J Glob Optim (2008) 42:221–253
DOI 10.1007/s10898-007-9239-9

A memetic algorithm for multi-objective dynamic location problems

Joana Dias · M. Eugénia Captivo · João Clímaco

Received: 10 September 2007 / Accepted: 13 September 2007 / Published online: 13 October 2007

Abstract This paper describes a new multiobjective interactive memetic algorithm applied to dynamic location problems. The memetic algorithm integrates genetic procedures and local search. It is able to solve capacitated and uncapacitated multi-objective single or multi-level 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 possible to distinguish the opening and reopening periods, assigning them different coefficient values in the objective functions. The algorithm is part of an interactive procedure that asks the decision maker to define interesting search areas by establishing limits to the objective function values or by indicating reference points. The procedure will be applied to some illustrative location problems.

Keywords Location problems · Genetic algorithms · Local search · Multi-objective

1 Introduction

Since the pioneering work of Schaffer [41], parent of the VEGA algorithm, much has been said and done about the use of evolutionary algorithms in a multi-objective scenario. The existence of a population of solutions motivates, immediately, the idea of simultaneously approximating several non-dominated solutions. Jones et al. [29], say that about 70% of the papers describing the use of metaheuristics for multi-objective problems are about genetic algorithms. According to the authors, the success of the known algorithms is justified by the fact that they are the perfect tools for the generation of sets of good quality efficient

J. Dias (✉) · J. Clímaco
Faculdade de Economia and INESC-Coimbra, Universidade de Coimbra, Av. Dias da Silva, 165, Coimbra 3004-512, Portugal
e-mail: joana@fe.uc.pt

M. E. Captivo
Faculdade de Ciências, Centro de Investigação Operacional, Universidade de Lisboa, Campo Grande, Bloco C6, Piso 4, Lisboa 1749-016, Portugal

solutions. Some of the most well known multi-objective evolutionary algorithms are MOGA [22], NSGA [44], SPEA [54].

There are several references in the literature that describe the use of metaheuristics, in particular genetic algorithms, in the location problems research field. Hosage and Goodchild [25], study the possibilities of applying genetic algorithms to location problems. Kratica [30]; Kratica et al. [31], use genetic algorithms for the simple plant location problem, and propose hybridization with an *ADD* heuristic (at each iteration only the facility that contributes to the maximum reduction of the overall cost is open). Filipovic et al. [21], introduce the grained tournament selection operator. Jaramillo et al. [28], study genetic algorithms as an alternative way of calculating good quality solutions to location problems, and conclude that these algorithms should not be used for capacitated location problems with fixed costs. Shimizu [43], mingles genetic algorithms and mathematical programming, solving the sanitary landfill for hazardous materials location problem, in a multi-objective environment. Correa et al. [9], apply genetic algorithms to a real problem that can be formulated as a p-median problem. Cheung et al. [4], describe a genetic algorithm, partially implemented in a parallel architecture, which is applied to several location and location-allocation problems in the oil industry. Cortinhal and Captivo [10], describe genetic algorithms applied to the capacitated location problem with total assignment. Domínguez-Marín et al. [19], solve location problems (p-centre and p-median) using genetic algorithms and variable neighbourhood search.

In this paper we will consider a dynamic location problem where facilities can be organized in a single or multi-level structure, and can be capacitated or uncapacitated. It is possible to open, close and reopen each facility more than once during the planning horizon. All the objective functions considered are linear (there are several examples of interesting objectives that can be represented by a linear function, as seen, for instance, in the works of [26,37,38]). The problem is tackled through the use of a memetic algorithm, executed inside an interactive method. In most evolutionary algorithms dedicated to multi-objective problems, the interaction with the decision maker (DM) happens before or after the calculation of non-dominated solutions (exceptions can be found in [1,2,40,42]). Coello Coello [5], states that little has been done in the evolutionary multi-objective algorithms research field considering explicitly the decision makers preferences, and allowing these preferences to change with time. Veldhuizen [47], considers surprising the lack of efforts put into the development of interactive methods. According to the author, no matter the algorithm, the interaction with the DM can only lead to better solutions. Coello Coello et al. [6], say that interactive methods are the best choice, especially when the decision maker is interested in a particular search area.

This paper describes a new multiobjective interactive memetic algorithm applied to dynamic location problems. The interaction with decision maker is based on either a reference points approach or on the establishment of upper bounds. The decision maker can force the algorithm to look for solutions that are placed within a region of interest.

This paper is organized as follows: in the next section we present the problem, in Sect. 2 the main characteristics of our algorithm are described, in Sect. 3 the interaction with the decision maker is explained, in Sect. 4 some computational results are presented and, finally, in Sect. 5, we point out some conclusions and possible future work.

2 The dynamic location problem

Consider that there are n clients, m facilities' possible locations, and T is the number of time periods considered in the planning horizon. A facility can be opened, closed and reopened more than once during the planning horizon. The decision maker will need to define which facilities will be open in each time period and for how long, and a way to assign clients to operating facilities in such a way that all the clients' demand is satisfied. The objective functions can represent the total cost, the risk of locating the facilities, equity concerns, or other, and will be considered as minimization functions. We developed a model considering two different types of facilities: (1) uncapacitated facilities; (2) facilities with maximum and/or minimum capacity restrictions. Each possible location has a set of location variables associated with it, that determine the opening, closing and reopening time periods. Considering location i, there are two sets of binary variables as follows:

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

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

There are also assignment variables. In the single-level case they are of the type x_{ij}^{t}, that define the flow between client j and facility i during time period t. In the multi-level case they are of the type x_{pj}^{t}, and represent the flow originated from client j, assigned to a path p of open facilities. A path of facilities can be composed of up to μ facilities (where μ represents the number of levels), with, at most, one facility of each level. The objective functions considered are linear with respect to location and assignment variables. The model assures that each client's demand in each time period has to be satisfied; each client can only be assigned to operating facilities or paths composed of open facilities only. A facility can only be reopened at the beginning of period t if it has been opened earlier and can only be opened once during the planning horizon. Only one facility can be open at each location, in each time period. Problems' formulations as Mixed Integer Linear Programming Problems can be found in [13–17].

It is interesting to note that, depending on the type of actual facilities, after fixing a set of feasible values for the location variables, 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 (or path of facilities), the cheapest one;
2. If there are capacitated facilities, then it is necessary to solve a transportation (or a transshipment) problem.

3 The memetic algorithm

According to Osman and Kelly [34], 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) evaluation and selection functions; (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. These components will be described in the following subsections.

a

t \ i	1	2	3	4	5
1	■	□	□	■	■
2	■	■	□	□	□
3	▨	■	□	▨	□

□	The facility is not operating.
■	The facility is operating.
▨	The facility is operating and it was reopened at the beginning of this time period.

b Chromosome L

t \ i	1	2	3	4	5
1	1	0	0	1	1
2	1	1	0	0	0
3	1	1	0	1	0

Chromosome F

t \ i	1	2	3	4	5
1	1	1	0	0	1
2	***0***	0	0	1	1
3	***1***	***0***	1	0	1

Fig. 1 (**a**) Visual representation of the solution (**b**) An individual's representation

3.1 Representation of solutions

Each individual is represented using two chromosomes (L and F), both composed of genes that can only take values 0 or 1. Gene in position $(t-1)m+i$ of the L-chromosome is equal to 1 if facility i is open during time period t, and equal to 0 otherwise. This information is not sufficient to build an admissible solution for the problem, because it is necessary to determine the open and reopen periods: a facility i can be operating from τ to ξ but have been closed at the end of period χ and reopened at the beginning of $\chi+1$, with $\tau \leq \chi \leq \xi$. The second chromosome (F-chromosome) will give exactly this information. Gene in position $(t-1)m+i$ will be equal to 1 if facility i is reopened at the beginning of period t, and 0 otherwise. The F-chromosome is less important than the L-chromosome (the F-chromosomes complement the information provided by L-chromosomes). Its genes' values will only be taken into account when strictly needed. Consider the following example (Figs. 1a, b), with five possible locations and three time periods (a matrix notation is used for ease of understanding). 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. All the other genes are simply ignored (they can have a 0 or 1 value).

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 both equal to one.*

Proposition 3 *It is possible to represent each and every admissible solution to the location problem using a pair of F- and L-chromosomes.*

It is straightforward to conclude that this representation is *redundant*, according to the definition of [39][1]: representations are redundant if the number of genotypes exceeds the number of phenotypes. This representation guarantees that the individuals codify a feasible solution, if the capacity restrictions are relaxed. These restrictions are the only ones that can be violated.

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 the location problem. Let us consider the following definitions:

Definition 2 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 3 Consider two individuals 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 4 Consider a population Psuch 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 5 If $x \in X$ is efficient, its image in the objective space, z, is non-dominated.

Proposition 4 *If x is P-efficient, for every set P considered, then x is said to be efficient.*

Proposition 5 *An individual x can be P_1-efficient and P_2-non efficient, if and only if P_2 has at least one individual $x' \notin P_1$ such that x' dominates x.*

3.2 Fitness and calculation of optimal assignments

The first question rose when dealing with multi-objective problems is how to calculate the individuals' fitness. The fitness of an individual x can be calculated in various forms (using linear or non-linear aggregating functions, considering the number of individuals that dominate or are dominated by x in the current population, asking the DM to make pair-wise comparisons between different individuals, etc[2]).

In the algorithm developed, the fitness of an individual can be calculated in two different ways, depending on the type of interaction with the decision maker: if the decision maker wants to establish upper bounds on the objective function values (considering that all objectives are of the minimization type), then a weighted objective function is considered. If the decision maker prefers to indicate reference points, then a particular achievement function is used.

There are several procedures that determine the weights to assign to each objective function value [48]. In this work, the weights used can be interpreted in two different ways. They can be related to the decision maker's preferences, if he/she wishes to explicitly indicate values for the weights. Nevertheless, the weights can be implicitly calculated by the algorithm, without the decision maker interference. These weights are mainly seen as a technical tool, not as a way of eliciting preferences from the DM.

[1] If there are q determinant genes within the F-chromosome, this means that there can be $2^{(Tm-q)}$ different individuals codifying exactly the same solution.

[2] See Coello et al. [6], for a review.

If the decision maker establishes aspiration and reservation levels (the first ones represent the situation the decision maker would like to achieve, while the latter represent a situation the decision maker wants to get away from — [45,49,50]), then the fitness of each individual is calculated based on these levels. Consider a function $\sigma_i\left(q_i, \overline{q}_i, \overline{\overline{q}}_i\right)$ [49], such that q_i represents the ith objective function value corresponding to individual q, $\overline{q}_i$ and $\overline{\overline{q}}_i$ are the ith aspiration and reservation levels, respectively, $q_{i,lo}$ and $q_{i,up}$ are the lower and upper limits known for the ith objective function value. Then:

$$\sigma_i\left(q_i, \overline{q}_i, \overline{\overline{q}}_i\right) = \begin{cases} 1 + \alpha\left(\overline{q}_i - q_i\right)/\left(\overline{q}_i - q_{i,lo}\right), & q_{i,lo} \le q_i \le \overline{q}_i \\ \left(\overline{\overline{q}}_i - q_i\right)/\left(\overline{\overline{q}}_i - \overline{q}_i\right), & \overline{q}_i < q_i < \overline{\overline{q}}_i \\ \beta\left(\overline{\overline{q}}_i - q_i\right)/\left(q_{i,up} - \overline{\overline{q}}_i\right), & \overline{\overline{q}}_i \le q_i \le q_{i,up} \end{cases} \quad (1)$$

α and β should be greater than zero, and chosen such that this function is monotonous and concave. This function shows how far a given individual is from the reference points defined. The fitness of an individual q, with relation to $\overline{q}$ and $\overline{\overline{q}}$, is calculated using function $\sigma\left(q, \overline{q}, \overline{\overline{q}}\right)$ defined in (2), where ν represents the total number of objective functions considered.

$$\sigma\left(q, \overline{q}, \overline{\overline{q}}\right) = \left(\min_{1 \le i \le \nu} \sigma_i\left(q_i, \overline{q}_i, \overline{\overline{q}}_i\right) + \varepsilon \sum_{i=1}^{\nu} \sigma_i\left(q_i, \overline{q}_i, \overline{\overline{q}}_i\right)\right) / (1 + \nu\varepsilon), \quad \varepsilon > 0 \quad (2)$$

The value given by (2) characterizes an individual with respect to the distance between it and the reference points defined, considering as possible the following situations: the individual can either dominate or be dominated by the aspiration or the reservation levels. An individual that is far from the aspiration point but dominates it is considered better than an individual that corresponds exactly to the aspiration point.

As the only variables that are represented by an individual are the location variables, for each individual one needs to calculate the assignment variables' values. This is a challenging task, because the calculation of optimal assignments is achieved through the resolution of a mono-objective problem. It is also possible to consider several objectives in the assignment problem, which would increase the computational time needed, and extra data structures would become necessary (because for two identical individuals, different assignment variables could be calculated). In the present version, we chose to use weights and to consider a weighted objective function value to calculate optimal assignment variables. If the interaction with the DM is based on the establishment of upper bounds on the objective function values, then the weighted objective function that is used to calculate the fitness of an individual is also used to solve the assignment problems. On the other hand, if the interaction is based on the definition of reference points and function (2) is used to calculate an individual's fitness, then the algorithm chooses randomly, for each generation, a set of weights to build the weighted objective function value for the resolution of each assignment problem. Once again, these weights are not interpreted as representing the DM's preferences. They are only used so that the algorithm is able to calculate an admissible assignment solution.

The individuals that represent unfeasible solutions are not deleted from the population and are given a fitness value equal to $+\infty$.

3.3 Genetic operators

The memetic algorithm developed uses the most common genetic operators found in the literature: selection, crossover and mutation. Selection is based on the binary tournament selection with sharing [11,33]. In every generation, two individuals i and j are randomly selected from the parent population. A sharing value $sh(i, j)$ is calculated using the distance between them given by the number of L-chromosome genes that are different in both individuals.

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

$$nc_i = \sum_{j \text{ belongs to the new population}} sh\,(i, j), sh\,(i, j) = \begin{cases} 1 - \left(\frac{d_{ij}}{\alpha_{share}}\right)^{\alpha}, & \text{if } d_{ij} \le \alpha_{share} \\ 0, & \text{otherwise} \end{cases}.$$

For each selected individual i, all values $sh(i,j)$[3] (calculated considering all individuals j belonging to the new—children—population) are summed up (nc_i). 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 $f(i)$ 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} (usually closer to one). 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 crossover operator does not guarantee that the children of two admissible parents are admissible. We also developed some special operators that take advantage of the known structure of the problem [15], namely a repair algorithm (to diminish the number of non-admissible individuals) and an algorithm that changes only the F-chromosome's genes (the *change openings*, to diminish the fixed costs). The structure of the repair algorithm for single-level problems is presented as Algorithm 1. Algorithm 2 shows the structure of the *change openings* procedure.

An individual represents an unfeasible solution if it violates any maximum or minimum capacity restrictions. These violations are caused by L-chromosome genes. The repair algorithm changes in a random but guided manner, L-chromosome genes. If, for instance, 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. 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 [7].

The *change openings* procedure studies the effect on the location variables' objective function coefficients 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 reopened during that interval (in a time period $t \le \xi$). 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

[3] The α_{share} value is calculated as described in [11], and α is considered equal to one.

value the procedure tries to change to zero. If the current objective function value diminishes, then the gene's value is changed, otherwise retains its original value.

Algorithm 1 Repair algorithm
Predefined parameters: *NMAXTRY*-total number of iterations in each time period.
d_j^t-demand of client j during time period t; Q_i-maximum capacity of facility i; Q_i'-minimum capacity of facility i

```
1. $t \leftarrow 1$.
2. $ntries \leftarrow 1$. If $t > T$ then stop.
3. Calculate $D \leftarrow \sum_j d_j^t$, $C_{max} \leftarrow$ total maximum capacity of facilities operating during $t$[4], $C_{min} \leftarrow$ total minimum capacity of facilities operating during $t$.
4. $ntries \leftarrow ntries + 1$. If $ntries > NMAXTRY$ then stop.
5. If $C_{min} > D$ then go to 6. If $C_{max} < D$ then go to 7.
6. Choose randomly a capacitated facility such that $L$-chromosome$[t,i]=1$ and $C_{max} - Q_i \geq D$. $L$-chromosome$[t,i] \leftarrow 0$, $C_{min} \leftarrow C_{min} - Q_i'$, $C_{max} \leftarrow C_{max} - Q_i$. Go to 5.
7. Choose randomly a facility $i$ such that $L$-chromosome$[t,i]=0$ and $C_{min} + Q_i' \leq D$. $L$-chromosome$[t,i] \leftarrow 1$, $C_{min} \leftarrow C_{min} + Q_i'$. Go to 5.
8. Solve a transportation problem. $t \leftarrow t + 1$ and go to 2.
```

Algorithm 2 *Change-openings* procedure
m-number of possible locations for facilities.

```
1. $i \leftarrow 1$;
2. If $i > m$, then stop.
3. Detect a pair of location variables equal to one of the form $\left(a_{i\tau}^{\xi}, r_{i\xi+1}^{\psi}\right)$ or $\left(r_{i\tau}^{\xi}, r_{i\xi+1}^{\psi}\right)$.
   If there are no pair of variables in this situation, then $i \leftarrow i + 1$ and go to 2.
4. $\Delta \leftarrow FA_{i\tau}^{\psi}\left(\text{or } FR_{i\tau}^{\psi}\right) - FA_{i\tau}^{\xi}\left(\text{or } FR_{i\tau}^{\xi}\right) - FR_{i\xi+1}^{\psi}$. 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 3.
```

3.4 Local search

Local search plays a very important role in the algorithm developed. Examples of genetic algorithms hybridized with local search can also be found in [27,32,35,51]. Every individual in the new population is a potential starting solution for the local search procedure, running with a given probability p_{ls}. If a child is equal to one of the parents that, in turn, resulted from local search, this probability is equal to zero. The procedure developed considers the weighted objective function value that is being used in the current generation for the calculation of assignments, and tries to improve an individual's fitness, calculated according to this weighted objective function value, by searching k-neighbourhoods, from $k = 1$ to T, where 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 to a single facility i. Whenever the fitness function is improved, the individual's genotype is immediately changed, and the search continues with this new individual as the starting-point. This procedure is very time consuming and can be responsible for 95% or more of the algorithm's total computational time. This makes it compulsory to improve its performance. We changed the local search procedure, by performing a sensitivity analysis (based on the dual optimal

[4] If there is at least one uncapacitated facility in operation during t, then $C_{max} \leftarrow +\infty$.

solution of the assignment problems) in order to estimate if the present neighbour is or is not better than the current individual. This sensitivity analysis is only performed if in presence of capacitated facilities, and can decrease the total computational time by 20% or more without a significant decrease in the final solution's quality.

3.5 Populations

The present implementation of our algorithm works with two populations. Borrowing the notation of [46], $P_{current}(ger)$ represents the population during the *ger*th generation. Population $P_{known}(ger)$ is composed of all *PU*-efficient individuals, where $PU = \bigcup_{g=1}^{ger} P_{current}\,(g)$. The first $P_{current}$ population is constituted by individuals randomly created that are modified by the *repair*, *change openings* and *local search* procedures (clones are allowed). We chose to work with small populations. As there is a risk of premature convergence to a poor quality solution, we overcame this disadvantage by changing on-line the total number of individuals in the current population. The number of individuals is increased whenever a predefined number of generations (*nimp*) are executed without improving the fitness of the best individual, until the maximum number of individuals in $P_{current}$ is reached. All new individuals are randomly initialized. The population is initialized with *npop* individuals calculated as described in [36]: it is the minimum value such that $\left(1 - \left(\frac{1}{2}\right)^{npop-1}\right)^{l} \geq 0.99$[5], 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 2*mT*. 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 2*mT* for the calculation of the maximum value *npop* can take.

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, [46]), 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 P_{known}-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 P_{known}-efficient solutions. Another interesting subject is to define how the two populations will interact between them and with the algorithm. In the present implementation, 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 [54]. One possible interaction between the two populations could be easily implemented: whenever the number of individuals in $P_{current}$ is increased, the new individuals could be randomly chosen from P_{known}, instead of randomly initialized.

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

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

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}[6]. If P_{known} has reached its maximum capacity and has only efficient individuals, some of them will have to be removed. In the present implementation of the algorithm the individual to be removed is randomly chosen. Many other procedures could be devised: asking the DM which individual he/she wishes to remove or maintain, applying clustering algorithms in order to insure that set P_{known} maintains a good diversified efficient solution set, etc.

The P_{known}-efficient individuals are, at each generation, an approximation of the true efficient solutions set (that is unknown).

3.6 The overall algorithm

Algorithms 3–5 describe the overall functioning scheme of the memetic algorithm.

Algorithm 3 Multi-objective memetic algorithm
ngenerations—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 ngenerations (using algorithm 2) 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.
8. Ask the DM to establish limits for each objective function, or weights for each objective
   function $p$, or indicate aspiration and reservation levels for each objective function.
9. Evolve $P_{current}$ using algorithm 2. 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.
```

Algorithm 4 Memetic algorithm
X_{best}—best solution known thus far, *f(x)*—fitness of individual x, *nimp*—maximum number of generation without improvement of $f(x_{best})$, *nmaxpop*—maximum number of individuals in $P_{current}$

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

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, *npop*—number of individuals in the current population.

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

1. $x_1 \leftarrow x_{best}$; $x_{best} \leftarrow x_1$; $j \leftarrow 2$; $Newpop \leftarrow \{x_1\}$.
2. If $j > npop$ then $P_{current} \leftarrow Newpop$.
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 $flag(x_j) \leftarrow flag(x_B)$;[7]
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.

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. Table 1 lists all parameters used within the algorithm, and the way in which we think they influence the algorithm's behaviour.

4 Interaction with the decision maker

Population $P_{current}$ evolves through a number of generations, being the evolution guided by the preferences of the DM. In the initial phase of the interactive method, a set of solutions is shown to the decision maker. If he/she feels that a good compromise solution has been found, the method stops. Otherwise, the decision maker is asked to express his/her preferences in the form of weighting values, limits to the objective function values or reference points. The interaction between the algorithm and the decision maker is depicted in Fig. 2, and follows the works of [12,18,20].

If the DM chooses to interact with the algorithm through the establishment of upper limits to the objective functions' values, then this will translate into the introduction of additional restrictions to the problem. An individual that violates this restrictions is penalized, being given a fitness equal to $+\infty$. Considering these additional restrictions, the algorithm is able to calculate non-dominated candidate solutions by solving a mono-objective problem [38]. This justifies the use of a weighting objective function. If the problem has only two objectives, the algorithm can calculate the objective functions' weighting values automatically, as described in [12,18]. In all other cases, with three objectives or more, the weights are randomly generated.

If the DM chooses to interact with the algorithm through the establishment of reference points, the decision maker is asked to indicate aspiration and reservation levels (Granat and Makowski [23,24], present a software application that illustrates the interactive use of a reference point based interactive approach).

The algorithm terminates when the DM is satisfied with the solutions calculated, and feels he/she has gained sufficient insight into the problem and feasible alternatives. The options taken by the DM in some iteration of the algorithm do not restrict future options: all the choices made are reversible. The DM may jump from one area of interest to another one.

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

Table 1 Algorithm's Parameters

Parameter	Description	Influence on the algorithm's behaviour and recommended values[a]
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.
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.
npop	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 [36]. In our algorithm we calculate the initial population as described in 3.5, and increase this value whenever there are *nimp* generations without improvement of the best objective function value.

Table 1 continued

nmaxpop	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 *nmaxpop*, 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%.
Nger	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.
nimp	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 *nimp* 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 *nmaxpop*, then the algorithm is terminated after *nimp* generations without improving the objective function. It is fixed to 5.

[a] 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.

Although the different interaction possibilities are available we agree with those that prefer the establishment of reservation and aspiration levels as the best procedures, namely taking into account cognitive psychology. Furthermore the computational tests of Sect. 5 confirm this point of view.

5 Computational results

The task of evaluating the quality of a given multi-objective evolutionary algorithm is not simple, and can be interpreted as a multi-objective problem by itself [52]. When the method we want to evaluate is interactive, things get even more complicated. When dealing with interactive approaches, the ideal situation would be to have a set of real decision makers willing to participate in a series of experiments. This is, most of the times if not always, not possible. Zitzler et al. [55], consider that to assess the quality of a given evolutionary algorithm, one should account for the following factors: the distance of the solutions set to the Pareto-optimum frontier; the distribution of solutions; the range covered by the solutions. If the solutions are calculated using an interactive method, the resulting set will be strongly

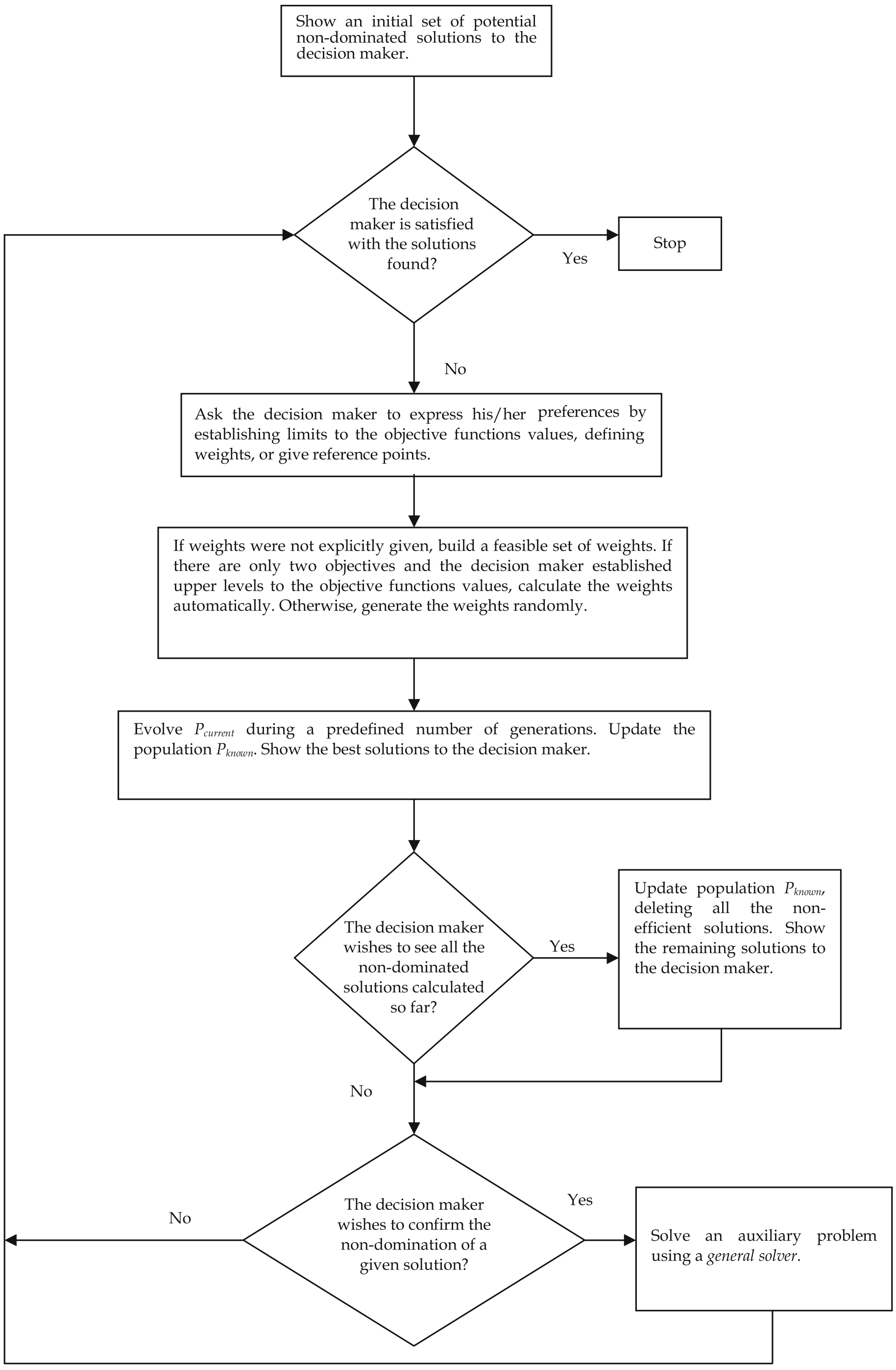
Show an initial set of potential non-dominated solutions to the decision maker.
The decision maker is satisfied with the solutions found?
Yes
Stop
No
Ask the decision maker to express his/her preferences by establishing limits to the objective functions values, defining weights, or give reference points.
If weights were not explicitly given, build a feasible set of weights. If there are only two objectives and the decision maker established upper levels to the objective functions values, calculate the weights automatically. Otherwise, generate the weights randomly.
Evolve $P_{current}$ during a predefined number of generations. Update the population P_{known}. Show the best solutions to the decision maker.
The decision maker wishes to see all the non-dominated solutions calculated so far?
Yes
Update population P_{known}, deleting all the non-efficient solutions. Show the remaining solutions to the decision maker.
No
The decision maker wishes to confirm the non-domination of a given solution?
No
Yes
Solve an auxiliary problem using a *general solver*.

Fig. 2 Interaction with the decision maker

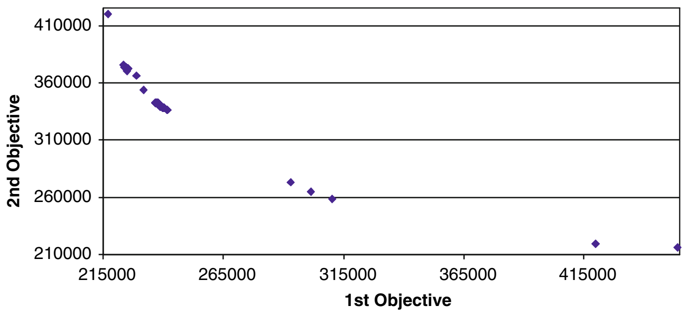

Fig. 3 First set of solutions

dependent on the decision makers choices and preferences. He/she can decide to explore the whole non-dominated region, or focus his/her search on only a tiny part of this region. These kinds of options can influence significantly the quality evaluation of a given solution set.

Our aim was to perform some computational tests in order to assess the memetic algorithm's capability of calculating non-dominated solutions, and to compare the two different interactive approaches. We will compare the solution set obtained with the two approaches, calculating the minimum, average and maximum distance of the non-dominated solutions to the Pareto-optimal frontier. These metrics can classify the algorithm with respect to its convergence capabilities. The spacing metric as defined in [8], measuring the spacing of points on a Pareto frontier, is also going to be calculated.

Three location problems were randomly generated: one uncapacitated single-level location problem, with 10 time periods, 25 possible locations for facilities and 100 clients; one capacitated single-level location problem of the same dimension and one uncapacitated two-level location problem with 10 time periods, 100 clients, 5 possible locations for facilities in the last level and 10 in the first level. These problems were generated as described in [13,16,17].

For the uncapacitated single-level location problem, the algorithm begins by showing to the decision maker a set of candidate non-dominated solutions, as depicted in Fig. 3. As can be seen, there are some areas of the non-dominated region that are not covered in this set. The DM will interact with the algorithm, establishing upper (and lower, if desired) bounds to the objective functions' values. These values can be defined explicitly, or by the selection of two already known solutions. Suppose that he/she establishes the following bounds: the first objective function has to be lower than 293191 and the second objective function has to be lower than 335809 (these values are indirectly given by the selection of two solutions, and lower bounds can also be considered). As can be seen in Fig. 4, the algorithm is capable of calculating some new solutions in the defined area, but these solutions are not well dispersed. After two interactions with the decision maker, where he/she defines new upper/lower levels, the solutions set calculated in the region of interest is shown in Fig. 5. The decision maker is free to define other areas of interest, as shown in Figs. 6 and 7. In all cases, the algorithm is capable of calculating solutions within the established upper bounds, but badly dispersed.

If the interaction now asks the decision maker to give aspiration and reservation levels, then the results are quite different. The values for $q_{i,lo}$ and $q_{i,up}$ are dynamically updated as the algorithm finds new solutions. They always represent the best lower and upper limits known for the ith objective function value. Figure 8 shows the first set of solutions calculated and the aspiration and reservation points given by the decision maker. Figure 9 depicts the

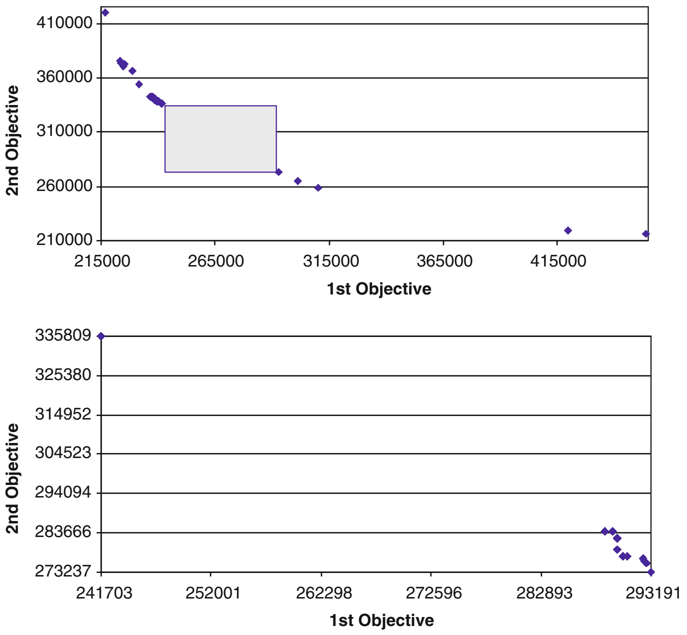

Fig. 4 Calculating solutions in a selected area

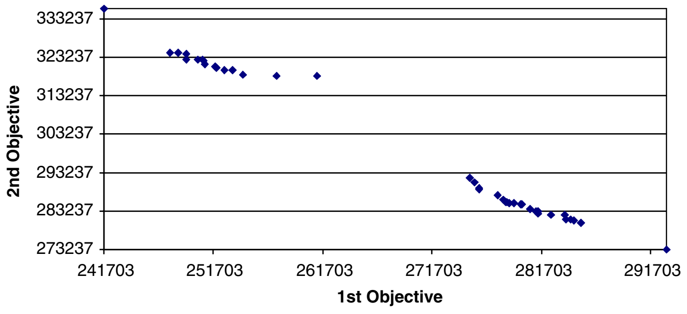

Fig. 5 New solutions after two more interactions with the decision maker

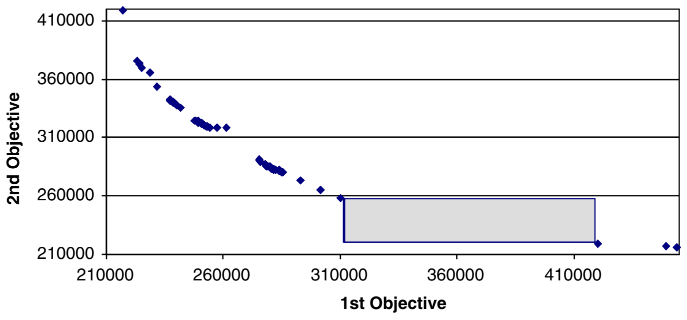

Fig. 6 Definition of a new area of interest

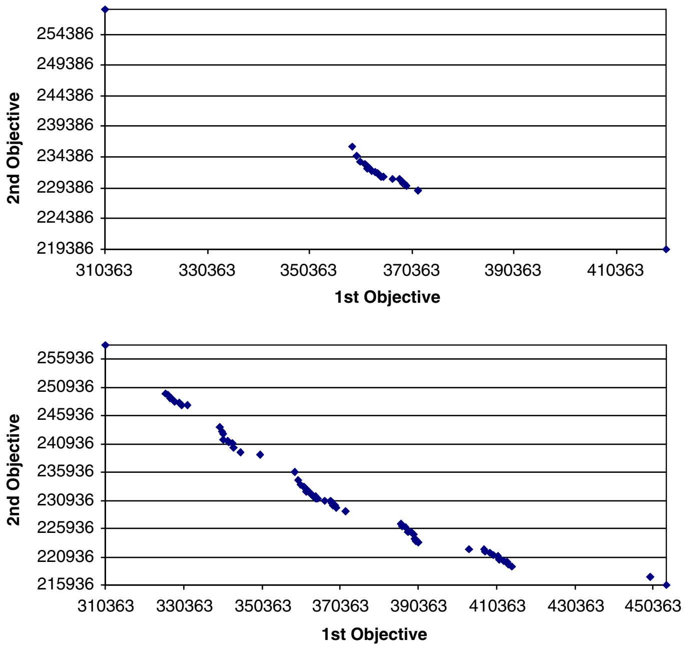

Fig. 7 New solutions in the new region of interest after one and five interactions

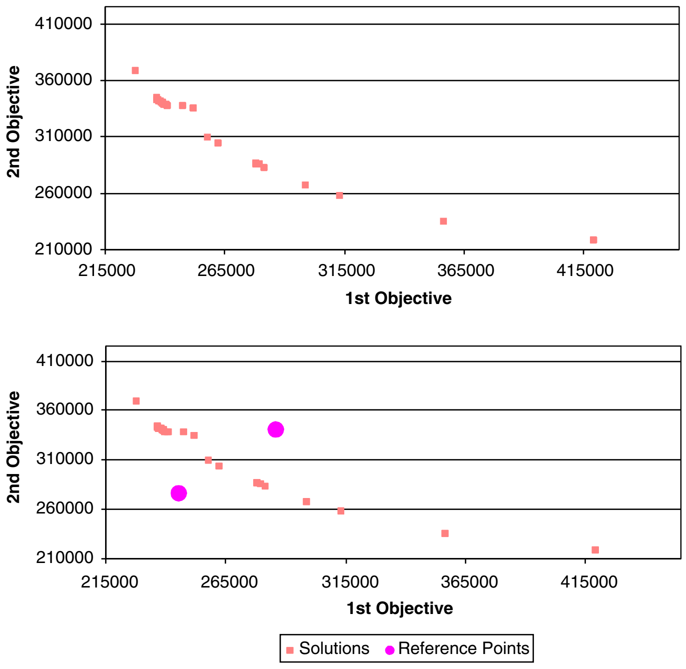

Fig. 8 First set of solutions and establishment of reference points

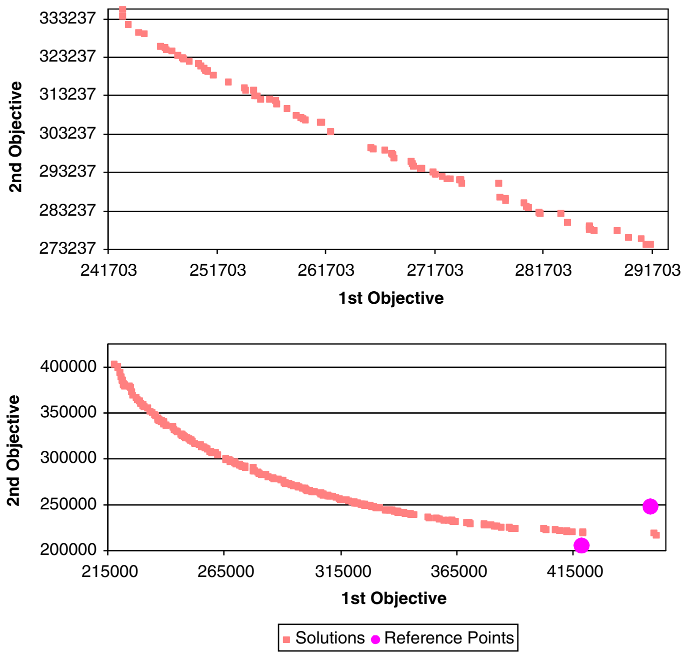

Fig. 9 Calculation of new solutions, and definition of new reference points

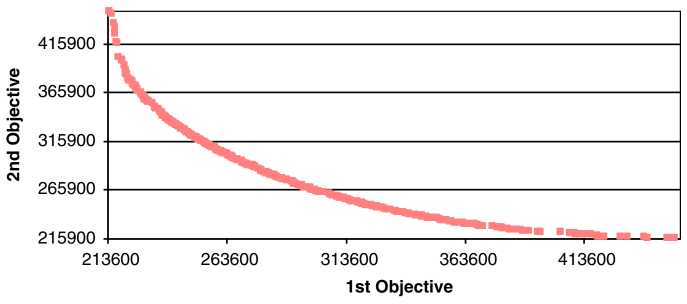

Fig. 10 Solutions set calculated by the definition of reference points

solutions calculated by the algorithm and the establishment of new reference points. With only two interactions with the decision maker, Fig. 10 shows the total set of solutions calculated, and Fig. 11 compares this set with the one obtained earlier, when the interaction was based in the establishment of upper bounds.

Similar results are shown in Figs. 12–21 for the capacitated single-level location problem. Figs. 12–15 depict the results obtained when the algorithm asks the decision maker to establish upper bounds to both objective functions. Figures 16–20 show the results when the interaction is based on the reference point approach. Figure 21 compares the two sets of solutions.

Figures 22–29 depict the results obtained when solving an uncapacitated two-level location problem, and Fig. 30 compares the two sets of solutions.

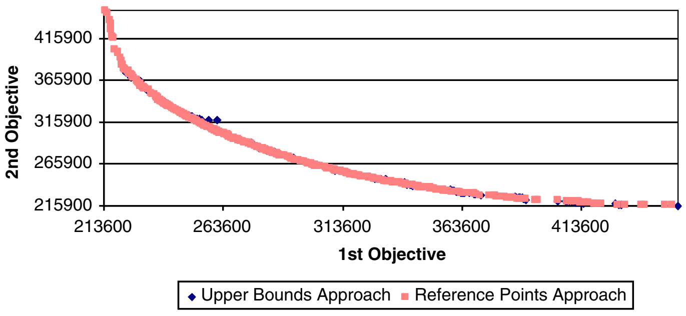

Fig. 11 Comparison with the set calculated by the establishment of upper/lower bounds

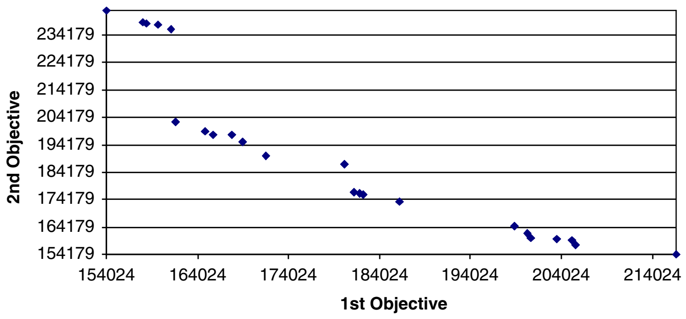

Fig. 12 First set of solutions

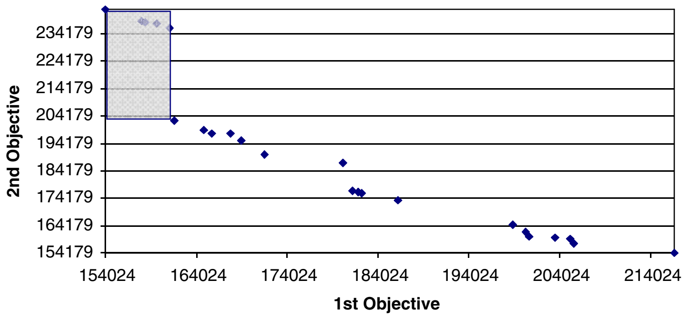

Fig. 13 Definition of the area of interest

The memetic algorithm developed was compared with the algorithm described in [3][8], and also with SPEA2[9] [53]. It is not easy to compare the behaviour and performance of an

[8] Notice that this algorithm does not guarantee the calculation of all the non-dominated solutions if in presence of capacitated facilities, because the dynamic location problems are formulated as mixed-integer linear programming problems and the solution may be composed of variables with real values.

[9] The parameters used in the SPEA2 algorithm were similar to the ones used in the memetic algorithm developed. The number of individuals in the population was considered equal to the maximum number of individuals in the population for the memetic algorithm (*nmaxpop*). The maximum number of iterations considered was equal to 5000.

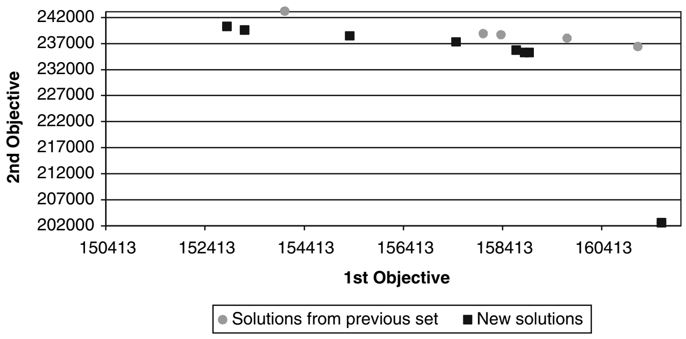

Fig. 14 Comparison between the existing and new solutions calculated

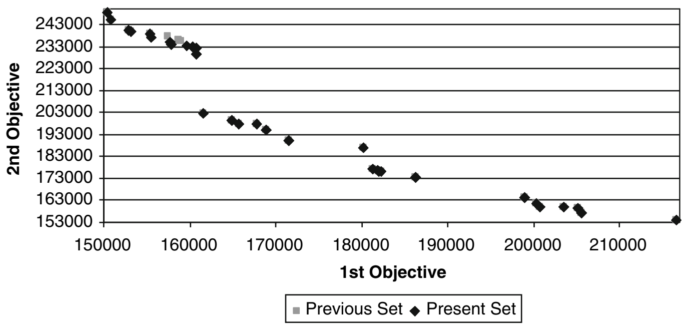

Fig. 15 After two interactions with the decision maker, the solutions set suffers few changes

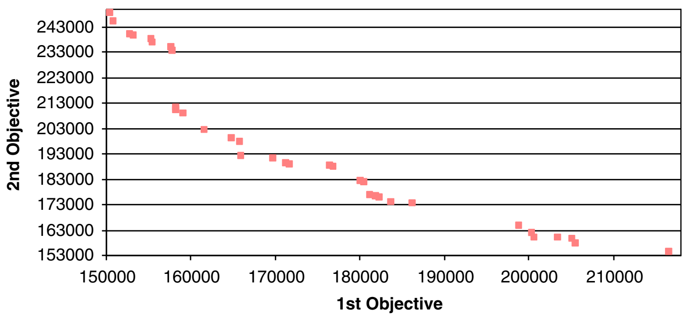

Fig. 16 First set of solutions

interactive algorithm with the performance of non-interactive algorithms. As a matter of fact, the behaviour of the memetic algorithm developed is deeply dependent upon the decision maker's choices. That is why we chose to compare only computational times and quality of solutions calculated.

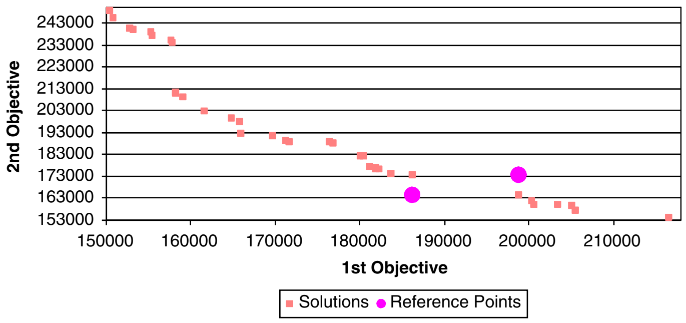

Fig. 17 Definition of the reference points

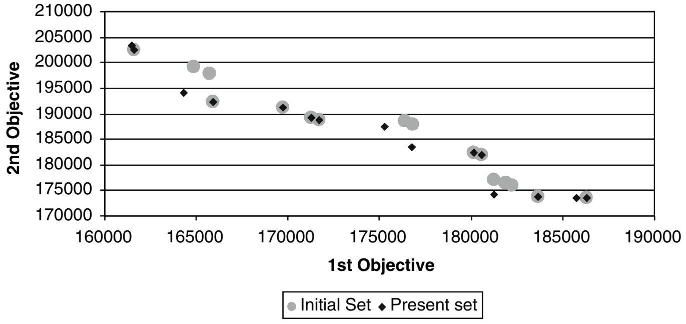

Fig. 18 New set of solutions

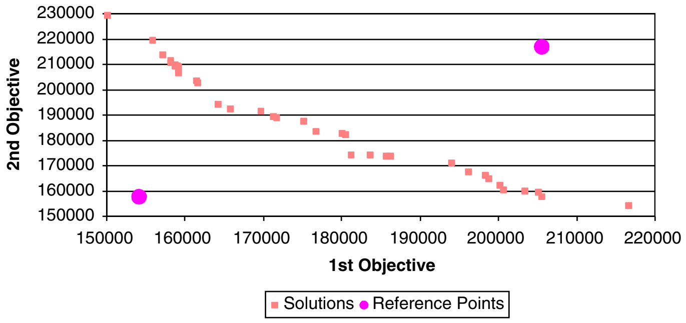

Fig. 19 Definition of new reference points

After testing the behaviour of SPEA2 algorithm, we chose to change it including the local search procedure. As shown in Figs. 31–33, the quality of the solutions calculated is greatly improved by the introduction of the local search. If the local search procedure had not been included in SPEA2 then the results would not be comparable with the ones obtained by executing the memetic algorithm.

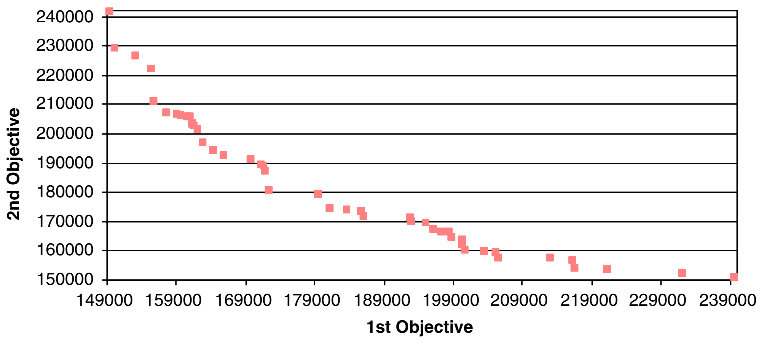

Fig. 20 Final set of solutions

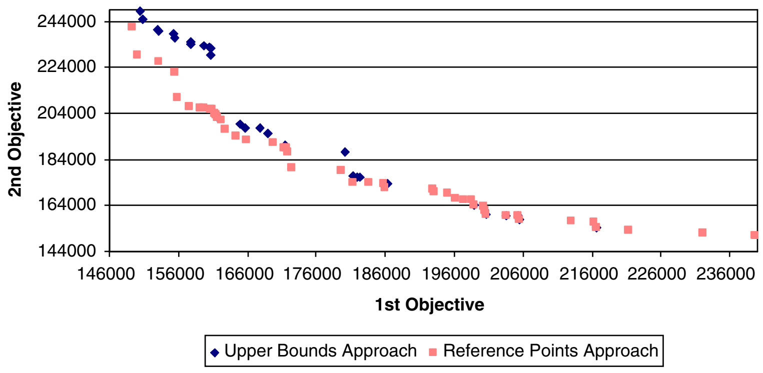

Fig. 21 Comparison of the two different sets

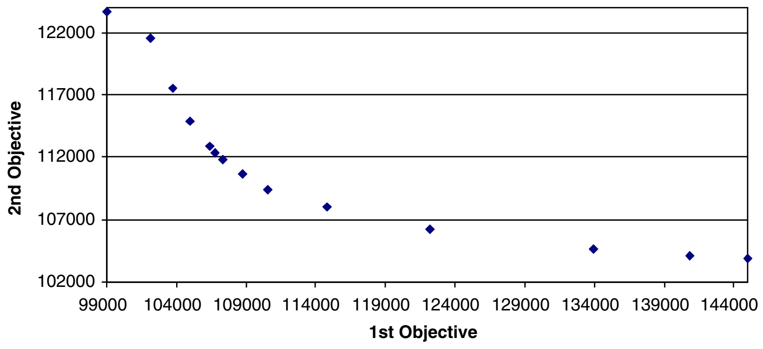

Fig. 22 First set of solutions

Figures 34–36 shows the sets of solutions calculated by SPEA2 for the three problems considered. These figures depict the whole set of solutions calculated during all the generations and the solutions that belong to the last set. In this way one can get an idea of the evolution that took place during the algorithm's execution.

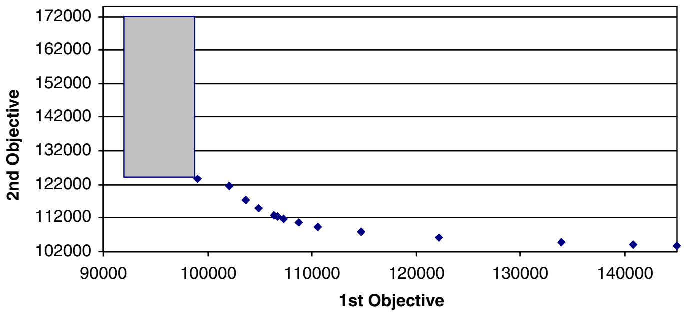

Fig. 23 Definition of the area of interest

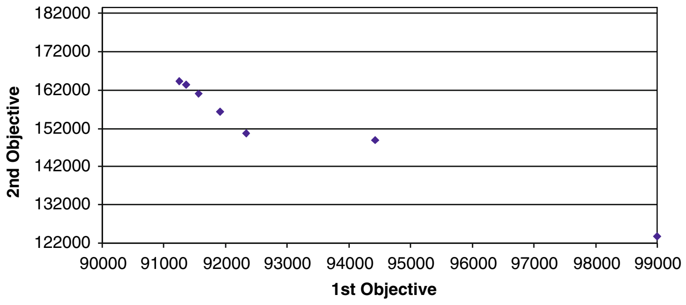

Fig. 24 Solutions calculated in the area of interest

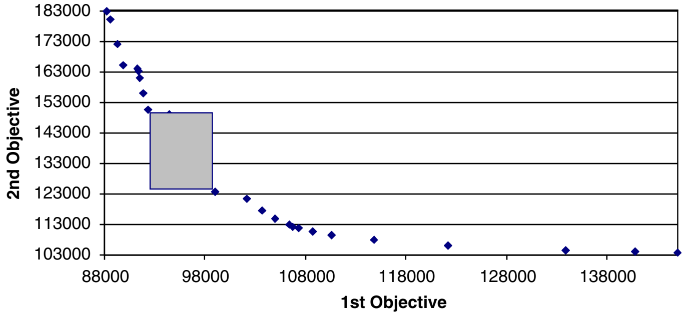

Fig. 25 New region of interest

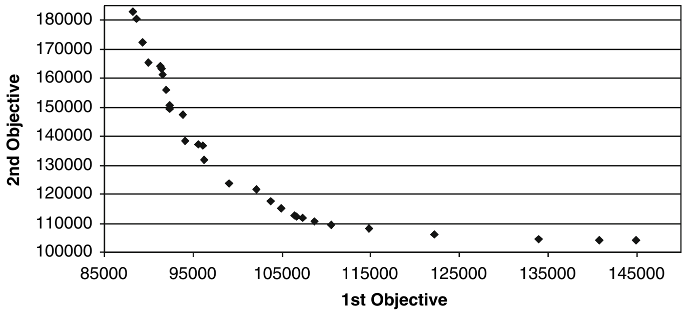

Fig. 26 Final set of solutions

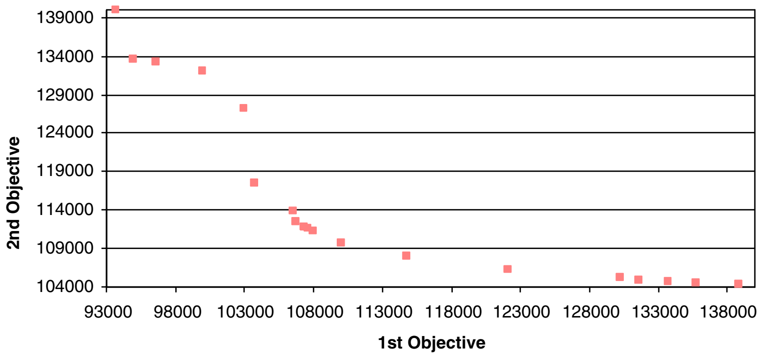

Fig. 27 First set of solution for the reference point approach interactive method

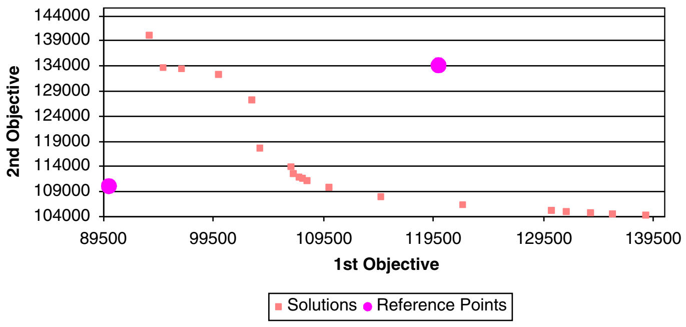

Fig. 28 Reference points

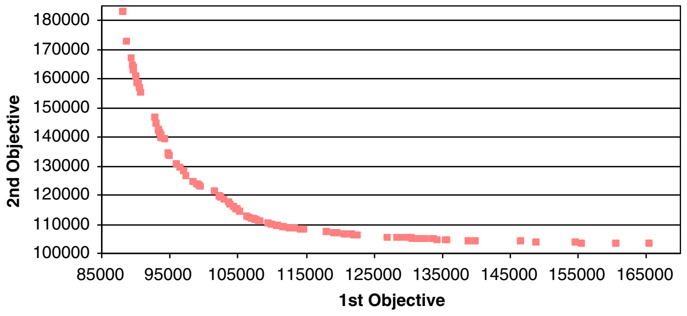

Fig. 29 Final set of solutions

Figures 37–39 compare different sets of solutions for each of the problems: true Pareto-optimal solutions calculated using the algorithm described in [3], the sets obtained with both approaches and also using SPEA2 algorithm.

The local search procedure is a fundamental piece of the memetic algorithm described. It is responsible for a significant improvement in the quality of solutions calculated. We wanted to test how this procedure would behave on its own. Figs. 40–42 show the results obtained by using the local search procedure only, considering as initial solutions randomly generated solutions. It can be seen that this procedure alone is not sufficient to generate good quality solutions.

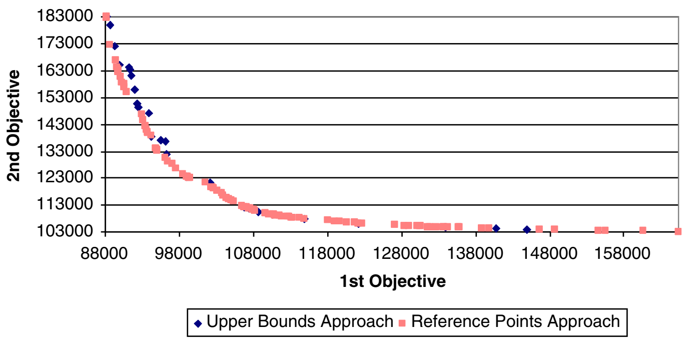

Fig. 30 Comparison of the two different sets of solutions

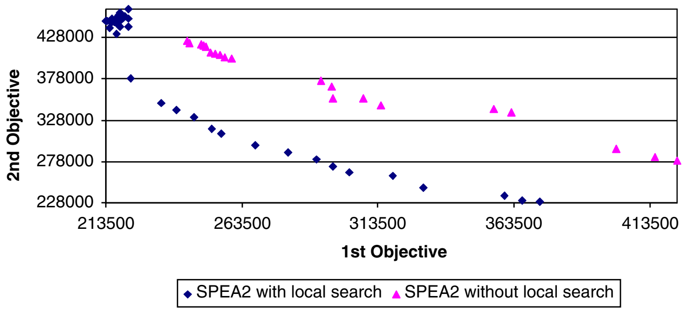

Fig. 31 SPEA2's sets of solutions for the uncapacitated problem with and without the local search procedure

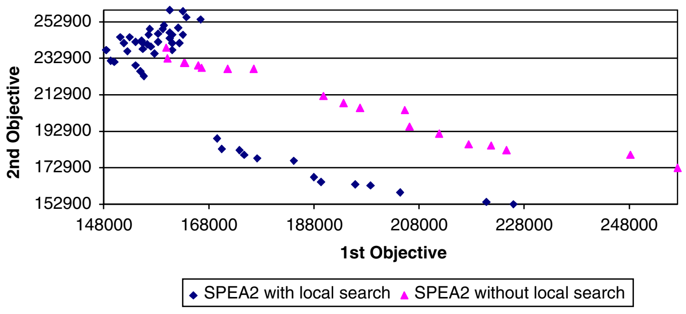

Fig. 32 SPEA2's sets of solutions for the capacitated problem with and without the local search procedure

Tables 2–4 summarize the computational results obtained for all problems solved. Table 2 shows, for each problem and for each algorithm tested, the number of solutions calculated, the distance from the solutions calculated to the true Pareto-optimal frontier, and the spacing metric. Table 3 compares the non-dominated sets calculated by each of the algorithms. Table 4 shows the computational times.

Trying to draw some conclusions from the results obtained, and considering first the two interactive approaches, how can be justified the fact that the reference point approach presents, in all three cases, better results? At a first glance, one could think that the main reason is due to the use of randomly generated weights that are more often changed. This can,

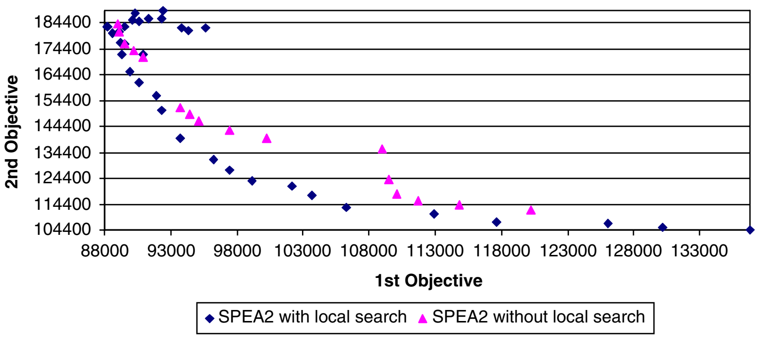

Fig. 33 SPEA2's sets of solutions for the two-level uncapacitated problem with and without the local search procedure

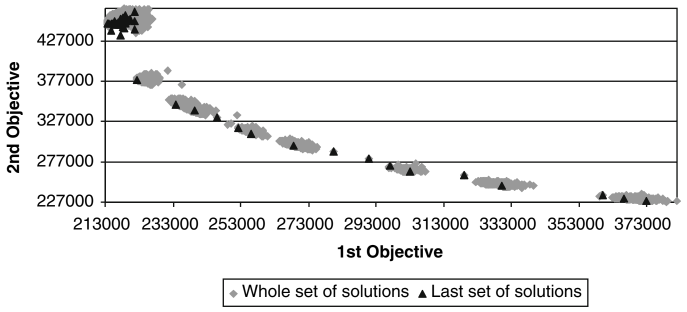

Fig. 34 SPEA2's sets of solutions for the uncapacitated problem

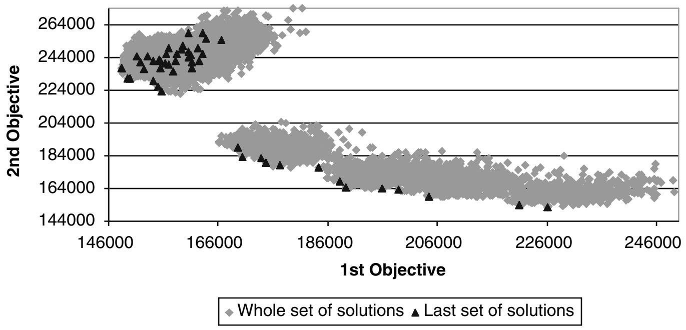

Fig. 35 SPEA2's sets of solutions for the capacitated problem

in fact, be an important part of the answer. But the true justification lies in the fitness function itself. As a matter of fact, if the weights were randomly chosen or changed more often, in the establishment of bounds approach, it would result in worse outcomes. The quality of a given individual is given by the value of the weighted objective function, considering additional restrictions that correspond to the bounds introduced. If the weights were calculated or changed in a different way, most of the generated individuals would not be feasible, with respect to the bounds established, so the overall results would be worse. The strength of the

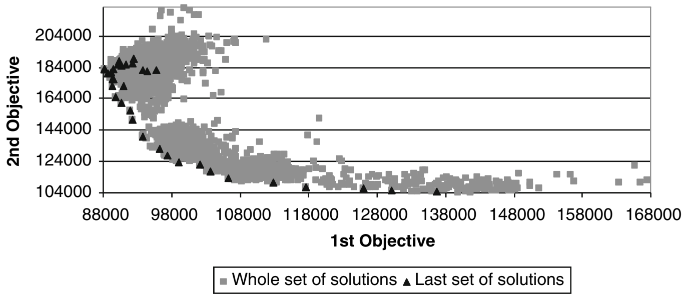

Fig. 36 SPEA2's sets of solutions for the two-level uncapacitated problem

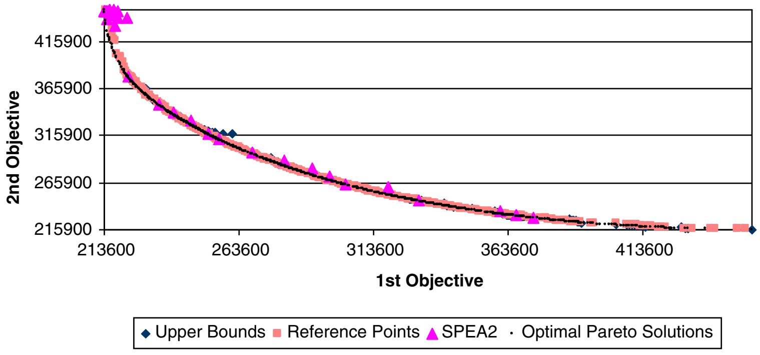

Fig. 37 Uncapacitated single-level location problem: comparison of different sets of solutions

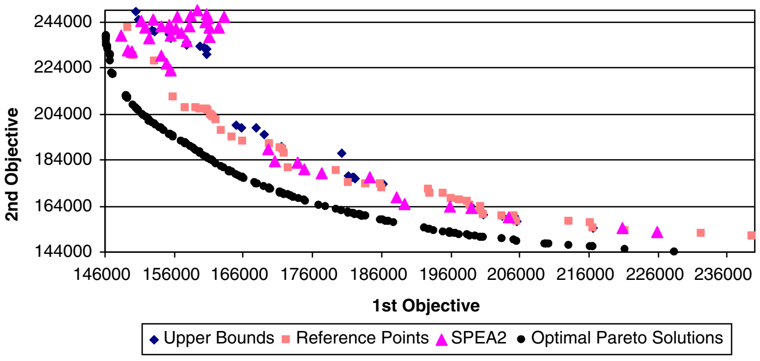

Fig. 38 Capacitated single-level location problem: comparison of different sets of solutions

reference point approach comes from the fitness function it considers, that is not dependent on a given set of weights.

Comparing now the SPEA2 algorithm with the interactive memetic algorithm, one can say that SPEA2 is capable of calculating more solutions per generation, but these solutions are generally more badly dispersed than the solutions calculated by both interactive approaches.

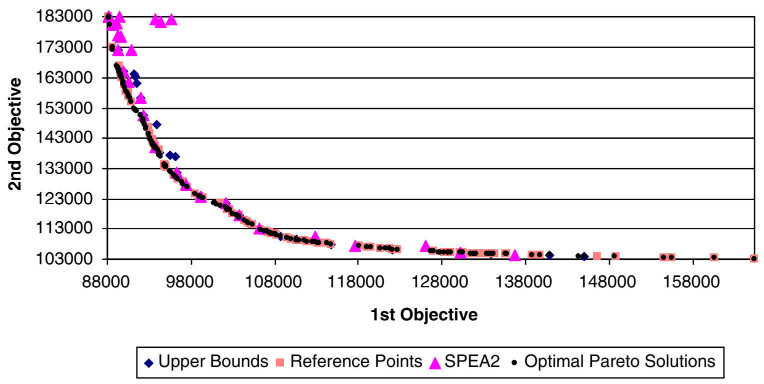

Fig. 39 Uncapacitated two-level location problem: comparison of different sets of solutions

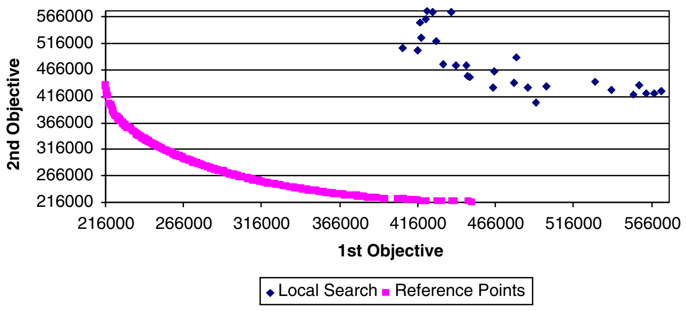

Fig. 40 Comparison between the solutions' sets calculated by the reference point approach and applying local search only: uncapacitated location problem

In Table 3 it is clear that most of the solutions calculated by SPEA2 are dominated by solutions calculated by the interactive algorithm.

The Chalmet et al. [3], algorithm is capable of calculating non-dominated solutions, but it takes a longer time per solution. Nevertheless we cannot forget that the solutions generated by this algorithm are non-dominated for sure. If the decision maker has the time and wants to generate a large set of non-dominated solutions, he/she should choose to use such an algorithm. If, on the other hand, the decision maker wants to have a glance over the whole non-dominated set, then he/she could prefer the memetic algorithm. Despite the fact that the solutions calculated are not all non-dominated, the decision maker can control the process of exploring chosen areas in the objective functions' space and visualize an approximation set of the real non-dominated solutions.

Regarding computational times (Table 4), the memetic algorithm takes longer than what would be desired. SPEA2 is also very time consuming (because of the local search procedure). The computational times do not differ significantly from one approach to the other, and are worse in the capacitated problem. This value diminishes in the case of the two-level location problem. These computational times have to be improved, especially through changes in the local search procedure that is the main responsible for these times.

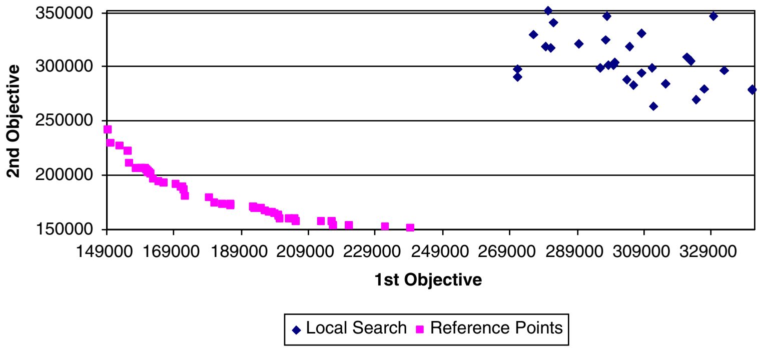

Fig. 41 Comparison between the solutions' sets calculated by the reference point approach and applying local search only: capacitated location problem

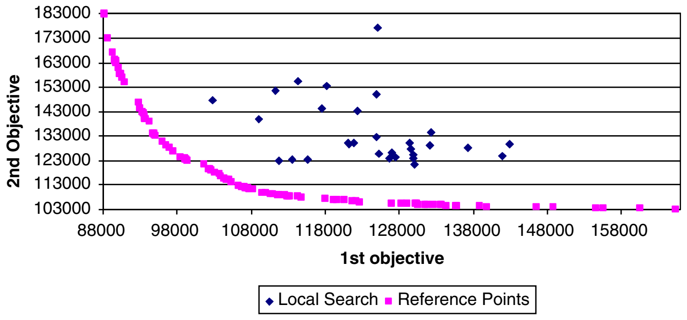

Fig. 42 Comparison between the solutions' sets calculated by the reference point approach and applying local search only: two-level location problem

6 Conclusions and future work

In this paper a memetic algorithm is described that can be used inside an interactive approach to calculate non-dominated solutions to dynamic location problems. Two different approaches are possible: the decision maker expresses his/her preferences through the establishment of upper bounds on the objective function values, or through the indication of reference points. Both approaches were applied to three bi-objective location problems. It is possible to conclude that, for all cases, the reference point approach is capable of finding more and better solutions in less iterations than the other approach. Both approaches are capable of finding solutions according to the decision maker's preferences, but the reference point approach builds better sets, as can be seen in Table 3. The worst results are obtained when capacity restrictions are introduced. This can be justified by the representation chosen (the assignment variables are calculated for each individual, and are not codified in any way), and also because the number of unfeasible individuals is much higher than in the uncapacitated cases. For the uncapacitated problems, the sets calculated are a good representation of the set of true Pareto optimal solutions.

The memetic algorithm shown can be used in problems with any number of objectives. With more than two objectives, the interaction with the decision maker cannot rely only on bi-dimensional charts, and other ways of illustrating the compromises between solutions have to be thought.

Table 2 Computational results

	Chalmet et al. [3]		TNS	MA_ED	AED	MI_ED	SM
Single-level Uncapacitated Problem	21175	RPS	417	5518.32	445.09	0.00	0.97
		UBS	134	7993.00	888.64	1.41	1.03
		SPEA2	12709	20463.00/ 15226.70	6031.42/ 3550.19	0.00	1.79
Single-level Capacitated Problem	1878	RPS	47	14973.90	9927.05	3373.00	0.48
		UBS	32	17911.70	11038.05	7082.50	0.83
		SPEA2	8612	44426.70/ 28290.80	14637.70/ 12443.00	2729.52	1.46
Two-level Uncapacitated Problem	16403	RPS	97	1623.28	65.85	0.00	0.59
		UBS	30	1811.2	549.16	0.00	0.62
		SPEA2	2694	40726.60/ 7468.86	7668.47/ 1538.28	0.00	1.26

RPS: Solutions' set generated by the reference point approach; UBS: Solutions' set generated by the establishment of bounds approach; SPEA2: Solutions' set generated by the SPEA2 algorithm; Chalmet et al. [3]: Total number of non-dominated solutions calculated; TNS: Total number of different solutions calculated; MA_ED, AED, MI_ED: Maximum, average and minimum euclidean distance between a solution and the real Pareto-optimal frontier. In the case of the SPEA2 algorithm two values are shown: the distance considering the set of all solutions calculated during the algorithm's execution and the final set of solutions, respectively; SM: spacing metric

Table 3 Comparison of non-dominated solutions' set

	Uncapacitated problem			Capacitated problem			Two-level problem		
	SPEA2	UBS	RPS	SPEA2	UBS	RPS	SPEA2	UBS	RPS
SPEA2	–	7685 (60%)	11775 (93%)	–	6853 (80%)	8476 (98%)	–	2640 (98%)	2665 (99%)
UBS	26 (19%)	–	70 (52%)	21 (68%)	–	24 (75%)	6 (20%)	–	11 (37%)
RPS	39 (9%)	32 (8%)	–	25 (54%)	0	–	0	2 (2%)	–

SPEA2: SPEA2 algorithm; UBS: Upper bounds approach; RPS: Reference points approach; Each cell in this table represents the number of solutions generated by the algorithm identified in the line that are dominated by solutions generated by the algorithm identified in the column

It is also interesting to consider the possibility of introducing new restrictions to the problem as the decision maker gains knowledge about the problem. He/she could, for instance, fix some facility open or close in one or more time periods. As most of the strategical location problems are group decision problems, we intend to work on this algorithm considering a multi decision maker context.

Table 4 Computational times

		Chalmet et al. [3]	SPEA2	Upper bounds approach	Reference points approach
Average computational time per solution (in seconds)	Uncapacitated problem	23.05	0.41	0.45	0.51
	Capacitated problem	143.62	1.24	2.37	2.65
	Two-level uncapacitated problem	4.67	0.04	0.04	0.04
Average computational time per generation (in seconds)	Uncapacitated problem	–	20.43	8.84	9.99
	Capacitated problem	–	145.00	42.47	42.88
	Two-level uncapacitated problem	–	1.89	0.67	0.73

Acknowledgements This research was partially supported by research project POCTI/ISFL-1/152.

References

1. Barbosa, H.J.C., Barreto, A.M.S.: An interactive genetic algorithm with co-evolution of weights for multiobjective problems. In: Proceedings of the Genetic and Evolutionary Computation Conference (2001)
2. Branke, J., Kaubler, T., Schmek, H.: Guidance in evolutionary multi-objective optimization. Adv. Eng. Softw. Elsevier Publisher **32**, 499–507 (2001)
3. Chalmet, L.G., Lemonidis, L., Elzinga, D.J.: An algorithm for the bi-criterion integer programming problem. Eur. J. Oper. Res. **25**, 292–300 (1986)
4. Cheung, B.K.-S., Langevin, A., Villeneuve, B.: High performing techniques for solving complex location problems in industrial system design. J. Intelligent Manufa. **12**, 455–466 (2001)
5. Coello Coello, C.A.: Handling preferences in evolutionary multiobjective optimization: a survey. In: Congress on Evolutionary Computation Proceedings (2000)
6. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey and the state of the art. Comput. Meth. Appl. Mech. Eng. **191**, 1245–1287 (2002)
7. Coello Coello, C.A., Van Veldhuizen, D., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic/Plenum Publishers (2002)
8. Collette, Y., Siarry, P.: Three new metrics to measure the convergence of metaheuristics towards the pareto frontier and the aesthetic of a set of solutions in biobjective optimization. Comput. Oper. Res. **32**, 773–792 (2005)
9. Correa, E.S., Steiner, M.T.A., Freitas, A.A., Carnieri, C.: A genetic algorithm for the P-median problem. Proceedings 2001 Genetic and Evolutionary Computation GECCO2001 (2001)
10. Cortinhal, M.J., Captivo, M.E.: Genetic algorithms for the single source capacitated location problem. In: Resende, M., Sousa, J.P.d. (eds.) Metaheuristics: Computer Decision-Making, pp. 187–216. Kluwer Academic (2003)
11. Deb, K.: Multi-objective Optimization using Evolutionary Algorithms. John Wiley & Sons (2001)
12. Dias, J., Captivo, M.E., Clímaco, J.: An interactive procedure dedicated to a bicriteria plant location model. Comput. Oper. Res. **30**, 1977–2002 (2003)
13. Dias, J., Captivo, M.E., Clímaco, J.: Dynamic multi-level capacitated and uncapacitated location problems: an approach using primal-dual heuristics. Research Report 26/2004 Inesc-Coimbra, available on http://www.inescc.pt/documentos/26_2004.pdf (2004)
14. Dias, J., Captivo, M.E., Clímaco, J.: Dynamic multi-level capacitated and uncapacitated location problems: an approach using primal-dual heuristics. In: INOC'05 Proceedings, Lisbon (2005a)
15. Dias, J., Captivo, M.E., Clímaco, J.: A Hybrid algorithm for dynamic location problems. Inescc Research Report n. 3/2005, available on http://www.inescc.pt/documentos/3_2005.pdf (2005b)

16. Dias, J., Captivo, M. E., Clímaco, J.: Capacitated dynamic location problems with opening, closure and reopening of facilites. In: Salhi, S., Drezner, Z. (eds.) IMA J. Manage. Math.: Models Appl. Location Anal. **17**(4) 317–348 (2006)
17. Dias, J., Captivo, M.E., Clímaco, J.: Efficient primal-dual heuristic for a dynamic location problem. Comput. Oper. Res. **34**, 1800–1823 (2007b)
18. Dias, J.M.: Localização Simples Multicritério: desenvolvimento de um algoritmo em ambiente interactivo. MsC thesis, Faculdade de Ciências da Universidade de Lisboa (2000)
19. Domínguez-Marín, P., Nickel, S., Hansen, P., Mladenovic, N.: Heuristic procedures for solving the discrete ordered median problem. Ann. Oper. Res. **136**, 145–173 (2005)
20. Ferreira, C.: Problemas de Localização e Distribuição Multicritério: aproximações e estudo de alguns casos com implicações ambientais. Departamento de Matemática (1997)
21. Filipovic, V., Kratica, J., Tosic, D., Ljubic, I.: Fine grained tornament selection for the simple plant location problem. In: Proceedings of the 5th Online World Conference on Soft Computing Methods in Indsutrial Applications WSC5, September 2000
22. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: formulation, discussion and generalisation. genetic algorithms: Proceedings of the Fifth International Conference, San Mateo, CA (1993)
23. Granat, J., Makowski, M.: ISAAP—Interactive Specification and Analysis of Aspiration-Based Preferences. IR-98–052 (1998)
24. Granat, J., Makowski, M.: Interactive specification and analysis of aspiration-based preferences. Eur. J. Oper. Res. **122**, 469–485 (2000)
25. Hosage, C.M., Goodchild, M.F.: Discrete space location-allocation solutions from genetic algorithms. Ann. Oper. Res. **6**, 35–46 (1986)
26. Hultz, J., Klingman, D., Ross, G.T., Soland, R.: An interactive computer system for multicriteria facility location. Comput. Oper. Res. **8**, 249–261 (1981)
27. Huntley, C., Brown, D.: Parallel genetic algorithms with local search. Comput. Oper. Res. **23**, 559–571 (1996)
28. Jaramillo, J., Bhadury, J., Batta, R.: On the use of genetic algorithms to solve location problems. Comput. Oper. Res. **29**, 761–779 (2002)
29. Jones, D.F., Mirrazavi, S.K., Tamiz, M.: Multi-objective meta-heuristics: an overview of the current sate-of-the-art. Eur. J. Oper. Res. **137**, 1–9 (2002)
30. Kratica, J.: Improvement of simple genetic algorithm for solving the uncapacitated warehouse location problem. Adv. Soft Comput. Eng. Design Manufact. 390–402 (1999)
31. Kratica, J., Tosic, D., Filipovic, V., Ljubic, I.: Solving the simple plant location problem by genetic algorithm. RAIRO Oper. Res. **35**, 127–142 (2001)
32. Murata, T., Ishibuchi, H., Gen, M.: Neighborhood structures for genetic local search algorithms. IEEE Trans. Syst. Man Cybernetics 259–263 (1998)
33. Oei, C.K., Goldberg, D.E., Chang, S.-J.: Tournament Selection, Niching and the Preservation of Diversity.91011 (1991)
34. Osman, I.H., Kelly, J.P.: Meta-Heuristics: Theory & Applications. Kluwer Academic Publishers (1996)
35. Reeves, C., Höhn, C.: Integrating local search into genetic algorithms. Modern Heuristic Search Meth. 99–115 (1996)
36. Reeves, C.R.: Using genetic algorithms with small populations. In: Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA (1993)
37. Revelle, C., Laporte, G.: The plant location problem: new models and research prospects. Oper. Res. **44**, 864–873 (1996)
38. Ross, T., Soland, R.: A multicriteria approach to the location of public facilities. Eur. J. Oper. Res. **4**, 307–321 (1980)
39. Rothlauf, F., Goldberg, D.: Redundant representations in evolutionary computation. Research Report n. 2002025, Illinois Genetic Algorithms Laboratory (IlliGAL) Report (2002)
40. Sakawa, M., Kato, K.: An interactive fuzzy satisficing method for general multiobjective 0–1 programming problems through genetic algorithms with double strings based on a reference solution. Fuzzy Sets Syst. **125**, 289–300 (2002)
41. Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. genetic algorithms and their applications: Proceedings of the First International Conference on Genetic Algorithms (1985)
42. Shibuya, M., Kita, H., Kobayashi, S.: Integration of multi-objective and interactive genetic algorithms and its application to animation design. IEEE Transactions on Syst. Man. Cybern. **3**, 646–651 (1999)
43. Shimizu, Y.: Multi-objective optimization for site location problems through hybrid genetic algorithm with neural networks. J. Chem. Eng. Jpn. **32**, 51–58 (1999)

44. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evol. Comput. **2**, 221–248 (1994)
45. Szczepanski, M., Wierzbicki, A.: Multiple criteria evolutionary algorithms in relation to reference point approaches and nadir estimation. International Conference on Multiple Criteria Decision Making, Austria 2002
46. Veldhuizen, D.A.V., Lamont, G.B.: Multiobjective evolutionary algorithms: analysing the state-of-the-art. Evol. Comput. **8**, 125–147 (2000)
47. Veldhuizen, D.V.: Multiobjective Evolutionary Algorithms: Classifications, Analysis and New Innovations, PhD Thesis, Air Force Institute of Technology, Faculty of the Graduate school of Engineering (1999)
48. Weber, M., Borcherding, K.: Behavioral influences on weight judgements in multiattribute decision making. Eur. J. Oper. Res. **67**, 1–12 (1993)
49. Wierzbicki, A.: Multi-objective and reference point optimization tools. In: Makowski, M., Wierzbicki, A.P., Wessels, J. (eds.) Model-Based Decision Support Methodology with Environmental Applications, pp. 215–247. Kluwer Academic Publishers (2000a)
50. Wierzbicki, A.: Reference point methodology. In: M. M. Andrzej Wierzbicki, Jaap Wessels (eds.) Model-Based Decision Support Methodology with Environmental Applications, pp. 71–89. Kluwer Academic Publishers (2000b)
51. Yagiura, M., Ibaraki, T.: The use of dynamic programming in genetic algorithms for permutation problems. Eur. J. Oper. Res. **92**, 387–401 (1996)
52. Zitzler, E., Laumanns, M., Thiele, L., Fonseca, C.M., Fonseca, V.G.D.: Why quality assessment of multiobjective optimizers is difficult. In: Proceedings of the Genetic and Evolutionary Computation Conference (2002a)
53. Zitzler, E. Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Giannakoglou, K., Tsahalis, D., Periaux, J., Papailiou, K., Fogarty, T. (eds) Evolutionary Methods for Design, 95–100. CIMNE Barcelona, Spain (2002b)
54. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evol. Comput. **3**, 257–271 (1999)
55. Zitzler, E., Thiele, L., Deb, K.: Comparison of multiobjective evolutionary algorithms: empirical results. Evol. Comput. **8**, 173–195 (2000)