2. The capacities are integers generated from a uniform distribution in the interval centered in B_{prom} that has $B_{\min}$ as lower limit. $B_{\min}$ is fixed to D_{prom} and B_{prom} is set to $(\textit{Total demand})/(D_{coef}K)$. K is randomly generated in the interval $[\lfloor(0.1)n\rfloor, n]$ and represents an estimation of the number of plants to be opened. D_{coef} is an user provided parameter that represents the average ratio between the capacity of K plants and the total demand. Unfortunately, we could not obtain information about the values of D_{coef} that were used to generate the problems. However, the actual values of the capacities indicate that they are approximately 1.01.
3. Transportation costs are integers from a uniform distribution in the interval centered in C_{prom} that has $C_{\min}$ as lower limit where $C_{\min} = 0$ and $C_{prom} = 50$. They are computed as $c_{ij} = a_{ij} + r_{ij}d_i$ assuming that they have two components, the first one a_{ij}, a fixed component that depends on the plant, determined randomly and the second $r_{ij}d_i$ proportional to the demand.
4. Fixed costs are integers from an uniform distribution in the interval centered in f_{prom} that has $f_{\min}$ as lower limit. Their values are $f_{\min} = RB_{\min}$ and $f_{prom} = RB_{prom}$ being R a proportionality factor fixed by the user. Hence, fixed costs can be seen as proportional to capacity plus a random component, the proportionality ratio being fixed by the user. The values of the proportionality factor vary for the different data sets and range from 10 to 100.

A new generator with the characteristics described in Barceló (1985) has been implemented to generate the second set of test problems. These problems have been generated to evaluate the performance of the proposed methods on larger instances. The new set is divided into three classes of eight problems each: C_5 contains (30×60) problems, C_6 (30×75) problems and C_7 (30×90) problems. Now the value of D_{coef} has been set to 1.01. Within each class of problems, the value of the proportionality factor R is 10 for the first two problems, 30 for the next two ones, 60 for the following two ones and 90 for the last two ones.

The experiments corresponding to the proposed algorithms were performed on a VAX-2000 system with an alphaserver 2000 4/275 CPU and the results presented correspond to average values over 25 runs. The parameters' values that have been used for the different procedures are the following: In Reactive GRASP, to generate the different values of α, $0 \leq \alpha \leq 1$ the interval $[0, 1]$ has been divided in 10 equal-sized intervals; the fixed number of iterations for each α in the initialization phase is 20. The procedure terminates when the best solution does not improve after 10 consecutive iterations. In Tabu Search and in the first hybrid approach the number k of generated initial solutions is 300. The number r of selected sets of plants is 5 in TS and in the two hybrid approaches. The number of cycles in the improving phase is $\ell = 6$ both in TS and in the first hybrid approach. In the case of the second hybrid approach, the number of such cycles is $\ell' = 3$. Within the cycles of the improving phase, the values of the parameters are the same for TS and the two hybrid heuristics: 1) For the intensification phase applied to the Allocation Subproblem, the range for the tabu tenure values are, $T_{\min} = 7$ and $T_{\max} = 12$; the stopping criterion is 750 iterations without improvement. 2) The number of iterations in the diversification phase applied to the Allocation Subproblem, is set to 10.

To evaluate the quality of the proposed methods, the obtained results have been compared to the ones reported in Delmaire *et al.* (1997) and to the results of three series of experiments performed with the version 4.0 of the CPLEX software package. The experiments with CPLEX have been performed on a Sun Sparc Station 10/30 with 4 HyperSparc at 100 Mhz using only one processor. The first two series of experiments with CPLEX are designed to compare the quality of the solutions obtained with the proposed heuristics to that of the solutions generated by CPLEX within different time-limits. The last series of experiments is designed to try to find the optimal solutions to the problems. In the first two series of experiments with CPLEX problems were solved from scratch. The time-limits were set to ten minutes (600 seconds) of CPU time for the first series and two hours (7200 seconds) of CPU for the second series. In the third series of experiments, the value of the best solution found with the proposed heuristics was used as an incumbent value to speed up the tree search, and a time-limit of two hours (7200 seconds) of CPU was fixed. In all the experiments with CPLEX an "*.ORD*" file is used to assign higher priority to the plants' selection variables in the branching process. No priorities are set within the different plants. The default value of the *strategy branch* parameter was changed to 1 to take first the up branch at each node. The default value of the *strategy variableselect* parameter was changed to 3 to use a strong branching strategy that seems to be effective on large difficult MIP problems. For all the remaining parameters, the default values were used.

In the first series of runs, problems P3, P4, P6, P25, P32 and P41 could be solved within the time-limit of 600 seconds of CPU time. In the second series of runs, CPLEX succeeded in optimally solving from scratch all problems in the C_1 class, 5 out of the 11 problems in the C_2 class, 5 out of 8 problems in the C_3 class, 2 out of the 8 problems of the C_4 class, within the given time-limit of 7200 seconds of CPU. For the larger

problems in the second set, the number of problems that could be optimally solved from scratch in this time bound are 2 in the C_5 class, 2 in the C_6 class and 1 in the C_7 class. The third series of experiments, was performed with those problems that could not be optimally solved with the first two series of experiments. Now, problems were solved with the additional information provided by the incumbent value associated with the best solution found with the heuristics. In this series of experiments, CPLEX could optimally solve all the problems in the C_2 class. However, in this last series of runs, problems P18, P20 and P21 of the C_3 class and problems P26, P30, P31 and P33 of C_4 could not be optimally solved. In the second set of test problems, optimality could not be achieved for problems P36, P39 and P40 of the C_5 class, problems P44, P46, P47, P48 and P49 of the C_6 class and problems P51, P52, P53 and P55 of the C_7 class.

For each of the proposed methods three different measures have been considered: a) the value of the best solution it provides, b) its robustness, measured in terms of the average deviation from the value of the best known solution and, c) its efficiency in terms of the required computational times. Table 1 shows the best results obtained with the different algorithms proposed in this paper as well as the results corresponding to the approaches to which our methods are compared. The contents of the columns of Table 1 are the following: *GRASP* and *RGRASP*, the results of the GRASP heuristic of Delmaire *et al.* (1997) and the Reactive GRASP proposed in Section 4, respectively; *TS1* and *TS2*, the results of the TS heuristic of Delmaire *et al.* (1997) and the TS heuristic proposed in Section 5, respectively. *HB1* and *HB2* show the results of the two hybrid proposals described in Section 6. Finally, *CPLEX1*, *CPLEX2* and *CPLEX3* contain the results of the three series of experiments performed with CPLEX as described above. In *CPLEX3*, the cells marked with an asterisk correspond to values for which optimality could not be proved. The shadowed cells correspond to the values of the optimal/best-known solution for the different problems.

Reactive GRASP clearly outperforms the standard GRASP of Delmaire *et al.* (1997) since it obtains a better solution in 22 out of the 33 test problems of the first set and never does worse. As can be seen, this improvement in the performance seems to increase with the size of the problems: *RGRASP* improves the best solution of GRASP in 6 out of the 8 C_3 problems and in all the C_4 problems. The number of test problems of the first set for which the optimal/best known solution is found, increases from 4 in GRASP to 16 in *RGRASP*. In this sense, the behavior of *RGRASP* is similar to that of *TS1* which, so far, was considered to be the best approach. For the problems of the second set the behavior of *RGRASP* decreases considerably; it finds the optimal/best known solution in only 5 of the 24 problems. *TS1* is clearly outperformed by *TS2* that provides better-quality solutions in 16 out of the 33 test problems of the first set. Yet, for problem P14, *TS1* performs better than *TS2*. *TS2* provides the optimal/best-known solution for 25 of the 33 test problems of the first set and for 16 of the 24 test problems of the second set. Regarding the hybrid approaches, both *HB1* and *HB2* outperform *TS2* in the problems of the first set. *HB1* provides better solutions than *TS2* in 15 problems while *HB2* provides better solutions than *TS2* in 11 problems. *HB1* provides the optimal/best-known solution in all the 33 test problems of the first set and in 20 of the 24 problems of the second set which we consider remarkable. In all but four problems of the first set, *HB2* also generates the optimal/best known solution. In the case of the second set of test problems, *HB2* generates the optimal/best known solution in 20 of the 24 problems.

The results of Table 1 show that all the proposed methods, but specially the two hybrid heuristics, give high-quality solutions. In our opinion, the results obtained with CPLEX permit one to further appreciate the quality of the results obtained with the

Prb. No	GRASP	RGRASP	TS1	TS2	HB1	HB2	CPLEX1	CPLEX2	CPLEX3
1	2.014	2.014	2.014	2.014	2.014	2.014	2.054	2.014	2.014
2	4.289	4.269	4.269	4.251	4.251	4.251	4.347	4.251	4.251
3	6.061	6.051	6.051	6.051	6.051	6.051	6.051	6.051	6.051
4	7.168	7.168	7.168	7.168	7.168	7.168	7.168	7.168	7.168
5	4.567	4.551	4.567	4.551	4.551	4.551	5.018	4.551	4.551
6	2.269	2.269	2.271	2.269	2.269	2.269	2.269	2.269	2.269
7	4.372	4.372	4.372	4.372	4.366	4.366	4.639	4.366	4.366
8	7.943	7.926	8.055	8.055	7.926	7.926	8.487	8.160	7.926
9	2.496	2.496	2.490	2.480	2.480	2.480	2.961	2.751	2.480
10	23.120	23.112	23.112	23.112	23.112	23.112	24.702	23.141	23.112
11	3.471	3.471	3.469	3.447	3.447	3.447	3.692	3.479	3.447
12	3.716	3.716	3.711	3.711	3.711	3.711	3.760	3.711	3.711
13	3.775	3.760	3.760	3.760	3.760	3.760	4.238	3.760	3.760
14	6.118	6.118	6.065	6.118	5.965	5.970	5.998	5.965	5.965
15	7.832	7.830	7.816	7.816	7.816	7.816	8.042	7.987	7.816
16	11.561	11.543	11.543	11.543	11.543	11.543	12.089	11.881	11.543
17	9.895	9.895	9.920	9.884	9.884	9.884	10.011	9.884	9.884
18	15.616	15.607	15.624	15.615	15.607	15.607	16.868	16.528	15.607*
19	18.687	18.683	18.683	18.683	18.683	18.683	20.444	18.683	18.683
20	26.609	26.578	26.593	26.577	26.561	26.570	31.701	31.701	26.561*
21	7.320	7.295	7.318	7.301	7.295	7.295	8.070	7.764	7.295*
22	3.314	3.298	3.271	3.271	3.271	3.271	3.517	3.271	3.271
23	6.042	6.042	6.036	6.036	6.036	6.036	6.134	6.036	6.036
24	6.327	6.327	6.330	6.327	6.327	6.327	6.441	6.327	6.327
25	8.976	8.947	8.950	8.947	8.947	8.947	8.947	8.947	8.947
26	4.474	4.471	4.467	4.448	4.448	4.448	4.744	4.744	4.448*
27	10.928	10.921	10.921	10.921	10.921	10.921	11.020	10.921	10.921
28	11.132	11.119	11.117	11.117	11.117	11.117	11.838	11.279	11.117
29	9.837	9.832	9.832	9.832	9.832	9.832	10.177	10.169	9.832
30	10.963	10.961	10.939	10.845	10.835	10.845	11.871	11.555	10.835*
31	4.519	4.488	4.484	4.466	4.466	4.466	4.913	4.913	4.466*
32	9.907	9.897	9.891	9.881	9.881	9.881	9.881	9.881	9.881
33	39.605	39.553	39.578	39.501	39.477	39.478	41.939	41.757	39.477*
34	n.a.	4.705	n.a.	4.701	4.701	4.701	4.919	4.859	4.701
35	n.a.	5.463	n.a.	5.456	5.456	5.456	5.928	5.456	5.456
36	n.a.	16.785	n.a.	16.785	16.785	16.785	17.534	17.534	16.785 *
37	n.a.	14.680	n.a.	14.668	14.668	14.668	15.233	15.233	14.668
38	n.a.	47.284	n.a.	47.249	47.249	47.249	50.926	50.716	47.249
39	n.a.	41.035	n.a.	41.014	41.011	41.014	43.536	42.347	41.011*
40	n.a.	61.642	n.a.	61.642	61.633	61.642	64.600	64.539	61.633 *
41	n.a.	17.246	n.a.	17.246	17.246	17.246	17.246	17.246	17.246
42	n.a.	7.888	n.a.	7.887	7.887	7.887	8.553	7.887	7.887
43	n.a.	5.157	n.a.	5.114	5.114	5.114	5.407	5.407	5.114
44	n.a.	36.169	n.a.	36.082	36.050	36.050	38.065	38.065	36.050 *
45	n.a.	17.676	n.a.	17.676	17.676	17.676	18.064	17.676	17.676
46	n.a.	48.742	n.a.	48.720	48.712	48.730	51.248	51.248	48.712 *
47	n.a.	66.264	n.a.	66.230	66.230	66.230	69.006	69.006	66.230*
48	n.a.	58.970	n.a.	59.045	59.045	58.964	59.427	59.427	58.964 *
49	n.a.	79.652	n.a.	79.621	79.615	79.614	85.507	85.507	79.614 *
50	n.a.	5.940	n.a.	5.937	5.937	5.937	6.032	6.032	5.937
51	n.a.	9.124	n.a.	9.123	9.123	9.123	9.452	9.452	9.123 *
52	n.a.	34.692	n.a.	34.661	34.660	34.664	37.961	36.520	34.660 *
53	n.a.	30.051	n.a.	30.038	30.038	30.038	32.061	32.061	30.038 *
54	n.a.	43.856	n.a.	43.853	43.853	43.853	44.714	44.212	43.853
55	n.a.	69.697	n.a.	69.655	69.632	69.621	76.526	76.526	69.621 *
56	n.a.	64.481	n.a.	64.474	64.474	64.474	69.950	64.800	64.474
57	n.a.	49.791	n.a.	49.791	49.791	49.791	49.854	49.791	49.791

Table 1 Values of the Best Solutions

heuristics. Note that, for the problems in the first set, CPLEX has never been able to improve the quality of the overall best known solution found with the heuristic approaches. In particular, it has proven the optimality of the best solutions found with the heuristics in 26 out of the 33 test problems of the first set. For the remaining 7 problems, even when an incumbent value was provided and with a tremendous amount of computation time, it could not be proved that the best solution found with the heuristic methods is not optimal. For the second set of test problems, CPLEX could prove the optimality of the best solution found in 12 out of the 24 problems. For none of the other 12 test problems of the second set, it could improve the best solution found. Again, it has to be pointed out that for this, an incumbent value was supplied and an extremely large time limit was allowed. Although we assumed that some best-known solutions could be nonoptimal, the results of CPLEX show the difficulty in obtaining better quality solutions for the considered sets of problems.

However, we believe that the performance of an algorithm should not be uniquely measured in terms of the number of problems for which the optimal/best-known solution is found but also in terms of the robustness of the different approaches and the required computational times. In our opinion, when a series of runs are performed over the same data set, the second measure that we consider (mean deviation from the best known value) is a very accurate evaluator of the performance of an algorithm over the series of runs. When this measure takes very small values this implies not only that the studied algorithm occasionally provides high-quality solutions, but also (and more important) that most of the times the algorithm generates high quality solutions.

Next, we summarize the results relative to the robustness and computational times of the different algorithms. Within each group of problems, say q, the following values have been used:

- A_q: mean deviation from the best known value. It measures the robustness with regard to the quality of the solutions.
- T_q: average running time. It measures the efficiency of the algorithms.

For each class q, the average, A_q is obtained with the expression:

$$A_q = \Big(\sum_{j \in C_q} \mu_j \Big) / |C_q| \qquad q = 1, \ldots, 7.$$

where μ_j is the mean deviation from the best know value, $best_j$, for problem j:

$$\mu_j = \Big[\sum_{i=1}^{25} (solution_{ij} - best_j \Big] / 25,$$

and $solution_{ij}$ is the value of the best solution obtained for problem j in run i.

The average running time, T_q, is calculated with the expression:

$$T_q = \Big(\sum_{j \in C_q} \tau_j \Big) / |C_q| \qquad q = 1, \ldots, 7,$$

where τ_j is the mean time over the 25 runs for problem j, *i.e.*

$$\tau_j = \Big(\sum_{i=1}^{25} t_{ij} \Big) / 25.$$

and t_{ij} is the running time needed to obtain a solution for problem j during run i.

Figures 15 and 16 depict bar graphs with the mean deviations from the best known values for the compared approaches, for each class of problems, for the two sets of test

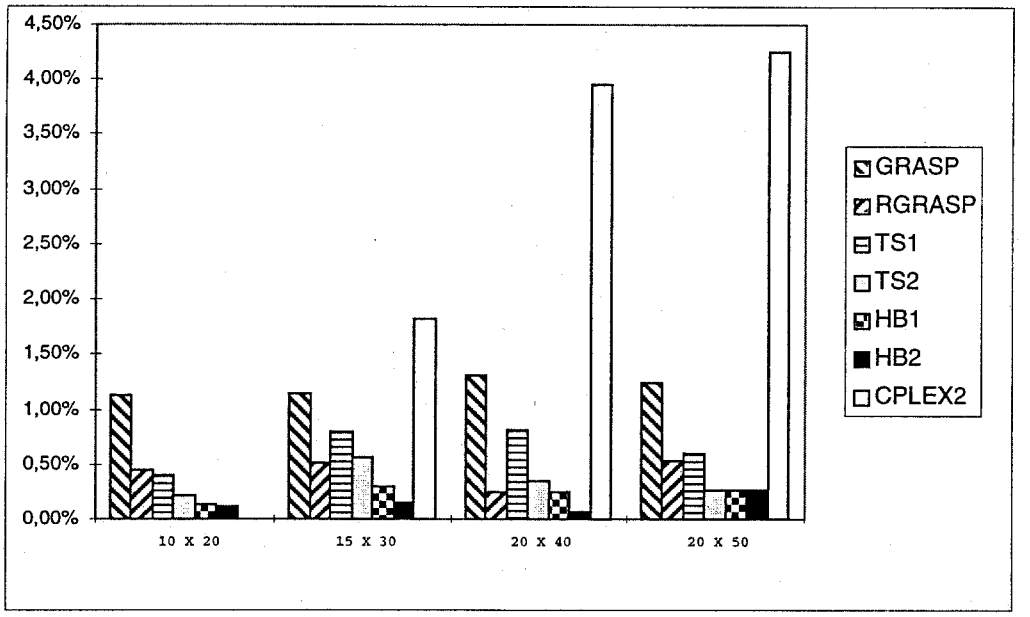

Figure 15: Average Percentage Deviation from Best Known Solution

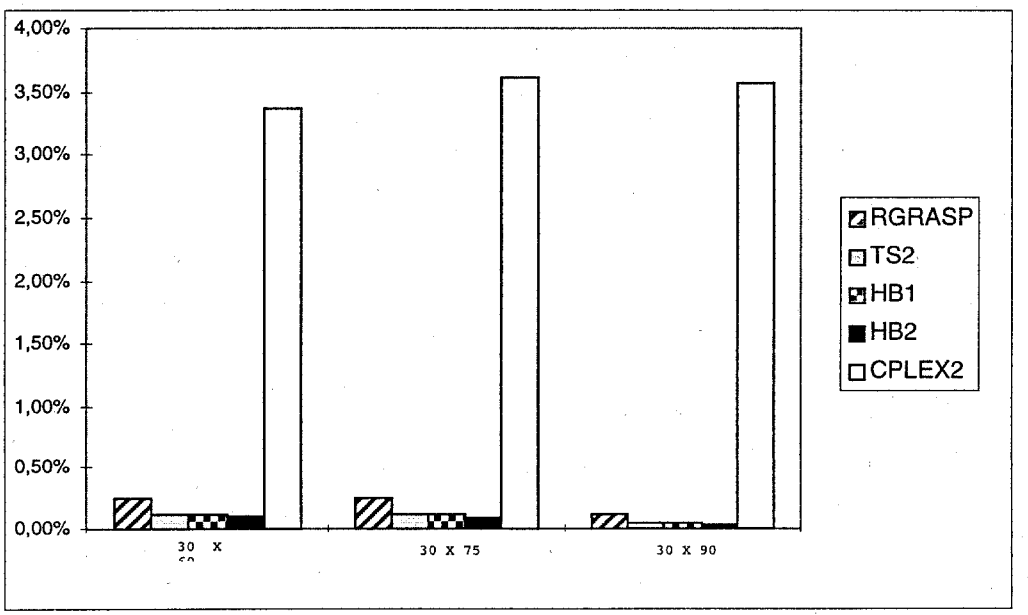

Figure 16: Average Percentage Deviation from Best Known Solution

Problem	GRASP		RGRASP		TS1		TS2	
Group	A_q	T_q	A_q	T_q	A_q	T_q	A_q	T_q
C_1	1.12%	0.09	0.44%	1.33	0.39%	2.95	0.21%	6.47
C_2	1.14%	0.22	0.51%	4.40	0.80%	6.55	0.56%	13.18
C_3	1.31%	0.38	0.25%	8.86	0.81%	13.27	0.34%	21.57
C_4	1.24%	0.58	0.55%	14.03	0.59%	19.75	0.27%	34.66

Problem	HB1		HB2		CPLEX1		CPLEX2	
Group	A_q	T_q	A_q	T_q	A_q	T_q	A_q	T_q
C_1	0.13%	6.65	0.12%	6.06	2.42%	600.00	0.00%	7200.00
C_2	0.30%	11.89	0.15%	13.68	6.38%	600.00	1.82%	7200.00
C_3	0.25%	20.88	0.07%	23.52	7.30%	600.00	3.96%	7200.00
C_4	0.27%	33.54	0.27%	36.20	5.42%	600.00	4.25%	7200.00

Table 2: Average Deviation from Best Known Values and Average Running Times for First Set of Test Problems

Problem	RGRASP		TS2		HB1	
Group	A_q	T_q	A_q	T_q	A_q	T_q
C_5	0.25%	21.30	0.11%	44.15	0.12%	40.42
C_6	0.24%	32.86	0.12%	59.14	0.11%	56.75
C_7	0.11%	40.35	0.05%	67.42	0.05%	65.26

Problem	HB2		CPLEX1		CPLEX2	
Group	A_q	T_q	A_q	T_q	A_q	T_q
C_5	0.10%	51.76	5.04%	600.00	3.37%	7200.00
C_6	0.08%	79.90	4.94%	600.00	3.61%	7200.00
C_7	0.03%	105.80	5.25%	600.00	3.57%	7200.00

Table 3: Average Deviation from Best Known Values and Average Running Times for Second Set of Test Problems

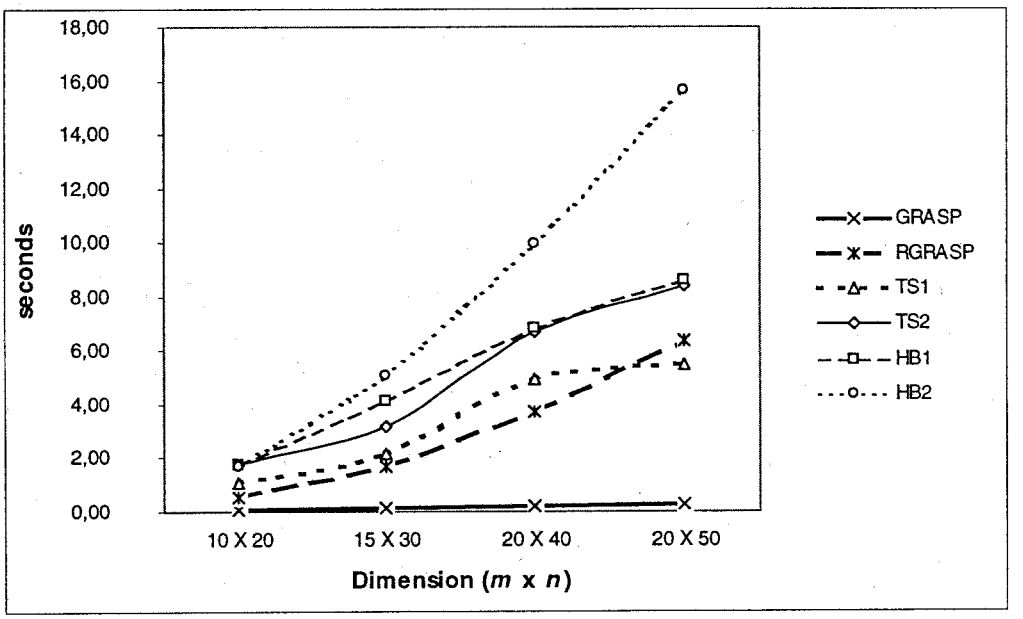

Figure 17: Average Time to Obtain Best Solution

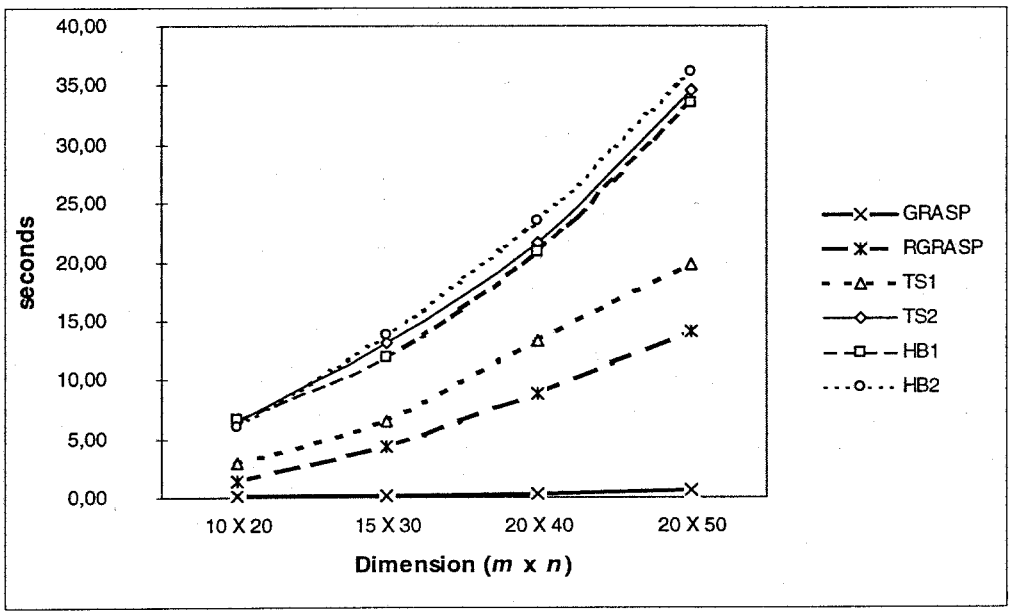

Figure 18: Average Total Running Time

problems. The actual values of the average deviations for the first set of test problems are depicted in Table 2; similarly, Table 3 shows the values for the second set of test problems.

As can be seen, the improvements in robustness obtained with our proposals are remarkable. For the reasons that have been explained above, we consider that the results of Figures 15 and 16 reflect the efficiency of the proposed algorithms better than

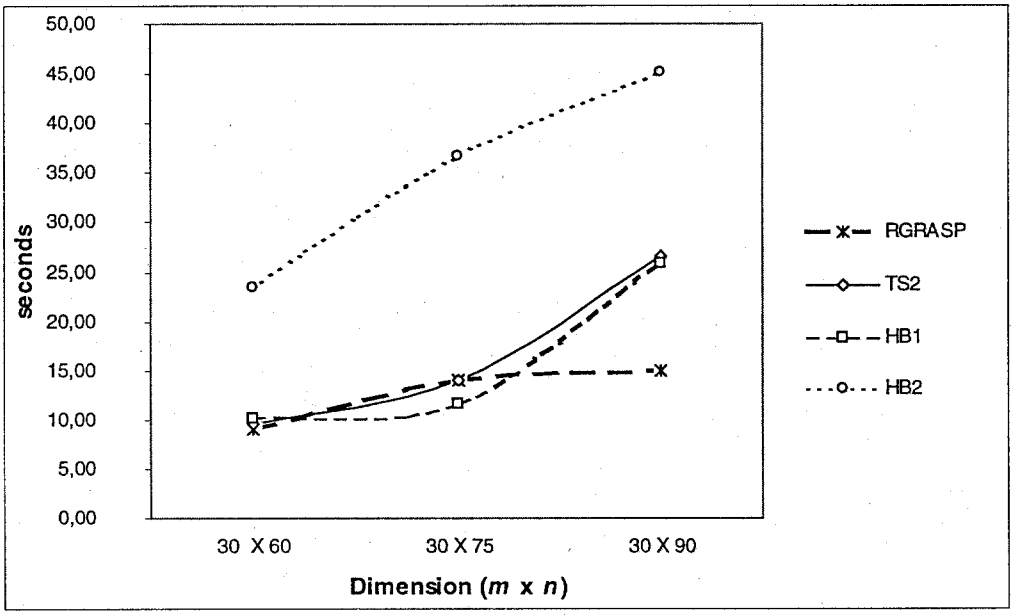

Figure 19: Average Time to Obtain Best Solution

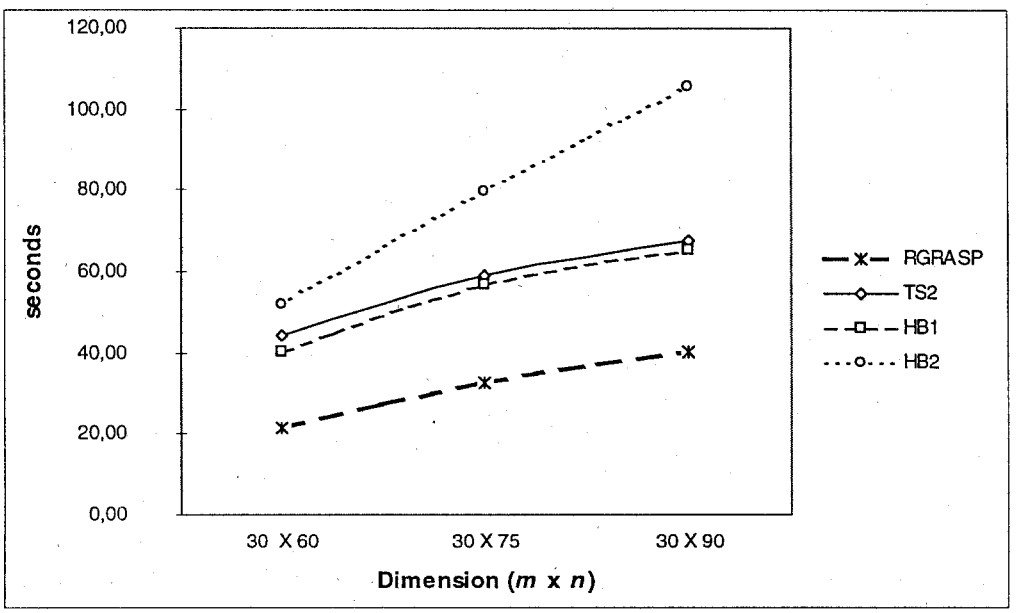

Figure 20: Average Total Running Time

those of Table 1. Moreover, these are efficient because the best known values, relative to which the average deviation has been measured, are either optimal or correspond to solutions that could not be improved by CPLEX on an enormous time limit. In fact, the results of Figures 15 and 16 confirm the interpretation given to the values of Table 1. For all the classes of test problems in the first set, *RGRASP* reduces the average deviation from the value of the best solution obtained with GRASP in, at least 50%. In particular, for the first set of test problems the average mean deviation of *RGRASP* never exceeds 0.5% while in GRASP it is always above 1%. Moreover, except for the C_1 class, the average deviations of *RGRASP* are better than those of *TS1*. In terms of the average deviation from the best known value, again *TS2* improves the performance of *TS1* for all classes of test problems of the first set. The obtained improvements range between 30% and 58%. This confirms that the strategies used in *TS2* are more appropriate for solving SSCPLP than those used in *TS1*. For the 4 classes of problems in the first set, the average deviation of *TS2* never exceeds 0.56%. Once more, the best results in terms of average behavior are provided by the hybrid approaches. In particular, in *HB1* the average deviation never exceeds 0.3% and, except for the C_4 class, the improvement with respect to *TS2* is over 25%, for the first set of problems. In the case of *HB2*, the average deviation from the best known value ranges from 0.07% to 0.27% for the test problems in the first set. Compared to *HB1*, the behavior of *HB2* is, on the average, similar for classes C_1 and C_4, but gives a reduction on the mean deviation from the value of the best-known solution of 50% for the C_2 class and of 72% for the C_3 class. The results relative to the second set of test problems, show the same tendency than in the first set of test problems. In particular, the average deviation from best known value never exceeds 0.25% in *RGRASP*, 0.12% in *TS2*, 0.12% in *HB1* and 0.10% in *HB2*. However it is somewhat surprising that the average deviation from the best known value be greater for the smaller test problems than the larger ones. This can only be interpreted in terms of the difficulty of the generated problems. Specifically, we think that these results reflect the fact that the generator of Barceló (1985) used to create the first set of test problems generates more difficult instances than the one that we have implemented. However, we have not found a satisfactory explanation for this difference in the two generators. Nevertheless, we consider that for both sets of test problems the obtained average deviations show the efficiency of the proposed methods. Additionally, Figure 16 shows that, for the second set of test problems, the proposed heuristics have an average deviation from the best known value which is at least 10 times smaller than the one of CPLEX.

Figures 17 – 20 show the average running times for each class of problems. Figures 17 and 19 give the average times required to obtain the best solution for each method, for the two set of test problems, while the times of Figures 18 and 20 correspond to the average total running times. The actual values of the required times for the two sets of problems are depicted in Tables 2 and 3. As can be seen in Figure 18, for the test problems in the first set, the improvement in the quality of the best found solutions and on the mean deviations implies, in some cases, a significant increase in running times. This is particularly true if the times required by *GRASP* are compared to those required by *RGRASP* or if the computation times of *TS1* are compared to those of *TS2*. Even though, in all the cases these times are still extremely small. For the 20×50 problems in the C_4 class (*i.e.* 1020 binary variables), the average computation time never exceeds 37 seconds. For the larger 30×90 instances, the average computation time raises a little over 100 seconds in *HB2*, which is the most time consuming heuristic. We consider this times very good for problems of this difficulty and sizes. Note that these times become insignificant compared to the ones required CPLEX, despite corresponding to

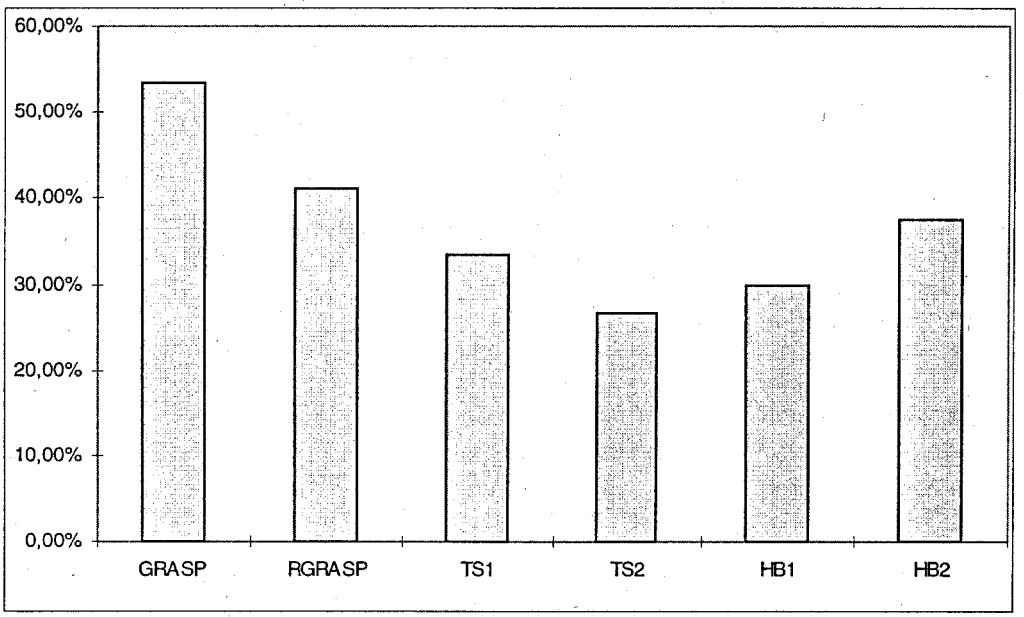

Figure 21: Average Percentage of Total Time to Obtain Best Solution (First Set of Test Problems)

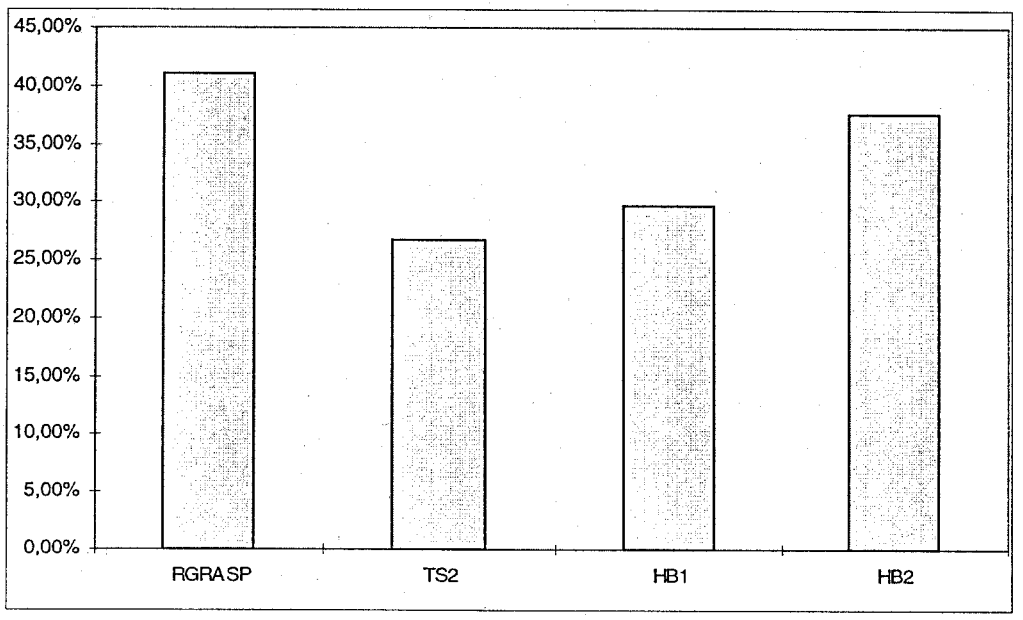

Figure 22: Average Percentage of Total Time to Obtain Best Solution (Second Set of Test Problems)

experiments performed on different computers. As could be expected, the heuristics that provide the best results are the ones that consume more CPU time. Note, however, that in the two sets of test problems, the running times required by *HB1* are, on the average, a little smaller than the ones of *TS2*. This may seem surprising, since *HB1* basically consists of *TS2* plus a LS applied at the inner iterations of the process. This reduction in times is due to the test that is applied to the selected solutions before entering the second stage. Since *HB1*, generates in general better quality solutions than *TS2*, usually at the same point of the procedure, the incumbent value is better in *HB1* than in *TS2*. Thus the mentioned test applied to the same set of selected solutions eliminates more solutions in *HB1* than in *TS2*. This is the cause for the obtained reduction in computation times.

For all the classes of test problems, *HB2* is the most time-consuming heuristic. For this procedure, the required computation times, can be seen as the sum of the times needed by *RGRASP* plus the times of a reduced Tabu Search phase applied at the second stage. For the first set of test problems, the times required by *HB2* are not significantly larger than the ones of *HB1* but, as the size of problems grows, the required computation times become much larger, relative to *HB1* (note that the times of *RGRASP* also become closer to the times of *TS2* for larger instances).

Finally, Figures 21 and 22 depict, for each set of test problems, the percentage of the total running time, required to obtain the best solution, on the average, for each of the compared methods. These two figures show that the criteria used as stopping rules are adequate since, on the average, the times needed to obtain the best solutions are below a 40% of the total running time for all the compared methods. This suggests that the compared methods could achieve higher performance in terms of the required computation times, if the number of iterations allowed in the different procedures were reduced. However, we consider that the overall running times are still small enough as not to have to resort to this possibility.

The outcome of the experiments carried out with CPLEX, permit one to judge the efficiency of the proposed methods. From these results several conclusions can be drawn:

1. The two sets of test problems contain instances that are very difficult to solve. Note that, when problems are solved from scratch, CPLEX usually fails to optimally solve the problems, even in a time limit of two hours of CPU. However, the larger instances of test problems in the second set seem to be somewhat easier than the smaller instances of the first set. This can be deduced since, for all the proposed methods, the average deviations from the best known values are better for the larger instances than for the smaller ones. The difference in the difficulty of the problems can be attributed to the fact that they have been created with different generators. Although the generator used to create the larger instances, was implemented according to the description of the generator of Barceló (1985) that created the smaller instances, there is some difference that, unfortunately, we cannot explain that makes the new problems easier to some extent than the ones of the first set.
2. All the proposed methods are in general more efficient than CPLEX for solving SSCPLP. Excepting for the smaller 10×20 problems, the average deviation from the best known value of all the proposed methods is always considerably smaller than the deviation from the best known value of the solutions obtained by CPLEX in two hours of CPU time. This means that, excepting the smaller instances, all the proposed methods produce, in general, better quality solutions than the ones obtained with CPLEX in very large time limits.

3. All the proposed methods, but specially *HB1* and *HB2*, produce high quality solutions. Remember that the best solution found with the proposed heuristics was optimal for 26 of the 33 test problems of the first set and for 12 of the 24 test problems of the second set. For the remaining problems CPLEX could not improve the value of the best solution found with the heuristic methods within a time limit of two hours, even when the value of the best know solution was used as an incumbent.
4. All the proposed methods are also efficient in terms of the required computation times. Note that compared to the times needed by CPLEX, the running times of the heuristics become insignificant, in spite of corresponding to experiments carried out in different computers.

Thus, from the results in the different figures and tables of this section, we can deduce that both *HB1* and *HB2* are very efficient for obtaining extremely high-quality solutions in small computation times. For smaller instances, *HB1* seems to give slightly higher quality solutions than *HB2*. For the larger instances they seem to perform equally well in terms of the quality of the generated solutions. However, *HB2*, seems to give better results in terms of the mean deviation from the best known solution, specially for larger instances. On the other hand, the required computation times of *HB1* are better than those of *HB2*, specially for larger instances. Thus, in our opinion, both *HB1* and *HB2* can be seen as equally attractive tools to obtain good quality solutions to SSCPLP in reduced computation times.

9. CONCLUSIONS

In this paper SSCPLP has been considered and several metaheuristic based methods have been proposed: a Reactive GRASP heuristic; a Tabu Search heuristic; and two hybrid approaches that combine the GRASP and the Tabu Search methodologies. The first hybrid heuristic is embedded within a TS framework and includes the Local Search phase of the GRASP algorithm within the inner iterations of the intensification/diversification cycle, as a means to produce elite solutions. The second hybrid approach is embedded within the framework of Reactive GRASP and applies a "reduced" Tabu Search to the best solutions obtained with Reactive GRASP.

Computational experiments have been performed over two different sets of test problems whose dimensions range from 10×20 to 30×90. The results obtained with the different approaches have been compared to the ones obtained with other known heuristics as well as to the ones obtained with three series of experiments carried out with CPLEX. The Reactive GRASP heuristic outperforms the standard GRASP heuristic of Delmaire *et al.* (1997), to which it has been compared, in terms of the quality of the solutions it provides and of the average deviation from the value of the best known solution. The average running times are very small, although they are higher than those of standard GRASP. Thus, this self-tuning procedure has proven to be a better alternative than standard GRASP for solving SSCPLP. The Tabu Search heuristic contains a diversification phase, in the procedure applied to the Allocation Subproblem, that consists of temporarily increasing the capacities of the open plants. This new diversification mechanism, together with the strategies used to manage the lists of candidate moves, have proven to be very effective since they produce a considerable improvement in the performance of the TS heuristic, as compared to the one of Delmaire *et al.* (1997), in terms of the best solutions found and the deviations from best known values. Again this produces a considerable increase in the average running times. However, taking into account the difficulty of the test problems, the average running times are still small.

Hence, the proposed Tabu Search heuristic can also be considered as a better alternative to solve SSCPLP than the one of Delmaire *et al.* (1997). The best results have been obtained with the two hybrid heuristics. Both of them outperform the Reactive GRASP heuristic as well as the proposed Tabu Search algorithm in terms of the quality of the solutions and the average deviations. The hybrid approach based on the TS methodology (*HB1*) also outperforms the proposed TS heuristic in terms of the required computation times. The most time consuming method, is the hybrid approach based on the Reactive GRASP methodology (*HB2*). It takes on the average 106 seconds for the larger 30×90 instances which can still be considered very good for problems of this size and difficulty. Hence, the two hybridation schemes have proven to be very effective for solving SSCPLP and it is difficult to establish a preference: the best solutions found with *HB1* are, in general, slightly better than those of the second hybrid heuristic but, on the average, *HB2* performs a little better in terms of the average deviation from the best known solutions.

ACKNOWLEDGEMENT

The rearch of H. Delmaire was partially supported by a Fellowship grant from Generalitat de Catalunya. The research of J.A. Díaz was partially supported by grant 110598 from Consejo Nacional de Ciencia y Tecnología, Méjico. The research of M. Ortega was partially supported by grant SAB-95-0474 from Dir. Gral. de Enseñanza Superior, España. The authors are grateful to Robert Garfinkel for his careful reading of the manuscript of this paper. Thanks are due to the referees for their valuable comments.

REFERENCES

1. Balas, E. 1982. A class of location, distribution and scheduling problems. Modeling and solution methods, *Revue Belge de Statistique, d'Informatique et de Recherche Opérationelle*, 22, 2, 36-65.
2. Barceló, J. and J. Casanovas. 1984. A heuristic lagrangean algorithm for the capacitated plant location problem, *European Journal of Operational Research*, 15, 212-226.
3. Barceló, J. 1985. Computational experiments with location problems using automatic constraint generation, Working paper RR85/06 FIB, Universitat Politècnica de Catalunya.
4. Barceló, J., A. Hallefjord, E. Fernández, and K. Jörnsten. 1990. Lagrangean relaxation and constraint generation procedures for capacitated plant location problems with single sourcing, *Operations Research Spektrum*, 12, 79-88.
5. Barceló, J., E. Fernández, and K. Jörnsten. 1991. Computational results from a new lagrangean relaxation algorithm for the capacitated plant location problem, *European Journal of Operational Research*, 53, 38-45.
6. Battiti, R., and G. Tecchiolli. 1994. The reactive Tabu Search, *ORSA Journal on Computing*, 6, 2, 126-140.
7. Battiti, R., and G. Tecchiolli. 1995. Local Search with memory: Benchmarking RTS, *Operations Research Spektrum*, 17, 2/3, 67-86.
8. Battiti, R. 1996. Reactive Search: Towards Self-Tuning Heuristics, in *Modern Heuristic Search Methods*, chapter 4, 61-83, V.J. Rayward-Smith, I.H. Osman, C.R. Reeves and G.D. Smith (eds.), John Wiley and Sons.
9. Beasley, J.E. 1988. An algorithm for solving large capacitated warehouse location problems, *European Journal of Operational Research*, 33, 314-325.
10. Beasley, J.E. 1990. Lagrangean heuristics for location problems, Working paper, Management School, Imperial College.
11. Delmaire, H., J. A. Díaz, E. Fernández and M. Ortega. 1997. Comparing New Heuristics for the Pure Integer Capacitated Plant Location Problem, Working paper DR97/10 Dpt.EIO, Universitat Politècnica de Catalunya.
12. Devine, M.D. and W.G. Lesso. 1972. Models for the minimum cost development of offshore oil fields, *Management Science*, 18, 8, 378-387.
13. Feo, T.A. and M.G.C. Resende. 1995. Greedy randomized adaptive search procedures, *Journal of Global Optimization*, 6, 109-133.
14. Gendreau, M., G. Laporte and R. Séguin. 1996. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers, *Operations Research*, 3, 44, 469-477.
15. Glover, F. 1989. Tabu Search-Part I, *ORSA Journal on Computing*, 3, 1, 190-206.

16. Glover, F. 1990. Tabu Search-Part II, *ORSA Journal on Computing*, 1, 2, 4-32.
17. Glover, F. and M. Laguna. 1997. *Tabu Search*, Kluwer Academic Publishers, 1997.
18. Guignard, M. and K. Opaswongkarn. 1990. Lagrangean dual ascent algorithms for computing bounds in capacitated plant location problems, *European Journal of Operational Research*, 46, 73-83.
19. Holland, J.H. 1975. *Adaptation in Natural and Artificial Systems*, University of Michigan Press.
20. Jacobsen, S.K. 1983. Heuristics for the capacitated plant location model, *European Journal of Operational Research*, 12, 253-261.
21. Klincewicz, J.G. and H. Luss. 1986. A Lagrangean Relaxation Heuristic for Capacitated Facility Location with Single-Source Constraints, *Journal of the Operational Research Society* 37, 495-500.
22. Kuehn, A.A. and M.J. Hamburger. 1963. A heuristic program for locating warehouses, *Management Science*, 9, 645-666.
23. Michalewicz, Z. 1992. *Genetic Algorithms + Data Structures = Evolution Programs*, Springer.
24. Mirchandani, P.B. and R.L. Francis (eds.). 1990. *Discrete Location Theory*, Wiley.
25. Neebe, A.W. and M.R. Rao. 1983. An algorithm for the fixed-charge assigning users to sources problem, *Journal of the Operational Research Society* 34, 1107-1113.
26. Pirkul, H. 1987. Efficient algorithms for the capacitated concentrator location problem, *Computers and Operations Research* 14, 197-208.
27. Prais, M. and C.C. Ribeiro. 1997. Reactive GRASP: An Application to a Matrix Decomposition Problem in TDMA Traffic Assignment, Second Metaheuristic International Conference, Sophia-Antipolis.
28. Sridharan, R.. 1995. The capacitated plant location problem, *European Journal of Operational Research*, 87, 203-213.
29. Yaguira, M., T. Yamaguchi and T. Ibaraki. 1997. A variable Depth search Algorithm for the Generalized Assignment Problem, Second Metaheuristic International Conference, Sophia-Antipolis.

Hugues Delmaire obtained his Ph.D. from the Ecole Polytechnique de Montréal in 1996. He received a Fellowship grant from the Generalitat de Catalunya in Spain for a Postdoctoral stay at the Universitat Politècnica de Catalunya in 1997. He then worked as mathematician for the company Mid Ocean Reinsurance in Bermuda.

Juan A. Díaz is a Ph.D. student in the Department of Statististics and Operations Research at the Universitat Politècnica de Catalunya. His research interest is focused on metaheuristic methods for discrete optimization problems such as generalized assignment and discrete location problems.

Elena Fernández is a Professor in the Department of Statistics and Operations Research at the Universitat Politècnica de Catalunya. Her research interest focuses on discrete optimization problems such as multiknapsack and discrete location problems.

Maruja Ortega is a Professor in the Department of Computer Science at Universidad Simón Bolívar in Caracas, Venezuela. Her main research interest is in Graph Algorithms and Discrete Optimization. The present work was developped while on sabbatical at the Universitat Politècnica de Catalunya.