

A Hybrid Algorithm for Dynamic Location Problems

JOANA DIAS⁽¹⁾, M. EUGÉNIA CAPTIVO⁽²⁾ * AND JOÃO CLÍMACO⁽¹⁾

(1)Faculdade de Economia and INESC-Coimbra

Universidade de Coimbra

Av. Dias da Silva, 165

3004-512 Coimbra

Portugal

(2) Universidade de Lisboa, Faculdade de Ciências

Centro de Investigação Operacional

Campo Grande, Bloco C6, Piso 4

1749-016 Lisboa

Portugal

Abstract: In this research report a hybrid algorithm integrating genetic procedures and local search will be described which is able to solve capacitated and uncapacitated dynamic location problems. These problems are characterized by explicitly considering the possibility of a facility being open, closed and reopen more than once during the planning horizon. It is also possible to explicitly consider different open and reopen fixed costs. The Decision Maker (DM) can include additional restrictions in the proposed model. The algorithm developed is prepared to solve both mono and multi-objective location problems. In the latter case, the DM has to interact with the algorithm, indicating desired search areas in the objective space.

Keywords: *location problems, genetic algorithms, local search, multi-objective.*

1 Introduction

When faced with a hard combinatorial optimisation problem, the first obvious question that has to be answered is which algorithm(s) should be used to find the optimal solution. Usually, the best approach to solve a problem depends strongly on the specific problem, and even on the specific instance of the problem at hand, and sometimes the best results are obtained by combining different approaches (Toth, 2000). Before answering this question, it is equally important to answer another one: is it necessary to find the optimal solution or is it sufficient to

* This research was partially supported by research project POCTI/ISFL-1/152 and POCTI/MAT/139/2001.

find good (non-optimal) quality solutions? And how much time can we afford to spend on the pursuing of an optimal or near optimal solution? These questions are even harder to answer when dealing with a multi instead of a mono-objective decision-making context.

Methods like branch and bound have long been used to find optimal solutions to combinatorial problems. Their main advantage is that they can calculate the optimal solution. Nevertheless this is generally achieved at the cost of an intensive and extensive use of resources like time and memory storage. When faced with really hard combinatorial optimisation problems, the use of exact methods becomes, most of the times, impracticable (in spite of the increase in computing power and hence speed of calculation). Quoting Goldberg, 1989, “...convergence to the best is not an issue in business or in most walks of life; we are only concerned with doing better relative to others. (...) Attainment of the optimum is much less important for complex systems. It would be nice to be perfect, meanwhile we can only strive to improve”.

Heuristic methods try to calculate good solutions, without guaranteeing their optimality, but offering good compromises between solution quality, computational time and storage. Reeves, 1993b, gives the following definition: *A heuristic is a technique which seeks good (i.e. near optimal) solutions at reasonable computational cost without being able to guarantee either feasibility or optimality, or even, in many cases, to state how close to optimality a particular feasible solution is.* In the location problems’ field, much work has been done in the development of heuristic methods (see, for instance, Jacobsen, 1983; Barceló and Casanovas, 1984; Domschke and Drexl, 1985; Klineciewicz and Luss, 1986; Tcha et al, 1988; Cornuejols et al, 1991; Sridharan, 1991, 1993; Galvão and Santibañez-Gonzalez, 1992; Beasley, 1993; Hansen et al, 1994; Klose, 1995, 1999; Salhi and Atkinson, 1995; Holmberg and Ling, 1997; Tragantalerngsak et al, 1997; Agar and Salhi, 1998; Pirkul and Jayaraman, 1998; Saldanha da Gama and Captivo, 1998; Rönnqvist et al, 1999; Rosing and Hodgson, 2002; Wu et al, 2002, Espejo et al, 2003, Levin and Ben-Israel, 2004).

Metaheuristics are general combinatorial optimisation techniques, designed with the aim of solving as many different combinatorial optimisation problems as possible (Hertz and Widmer, 2003). Metaheuristics are naturally discrete. They have a number of disadvantages, one of them being the fact that it is generally necessary to tune several parameters (Jones et al, 2002), but have proven their capability of calculating high quality solutions for complex problems. However the design of a good metaheuristic remains an art (Osman and Kelly, 1996). Examples of widely used metaheuristics are tabu search, simulated annealing, scatter search, evolutionary computing (including genetic algorithms, evolution strategies, evolutionary

programming and genetic programming – Eiben and Smith, 2003)¹. They have been widely used in solving combinatorial problems in general (see, for instance, Reeves, 1993b and Taillard et al, 2001), and location problems in particular.

Alves and Almeida, 1992, use simulated annealing to solve simple plant location problems. Houck et al, 1996, use genetic algorithms to solve large location-allocation problems, where each location is a point in a continuous two-dimensional space. Rolland et al, 1996, apply a tabu search procedure to the p -median problem. Kratica et al, 1996, 2001 and Kratica, 1999, use simple genetic algorithms to solve simple location problems, and propose the hybridisation of the genetic algorithm with an ADD-heuristic to improve its performance. Filipovic et al, 2000, improves the performance of a genetic algorithm that solves simple plant location problems by applying a grained tournament selection operator. Vaithyanathan et al, 1996, develop a neural network combined with tabu search algorithm for combinatorial problems and applied it to the simple plant location problem. Lorena and Lopes, 1997, apply genetic algorithms to computationally difficult set covering problems. Bornstein and Azlan, 1998, study the capacitated plant location problem and use simulated annealing to calculate the optimal values of location variables whose values was not possible to set through the use of reduction tests. Owen and Daskin (1998a, b) use evolution programs to solve strategic facility location problems (in a scenario planning context). Sun et al, 1998, develop a tabu search heuristic for the fixed charge transportation problem, and report some computational results. Abdinnour-Helm, 1998, describes a hybrid heuristic that uses genetic algorithms to solve the location problem and tabu search to solve the assignment problem for the uncapacitated hub location problem. The idea of considering two different processes of optimisation (one for location and another for the allocation problem) is also present in the work of Righini, 1995. The author describes a double annealing algorithm that works with two mutually dependent variable sets. Jaramillo, 1998, and Jaramillo et al, 2002, study genetic algorithms as an alternative procedure to generate optimal or near-optimal solutions for location problems. The authors study the capacitated and uncapacitated fixed charge location problems, the maximum covering problem and competitive location problems, and conclude that genetic algorithms should not be adopted for solving capacitated fixed charge problems. Filho and Galvão, 1998, develop a tabu search heuristic for the concentrator location problem. Rosing et al, 1999, describe a two-stage metaheuristic that is particularly suited to solve location problems like p -median, where the number of facilities is given in advance. Maniezzo et al, 1998, propose a bionomic heuristic as an effective method to solve the p -median problem and Alp et al, 2003,

¹ For information on several metaheuristics see, for instance, Glover and Kochenberger, 2003.

propose an efficient and simple genetic algorithm for the same problem. Shimizu and Wada 2003, formulate a site location and route selection problem as a capacitated p -hub problem, and develop a hybrid tabu search algorithm. Shimizu, 1999, combines genetic algorithms with mathematical programming and considers the problem of locating a hazardous waste disposal plant problem under multiple objectives. Correa et al, 2001, models a real-world problem (the selection of facilities for a university's admission examinations) as a capacitated p -median location problem and develops a genetic algorithm hybridised with a heuristic approach. The heuristic is responsible for determining admissible allocations. Antunes and Peeters, 2001, apply simulated annealing to a real-world multi-period location problem. Cortinhal and Captivo, 2003, study the total assignment capacitated location problem using genetic algorithms.

In the present research report, we will describe an algorithm that hybridises genetic algorithms and local search, and that is able to solve dynamic capacitated and uncapacitated location problems, with opening, closing and reopening of facilities. The authors have already studied this problem and developed several efficient primal-dual heuristics² (Dias et al, 2004a, b). The primal-dual heuristics developed calculate good quality solutions as well as lower bounds on the optimal objective function value. Nevertheless, their structure makes it difficult and time consuming to adapt these heuristics to even minor changes in the problem formulation. Thus, we can conclude that the previous developed heuristics are not particularly suited for situations where the DM wishes to introduce additional restrictions to the model or when the DM considers explicitly more than one objective. These observations motivated the development of metaheuristics, in particular genetic algorithms, to solve dynamic location problems.

This research report is organized as follows: in the next section we present the problem, in section 3 the main characteristics of our algorithm are described, in section 4 the introduction of additional restrictions is considered, in section 5 the usage of the algorithm is extended to a multi-objective context and, finally, in section 6 we point out some conclusions and possible future work.

We consider that the reader is familiarized with evolutionary algorithms, location problems and multi-objective programming.

² For the uncapacitated case a branch and bound procedure based on the heuristic was also developed, that guarantees the calculation of the optimal solution.

2 The Dynamic Location Problem

Consider the following notation:

$J = \{1, \dots, j, \dots, n\}$ set of indexes corresponding to the clients' locations;

$I = \{1, \dots, i, \dots, m\}$ set of indexes corresponding to facilities' possible locations;

$T =$ number of time periods considered in the planning horizon ($1 \leq t \leq \xi \leq T$);

$c_{ij}^t =$ cost of fully assigning client j to facility i in period t ;

$FA_{it}^\xi =$ fixed cost of opening a facility i at the beginning of period t , and closing it at the end of period ξ (the facility will be in operation from the beginning of t to the end of ξ);

$FR_{it}^\xi =$ fixed cost of reopening a facility i at the beginning of period t , and closing it at the end of period ξ (the facility will be in operation from the beginning of t to the end of ξ);

$d_j^t =$ demand of client j at period t ;

$Q_i =$ maximum capacity of a facility located at i .

$Q'_i =$ minimum capacity of a facility located at i .

and let us define the variables:

$$a_{it}^\xi = \begin{cases} 1 & \text{if facility } i \text{ is open at the beginning of period } t \text{ and stays open until the end} \\ & \text{of period } \xi \\ 0 & \text{otherwise} \end{cases}$$

$$r_{it}^\xi = \begin{cases} 1 & \text{if facility } i \text{ is reopen at the beginning of period } t \text{ and stays open until the end} \\ & \text{of period } \xi \\ 0 & \text{otherwise} \end{cases}, t > 1$$

$x_{ij}^t =$ fraction of customer j 's demand that is served by facility i during period t .

In our model we consider possible that a facility is open, closed and reopen more than once during the planning horizon. The fixed open and reopen costs can be quite different and the model explicitly considers that difference. We defined three different types of capacity restrictions: maximum capacity restrictions, minimum capacity restrictions and maximum decreasing capacity restrictions.

Maximum and minimum capacity restrictions establish lower and upper bounds on the total flow that reaches a facility in each time period. Generally a facility should not operate

under a minimum threshold (for economic reasons, for instance), and has physical limitations that impose maximum limits to the number of clients it can serve.

Maximum decreasing capacity restrictions deal with a special kind of facilities that have a certain maximum capacity when they are (re) open. This maximum capacity diminishes as the facility serves clients. Examples of such facilities are, for instance, sanitary landfills that are open with a maximum capacity that diminishes, as waste is disposed, or a warehouse that has an initial stock that is “consumed” by clients³.

Instead of considering that, conceptually, all facilities are composed of a single element, we can also consider that a facility is composed of several elements that can be of different dimensions⁴. If this is the case, it is possible to increase (decrease) the maximum capacity of an already open facility by locating a new element (closing an existing element). Examples of facilities that have this kind of behaviour are found, for instance, in the urban waste treatment systems, where transfer stations can be constituted by one or more elements⁵.

We can now develop a general model that considers four different types of facilities:

- 1- uncapacitated facilities;
- 2- facilities with maximum and/or minimum capacity restrictions⁶;
- 3- facilities with maximum decreasing capacity restrictions;
- 4- facilities composed of one or more elements that can be of different dimensions.

Consider the additional notation, necessary for facilities of type 4:

Q_s = maximum capacity of a facility of dimension s ⁷;

$Nmax$ = maximum number of elements that can be operational simultaneously at facility i of type 4;

³ In this case the (re) opening of a warehouse can be interpreted as supply of goods that are treated as being of a single kind.

⁴ Elements of different dimensions mean elements with different maximum capacities.

⁵ There are some papers in the literature that address location problems of facilities similar to these: Luss, 1982; Min, 1988; Shulman, 1991.

⁶ In this model it is considered that the maximum and minimum capacities of a facility located at i will remain constant during the planning horizon. This means that the maximum and minimum capacities of a given facility are the same during all its operating periods. It is also possible to consider that these maximum and/or minimum capacities can change over the planning horizon. In this case capacities Q_i^t and Q'_i^t should be considered. This change can easily be incorporated in the model and in the procedures that are going to be presented.

⁷ In this case we are not considering minimum capacities, but the model and the procedures developed could be easily changed to accommodate their existence.

$S = \{1, \dots, q\}$ set of indexes corresponding to elements' possible dimensions, ordered by ascending order of the corresponding capacities;

c_{ijs}^t = cost of fully assigning client j to an element of dimension s located at i in period t ;

FA_{ist}^ξ = fixed cost of opening an element of dimension s at facility i of type 3 at the beginning of period t , and closing it at the end of period ξ (knowing that this is the first element to be located at i);

FR_{ist}^ξ = fixed cost of locating one element of dimension s at i at the beginning of period t , and closing it at the end of period ξ (knowing that at least one element has been previously located at i).⁸

And also the additional decision variables:

$$a_{ist}^\xi = \begin{cases} 1 & \text{if an element of dimension } s \text{ located at } i \text{ is open at the beginning of period } t \\ & \text{and stays open until the end of period } \xi, \text{ knowing that this is the first} \\ & \text{element to be located at } i \\ 0 & \text{otherwise} \end{cases}$$

r_{ist}^ξ = number of elements of dimension s located at i at the beginning of period t and staying open until the end of period ξ , knowing that there has been already at least one element located at i ;

x_{ijs}^t = fraction of customer j 's demand that is served by an element of dimension s located at facility i during period t .

Notice that it is possible to distinguish not only different open and reopen fixed costs but also different assignment costs, depending on the dimension of the element (generally, elements of greater dimension have smaller operating costs).

We can define four different sets:

$$I_1 = \{i \in I \text{ and } i \text{ is of type 1}\};$$

$$I_2 = \{i \in I \text{ and } i \text{ is of type 2}\};$$

$$I_3 = \{i \in I \text{ and } i \text{ is of type 3}\};$$

$$I_4 = \{i \in I \text{ and } i \text{ is of type 4}\}.$$

⁸ The fixed cost FA_{ist}^ξ should be equal to FR_{ist}^ξ plus the additional cost of installing for the first time an element at location i . This additional cost may represent costs of land acquisition, development of infra-structures, etc. Let us define this additional cost as f_i^t . Then $FA_{ist}^\xi = FR_{ist}^\xi + f_i^t, \forall i, s, t, \xi \geq t$.

The capacitated dynamic location problem can be formulated as CDLPOCR:

CDLPOCR

$$\begin{aligned} \text{Min} \sum_t \sum_{i \in I_4} \sum_j \sum_s c_{ijs}^t x_{ijs}^t + \sum_t \sum_{i \in I_4} \sum_s \sum_{\xi=t}^T FA_{ist}^\xi a_{ist}^\xi + \sum_t \sum_{i \in I_4} \sum_s \sum_{\xi=t}^T FR_{ist}^\xi r_{ist}^\xi + \\ \sum_t \sum_{i \in I/I_4} \sum_j c_{ij}^t x_{ij}^t + \sum_t \sum_{i \in I/I_4} \sum_{\xi=t}^T FA_{it}^\xi a_{it}^\xi + \sum_t \sum_{i \in I/I_4} \sum_{\xi=t}^T FR_{it}^\xi r_{it}^\xi \end{aligned} \quad (1)$$

subject to:

$$\sum_{i \in I/I_4} x_{ij}^t + \sum_{i \in I_4} \sum_s x_{ijs}^t = 1, \quad \forall j, t \quad (2)$$

$$\sum_{\tau=1}^t \sum_{\xi=\tau}^T (a_{i\tau}^\xi + r_{i\tau}^\xi) - x_{ij}^t \geq 0, \quad \forall i \in I/I_4, j, t \quad (3)$$

$$\sum_{\tau=1}^t \sum_{\xi=\tau}^T (a_{is\tau}^\xi + r_{is\tau}^\xi) - x_{ijs}^t \geq 0, \quad \forall i \in I_4, j, s, t \quad (4)$$

$$\sum_{\tau=1}^{t-1} \sum_{\xi=\tau}^{t-1} a_{i\tau}^\xi - \sum_{\xi=t}^T r_{it}^\xi \geq 0, \quad \forall i \in I/I_4, t \quad (5)$$

$$Nmax \sum_{s'} \sum_{\tau=1}^t \sum_{\psi=\tau}^T a_{is'\tau}^\psi - r_{ist}^\xi \geq 0, \quad \forall i \in I_4, s, t, \xi \geq t \quad (6)$$

$$\sum_{t=1}^T \sum_{\xi=t}^T a_{it}^\xi \leq 1, \quad \forall i \in I/I_4 \quad (7)$$

$$\sum_s \sum_{t=1}^T \sum_{\xi=t}^T a_{ist}^\xi \leq 1, \quad \forall i \in I_4 \quad (8)$$

$$\sum_{\tau=1}^t \sum_{\xi=\tau}^T (a_{i\tau}^\xi + r_{i\tau}^\xi) \leq 1, \quad \forall i \in I/I_4, t \quad (9)$$

$$\sum_s \sum_{\tau=1}^t \sum_{\xi=\tau}^T (a_{is\tau}^\xi + r_{is\tau}^\xi) \leq Nmax, \quad \forall i \in I_4, t \quad (10)$$

$$Q_i \sum_{\tau=1}^t \sum_{\xi=\tau}^T (a_{i\tau}^\xi + r_{i\tau}^\xi) - \sum_j d_j^t x_{ij}^t \geq 0, \quad \forall i \in I_2, t \quad (11)$$

$$\sum_j d_j^t x_{ij}^t - Q_i \sum_{\tau=1}^t \sum_{\xi=\tau}^T (a_{i\tau}^\xi + r_{i\tau}^\xi) \geq 0, \quad \forall i \in I_2, t \quad (12)$$

$$Q_i \sum_{\tau=1}^t \sum_{\xi=\tau}^T (a_{i\tau}^\xi + r_{i\tau}^\xi) - \sum_{\tau=1}^t \sum_j d_j^\tau x_{ij}^\tau \geq 0, \quad \forall i \in I_3, t \quad (13)$$

$$Q_s \sum_{\tau=1}^t \sum_{\xi=\tau}^T (a_{is\tau}^\xi + r_{is\tau}^\xi) - \sum_j d_j^t x_{ijs}^t \geq 0, \quad \forall i \in I_4, s, t \quad (14)$$

$$\begin{aligned} a_{it}^\xi &\in \{0,1\}, & \forall i \in I / I_4, t, \xi \geq t \\ r_{it}^\xi &\in \{0,1\}, & \forall i \in I / I_4, t > 1, \xi \geq t \end{aligned} \quad (15)$$

$$\begin{aligned} a_{ist}^\xi &\in \{0,1\}, & \forall i, s, t, \xi \geq t \\ r_{ist}^\xi &\geq 0 \text{ and integer}, & \forall i, s, t, \xi \geq t \end{aligned} \quad (16)$$

The objective function considers the minimization of all fixed and assignment costs. Constraints (2) guarantee that, in every time period, each client's demand is satisfied; constraints (3) and (4) assure that, in every time period, a client can only be assigned to facilities that are operational in that time period; constraints (5) and (6) impose that a facility can only be reopened at the beginning of period t if it has been open earlier and, for $i \notin I_4$, i is not in operation at the beginning of period t ; constraints (7) and (8) guarantee that a facility can only be open once during the planning horizon; constraints (9) and (10) assure that, in every time period, only one facility of types 1 to 3 or N_{max} elements of type 4 can be open in each location; constraints (11) and (14) guarantee that the facilities' maximum capacity will not be exceeded in any time period; constraints (12) are the minimum capacity constraints for facilities of type 2; constraints (13) are the maximum capacity constraints for facilities of type 3.

It is interesting to note that, depending on the type of facilities present, after fixing a set of feasible values for the location variables $a_{i\tau}^\xi$, $r_{i\tau}^\xi$, $a_{is\tau}^\xi$ and $r_{is\tau}^\xi$ it is rather simple to calculate the optimal allocation variables in each time period:

- 1- If all facilities are uncapacitated then each client is assigned to exactly one facility (the cheapest one);
- 2- If all facilities are of type two, then it is necessary to solve a transportation problem. The transportation problem will have $2m$ destinations and $n+1$ sources. The $(n+1)$ -th source is fictitious and its supply is used to balance the problem. Destinations 1 to m correspond to the m

facilities, and will have demand equal to zero if the facility is closed during period t and equal to the maximum minus minimum capacity if the facility is open during t . Destinies i' between $m+1$ and $2m$ will have demands equal to zero if the corresponding facility $i = i' - m$ is closed during t , and demands equal to the facility's minimum capacity otherwise. The costs of assigning origin j to destinies i and $i+m$ are the same, for all i and for all j , except j equal to $n+1$ (the fictitious origin). The costs of assigning origin $n+1$ to any destiny 1 to m are equal to zero. The costs of assigning origin $n+1$ to destinies $m+1$ to $2m$ are equal to $+\infty$. In this way we guarantee the satisfaction of all the minimum capacity restrictions.

- 3- If there are facilities of type two and four, then the assignment problem can be solved as described in 2 but considering q destinies for each facility $i \in I_4$ (one for each possible dimension). The demand of each of these q destinies will be given by the number of elements of the corresponding dimension that are open at i .
- 4- If there is at least one facility of type 3, then it is necessary to solve a linear programming problem. The optimal allocation is more difficult to calculate because it is not possible to disaggregate the allocation problem in T independent linear problems (one for each time period), due to restrictions (13). The total assignment problem (TAP) can be solved using a general solver.

TAP:

$$\text{Min} \sum_t \sum_{i \in I_4} \sum_j \sum_s c_{ijs}^t x_{ijs}^t + \sum_t \sum_{i \in I/I_4} \sum_j c_{ij}^t x_{ij}^t \quad (17)$$

Subject to: (2)-(4), (11)-(14), with all location variables $a_{i\tau}^{\xi}$, $r_{i\tau}^{\xi}$, $a_{is\tau}^{\xi}$ and $r_{is\tau}^{\xi}$ fixed.

3 The Hybrid Genetic Algorithm

We called “hybrid genetic algorithm” to the algorithm developed because it integrates both genetic algorithms and local search. Another possible name would be “memetic” algorithm, if interpreted as an algorithm that tries to take advantage of the powerful characteristics of genetic algorithms, incorporating all available knowledge about the problem under study.

Our first experiences began with a simple genetic algorithm, without local search, but the results were far from being satisfactory. So, we followed Reeves, 1993b, advice: Hybridise whenever possible! Moscato and Cotta, 2003, feel that the success of memetic algorithms can probably be explained as being a direct consequence of the synergy of the different search approaches they incorporate.

According to Osmann and Kelly, 1996, an evolutionary algorithm is composed by five basic components: 1. a genetic representation of solutions to a problem; 2. a way to create an initial population of solutions; 3. an evaluation function; 4. genetic operators that alter the genetic composition of children during reproduction and 5. values for the parameters. These authors say that the data structure used for representation of solutions to the problem and the set of genetic operators constitute the algorithm's most essential components.

In the following subsections, all these components will be described for our particular algorithm.

3.1 Representation of solutions

Genetic algorithms work with populations of individuals, each representing a solution. So, the first step in designing a genetic algorithm for a particular problem is to devise a suitable representation scheme (Jaramillo et al, 2002). The way in which candidate solutions are encoded is a central factor in the success of a genetic algorithm. Sometimes, coming up with the best encoding is almost as difficult as solving the problem itself (Mitchell, 1996), especially because genetic representation is a component of genetic algorithms that is limited only by the implementer's imagination (Van Veldhuizen, 1999).

In this field of research, most authors borrowed the notions and definitions of biologists to refer to the codification of solutions as *chromosomes*, to chromosomes' individual elements as *genes*, to genes' possible values as *alleles*. Each gene is located at a particular position (*locus*) of the chromosome. If a solution is coded using one single chromosome, then each individual in the population is *haploid*. If a solution is coded using two chromosomes (similarly to human beings), it is called *diploid*. Most applications of genetic algorithms employ haploid individuals (Mitchel, 1996). It is also usual to define two different spaces: the genotype and phenotype space. The genotype space is the space where the whole evolutionary search takes place (Davis, 1996). The phenotype space can be very different and is the space of solutions to the problem under study.

In most genetic algorithms applied to location problems, the solutions are represented by chromosomes such that genes are a direct translation of the decision variables. For example, in simple plant location problems with decision variables y_i (equal to one if facility i is open or equal to zero otherwise), solutions can be coded as chromosomes such that each gene corresponds to a decision variable y_i .

As noticed in section 2 once the location variables are fixed, it is usually rather simple to find the optimal allocation variables⁹. This simple observation justifies the choice of codifying only part of the solution (the location variables).

If all facilities are of types 1 to 3, then all location variables are binary. If there is at least one facility of type 4, and N_{max} is greater than one, then location variables $r_{is\tau}^{\xi}$ can take integer values different from one or zero, which adds to the difficulty of finding a valid representation. For the time being let us consider that there are no facilities of type 4, meaning that all location variables are binary.

Our first attempt to codify a solution to problem CDLPOCR was considering a gene for each variable $a_{i\tau}^{\xi}$ and $r_{i\tau}^{\xi}$ (equal to one if the corresponding variable is equal to one and zero otherwise). The allocation variables were calculated as explained in section 2. This representation had a major drawback: the chromosomes' length. With simple calculations is easy to conclude that even in small instances of the problem the chromosomes would have an enormous number of genes¹⁰ (most of which would be equal to zero). Another disadvantage of this representation is the difficulty in generating admissible solutions (in both the initialisation phase and in each generation).

The observation of the algorithm's behaviour drove us to another representation: two chromosomes (each of size mT) represent each individual. The first chromosome (let us called it the *L*-chromosome) is composed of mT genes that can take values zero or one. Gene in position $(t-1)m+i$ is equal to one if facility i is open during time period t , and equal to zero otherwise¹¹. This information is not sufficient to build an admissible solution for problem CDLPOCR, because it is necessary to determine the open and reopen periods. If a facility i is continuously operating from time period t_1 to time period t_2 it is necessary to know if there were any reopenings during that time interval. The second chromosome (let us called it the *F*-chromosome) will give exactly this information. Gene in position $(t-1)m+i$ ¹² will be equal to

⁹ We are not considering here the problems where the assignment variables are also integer (total assignment problems). These problems are generally more difficult to solve (see, for instance, Cortinhal and Captivo, 2003).

¹⁰ For a problem with m potential facilities and T time periods the number of location variables could reach $2mT \frac{1+T}{2}$.

¹¹ Each *L*-chromosome can also be interpreted as a matrix with T rows and m columns such that gene in position $[t,i]$ is equal to one if facility i is open during period t and zero otherwise.

¹² Similar to the *L*-chromosome, this chromosome could be considered as a matrix with T rows and m columns.

one if facility i is reopened at the beginning of period t , and zero otherwise. The F -chromosome is less important than the L -chromosome (we can say that F -chromosomes complement the information provided by L -chromosomes). Its genes' values will only be taken into account when strictly needed. We will refer to the $(t-1)m+i$ -th gene in a chromosome as F or L -chromosome $[t,i]$ gene.

Consider the following example, with m equal to 5 and T equal to 3 (the matrix notation is used for ease of understanding):

Chromosome L						Chromosome F					
$t \backslash i$	1	2	3	4	5	$t \backslash i$	1	2	3	4	5
1	1	0	0	1	1	1	1	1	0	0	1
2	1	1	0	0	0	2	<i>0</i>	0	0	1	1
3	1	1	0	1	0	3	<i>1</i>	<i>0</i>	1	0	1

Figure 1: An individual's representation

In terms of location variables, these two chromosomes would be interpreted as all variables equal to zero except $a_{11}^2, r_{13}^3, a_{22}^3, a_{41}^1, r_{43}^3, a_{51}^1$. The three F -chromosome genes represented in bold italic are the only genes (from this chromosome) that really matter for building the solution.

Definition 1: Consider two individuals that differ only in one L (F) - chromosome gene. If the solutions they represent in the phenotype space are different then the L (F) - chromosome gene is called *determinant*, otherwise is called *non-determinant*.

Proposition 1: All L -chromosome genes are determinant.

Proposition 2: The only F -chromosome genes that are determinant are genes in position $(t-1)m+i$, for some i and $t > 1$, such that L -chromosome genes $(t-1)m+i$ and $(t-2)m+i$ are equal to one.

Proposition 3: It is possible to represent each and every admissible solution to CDLPOCR using a pair of F and L -chromosomes.

It is straightforward to conclude that this representation is *redundant*, according to the definition of Rothlauf and Goldberg, 2002: representations are redundant if the number of genotypes exceeds the number of phenotypes. Looking at the previous example, there are a number of different individuals (in the genotype space) that will be mapped to the same solution (in the phenotype space): all individuals that differ from the individual depicted in figure 1 in, at least, one non-determinant F -chromosome gene. Nevertheless, two individuals

are mapped to the same solution if and only if their L -chromosomes are exactly the same (due to proposition 1). Rothlauf and Goldberg, 2002 study the effect of redundant representations in the performance of genetic and evolutionary algorithms. Some authors have the opinion that each solution should be coded by exactly the same number of different individuals. The justification is obvious: if some solutions are “super-represented” by several different individuals then the genetic search can be biased. Rothlauf and Goldberg say that synonymously redundant representations, i.e, representations where the genotypes that represent the same phenotype are very similar to each other, do not change the performance of genetic algorithms as long as all phenotypes are represented on average by the same number of different genotypes. We have not studied deeply this problem, but the computational experiments already made let us believe that this is happening in our algorithm (maybe because the L -chromosome genes are all determinant).

As can be easily observed, this representation guarantees that restrictions (5), (7) and (8) are satisfied for every individual in the population. The only restrictions that can be violated are the capacity restrictions (11), (12), and (13).

Let us now return to problem CDLPOCR and consider the existence of facilities of type 4. Instead of creating another representation for non-binary location variables, we chose to adopt the representation just described to this new situation: a facility i can be composed of up to $Nmax$ elements of equal or different dimensions. This means that, at each time period, there are at most $Nmax$ elements of each possible dimension. Therefore, to extend the representation to this kind of facilities, each facility i will be represented by $q \times Nmax$ genes at each time period t . Each facility $i \in I_4$ is transformed in $q \times Nmax$ facilities called *dummy facilities*.

As an example, consider that all facilities are of type 4, T is equal to one, m is equal to 3, q is equal to 4 and $Nmax$ is equal to 2. Then each chromosome would have $m \times q \times Nmax$ genes, i.e, would have 24 genes and organized as follows:

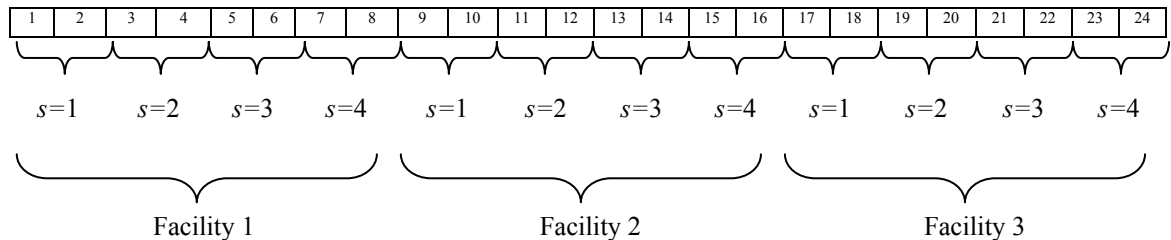


Figure 2: schematic representation of a chromosome when there are facilities of type 4

This representation can be easily translated to decision variables' values: variables $r_{is\tau}^{\xi}$ are equal to the number of *dummy facilities* corresponding to facility i and dimension s operating from τ to ξ . From all variables $r_{is\tau}^{\xi}$ with values greater than zero, we choose $r_{is\tau'}^{\xi}$, such that $\tau' = \min_{\tau, s} \{r_{is\tau}^{\xi} : r_{is\tau}^{\xi} > 0\}$, and decrease this variable in one unit, changing $a_{is\tau'}^{\xi}$ from zero to one. Notice that for each *dummy facility* i' corresponding to facility $i \in I_4$ and dimension $s \in S$, we can define $FR_{i'\tau}^{\xi}$ equal to $FR_{is\tau}^{\xi}$ and $FA_{i'\tau}^{\xi}$ equal to $FA_{is\tau}^{\xi}$. The problem CDLPOCR could be formulated using only variables $a_{i\tau}^{\xi}$ and $r_{i\tau}^{\xi}$, with $i=1, \dots, \#I_1 + \#I_2 + \#I_3 + q \times Nmax \#I_4$.

This representation does not guarantee the satisfaction of restrictions (10) nor (14). It has the advantage of maintaining the use of a binary alphabet, and allows the use of simple genetic operators. It has the disadvantage of increasing the number of genes in each chromosome.

3.2 Evaluation Function

In our algorithm, the fitness of each and every individual in a population is equal to the objective function value of the corresponding solution in the phenotype space. The calculation of the fixed open and reopen costs comes out directly from the solution's representation. Algorithm 1 describes this procedure for facility i . If $i \in I_4$ then this algorithm will be used for each of the $q \times Nmax$ *dummy facilities* and the costs summed up. After deciding which variable $a_{is\tau'}^{\xi}$ is equal to one (as described in the preceding section), then the total cost is changed by summing $FA_{is\tau'}^{\xi}$ minus $FR_{is\tau'}^{\xi}$.

The assignment costs are calculated separately as described in section 2. If the allocation problem (that has to be solved if in presence of capacitated facilities) is impossible, then the solution will have fitness equal to $+\infty$. The same happens if restrictions (10) are violated.

Algorithm 1: Calculation of fixed open and reopen costs for facility i

1. $cost \leftarrow 0$; $open \leftarrow false$; $t \leftarrow 1$.
2. If $t > T$ then stop, else go to 3.
3. if $L\text{-chromosome}[t, i] = false$ then $t \leftarrow t + 1$ and go to 2; else go to 4.
4. $tin \leftarrow t$; $t \leftarrow t + 1$;
5. If $L\text{-chromosome}[t, i] = 1$ then go to 6; else go to 7.
6. If $F\text{-chromosome}[t, i] = 1$ and $tin \neq t$ then go to 7, else $t \leftarrow t + 1$ and go to 5.
7. $tend \leftarrow t - 1$.

8. If *not open* and $i \notin I_4$ then $open \leftarrow true$, $cost \leftarrow cost + FA_{i\ tin}^{tend}$; else $cost \leftarrow cost + FR_{i\ tin}^{tend}$.
Go to 2.
-

3.3 Genetic Operators

The genetic algorithm developed uses the most common genetic operators found in the literature: selection, crossover and mutation. We also developed some special operators that take advantage of the known structure of the problem, namely a repair algorithm (that diminishes the number of non-admissible individuals) and an algorithm that changes only the *F*-chromosome's genes (and tries to diminish the solution's fixed open and reopen costs).

In the next sub-sections all these genetic operators will be described.

3.3.1 Selection

The selection operator used is based on binary tournament selection with sharing (Goldberg, 1989, Oei et al, 1991, Deb, 2001).

In every generation, two individuals are randomly selected from the parent population. A sharing value is calculated for each of them. This sharing value is used to prevent the early convergence of the population towards a single solution, and is calculated as follows: given two individuals *i* and *j*, the distance between them is given by the number of *L*-chromosome genes that are different in both individuals.

$$d_{ij}^g = \begin{cases} 1, & \text{if the } g \text{ - gene in the } L \text{ - chromosome is different in } i \text{ and } j \\ 0, & \text{otherwise} \end{cases}$$

$$d_{ij} = \sum_{g=1}^{mT} d_{ij}^g$$

$$sh(i, j) = \begin{cases} 1 - \left(\frac{d_{ij}}{\alpha_{share}} \right)^\alpha, & \text{if } d_{ij} \leq \alpha_{share} \\ 0, & \text{otherwise} \end{cases}$$

For each selected individual *i*, all values $sh(i, j)$ ¹³ (calculated considering all individuals *j* belonging to the new – children – population) are summed up:

$$nc_i = \sum_{j \text{ belongs to the new population}} sh(i, j)$$

If, at the moment of the selection, there are already *num* individuals in the new population, then $nc_i = num - nc_i$. The individual's fitness value will be divided by nc_i , and the resulting value

¹³ The α_{share} value is calculated as described in Deb, 2001 and α is considered equal to one.

$f(i)$) is used in the binary tournament selection. In the presence of two randomly chosen individuals x_1 and x_2 , if $f(x_1) < f(x_2)$ then individual x_1 wins the binary tournament with a given probability p_{bt} ¹⁴.

3.3.2 Crossover

The crossover operator used is an adaptation of the one-point crossover. Two parent solutions will be recombined yielding two children. A value κ between 1 and T is randomly chosen. The first child will have all L and F -chromosome genes $(t-1)m+i$, with $t < \kappa$, equal to the first parent, and all the other genes equal to the second parent. The opposite happens with the second child. This operator guarantees that if two parents satisfy restrictions (10), then so will their children. However, this crossover operator does not guarantee that the children of two admissible parents are admissible.

3.3.3 Mutation

The mutation operator is responsible for random changes in an individual's genotype. According to Goldberg, 1989, mutation is simply an insurance policy against the loss of genetic material.

Each L and F -chromosome gene is changed with a given mutation probability p_μ (usually close to zero). For each facility of type 4 only one L and one F -chromosome gene (randomly chosen) can be changed by the mutation operator in each time period.

3.3.4 Additional genetic operators

The algorithm developed tries to take advantage of all the existing knowledge about the problem at hand. This is why two more operators were designed: one operator is a repair algorithm that tries to diminish the number of inadmissible solutions in every generation. The computational experiments showed that, without a repair algorithm, the populations had a large number of inadmissible solutions (especially in presence of facilities of type 3 or 4) that were responsible for a weak performance (this is in accordance with Davis, 1996: a genetic algorithm that generates many illegal - not admissible - solutions will always perform worse than an algorithm that generates no illegal solutions). The other operator tries to change F -chromosome determinant genes in order to diminish the fixed open and reopen costs.

¹⁴ This probability is usually close to 1.

3.3.4.1 The Repair Procedure

An individual represents an inadmissible solution if: 1. it violates any maximum or minimum capacity restrictions; 2. the total number of elements located at a facility i of type 4 is greater than N_{max} . These violations are caused by L -chromosome genes.

The repair algorithm changes in a random but guided manner L -chromosome genes, as depicted in algorithm 2. If maximum capacity restrictions are violated at period t it randomly opens more facilities (changes genes from zero to one) such that the minimum capacity restrictions remain satisfied. If minimum capacity restrictions are violated at period t it randomly closes facilities (changes genes from one to zero) such that the maximum capacity restrictions remain satisfied. If there are facilities of type 3, then the allocation problem cannot be split into T separate problems. In this case, a heuristic procedure is used to achieve admissibility.

If the maximum number of equipments placed at i (being i a facility of type 4) is exceeded, then the repair algorithm randomly chooses genes i' equal to one that correspond to elements of i and change their values to zero.

As all changes are performed in a random manner, the repair algorithm cannot guarantee to find an admissible solution. That is why a maximum number of tries had to be imposed. We chose to repair an infeasible solution in a random manner because the use of a more structured algorithm (like a greedy heuristic) can introduce a strong bias in the search (Coello Coello, 2002).

3.3.4.2 The *Change Opening* procedure

This procedure studies the effect on fixed (re)open costs of changes in some of the determinant F -chromosome genes. As stated in proposition 2, we can identify an F -chromosome determinant gene if the L -chromosome genes in the same and in the immediately previous time position are equal to one for some facility i .

The procedure does not try to change every determinant F -chromosome gene, because that would be very time consuming. It only identifies situations such that a facility i is open from the beginning of time period τ to the end of time period ξ , $\xi > \tau$, and is reopen during that interval (in a time period $t \leq \xi$). This means that there is a determinant F -chromosome gene in position $(t-1)m+i$ that is equal to one, and this is the gene whose value the procedure tries to change to zero. If the fixed open and reopen costs diminish, then the gene's value is changed, otherwise retains its original value.

Algorithm 2: Repair Algorithm

Predefined parameters: NMAXTRY-total number of iterations in each time period.

1. If there are facilities of type 3 then go to 2. Else go to 3.
2. Solve the total assignment problem using a general solver. If the problem is impossible then go to 3, else stop (the solution is admissible).
3. $t \leftarrow 1$. For each facility $i \in I_3$, calculate $Cap_i^0 \leftarrow 0$.
4. $ntries \leftarrow 1$. If $t > T$ then stop. Else go to 5.
5. For each facility $i \in I_3$ update:

$$Cap_i^t \leftarrow \begin{cases} Cap_i^{t-1} + Q_i, & \text{if } L\text{-chromosome}[t,i] = 1 \text{ and } F\text{-chromosome}[t,i] = 1 \\ Cap_i^{t-1}, & \text{otherwise} \end{cases}$$

This value represents the maximum capacity of facility i during period t .

6. Calculate $D \leftarrow \sum_j d_j^t$, $C_{max} \leftarrow$ total maximum capacity of facilities operating during t ¹⁵,
 $C_{min} \leftarrow$ total minimum capacity of facilities operating during t , $Num_i \leftarrow$ total number of elements operating at $i \in I_4$, during t , $\forall i \in I_4$.
7. $ntries \leftarrow ntries + 1$. If $ntries > NMAXTRY$ then stop. Else go to 8.
8. If $C_{min} > D$ then go to 9. If $C_{max} < D$ then go to 10. If $\exists i \in I_4: Num_i > Nmax$ then go to 11. Else go to 13.
9. Choose randomly a facility $i \in I_2$ such that $L\text{-chromosome}[t,i] = 1$ and $C_{max} - Q_i \geq D$.
 $L\text{-chromosome}[t,i] \leftarrow 0$, $C_{min} \leftarrow C_{min} - Q'_i$, $C_{max} \leftarrow C_{max} - Q_i$. Go to 7.
10. Choose randomly a facility i (including *dummy facilities*) such that $L\text{-chromosome}[t,i] = 0$ and $C_{min} + Q'_i \leq D$ ¹⁶. $L\text{-chromosome}[t,i] \leftarrow 1$, $C_{min} \leftarrow C_{min} + Q'_i$. If $i \in I_3$ then $C_{max} \leftarrow C_{max} + Cap_i^t$, else $C_{max} \leftarrow C_{max} + Q_i$ ¹⁷. If i is a *dummy facility* then $Num_{i'} \leftarrow Num_{i'} + 1$, with i' the corresponding facility belonging to I_4 . Go to 7.
11. Choose randomly a facility $i \in I_4$ such that $Num_i > Nmax$.
12. Choose randomly a *dummy facility* i' corresponding to one open element in i of dimension s , $s \in S$. $L\text{-chromosome}[t,i'] \leftarrow 0$, $Num_i \leftarrow Num_i - 1$, $C_{max} \leftarrow C_{max} - Q_s$.

¹⁵ If there is at least one facility of type 1 in operation during t , then $C_{max} \leftarrow +\infty$.

¹⁶ If $i \in I_3$ or is a *dummy facility*, then Q'_i is equal to zero.

¹⁷ If i is a *dummy facility* corresponding to dimension s , then Q_i will be equal to Q_s .

13. If there are no facilities of type 3, then $t \leftarrow t+1$ and go to 4. Else go to 14.
 14. Solve one transportation problem as described in section 2, treating facilities of type 3 as facilities of type 2 with maximum capacities equal to Cap_i^t and minimum capacities equal to zero. Update $Cap_i^t, \forall i \in I_3$, subtracting the total flow that reaches demand point i . $t \leftarrow t+1$ and go to 4.
-

This procedure does not change F -chromosome genes corresponding to facilities of type 3, because it could not only change the fixed costs value but also the allocation problem's optimal solution.

In algorithm 3, the *change opening* procedure is formally described.

Algorithm 3: Change-Opening procedure

1. $i \leftarrow -1$;
 2. If $i \in I_3$ then $i \leftarrow i + 1$.
 3. If $i > m$, then stop. If $i \notin I_4$ go to 4, else go to 6.
 4. Detect a pair of location variables equal to one of the form $(a_{i\tau}^\xi, r_{i\xi+1}^\psi)$ or $(r_{i\tau}^\xi, r_{i\xi+1}^\psi)$. If there are no pair of variables in this situation, then $i \leftarrow i + 1$ and go to 3. Else go to 5.
 5. $\Delta \leftarrow FA_{i\tau}^\psi$ (or $FR_{i\tau}^\psi$) $- FA_{i\tau}^\xi$ (or $FR_{i\tau}^\xi$) $- FR_{i\xi+1}^\psi$. If $\Delta < 0$ then F -chromosome $[\xi+1, i] \leftarrow 0$, $a_{i\tau}^\xi$ (or $r_{i\tau}^\xi$) $\leftarrow 0$, $r_{i\xi+1}^\psi \leftarrow 0$ and $a_{i\tau}^\psi$ (or $r_{i\tau}^\psi$) $\leftarrow 1$. Go to 4.
 6. Detect a pair of location variables greater than zero of the form $(a_{is\tau}^\xi, r_{is\xi+1}^\psi)$ or $(r_{is\tau}^\xi, r_{is\xi+1}^\psi)$. If there are no pair of variables in this situation, then $i \leftarrow i + 1$ and go to 3. Else go to 7.
 7. $\Delta \leftarrow FA_{is\tau}^\psi$ (or $FR_{is\tau}^\psi$) $- FA_{is\tau}^\xi$ (or $FR_{is\tau}^\xi$) $- FR_{is\xi+1}^\psi$. If $\Delta < 0$ then F -chromosome $[\xi+1, i'] \leftarrow 0$, such that i' is one *dummy facility* corresponding to facility i , dimension s , with F -chromosome $[\xi+1, i'] = 1$, $a_{is\tau}^\xi \leftarrow 0$ or $r_{is\tau}^\xi \leftarrow r_{is\tau}^\xi - 1$, $r_{is\xi+1}^\psi \leftarrow r_{is\xi+1}^\psi - 1$, and $a_{is\tau}^\psi \leftarrow 1$ or $r_{is\tau}^\psi \leftarrow r_{is\tau}^\psi + 1$. Go to 4.
-

3.4 Local Search

Local search plays a very important role in the algorithm developed. The first versions of this algorithm did not use local search to improve the individuals' fitness, and the results were far from satisfactory. Examples of genetic algorithms hybridized with local search can also be found in Huntley and Brown, 1996; Yagiura and Ibaraki, 1996; Reeves and Höhn, 1996.

We did not include local search in the genetic operators section because operators like selection and crossover can be interpreted as evolution operators that change the population as a whole. Local search works with a single individual at a time, and does not have a global perspective of the population where the individual is inserted. Sinha and Goldberg, 2003, state that genetic operators like crossover are responsible for global search, whilst local search can be seen as an individual's own learning path.

In our algorithm, local search is executed after the crossover and the mutation operators. Every individual in the new population is a potential starting solution for the local search procedure, that is ran with a given probability p_{ls} . If a child is equal to one of the parents that, in turn, had already been the result of a local search procedure, this probability is equal to zero.

Consider the following definition:

Definition 2: An individual x' is said to be in the k -neighbourhood of individual x if and only if x' differs from x by the insertion or removal of at most k continuous operating time periods (without reopenings) to a single facility i ¹⁸. This means that the genotype of x' and x differ in exactly k L -chromosome genes and at most k F -chromosome genes.

Among all x' individuals belonging to the k -neighborhood of x with the same values in the k L -chromosome genes, the local search procedure visits only individuals such that their corresponding F -chromosome genes represent the minimum possible fixed open and reopen costs. Consider an individual x such that facility i is operating from τ to ξ and from $\xi+k+1$ to φ . There are several different individuals x' in the k -neighborhood of x such that i is continuously operating from τ to φ . These individuals differ one from another due to the differences in determinant F -chromosomes' genes in positions $[\xi+1, i]$ to $[\xi+k+1, i]$. The local search procedure will only consider individual x' that corresponds to the solution with the least fixed costs. Exemplifying with k equal to two, the situation is depicted in figure 3, where the local search will try to put facility i operating during time periods $\xi+1$ and $\xi+2$.

The local search considers four possibilities, depending on the determinant L -chromosome's genes:

¹⁸ If facility i is of type 4, then *facility* should be replaced by *element of a facility* in this definition.

- Operating continuously from τ to φ without reopenings;
- Reopening at $\xi + 1$ and $\xi + 3$;
- Reopening only at $\xi + 1$;
- Reopening only at $\xi + 3$.

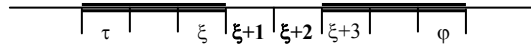


Figure 3: **====** operating periods

The individual that corresponds to the least costly solution in terms of fixed costs will be the only one visited. If there are no facilities of type 3, visiting only this subset of neighbors will give exactly the same results as visiting all k -neighbors. Nevertheless, if there are facilities of type 3, the reopening periods will affect not only the fixed costs but also the total assignment cost, so it is possible that the local search will not visit some interesting k -neighbors. This disadvantage is compensated by the decrease in the computational cost of the procedure.

The local search procedure developed tries to improve an individual's fitness by searching k -neighborhoods, from $k = 1$ to T ¹⁹. Whenever the fitness function is improved, the individual's genotype is immediately changed, and the local search procedure continues with this new individual as the starting-point (the first improvement strategy was chosen instead of best improvement). If the local search procedure does not find a better individual among the first z k_k -neighbors visited, then it stops searching the k_k -neighborhood and continues with k_{k+1} -neighborhood.

The local search is performed in the phenotype space, but if an improved fitness value is found then the individual's genotype is changed (we adopted the Lamarckian learning paradigm). There are authors that prefer to consider that *learning* (here interpreted as the local search procedure) cannot influence an individual's genetic structure (Baldwinian learning paradigm) (Goldberg, 1989). In the latter case, the individual retains the best fitness value found in its neighborhoods, but its genotype is not changed (or is changed with a given small probability).

The local search procedure is very time consuming and can be responsible for about 95% of the algorithm's total computational time. This makes it compulsory to improve its

¹⁹ The present implementation of the algorithm considers k -neighbourhoods, from $k=1$ to T , sequentially. It is possible that the procedure can be improved if the neighbourhood is changed as in Variable Neighbourhood Search Metaheuristic (Glover and Kochenberger, 2003).

performance. As observed in Ishibuchi et al, 2002, it is very important (and most of the times difficult) to find a good balance between genetic search and local search in the implementation of hybrid evolutionary algorithms.

In each iteration of the local search procedure, k L -chromosome genes and l , $0 \leq l \leq k$, F -chromosome genes are changed (all referring to the same facility i). This means that it is necessary to recalculate the fixed open and reopen costs for this facility, and also the optimal solution for k assignment problems²⁰. The latter recalculations are the ones that most contribute to the local search procedure computational time, especially when there are capacitated facilities²¹. We changed the local search procedure in order that, before calculating the k new optimal assignment solutions²² (in the presence of capacitated facilities), the procedure performs a sensitivity analysis in order to estimate if the present neighbor is or is not better than the current individual. This is done by observing the dual optimal solution of the assignment(s) problem(s).

Consider that the local search procedure is visiting a k -neighbor, and studying the effect of opening (closing) facility i from periods τ to $\tau + k - 1$. Let us first assume that there are no facilities of type 3. If this is the case, the local search procedure begins by testing the admissibility of the neighbor: it is necessary to test the minimum capacity restrictions and restrictions (10) if service i is going to be open and the maximum capacity restrictions if it is going to be closed. If the neighbor is not admissible, the local search will continue visiting other neighbors. Otherwise it will perform a sensitivity analysis to each and every k assignment problem.

Each assignment problem can be solved as a transportation problem as described in section 2. For a given period t consider, as usual, dual variables u_j^t associated to source j constraints and variables v_d^t associated to destiny d constraints. The change in the objective function value of increasing (decreasing) the demand of a destiny d in δ units is calculated by

²⁰ Notice that if there is a facility of type 3 then the optimal solution for the total assignment problem has to be recalculated.

²¹ If all facilities are uncapacitated then it is straightforward to find the new optimal allocations: if a facility is closed in period t then all the clients assigned to this facility will be reassigned to the nearest open facility. The other assignments are not changed. If a facility i is open in period t , all clients j such that $c_{ij}^t < c_{i'j}^t$ (being i' the present assigned facility) will be assigned to facility i .

²² Or the optimal solution to the total assignment problem, if that is the case.

$\delta(u_{n+1}^t + v_d^t)$, as long as the present dual solution remains the unique optimum dual solution for the problem with the modified data (Murty, 1983)²³.

The local search procedure studies the effect of increasing (decreasing) one unit of the facility's demand when the facility is considered open (close). If the facility has minimum capacity restrictions then, for each $\tau \leq t \leq \tau+k-1$, it calculates $\Delta_t = u_{n+1}^t + v_{i+m}^t$. If the facility has only maximum restrictions, then it calculates $\Delta_t = u_{n+1}^t + v_i^t$.

Then, for each k -neighbor, the local search procedure calculates $\Delta_A = \sum_{t=\tau}^{\tau+k-1} \Delta_t$ and compares this value with the change, by unit of maximum capacity, of the total fixed open and reopen costs (Δ_F). If $\Delta_F + \Delta_A$ (or $\Delta_F - \Delta_A$) is greater than zero than the local search procedure visits this neighbor with probability p_v , otherwise visits the neighbor with probability p_{nv} , $p_v \gg p_{nv}$.

If there is at least one facility of type 3, then the value of Δ_A is calculated by summing up the optimal dual variables' values associated with restrictions (11), (13) and (14).

In comparison with the local search procedure where all neighbors were visited, this modified procedure produces in almost all cases the same improved objective function value with a significant reduction in the computational time spent. The reduction in the computational time needed is reflected in the hybrid algorithm's total computational time, and also in the percentage of this time that is consumed by local search.

It is interesting to note that is much more difficult to study the changes in the objective function value of opening or closing a given facility during a time period t than in the static capacitated plant location problem. In the latter problem, it is known that opening a plant will increase the fixed costs and decrease the assignment costs. The closing of a facility will have the opposite effect. In the present problem, and particularly when considering the existence of facilities with minimum capacities greater than zero, opening or closing a facility during period t can increase or decrease the fixed costs and can increase or decrease the assignment costs. This makes it impossible to use the traditional methods of fixing *a priori* the values of some location variables to one or zero using what is usually called *reduction tests* (see, for instance, Bornstein and Azlan, 1998, Saldanha-da-Gama and Captivo, 2002). These reduction tests are based in a property of the transportation problem that is lost if there are facilities with minimum

²³ Notice that the increase (decrease) in one destiny's demand has to be accompanied by an equal increase (decrease) in the fictitious origin's supply.

capacities greater than zero. Consider a set O_1 of open facilities (destinies) and let us define $W(O_1)$ as the objective function value of the respective optimal assignment problem (as in Bornstein and Azlan, 1998). Consider also a facility $i \notin O_1$. If facility i is added to O_1 then: $W(O_1) - W(O_1 \cup \{i\}) \geq 0$ and $W(O_1) - W(O_1 \cup \{i\}) \leq W(O_2) - W(O_2 \cup \{i\})$, $\forall O_2 \subseteq O_1$ ²⁴. It is easy to demonstrate that if there are facilities with minimum capacities greater than zero, then none of these inequalities are guaranteed to be true. The possibility of extending the use of reduction tests and of ADD and DROP heuristics to fix the value of some location variables can only be thought if all facilities are uncapacitated or have only maximum capacities (we have not yet studied this possibility).

Algorithm 4: Local Search Procedure

x – starting solution; $f(x)$ – fitness of individual x .

1. $k \leftarrow 1$.
 2. if $k > T$ then stop. Else $count \leftarrow 0$ and go to 3.
 3. $flag(i) \leftarrow false$, $\forall i$.
 4. If $flag(i) = true$, $\forall i$ or $count > z$ then $k \leftarrow k + 1$ and go to 2. Else choose randomly a facility i such that $flag(i) = false$.
 5. $flag(i) \leftarrow true$; $t \leftarrow 1$;
 6. If $t > T - k + 1$ or $count > z$ then go to 4, else go to 7.
 7. If facility i is operating during periods t to $t + k - 1$, then study k -neighbor x' obtained from x by the removal of operating periods t to $t + k - 1$ and go to 9. Else go to 8.
 8. If facility i is not operating during periods t to $t + k - 1$, then study k -neighbor x' obtained from x by the insertion of operating periods t to $t + k - 1$ and go to 9. Else $t \leftarrow t + 1$ and go to 6.
 9. If $f(x) > f(x')$ then $x \leftarrow x'$ and $count \leftarrow 0$, else $count \leftarrow count + 1$. $t \leftarrow t + 1$ and go to 6.
-
-

Instead of solving each and every k transportation problem (or the total assignment problem) for every k -neighborhood visited, it is also possible to consider approximations: calculations of valid upper or lower bounds for the assignment problems. When opening a new facility (of type different than 3), we can calculate the flows that will be redirected to this facility by calculating the optimal solution of a modified knapsack problem²⁵. The optimal

²⁴ This property is called supermodularity.

²⁵ Jacobsen, 1983, uses a similar procedure in solving the static capacitated simple plant location problem.

objective function value is a valid upper bound for the optimal solution of the transportation problem. If a facility i is closed during t then it is possible to use a Lagrangean relaxation to calculate a lower bound on the optimal objective function value of the assignment problem. A Lagrangean relaxation could also be used if there is at least one facility of type 3. The true optimal values of the assignment problem(s) could only be calculated at the end of the local search procedure. These possibilities have not yet been tested, but can reduce significantly the computational time spent by the local search, with (possibly) the additional cost of decreasing the quality of the individual returned by this procedure.

3.5 Population

Our initial population is constituted by individuals randomly created that are modified by the *repair*, *change openings* and *local search* procedures. We have not tried to initialise the population with chosen individuals that guarantee the presence of a sound genetic diversity. We believe that the initialisation of the population will only affect the first generations, and will have a minor influence in the final outcome.

The algorithm developed is quite time demanding, especially due to the local search procedure. Working with very large populations, like most genetic algorithms do, could jeopardize the possibilities of obtaining good performances. So, we chose to use small populations. With small populations the algorithm simulates a greater number of generations per unit of time. There is a risk of premature convergence to a poor quality solution, nevertheless empirical results have shown that it is possible to achieve good results with small populations of 30 or less elements (Reeves, 1993a).

One way of overcoming this disadvantage is to consider a structured population, like the one described in Cortinhal and Captivo, 2003. The population is structured as a ternary tree, with hierarchy relations established between individuals through their fitness values. This approach gave poor results and was abandoned.

Another way of overcoming this disadvantage is by changing on-line the total number of individuals in the current population. The number of individuals is increased whenever *nimp* (a predefined parameter) generations are run without improving the best objective function value. The new individuals are randomly initialized.

The population is initialized with $npop$ individuals. The value for $npop$ is calculated as described in Reeves, 1993a: it is the minimum value such that $\left(1 - \left(\frac{1}{2}\right)^{npop-1}\right)^l \geq 0.99$ ²⁶,

where l is the number of genes of each individual. Each individual has two chromosomes, each with mT genes, so l should be equal to $2mT$. As the F -chromosome has very few determinant genes, we chose to consider l equal to mT for the calculation of the initial value of $npop$, and equal to $2mT$ for the calculation of the maximum value $npop$ can take.

3.6 Parameters' Values

As can be seen by the previous descriptions, this algorithm has many parameters whose values have to be fixed and that can influence the algorithm's behaviour. It is sometimes difficult to understand the influence of a single parameter over the whole algorithm, and even more complicated to understand the interaction between parameters.

There can be two forms of setting parameter values: parameter tuning and parameter control (Eiben et al, 1999). Parameter tuning refers to the process of finding the parameter values before executing the algorithm. Parameter control allows the parameter values to change during the execution of the algorithm (see, for instance, Bäck et al, 2000). Aside from the number of individuals in the population, that can change during the execution, all other parameters in our algorithm are fixed before the run and do not change during the run.

We have not studied in deep the influence of the parameters in our algorithm. Here is a list of all parameters, and the way in which we think they can influence its behaviour.

Parameter	Description	Influence on the algorithm's behaviour and Recommended Values ²⁷
p_{bt}	Probability of choosing the most fitness individual in the binary tournament selection	The greater the probability, the more difficult it is for less fit individuals to be passed on to the next generation. It can be used to influence the diversity of the population. If controlled on line, this parameter could be increased as the number of generations increases, to ensure diversity in the beginning and convergence in the end. In our algorithm this value is fixed at 0.9.

²⁶ The value 0.99 represents the probability of at least one allele being present at each locus in the initial population.

²⁷ We have not yet studied deeply the influence of all these parameters in our algorithm. These "recommended values" are indicated according to the computational experiments made so far.

p_{μ}	Probability of changing one gene in the mutation operator	This parameter can influence the diversity of the population. If controlled on-line, it could be decreased as the number of generations increases, or increased when the best fitness value does not improve in a given number of generations. In our algorithm this parameter is fixed at 0.002.
p_{ls}	Probability of executing the local search procedure for each individual	This parameter influences both the computational time and the quality of the best solution found. With values near 1, the algorithm will converge quicker and with good quality solutions. It is difficult to estimate how this parameter influences computational time because with smaller values each generation is executed in less computational time but the convergence towards a good solution is slower, so the total algorithm's computational time can increase. It is advised that p_{ls} should be equal to 1 at least in the last generation. In our algorithm this value is fixed to 1.
z	Maximum number of k -neighbours visited without improving the individual's fitness	This parameter influences the algorithm's behaviour in a way similar to the previous one. It should consider the total number of neighbours of a given solution which is hard to compute. In our algorithm we consider z equal to 10000.
p_v	Probability of visiting a neighbour that is expected to improve the individual's fitness	This parameter influences the algorithm's behaviour in a way similar to the previous two parameters. This probability should be always a value near to 1. In our algorithm it is fixed to one.
p_{nv}	Probability of visiting a neighbour that is not expected to improve the individual's fitness	This probability influences the computational time and also the quality of the final solution. To obtain a good compromise value, we recommend it should be fixed to a value between 0 and 0.1.
n_{pop}	Number of individuals in the current population	This parameter influences the computational time and the quality of the final solution: populations with more individuals will take longer to generate their children but are genetically more powerful. Small populations run the risk of under-covering the solution space (Reeves, 1993a). In our algorithm we calculate the initial population as described in 3.5, and increase this value whenever there are n_{imp} generations without improvement of the best objective function value.

<i>Nmaxpop</i>	Maximum number of individuals in the current population	The number of individuals in the current population is increased whenever there are a predefined number of generations without improving the objective function value. This parameter influences the total execution time of the algorithm, and can influence the quality of the best solution found.
β	Percentage of increase in the number of individuals	This parameter, along with parameter <i>Nmaxpop</i> , controls the number of times the population is increased. It is hard to predict how it will influence the quality of the solution or the total computational time: greater values will correspond to fewer generations but with longer computational times per generation. In our algorithm this value is equal to 25%.
<i>Nger</i>	Total maximum number of generations	It is a parameter that can be used to terminate the algorithm. If it is completely blind to the algorithm's performance, it can be responsible for premature terminations as well as for unnecessary generation runs. In our algorithm this parameter is not important, because it uses other termination rules.
<i>nimp</i>	Maximum number of generations without improving the best objective function value found	This parameter is used to indicate that the algorithm is converging. In our algorithm, the number of individuals in the population is increased whenever there are no improvements in the objective function during <i>nimp</i> generations, as a way of increasing the genetic diversity, and to avoid getting trapped in local minimums. If the current number of individuals is equal to <i>Nmaxpop</i> , then the algorithm is terminated after <i>nimp</i> generations without improving the objective function. It is fixed to 5.

3.7 Putting it all together

All the described procedures are now joined to build the hybrid algorithm. We opted to use a generational replacement with elitism. This means that the whole generation is replaced by their children, with the exception of some individuals (the best) that are directly passed on to the next generation. In our algorithm only the best individual is considered. We chose the generational replacement instead of steady-state replacement mainly due to the computational time requirements: the steady-state replacement strategy needs much more comparisons between solutions, and our algorithm is very time demanding already (due to the local search procedure). Algorithm 5 describes one generation, and algorithm 6 describes the whole hybrid algorithm.

Algorithm 5: Generation

x_{best} – represents the best individual in the preceding generation;
 $flag(x)$ – is equal to *true* if x has already passed through the local search procedure, *false* otherwise.
 $P_{current}$ – the current population.
 $f(x_j)$ – fitness of individual x .

1. $x_1 \leftarrow x_{best}; x_{best} \leftarrow x_1; j \leftarrow 2; Newpop \leftarrow \{x_1\}$.
 2. If $j > npop$ then $P_{current} \leftarrow Newpop$, else go to 3.
 3. Select parents x_A and x_B using binary tournament selection.
 4. Crossover to generate two children: x_j and x_{j+1} . $flag(x_j) \leftarrow false; flag(x_{j+1}) \leftarrow false$.
 5. Apply the mutation operator to x_j .
 6. If $x_j = x_A$ then $flag(x_j) \leftarrow flag(x_A)$; if $x_j = x_B$ then $flag(x_j) \leftarrow flag(x_B)$,²⁸
 7. Calculate the fitness of $x_j : f(x_j)$. If $f(x_j) = +\infty$, then apply the repair procedure to x_j . If $f(x_j) < f(x_{best})$ then $x_{best} \leftarrow x_j$.
 8. Apply the *change openings* procedure to x_j .
 9. If *not* $flag(x_j)$ then apply the local search procedure.
 10. If $(j+1) \leq npop$ then repeat steps 5 to 9 with child x_{j+1} .
 11. $Newpop \leftarrow Newpop \cup \{x_j\}$. If $(j+1) \leq npop$ then $Newpop \leftarrow Newpop \cup \{x_{j+1}\}$. $j \leftarrow j + 2$. Go to 2.
-
-

Algorithm 6: global hybrid algorithm

1. Initialise $P_{current}$.
 2. Initialise $x_{best}, best \leftarrow f(x_{best}), ngen \leftarrow 1, count \leftarrow 0$.
 3. If $ngen > Nger$ or $count > nimp$ then stop. Else go to 4.
 4. $ngen \leftarrow ngen + 1$. Call procedure *generation*.
 5. If $f(x_{best}) \geq best$ then $count \leftarrow count + 1$. Else $count \leftarrow 0$.
 6. If $count < nimp$ then $ngen \leftarrow ngen + 1$, go to 3. If $\min\{\lceil npop(1 + \beta) \rceil, Nmaxpop\} > npop$ then $npop \leftarrow \min\{\lceil npop(1 + \beta) \rceil, Nmaxpop\}$, initialise randomly the new individuals and $count \leftarrow 0$. Go to 3.
-
-

²⁸ x_j is considered equal to x_A if all the L -chromosome's genes are equal.

4 Additional Restrictions

There are a number of additional restrictions that immediately come to mind when looking at problem CDLPOCR: establish a maximum number of opening facilities, consider a limited budget, guaranteeing that a particular facility is open or close during a particular time period, etc. These kind of restrictions are difficult to handle by the primal-dual heuristics developed by the authors (Dias et al, 2004a,b): the additional restrictions will change the primal and also the dual problem and it would be necessary to modify the procedures developed, or embed the heuristic in some kind of branch and bound mechanism.

There are many different ways of dealing with restrictions within genetic algorithms. In the described hybrid algorithm, one way of dealing with problem restrictions was through the use of an adequate representation and also the development of a repair algorithm. There is a kind of additional restrictions that can be handled efficiently by the developed algorithm, and with only minor changes: fixing some facilities open or closed during some time periods. All other possible additional restrictions will be treated using a penalty function.

4.1 Fixing facilities open or closed

We consider the DM wishes to fix open or closed some of the facilities during some time periods. This kind of preferences can be modelled by linear restrictions of the form²⁹:

$$1. \text{ Fixing a facility } i \text{ open during time period } t: \sum_{\tau=1}^t \sum_{\xi=t}^T (a_{i\tau}^{\xi} + r_{i\tau}^{\xi}) = 1; \quad (18)$$

$$2. \text{ Fixing a facility closed during time period } t: \sum_{\tau=1}^t \sum_{\xi=t}^T (a_{i\tau}^{\xi} + r_{i\tau}^{\xi}) = 0; \quad (19)$$

Thanks to the representation chosen, these additional restrictions are treated in a very efficient manner: if the DM wants facility i open (closed) during period t , then the L -chromosome gene in position $(t-1)m+i$ will have to be equal to one (zero) in all the individuals of the population. This is achieved in a straightforward way: all individuals pass through a filter that changes their genotype in accordance with the DM preferences. This filter is used in the initialisation of the population, in the mutation operator (it will not be possible to change a gene whose value is fixed) and in the local search procedure (the procedure will only visit individuals that satisfy the DM preferences).

²⁹ Similar restrictions can be used to fix the number of elements of each possible dimension open or closed in facilities of type 4.

The computational tests performed show that this kind of additional restrictions do not worsen the algorithm's performance.

4.2 Linear additional restrictions

In this section we consider the possibility of introducing one or several additional restrictions r of the form:

$$\begin{aligned} & \sum_t \sum_{i \in I_4} \sum_j \sum_s c^{r^t}_{ijs} x^t_{ijs} + \sum_t \sum_{i \in I_4} \sum_s \sum_{\xi=t}^T FA^{r^\xi}_{ist} a^\xi_{ist} + \sum_t \sum_{i \in I_4} \sum_s \sum_{\xi=t}^T FR^{r^\xi}_{ist} r^\xi_{ist} + \\ & \sum_t \sum_{i \in I \setminus I_4} \sum_j c^{r^t}_{ij} x^t_{ij} + \sum_t \sum_{i \in I \setminus I_4} \sum_{\xi=t}^T FA^{r^\xi}_{it} a^\xi_{it} + \sum_t \sum_{i \in I \setminus I_4} \sum_{\xi=t}^T FR^{r^\xi}_{it} r^\xi_{it} \geq B^r \\ & \leq \end{aligned} \quad (20)$$

These linear additional restrictions are much harder to handle than those treated in the preceding section. They raise two different questions: 1– how to treat solutions that are admissible with respect to constraints (2) – (14), but do not satisfy the additional restrictions; 2– how to solve the assignment problems when $\exists r: c^{r^t}_{ijs} > 0$ or $c^{r^t}_{ij} > 0$.

There are many different ways of handling restrictions within genetic algorithms (see, for instance, Michalewicz, 1995; Coello Coello, 2000a, 2000b, 2002; Coello Coello and Cortés, 2001; Eiben, 2001; Coello Coello and Montes, 2002).

In section 3.2 we have treated infeasible individuals with a penalty value of $+\infty$: they are considered worse than any other individual in the population. This procedure could be extended to individuals that violate at least one additional restriction. Nevertheless, we opted to let the DM choose the way in which these individuals will be penalized, giving him four options: a) their fitness value is equal to $+\infty$; b) their fitness value is equal to the objective function value (1), penalised by a fixed percentage; c) their fitness value is equal to (1) plus the amount of violation of each additional restriction; d) their fitness value is equal to (1) and is not penalised³⁰.

The representation used to code the solutions considers a unique set of optimal assignment variables associated with each individual. If the assignment variables are calculated as described in section 2, then there can be situations when the algorithm will not be able to identify the optimal solution even if in presence of an individual that codifies the optimal

³⁰ This fourth option can be of use in the early phases of solving a complex problem, when a DM knows that he will have to consider a restriction but does not want to restrict the search in order to learn more about the problem.

location variables' values. If we consider the additional restrictions in the calculation of optimal assignment variables (notice that the right hand side will be changed taking into account the already fixed location variables) the assignment problem becomes much harder to solve and it will have to be solved as a linear programming problem (even when there are no facilities of type 3). Another way of dealing with this situation is by solving the assignment problems as described in section 2, but considering several different objective function coefficients and trying to find an assignment solution that will make the individual admissible with respect to the additional restrictions. This can be done using lagrangean multipliers associated with the modified additional restrictions and adjust their values through the multiplier adjustment procedure (Reeves, 1993b).

Algorithm 7: Multiplier Adjustment

We will consider that there are R additional restrictions (20), and all are of the \leq type³¹. We consider each restriction (20) represented as $A_r^1(\mathbf{l}) + A_r^2(\mathbf{x}) \leq B^r$, such that \mathbf{l} and \mathbf{x} represent the vector of all location and assignment variables, respectively. The location variables are fixed (they are given by the L and F -chromosomes' genes).
Nmaxtries – maximum number of iterations.

1. $r \leftarrow 1$.
 2. For each additional restriction r calculate $A_r^1(\mathbf{l})$, $B^r \leftarrow B^r - A_r^1(\mathbf{l})$, $\mu_r \leftarrow 0$.
 3. $admin \leftarrow false$, $c_{ij}^t \leftarrow c_{ij}^t + \sum_r \mu_r c_{ij}^{rt}$, $\forall i \notin I_4, j, t$, $c_{ijs}^t \leftarrow c_{ijs}^t + \sum_r \mu_r c_{ijs}^{rt}$, $\forall i \in I_4, s, j, t$.
 4. Solve T transportation problems (as described in section 2) considering the modified assignment costs given by c_{ij}^t and c_{ijs}^t . Calculate $A_r^2(\mathbf{x})$, $\forall r$.
 5. If $A_r^2(\mathbf{x}) \leq B^r$, $\forall r$ then $admin \leftarrow true$ and go to 8. Else go to 6.
 6. If $A_r^2(\mathbf{x}) > B^r$ then $\delta_r \leftarrow \min \left\{ \min_{i,j,t} \left\{ \frac{c_{ij}^t}{c_{ij}^{rt}} \right\}, \min_{i,j,s,t} \left\{ \frac{c_{ijs}^t}{c_{ijs}^{rt}} \right\} \right\}$, else $\delta_r \leftarrow 0$, $\forall r$.
 7. $\mu_r \leftarrow \mu_r + \delta_r$, $\forall r$, $count \leftarrow count + 1$.
 8. If $count > Nmaxtries$ or $admin$ then stop, else go to 3.
-

The algorithm developed leaves to the DM the decision of dealing with the assignment problems in presence of additional restrictions in one way or another. Our computational

³¹ It is trivial to change this procedure to accommodate \geq restrictions.

experience shows that the first alternative generates better solutions but at the cost of greater computational execution times.

In comparison with the situation of no additional restrictions, the performance of the algorithm is clearly worse, both in terms of solution quality and computational time spent. This is especially true when the right hand side values of the additional restrictions are very different from the left hand side values of (20) for the optimal solution to CDLPOCR.

5 Multi-objectives

Since the pioneering work of Schaffer, 1985, (that developed the VEGA algorithm), much has been done and said about the use of genetic algorithms (or evolutionary algorithms in general) in a multi-objective scenario. The use of a population of individuals reminds us immediately the possibility of simultaneously approximating several non-dominated solutions. Jones et al, 2002, say that about 70% of papers describing the use of metaheuristics in solving multi-objective problems were about genetic algorithms! Some of the most referred to multi-objective evolutionary algorithms are MOGA (Fonseca and Fleming, 1993), NSGA (Srinivas and Deb, 1994), SPEA (Zitzler e Thiele, 1999), PAES (Knowles e Corne, 2002). More examples of the use of genetic or hybrid genetic algorithms in a multi-objective context can be found in Fonseca and Fleming, 1995a, 1995b; Lis and Eiben, 1996; Valenzuela-Rendón and Uresti-Charre, 1997; Coello Coello and Christiansen, 1998; Laumanns et al, 1998, 2001, 2002a,b; Rudolph, 1998; Azar et al, 1999; Deb, 1999a,b; Coello Coello, 1999a,b; Hanne, 1999; Van Veldhuizen, 1999; Van Veldhuizen and Lamont, 2000; Sbalzarini et al, 2000; Bingul et al, 2000; Rudolph and Agapie, 2000; Teghem, 2000; Zitzler e Thiele, 1998; Zitzler et al, 2000, 2002; Zitzler 2002; Jaszkiwicz, 2002; Kumar and Rockett, 2002; Sarker et al, 2002; Coello Coello et al, 2002; Deb et al, 2003; Coello Coello and Sierra, 2003; Coello and Becterra, 2003; Mirrazavi et al, 2003; Yen and Lu, 2003.

We will consider that the DM is interested in studying problem CDLPOCR under a multi-objective scenario, and that all the objective functions considered are linear and can be represented as (1). There are several examples in the location's field literature of interesting objective functions that can be represented in this form (Ross and Soland, 1980; Hultz et al, 1981; Reville and Laporte, 1996).

Most of the literature on multi-objective genetic algorithms chooses as the most important characteristic of the algorithm to be able to calculate a representative subset of the set of non-dominated solutions. It is important that the algorithm converges to the non-dominated set, without losing diversity (or else only a tiny part of the non-dominated set would be known).

Most of the times the interaction with the DM takes place *a priori* or *a posteriori*, i.e., before or after the calculation of the non-dominated solutions (exceptions are, for instance, Branke et al, 2001). As stated by Coello Coello, 2000c, there is very little work in the evolutionary algorithms' literature that handles explicitly with the DM preferences, assuming that preferences can change over time.

We share the opinion that the genetic algorithm should be used in interaction with the DM, converging to non-dominated solutions placed at a regions of interest for the DM (generally defined in the objective function space), and not loosing diversity so that it will be rather easy to jump from one area of interest into another. The DM can define areas of interest by establishing upper limits on the objective function values³², or explicitly defining weights for each objective function.

5.1 Algorithm's Details

Consider a population P of individuals x . Consider that $f_i(x)$ represents the fitness of x with respect to the i -th objective function, X is the set of all admissible solutions to CDLPOCR defined by restrictions (2)-(16) and, possibly, (18)-(20). Let us consider the following definitions:

Definition 3: An individual $x \in P$ is P -efficient if and only if there is no other individual $x' \in P$ such that $f_i(x') \leq f_i(x)$, for all objective functions i , and $f_i(x') < f_i(x)$ for at least one objective function i . Otherwise the individual x is said to be P -non efficient.

Definition 4: Consider two individual x and x' such that $f_i(x) \leq f_i(x')$, for all objective functions i , and $f_i(x) < f_i(x')$ for at least one objective function i . Then x is said to dominate x' .

Definition 5: Consider a population P such that all solutions $y \in X$ are represented by, at least, one individual x . If x is P -efficient, then x is efficient and the corresponding solution y is efficient. Otherwise y is not efficient.

Definition 6: If $y \in X$ is efficient, its image in the objective space, z , is non-dominated.

Proposition 4: If x is efficient then x is P -efficient for every possible population P considered.

Proposition 5: An individual x can be P_1 -efficient and P_2 -not efficient, if and only if P_2 has at least one individual $x' \notin P_1$ such that x' dominates x .

³² Considering all the objective functions are of the *minimization* type.

5.1.1 Fitness and Calculation of Optimal Assignments

The first problem faced when extending the hybrid algorithm to several objective functions is how to calculate the individuals' optimal assignments and their fitness.

To calculate optimal assignments one needs to have one objective function. It is also possible to consider several objectives in the assignment problem, but that would increase the computational time needed, and extra data structures would become necessary (because for two identical individuals, different assignment variables could be calculated).

The fitness of an individual x can be calculated in various forms (using linear or non-linear aggregating functions, consider the number of individuals that dominate or are dominated by x in the current population, ask the DM to make pair-wise comparisons between different individuals, etc³³).

We chose to use weights and to consider the fitness of an admissible individual equal to the weighted objective function value. The weighted function value is used to calculate optimal assignment variables, and is also used in the local search procedure. In local search procedure we consider the same weighted objective function used to calculate the fitness of the parents. Ishibuchi et al, 2003, consider that this is not a good approach, because the direction of search should be dependant of the relative position in the objective space of the starting solution. They describe an alternative way of performing the local search: generate a set of weights randomly (independent of the weights used in the parents' selection); apply the local search to a set of children selected from the population using tournament selection, with the fitness function equal to the weighted objective function. The authors argue that it is not worth to spend the time with local search around a poor quality individual. So, they only apply local search to individuals that have already passed the tournament selection test. This approach will also be tested in our algorithm and results compared with the present implementation.

The weights can have two different interpretations: if the DM wants, he can decide what weights to give to each objective function. Otherwise, the weights are calculated by the algorithm and are only seen as a technical tool.

5.1.2 Populations

The present implementation of our algorithm works with two populations. Borrowing the notation of Van Veldhuizen and Lamont, 2000, $P_{current}(ger)$ represents the population at generation ger . Population $P_{known}(ger)$ is composed of all efficient individuals in the set

³³ See Coello Coello et al, 2002, for a review.

$\bigcup_{g=1}^{ger} P_{current}(g)$. Notice that there can be individuals that are efficient with respect to $P_{current}$ and

not efficient with respect to P_{known} . Furthermore, there can be individuals that are efficient with respect to $P_{known}(ger)$ and not efficient with respect to $P_{known}(ger+g)$, $g>0$. This means that it is not sufficient to copy all efficient individuals from $P_{current}(ger)$ to $P_{known}(ger)$: it is also necessary to verify the efficiency of all the elements of $P_{current}(ger)$ and $P_{known}(ger-1)$. This procedure is an $O(n^2)$ algorithm (n represents the number of individuals in the population, Van Veldhuizen and Lamont, 2000), so it should not be performed too often. It is also important to notice that population P_{known} has limited capacity: if the algorithm finds many efficient solutions, then it is necessary to update the set, possibly eliminating some individuals and inserting others. Even considering that P_{known} could have an unlimited number of individuals, it would not be reasonable to show a great number of solutions to the DM. So, it will always be necessary to devise some kind of procedure to choose a subset of efficient solutions from P_{known} . Another interesting subject is to define how the two populations will interact between them and with the algorithm.

In our algorithm, the set $P_{known}(ger)$ is equal to set $P_{known}(ger-1) \cup P_{current}(ger)$. The non efficient individuals in P_{known} are only deleted from this population if the DM wants to see all P_{known} -efficient solutions calculated thus far or if the number of elements in $P_{current}(ger)$ is greater than the remaining free capacity of P_{known} . In the latter case only efficient individuals are inserted and all non-efficient individuals are deleted from P_{known} ³⁴. If P_{known} has reached its maximum capacity and has only efficient individuals, some of them will have to be removed. This is done in the following way: if there are efficient individuals belonging to P_{known} that are violating the present DM preferences, then these are candidates for deletion. The algorithm chooses the one that is in P_{known} for a longer number of generations. If none of the individuals violates the preferences of DM, then the algorithm chooses randomly an individual. Many other procedures could be devised: asking the DM which individual he wishes to remove or maintain, applying clustering algorithms in order to insure that set P_{known} maintains a good diversified efficient solution set, etc.

In our algorithm, population P_{known} has no participation in crossover or selection operators. None of its members interact with individuals in $P_{current}$. There are authors who feel that a close interaction between the two populations can only benefit the algorithm (Zitzler and Thiele, 1999). One possible interaction between the two populations could be easily implemented:

³⁴ We do not allow the existence of replicated individuals in population P_{known} .

whenever the number of individuals in $P_{current}$ is increased, the new individuals could be randomly chosen from P_{known} , instead of randomly initialized.

The P_{known} -efficient individuals are, at each generation, an approximation of the true efficient solutions set (that is unknown). There are many different (and non consensual) ways of evaluating the quality of approximations to the non-dominated solutions set (Hansen and Jaskiewicz, 1998). In our algorithm that is of secondary importance, due to the interaction with the DM. It is possible to assess if a given individual x represents an efficient solution $y \in X$ by the resolution of a linear mixed-integer programming problem that is obtained by the insertion of as many additional restrictions as there are objectives, and considering an objective function equal to a weighted sum of all objective functions. If there are p objective functions, then the mixed-integer linear programming problem would be:

CDLPOCR-1:

$$\begin{aligned} \text{Min} \sum_p \lambda_p \left(\sum_{t \in I_4} \sum_j \sum_s c^{p,t}_{ijs} x^t_{ijs} + \sum_{t \in I_4} \sum_s \sum_{\xi=t}^T FA^{p,\xi}_{ist} a^{\xi}_{ist} + \sum_{t \in I_4} \sum_s \sum_{\xi=t}^T FR^{p,\xi}_{ist} r^{\xi}_{ist} + \right. \\ \left. \sum_{t \in I/I_4} \sum_j c^{p,t}_{ij} x^t_{ij} + \sum_{t \in I/I_4} \sum_{\xi=t}^T FA^{p,\xi}_{it} a^{\xi}_{it} + \sum_{t \in I/I_4} \sum_{\xi=t}^T FR^{p,\xi}_{it} r^{\xi}_{it} \right), \end{aligned} \quad (21)$$

with $\sum_p \lambda_p = 1, \lambda_p > 0, \forall p$

Subject to: (2)-(16) and, possibly, (18)-(20) and

$$\begin{aligned} \sum_{t \in I_4} \sum_j \sum_s c^{p,t}_{ijs} x^t_{ijs} + \sum_{t \in I_4} \sum_s \sum_{\xi=t}^T FA^{p,\xi}_{ist} a^{\xi}_{ist} + \sum_{t \in I_4} \sum_s \sum_{\xi=t}^T FR^{p,\xi}_{ist} r^{\xi}_{ist} + \\ \sum_{t \in I/I_4} \sum_j c^{p,t}_{ij} x^t_{ij} + \sum_{t \in I/I_4} \sum_{\xi=t}^T FA^{p,\xi}_{it} a^{\xi}_{it} + \sum_{t \in I/I_4} \sum_{\xi=t}^T FR^{p,\xi}_{it} r^{\xi}_{it} \leq M_p - \varepsilon \end{aligned}, \quad \forall p \quad (22)$$

Here, M_p should be equal to the p -th objective function value for the particular individual whose efficiency the DM wants to confirm and $\varepsilon \rightarrow 0$. If this problem has an admissible solution, then y was not an efficient solution and a true efficient solution with respect to X is found. If the problem is impossible then y is an efficient solution.

Ross and Soland, 1980, prove that the optimal solution to problem CDLPOCR-1 is always an efficient solution to the multi-objective version of CDLPOCR, and by changing the right hand side of the additional restrictions (22) it is possible to calculate all the efficient solutions,

even the non-supported ones. The right hand sides M_p of these solutions are also of use in the interaction with the DM.

Population P_{known} is initialized as an empty set. Population $P_{current}$ is initialized randomly, as in the mono-objective case, but considering several different sets of weights, also randomly generated by the algorithm.

5.1.3 Interaction with the DM

After initializing population $P_{current}$, this population evolves during a predefined number of generations, considering different randomly generated sets of weights. The calculated P_{known} -efficient solutions are showed to the DM. Showing to the DM an initial set of solutions instead of only one can be advantageous, because it can reduce the *anchoring* effect. If the DM feels that he has already found a good compromise solution, then the algorithm stops. Otherwise, the algorithm asks him to express his preferences. He can do this in two different ways: by establishing upper limits to each objective function values and/or by giving explicitly the objective function weights. The additional restrictions (22) are treated by the algorithm as described in section 4.2.

If there are only two objective functions, establishing upper limits for each objective function can be done in a straightforward way, as described in Ferreira, 1997, Dias, 1999 and Dias et al, 2003. If this is the case, the algorithm calculates the weights automatically, considering the limits imposed by the DM and trying to improve the algorithm's performance (Dias, 1999, Dias et al, 2003).

The weights are calculated considering the previous right hand side values and the objective function values of the best individual created. If there are more than three objectives, the weights are randomly generated.

At each iteration, a predefined number of generations are executed and the best solution found considering the actual weighted objective function is presented to the DM. Whenever he wants, he can see all the P_{known} -efficient solutions.

The algorithm does not consider the existence of irrevocable decisions from the DM: he can change his preferences, search areas of solutions that he had abandoned in earlier iterations, present an incoherent behavior during the iterations. The DM is the only responsible for the termination of the algorithm. The interaction between the DM and the algorithm is depicted in fig. 4, and is motivated by the work of Dias, 1999 and Ferreira, 1997. This interaction can be facilitated through the use of visual tools, especially in the biobjective case. Algorithm 8 depicts the operating scheme of the hybrid algorithm for multi-objective problems.

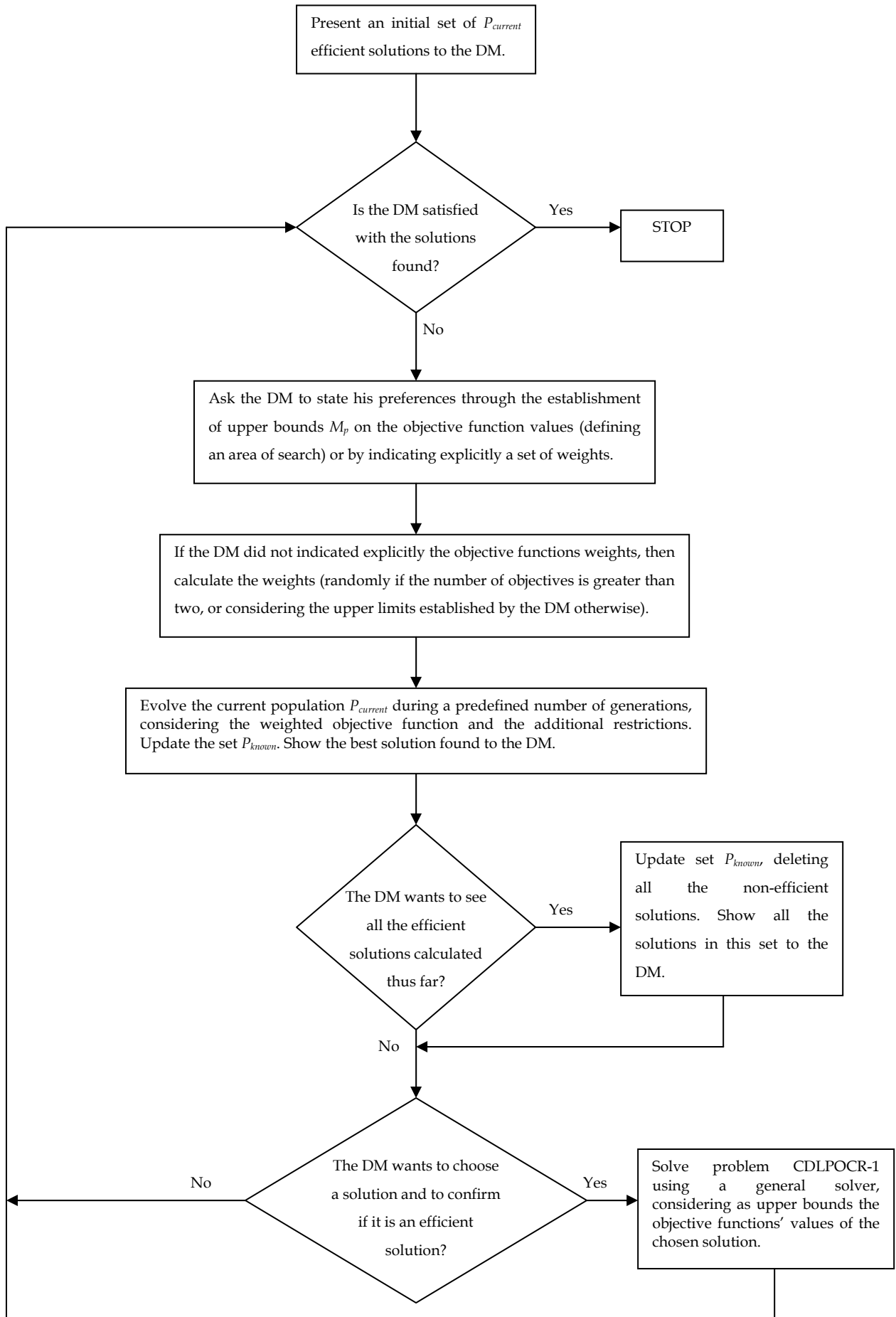


Figure 4: Scheme of the interaction with the DM.

Algorithm 8: Multi-objective Hybrid Algorithm

*n*generations – maximum number of generations population $P_{current}$ is evolved in each iteration in the first phase of the algorithm.
N – Maximum number of iterations in the first phase of the algorithm.

1. $P_{current} \leftarrow \emptyset, P_{known} \leftarrow \emptyset$.
 2. Initialise randomly population $P_{current}$, calculating the fitness of each individual using a random weighted objective function.
 3. $n \leftarrow 1$.
 4. Randomly generate a valid set of weights for the objective functions.
 5. Evolve $P_{current}$ during *n*generations and update population P_{known} .
 6. $n \leftarrow n + 1$. If *n* is greater than *N* then go to 7. Else go to 4.
 7. Show population P_{known} to the DM. If the DM is satisfied then stop, else go to 8.
 8. Ask the DM to establish limits M_p and/or weights for each objective function *p*.
 9. Evolve $P_{current}$ using algorithm 6, steps 2-6. Update set P_{known} .
 10. Show solution represented by x_{best} to the DM. If the DM wants, show all P_{known} -efficient solutions. If the DM is satisfied stop else go to 8.
-
-

6 Conclusions

The previous work on dynamic location problems (Dias et al, 2004a,b) motivated the development of a genetic algorithm capable of overcoming the weaknesses of primal-dual heuristics. It is a challenging task to move from a very structured and rigid heuristic to a metaheuristic where everything is possible and is impossible to test all different possibilities.

We can conclude that, for this kind of problems, genetic algorithms have to be hybridised with local search procedures, otherwise they will have a weak performance. We were able to obtain good results even in the presence of capacitated facilities.

From the experiences already made, we can say that:

- 1- In comparison with primal-dual heuristics, the hybrid algorithm needs much longer computational times, and sometimes reaches better solutions. Nevertheless, the primal-dual heuristics present a better compromise between solution quality and computational time needed.
- 2- The consideration of facilities of type 3 makes the problem much harder to solve. So, whenever possible, is better to consider facilities of type 3 as facilities of type 2, with a given maximum and/or minimum capacities that can differ from one time period to another.

- 3- The hybrid algorithm deals quite efficiently with additional restrictions that fix some facilities open or close during some time periods. It would be difficult to include this kind of restrictions in the primal-dual heuristics.
- 4- The hybrid algorithm has some difficulties dealing with general linear additional restrictions, especially due to the representation of solutions chosen: only location variables are represented and the allocation variables have to be calculated separately. It is interesting to note that if the additional constraints only involve the location variables, then the performance of the algorithm does not deteriorate in presence of these restrictions.
- 5- The primal-dual heuristics developed previously by the authors could only be used in a multi-objective context if the several objective functions were weighted. The resolution of a series of problems with different objective function weights would calculate a set of non-dominated (or near non-dominated) solutions. The hybrid algorithm is easily extended to be able to calculate sets of (possibly) non-dominated solutions in regions of interest for the DM.

The results obtained thus far encourage us to follow this line of work and extend the use of this hybrid algorithm to the resolution of multi-level dynamic facility location problems (Dias et al, 2004c), as well as to situations with several decision makers.

7 References

- Abdinnour-Helm, S. 1998. A Hybrid Heuristic for the Uncapacitated Hub Location Problem, *European Journal of Operational Research*, 106, pp 489-499
- Agar, M. C., Salhi, S. 1998. Lagrangean Heuristics Applied to a Variety of Large Capacitated Plant Location Problems, *Journal of the Operational Research Society*, 49, pp 1072-1084
- Alp, O., Drezner, Z., Erkut, E. 2003. An Efficient Genetic Algorithm for the P-Median Problem, *Annals of Operations Research*, 122, pp 21-42
- Alves, M. L., Almeida, M. T. 1992. Simulated Annealing Algorithm for the Simple Plant Location Problem: a Computational Study, *Investigação Operacional*, 12, pp 145-157
- Antunes, A., Peeters, D. 2001. On solving complex multi-period location models using simulated annealing, *European Journal of Operational Research*, 130, pp 190-201
- Azar, S., Reynolds, B. J., Narayanan, S. 1999. Comparison of Two Multiobjective Optimization Techniques with and within Genetic Algorithms, *Proceedings of the 1999 ASME Design Engineering Technical Conferences*, September 12-15, Las Vegas, Nevada
- Bäck, T., Eiben, A. E., Van der Vaart, N.A.L. 2000. An Empirical Study on Gas "Without Parameters", *Lecture Notes in Computer Science*, 1917, pp 315-324

- Barceló, J., Casanovas, J. 1984. A Heuristic Lagrangean Algorithm for the Capacitated Plant Location Problem, *European Journal of Operational Research*, 15, pp 212-226
- Beasley, J. E. 1993. Lagrangean Heuristics for Location Problems, *European Journal of Operational Research*, 65, pp 383-399
- Bingul, Z., Sekmen, A. S., Palaniappan, S., Sabatto, S. 2000. Genetic Algorithms Applied to Real Time Multiobjective Optimization Problems, *Proceedings of the 2000 IEEE Southeast Conference*, April 2000 Nashville, TN, USA, pp 95-103
- Bornstein, C. T., Azlan, H. B. 1998. The Use of Reduction Tests and Simulated Annealing for the Capacitated Plant Location Problem, *Location Science*, 6, pp 67-81
- Branke, J., Kaussler, T., Schmeck, H. 2001. *Guidance in Evolutionary Multi-Objective Optimization*, *Advances in Engineering Software*, Elsevier Publisher, 32, pp 499-507
- Coello Coello, C. A. 1999a. An Updated Survey of Evolutionary Multiobjective Optimization Techniques: State of the Art and Future Trends, *Special Session on Multiobjective Optimization at the 1999 Congress on Evolutionary Computation*, pp 3-13
- Coello Coello, C. A. 1999b. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques, *Knowledge and Information Systems*, 1, pp 269-308
- Coello Coello, C. A., 2000a. Treating Constraints as Objectives for Single-Objective Evolutionary Optimization, *Engineering Optimization*, 32, pp 275-308
- Coello Coello, C. A., 2000b. Constraint Handling using an Evolutionary Multiobjective Optimization Technique, *Civil Engineering Systems*, Gordon and Breach Science Publishers, 17, pp 319-346
- Coello Coello, C. A. 2000c. Handling Preferences in Evolutionary Multiobjective Optimization: A Survey, *2000 Congress on Evolutionary Computation*, 1, pp 30-37
- Coello Coello, C. A. 2002. Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey and the State of the Art, *Computer Methods in Applied Mechanics and Engineering*, 191, pp 1245-1287
- Coello Coello, C. A., Bécerra, R. L. 2003, *Evolutionary Multobjective Optimization using a Cultural Algorithm*, 2003 IEEE Swarm Intelligence Symposium, IEEE Service Center, Indianapolis, USA, April 2003, pp 6-13
- Coello Coello, C. A., Christiansen, A. D. 1998. Two new GA-based Methods for Multiobjective Optimization, *Civil Engineering Systems*, Gordon and Breach Science Publishers, 15, pp 207-243
- Coello Coello, C. A., Cortés, N. C. 2001. Constraint-Handling in Genetic Algorithms through Emulations of the Immune System, in *Tercer Encuentro Internacional de Ciencias de la Computación (ENC'01)*, Tomo I, Zozaya, C., Mejia, M., Noriega, P. and Sánchez, A. (Edts), pp 115-124
- Coello Coello, C. A., Montes, E. M. 2002. Constraint-Handling in Genetic Algorithms Through the Use of Dominance-based Tournament Selection, *Advanced Engineering Informatics*, 16, pp193-203
- Coello Coello, C. A., Sierra, M. R. 2003. A Multi-Objective Evolutionary Algorithm Based on Coevolutionary Concepts, *Technical Report Evolutionary Computation Group at CINVESTAV Sección de Computación, Departamento de Ingeniería Eléctrica, México*
- Coello Coello, C. A., Van Veldhuizen, D. A., Lamont, G. A. 2002. *Evolutionary Algorithms for Solving Multiobjective Problems*, Kluwer Academic Publishers
- Cornuejols, G., Sridharan, R., Thizy, J. M. 1991. A Comparison of Heuristics and Relaxations for the Capacitated Plant Location Problem, *European Journal of Operational Research*, 50, pp 280-297
- Correa, E. S., Steiner, M. T. A., Freitas, A. A., Carnieri, C. 2001. A Genetic Algorithm for the P-Median Problem, *Proceedings of 2001 Genetic and Evolutionary Computation GECCO2001*, pp 1268-1275

- Cortinhal, M.J., Captivo, M.E. (2003) Genetic Algorithms for the single source capacitated location problem, in *Metaheuristics: Computer Decision Making*, Mauricio G. C. Resende and Jorge P. de Sousa Edts, *Combinatorial Optimization Book Series*, D.-Z.Du and P.M.Pardalos Series Editor, Kluwer Academic Publishers, pp 187-216
- Davis, L. 1996. *Handbook of Genetic Algorithms*, Van Nostrand Reinhold
- Deb, K. 1999a. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems, *Evolutionary Computation*, 7, pp 205-230
- Deb, K. 1999b. Multi-Objective Evolutionary Algorithms: Introducing Bias Among Pareto-Optimal Solutions, Indian Institute of Technology Kanpur, KanGAL Report
- Deb, K. 2001. *Multi-objective optimization using evolutionary algorithms*, John Wiley & Sons, Wiley Interscience Series in Systems and Optimization
- Deb, K., Mohan, M., Mishra, S. 2003. A Fast Multi-Objective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions, Indian Institute of Technology Kanpur, KanGAL Report
- Dias, J. 1999. *Localização Simples Multicritério – Desenvolvimento de um algoritmo em ambiente interactivo*, MsC thesis, University of Lisbon (in portuguese)
- Dias, J. , Captivo, M. E., Clímaco, J. 2003 An Interactive Procedure Dedicated to a Bicriteria Plant Location Model, *Computers & Operations Research*, 30, pp 1977-2002
- Dias, J. , Captivo, M. E., Clímaco, J. 2004a. Efficient Primal-Dual Heuristic for a Dynamic Location Problem *submitted*
- Dias, J. , Captivo, M. E., Clímaco, J. 2004b. Capacitated Dynamic Location Problems with Opening, Closure and Reopening of Facilities, INESC-Coimbra Research Report, 2/2004
- Dias, J., Captivo, M. E., Clímaco, J. 2004c. Dynamic Multi-Level Capacitated and Uncapacitated Location Problems: an approach using primal-dual heuristics, INESC-Coimbra Research Report, 26/2004
- Domschke, W., Drexl, A. 1985. ADD-Heuristics' Starting Procedures for Capacitated Plant Location Models, *European Journal of Operational Research*, 21, pp 47-53
- Eiben, A. E. 2001. Evolutionary Algorithms and Constraint Satisfaction: Definitions, Survey, Methodology and Research Directions, in *Theoretical Aspects of Evolutionary Computing*, L- Kallel, Nandts, B., Rogers, A. (Edts), *Natural Computing Series*, Springer, pp 13-58
- Eiben, A. E., Hinterding, R., Michalewicz, Z. 1999. Parameter Control in Evolutionary Algorithms, *IEEE Transactions on Evolutionary Computation*, 3, pp 124-141
- Eiben, A. E., Smith, J. E. 2003. *Introduction to Evolutionary Computing*, Springer
- Espejo, L. G. A., Galvão, R. D., Boffey, B. 2003. Dual-based Heuristics for a Hierarchical Covering Location Problem, *Computers & Operations Research*, 30, pp 165-180
- Ferreira, C. 1997. *Problemas de Localização e Distribuição Multicritério – Aproximações e Estudos de Alguns Casos com Implicações Ambientais*, PhD thesis, University of Aveiro (in Portuguese)
- Filho, V. J. M. F., Galvão, R. D. 1998. A Tabu Search Heuristic for the Concentrator Location Problem, *Location Science*, 6, pp 189-209
- Filipovic, V., Kratica, J., Tasic, D., Ljubic, I. 2000, Fine Grained Tournament Selection for the Simple Plant Location Problem, *Proceedings of the 5th Online World Conference on Soft Computing Methods in Industrial Applications WSC5*, pp152-158
- Fonseca, C. M., Fleming, P. J. 1993. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, *Genetic Algorithms: Proceedings of the Fifth International Conference*, San Mateo, CA

- Fonseca, C. M., Fleming, P. J. 1995a. Multiobjective Genetic Algorithms Made Easy: Selection, Sharing and Mating Restriction, The Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95), pp 45-52
- Fonseca, C. M., Fleming, P.J. 1995b. An Overview of Evolutionary Algorithms in Multiobjective Optimization, *Evolutionary Computation*, 3, pp 1-16
- Galvão, R. D., Santibañez-Gonzalez, E. R. 1992. A Lagrangean Heuristic for the P-Median Dynamic Location Problem, *European Journal of Operational Research*, 58, pp 250-262
- Glover, F. Kochenberger, G. 2003. *Handbook of Metaheuristics* Kluwer Academic Publishers
- Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA
- Hanne, T. 1999. On the Convergence of Multiobjective Evolutionary Algorithms, *European Journal of Operational Research*, 117, pp 553-564
- Hansen, M., Jazskiewicz, A. 1998. Evaluating the Quality of Approximations to the Non-Dominated Set, Technical University of Denmark, IMM Technical Report IMMREP1998-7
- Hansen, P. H., Hegedahl, B., Hjortkjaer, S., Odel, B. 1994. A Heuristic Solution to the Warehouse Location-Routing Problem, *European Journal of Operational Research*, 76, pp 111-127
- Hertz, A., Widmer, M. 2003. Guidelines for the Use of Metaheuristics in Combinatorial Optimization, *European Journal of Operational Research*, 151, pp 247-252
- Holmberg, K., Ling, J. 1997. A Lagrangean Heuristic for the Facility Location Problem with Staircase Costs, *European Journal of Operational Research*, 97, pp 63-74
- Houck, C., Joines, J., Kay, M. 1996. Comparison of Genetic Algorithms, Random Restart and Two-Opt Switching for Solving Large Location-Allocation Problems, *Computers & Operations Research*, 23, pp 587-596
- Hultz, J., Klingman, D., Ross, G. T., Soland, R. 1981. An Interactive Computer System for Multicriteria Facility Location, *Computers & Operations Research*, 8, pp 249-261
- Huntley, C., Brown, D. 1996. Parallel Genetic Algorithms with Local Search, *Computers & Operations Research*, 23, pp 559-571
- Ishibuchi, H., Yoshida, T., Murata, T. 2002. Balance between Genetic Search and Local Search in Hybrid Evolutionary Multi-Criterion Optimization Algorithms, *Evolutionary Computation*, CSE 848, Fall 2002
- Ishibuchi, H., Yoshida, T., Murata, T. 2003. Balance Between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling, *IEEE Transactions on Evolutionary Computation*, 7, pp 204-223
- Jacobsen, S. K. 1983. Heuristics for the Capacitated Plant Location Model, *European Journal of Operational Research*, 12, pp 253-261
- Jaramillo, J. 1998. *Genetic Algorithms for Location Problems*, MsC Thesis, Department of Industrial Engineering, State University of New York at Buffalo
- Jaramillo, J., Bhadury, J., Batta, R. 2002. On the Use of Genetic Algorithms to Solve Location Problems, *Computers & Operations Research*, 29, pp 761-779
- Jazskiewicz, A. 2002. Genetic Local Search for Multi-objective Combinatorial Optimization, *European Journal of Operational Research*, 137, pp 50-71
- Jones, D. F., Mirrazavi, S. K., Tamiz, M. 2002. Multi-Objective Metaheuristics: An Overview of the Current State-of-the-art, *European Journal of Operational Research*, 137, pp 1-9

- Klincewicz, J., Luss, H. 1986. A Lagrangian Relaxation Heuristic for Capacitated facility Location with Single-Source Constraints, *Journal of the Operational Research Society*, 37, pp 495-500
- Klose A. 1999. An LP-based heuristic for two-stage capacitated facility location problems, *Journal of the Operational Research Society*, 50, pp 157-166
- Klose, A. 1995. A Lagrangean Heuristic to Solve the Two-stage Capacitated Facility Location Problem, Working Paper - Institut für Unternehmensforschung - Universität St. Gallen
- Knowles, J. D., Corne, D. W. 2002. Properties of an Adaptive Archiving Algorithm for Storing Nondominated Vectors, *IEEE Transactions on Evolutionary Computation*, 7, pp 100-116
- Kratka, J. 1999. Improvement of Simple Genetic Algorithm for Solving the Uncapacitated Warehouse Location Problem, *Advances in Soft Computing, Engineering Design and Manufacturing*, pp 390-402
- Kratka, J., Filipovic, V., Sesum, V., Tomic, D. 1996. Solving the Uncapacitated Warehouse Location Problem Using a Simple Genetic Algorithm, *Proceedings of the XIV International Conference on Material Handling and Warehousing, Belgrade*, pp 3.33-3.37
- Kratka, J., Tomic, D., Filipovic, V., Ljubic, I. 2001. Solving the Simple Plant Location Problem by Genetic Algorithm, *RAIRO Operations Research*, 35, pp 127-142
- Kumar, R., Rockett, P. 2002. Improved Sampling of the Pareto-Front in Multiobjective Genetic Optimizations by Steady-State Evolution: A Pareto Converging Genetic Algorithm, *Evolutionary Computation*, 10, pp 283-314
- Laumanns, M., Rudolph, G., Schwefel, H.-P. 1998. A Spatial Predator-Key Approach to Multi-Objective Optimization: A Preliminary Study, *PPSN V*, pp 241-249
- Laumanns, M., Thiele, L., Deb, K., Zitzler, E. 2001. On the Convergence and Diversity-Preservation Properties of Multi-Objective Evolutionary Algorithms, Technical Report, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich
- Laumanns, M., Thiele, L., Deb, K., Zitzler, E. 2002a. Combining Convergence and Diversity in Evolutionary Multi-Objective Optimization, *Evolutionary Computation*, 10, pp 263-282
- Laumanns, M., Thiele, L., Zitzler, E., Deb, K. 2002b. Archiving with Guaranteed Convergence and Diversity in Multi-Objective Optimization, *GECCO 2002, Proceedings of the Genetic and Evolutionary Computation Conference, July 2002*, pp 439-447
- Levin, Y., Ben-Israel A., 2004. A Heuristic Method for Large-Scale Multi-Facility Location Problems, *Computers & Operations Research*, 31, pp 257-272
- Lis, J., Eiben, E. E. 1996. Multi-Sexual Genetic Algorithm for Multiobjective Optimization, *Proceedings of the 1996 International Conference on Evolutionary Computation*
- Lorena, L., Lopes, L. de S. 1997. Genetic Algorithms Applied to Computationally Difficult Set Covering Problems, *Journal of the Operational Research Society*, 48, pp 440-445
- Luss, H. 1982. Operations Research and Capacity Expansion Problems, *Operations Research*, 30, pp 907-947
- Maniezzo, V., Mingozzi, A., Baldacci, R. 1998 A Bionomic Approach to the Capacitated p-Median Problem, *Journal of Heuristics*, 4, pp 263-280
- Michalewicz, Z. 1995. A Survey of Constraint Handling Techniques in Evolutionary Computation Methods, *Proceedings of the Fourth Annual Conference on Evolutionary Programming*
- Min, H. 1988. The Dynamic Expansion and Relocation of Capacitated Public Facilities: a Multi-Objective Approach, *Computers & Operations Research*, 15, pp 243-252
- Mirrazavi, S. K., Jones, D. F., Tamiz, M. 2003. MultiGen: an Integrated Multiple-objective Solution System, *Decision Support Systems*, 36, pp 177-187

- Mitchel, M. 1996 An Introduction to Genetic Algorithms, The MIT Press
- Moscato, P., Cotta, C. 2003. A Gentle Introduction to Memetic Algorithms, F. Glover, G. Kochenberger (eds.), Handbook of Metaheuristics, pp. 105-144, Kluwer Academic Publishers, Boston MA
- Murty, K. 1983. Linear Programming, John Wiley & Sons
- Oei, C. K., Goldberg, D. E., Chang, S.-J. 1991. Tournament Selection, Niching and the Preservation of Diversity, Illinois Genetic Algorithms Laboratory (IlligAL) Report 91011
- Osman, I. H., Kelly, J. P. (Edts) 1996. MetaHeuristics: Theory & Applications, Kluwer Academic Publishers
- Owen, S. H., Daskin, M. 1998a. Strategic Facility Location via Evolutionary Programming, Working Paper, Department of Industrial Engineering and Management Sciences, Northwestern University
- Owen, S. H., Daskin, M. 1998b. A Note on Evolution Programs for Solving Multi-Objective Strategic Facility Location Problems, Working Paper, Department of Industrial Engineering and Management Sciences, Northwestern University
- Pirkul, H., Jayaraman, V. 1998. A Multi-Commodity, Multi-Plant, Capacitated Facility Location Problem: Formulation and Efficient Heuristic Solution, Computers & Operations Research, 25, pp 869-878
- Reeves, C., Höhn, C. 1996. Integrating Local Search into Genetic Algorithms, *in* Modern Heuristic Search Methods, pp 99-115
- Reeves, C. R. 1993a. Using Genetic Algorithms With Small Populations, Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA
- Reeves, C. R. 1993b. Modern Heuristic Techniques for Combinatorial Problems, Blakwell Scientific Publications
- Revelle, C., Laporte, G. 1996. The Plant Location Problem: New Models and Research Prospects, Operations Research, 44, pp 864-873
- Righini, G. 1995. A Double Annealing Algorithm for Discrete Location/Allocation Problems, European Journal of Operational Research, 86, pp 452-468
- Rolland, E., Schilling, D., Current, J. 1996. An Efficient Tabu Search Procedure for the p-Median Problem, European Journal of Operational Research, 96, pp 329-342
- Rönqvist, M., Tragantalermsak, S., Holt, J. 1999. A Repeated Matching Heuristic for the Single-Source Capacitated Facility Location Problem, European Journal of Operational Research, 116, pp 51-68
- Rosing, K. E., Hodgson, M. J. 2002. Heuristic Concentration for the P-Median: an Example Demonstrating how and why it works, Computers & Operations Research, 29, pp 1317-1330
- Rosing, K. E., ReVelle, C. S., Schilling, D. A. 1999. A Gamma Heuristic for the p-Median Problem, European Journal of Operational Research, 117, pp 522-532
- Ross, T., Soland, R. 1980. A Multicriteria Approach to the Location of Public Facilities, European Journal of Operational Research, 4, pp 307-321
- Rothlauf, F., Goldberg, D. 2002. Redundant Representations in Evolutionary Computation, Illinois Genetic Algorithms Laboratory (IlligAL) Report
- Rudolph, G. 1998. On a Multi-Objective Evolutionary Algorithm and Its Convergence to the Pareto Set, Proceedings of the Fifth IEEE Conference on Evolutionary Computation, pp 511-516
- Rudolph, G., Agapie, A. 2000. Convergence Properties of Some Multi-Objective Evolutionary Algorithms, Proceedings of the 2000 Conference on Evolutionary Computation, 2, pp 1010-1016
- Saldanha da Gama, F., Captivo, M. E. 1998. A Heuristic Approach for the Discrete Dynamic Location Problem, Location Science, 6, pp 211-223

- Saldanha da Gama, F., Captivo, M. E. 2002. A Branch-and-Bound Procedure for the Multi-Period Capacitated Location Problem, Research Report 8/2002, Centro de Investigação Operacional, Faculdade de Ciências da Universidade de Lisboa
- Salhi, S., Atkinson, R. A. 1995. Subdrop: A Modified Drop Heuristic for Location Problems, *Location Science*, 3, pp 267-273
- Sarker, R., Liang, K.-H., Newton, C. 2002. A New Multiobjective Evolutionary Algorithm, *European Journal of Operational Research*, 140, pp 12-23
- Sbalzarini, I. F., Müller, S., Koumoutsakos, P. 2000. Multiobjective Optimization Using Evolutionary Algorithms, Center for Turbulence Research, Proceedings of the Summer Program
- Schaffer, J. D., 1985. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, *Genetic Algorithms and Their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pp 93-100
- Shimizu, Y. 1999. Multi-Objective Optimization for Site Location Problems through Hybrid Genetic Algorithm with Neural Networks, *Journal of Chemical Engineering of Japan*, 32, pp 51-58
- Shimizu, Y., Wada, T. 2003. Logistic Optimization for Site Location and Route Selection Under Capacity Constraints Using Hybrid Tabu Search Proceedings of the 8th International Symposium Process Systems Engineering, China (PSE'03)
- Shulman, A. 1991. An Algorithm for Solving Dynamic Capacitated Plant Location Problems with Discrete Expansion Sizes, *Operations Research*, 39, pp 423-436
- Sinha, A., Goldberg, D. 2003. A Survey of Hybrid Genetic and Evolutionary Algorithms, Illinois Genetic Algorithms Laboratory (IlliGAL) Report, 2003004
- Sridharan, R. 1991. A Lagrangian Heuristic for the Capacitated Plant Location with Side Constraints, *Journal of the Operational Research Society*, 42, pp 579-585
- Sridharan, R. 1993. A Lagrangian Heuristic for the Capacitated Plant Location Problem with Single Source Constraints, *European Journal of Operational Research*, 66, pp 305-312
- Srinivas, N., Deb, K. 1994. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, *Evolutionary Computation*, 2, pp 221-248
- Sun, M., Aronson, J., McKeown, P., Drinka, D. 1998. A Tabu Search Heuristic Procedure for the Fixed Charge Transportation Problem, *European Journal of Operational Research*, 106, pp 441-456
- Taillard, E., Gambardella, L., Gendreau, M., Potvin, J.-Y. 2001. Adaptive memory Programming: a Unified View of Metaheuristics, *European Journal of Operational Research*, 135, pp 1-6
- Tcha, D.-W., Ro, H.-B., Yoo, C.-B. 1988. A Dual-Based Add Heuristic for Uncapacitated Facility Location, *Journal of the Operational Research Society*, 39, pp 873-878
- Teghem, J. 2000. Multiobjective Combinatorial Optimization, Parallel Problem Solving from Nature, workshop on Multiobjective Problem Solving from Nature, Paris, France, September 2000
- Toth, P. 2000. Optimization engineering techniques for the exact solution of NP-hard combinatorial optimization problems, *European Journal of Operational Research*, 125, pp 222-238
- Tragantalerngsak, S., Holt, J., Rönnqvist, M. 1997. Lagrangian Heuristics for the Two-Echelon, Single-Source, Capacitated Facility Location Problem, *European Journal of Operational Research*, 102, pp 611-625
- Vaithyanathan, S., Burke, L., Magent, M. 1996. Massively Parallel Analog Tabu Search Using Neural Networks Applied to Simple Plant Location Problems, *European Journal of Operational Research*, 93, pp 317-330

- Valenzuela-Rendón, M., Uresti-Charre, E. 1997, A Non-Generational Genetic Algorithm for Multiobjective Optimization, Proceedings of the Seventh International Conference on Genetic Algorithms
- Van Veldhuizen, D. 1999. Multiobjective Evolutionary Algorithms: Classifications, Analysis and New Innovations, Phd Thesis, Air Force Institute of Technology, Faculty of the Graduate School of Engineering
- Van Veldhuizen, D. A., Lamont, G. B. 2000. Multiobjective Evolutionary Algorithms: Analysing the State-of-the-Art, *Evolutionary Computation*, 8, pp 125-147
- Wu, T.-H., Low, C., Bai, J.-W. 2002. Heuristic Solutions to Multi-Depot Location-Routing Problems, *Computers & Operations Research*, 29, pp 1393-1415
- Yagiura, M., Ibaraki, T. 1996. The Use of Dynamic Programming in Genetic Algorithms for Permutation Problems, *European Journal of Operational Research*, 92, pp 387-401
- Yen, G. G., Lu, H. 2003. Dynamic Multiobjective Evolutionary Algorithm: Adaptive Cell-Based Rank and Density Estimation, *IEEE Transactions on Evolutionary Computation*, 7, pp 253-274
- Zitzler, E. 2002. Evolutionary Algorithms for Multiobjective Optimization, in *Evolutionary Methods for Design, Optimisation and Control*, Giannakoglou, K., Tsahalis, D., Periaux, J., Papailou, K., Fogarty, T. (Edts)
- Zitzler, E., Laumanns, M., Thiele, L. 2002. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization, in *Evolutionary Methods for Design, Optimisation and Control*, Giannakoglou, K., Tsahalis, D., Periaux, J., Papailou, K., Fogarty, T. (Edts)
- Zitzler, E., Thiele, L. 1998. Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study, *Parallel Problem Solving from Nature*, PPSN, pp 292-301
- Zitzler, E., Thiele, L. 1999. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, *IEEE Transactions on Evolutionary Computation*, 3, pp 257-271
- Zitzler, E., Thiele, L., Deb, K. 2000. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results, *Evolutionary Computation*, 8, pp 173-195