Multi-objective Memetic Optimization for the Obnoxious p-Median Problem

J. M. Colmenar josemanuel.colmenar@urjc.es
Dept. de Ciencias de la Computación, Universidad Rey Juan Carlos, Móstoles, 28933 (Madrid), Spain

R. Martí rafael.marti@uv.es
Dept. de Estadística e Investigación Operativa, Universidad de Valencia, Burjassot, 46100 (Valencia), Spain

A. Duarte abraham.duarte@urjc.es
Dept. de Ciencias de la Computación, Universidad Rey Juan Carlos, Móstoles, 28933 (Madrid), Spain

Abstract

Location problems have been extensively studied in the optimization literature, being probably the p-median one of the most tackled models. The Obnoxious p-median is an interesting variant that appears in the context of hazardous location. The aim of this paper is to formally introduce a bi-objective optimization model for this problem, in which a solution consists of a set of p locations, and two conflicting objectives arise. On the one hand, the sum of the minimum distance between each client and its nearest open facility and, on the other hand, the dispersion among facilities. Both objective values should be kept as large as possible for a convenient location of dangerous facilities.

We propose a Multi-Objective Memetic Algorithm (MOMA) to obtain high-quality approximations to the efficient front of the bi-objective obnoxious p-median problem (*Bi-OpM*). In particular, we introduce efficient crossover and mutation mechanisms. Additionally, we present several multi-objective local search methods. All the strategies are finally incorporated in a memetic algorithm, which limits the search to the feasible region, thus performing an efficient exploration of the solution space. Our experimentation includes two well established multi-objective methods, NSGA-II and SPEA2, favors the proposed memetic algorithm.

Keywords

Memetic algorithms, local search, multi-objective optimization, combinatorial problems, obnoxious p-median.

1 Introduction

Multi-objective optimization aims at simultaneously optimizing several conflicting objectives. For such kind of problems, a single optimal solution may not exist, and the optimization search must deal with all the objectives at the same time. Thus, without any loss of generality, we can assume the following formulation of the m-objective maximization problem:

$$
\begin{aligned}
&\text{Maximization} \\
\mathbf{y} = \mathbf{f}(\mathbf{x}) &= [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x})] \\
&\text{Subject to} \\
\mathbf{x} &= (x_1, x_2, \ldots, x_n) \in \mathbf{X} \\
\mathbf{y} &= (y_1, y_2, \ldots, y_m) \in \mathbf{Y}
\end{aligned}
$$

where $\mathbf{x}$ is the vector of n decision variables, $\mathbf{f}$ is the vector of m objective functions, $\mathbf{X}$ is the feasible region in the decision space, and $\mathbf{Y}$ is the feasible region in the objectives space.

Comparing alternative solutions to a given problem is one of the first difficulties encountered when moving from single-objective optimization to multi-objective optimization. In other words, how do we know that one solution is better than another when the performance is evaluated by using more than one objective function value? Given that the merit of a solution is represented by a vector of objective function values, it is not always possible to determine when a vector is strictly larger or smaller than another. Multi-objective optimization considers the concept of efficiency, introduced by Vilfredo Pareto in 1896 (Pareto, 1896). Essentially, efficiency means that a solution to a multi-objective function is such that no single objective can be improved without deteriorating another one.

A solution $\mathbf{x_1} \in \mathbf{X}$ is said to dominate another solution $\mathbf{x_2} \in \mathbf{X}$ (denoted as $\mathbf{x_1} \succ \mathbf{x_2}$) if the following two conditions are satisfied when considering the maximization of the objectives:

$$
\begin{aligned}
&\forall i \in \{1, 2, ..., m\}, \qquad f_i(\mathbf{x_1}) \geq f_i(\mathbf{x_2}) \\
&\exists j \in \{1, 2, ..., m\}, \qquad f_j(\mathbf{x_1}) > f_j(\mathbf{x_2})
\end{aligned}
$$

If there is no solution which dominates $\mathbf{x} \in \mathbf{X}$ it is said that $\mathbf{x}$ is a non-dominated or efficient solution. The non-dominated set of the entire feasible search space $\mathbf{X}$ is the *Pareto Optimal Set (POS)*. The image of the *POS* in the objective space is the *Pareto Optimal Front (POF)* of the multi-objective problem at hand. A multi-objective optimization problem is solved when the entire *POS* is found.

In this paper we deal with the bi-objective Obnoxious p-median problem (*Bi-OpM*) (Belotti et al., 2007), which can be formally described as follows. Let I be a set of clients, let J be a set of facilities, let $d_c(i, j)$ be a distance function between the client $i \in I$ and the facility $j \in J$, and let $d_f(j_1, j_2)$ be a distance function between each pair of facilities ($j_1, j_2 \in J$). Any subset S of size p from the set J is a feasible solution. Facilities in S are called open facilities, while facilities in $J \setminus S$ are known as closed or unopened facilities. It is worth mentioning that in this problem each client is assigned to the nearest facility in S.

Associated to each solution, we define two objective functions. The first one, f_1, represents the distance from clients to the obnoxious facilities and it is computed as the sum of the minimum distances between each client and the nearest facility in S. More formally:

$$
f_1 = \sum_{i \in I} \min\{d_c(i, j) : j \in S\} \tag{1}
$$

The second objective function, f_2, represents the dispersion of the facilities, which is computed as the sum of the minimum distances from each facility to the rest of the facilities in S. In mathematical terms:

$$f_2 = \sum_{j_1 \in S} \min\{d_f(j_1, j_2) : j_2 \in S\} \tag{2}$$

As stated before, one of the goals of this paper is to propose a formal bi-objective optimization model for the OpM problem. Hence, after all the definitions and features of the problems detailed above, we are able to define the model in the following way:

Maximize

$$f_1 = \sum_{i \in I} \min\{dc_{ij} : j \in S\}$$

$$f_2 = \sum_{j_1 \in S} \min\{df_{j_1 j_2} : j_2 \in S\}$$

Subject to

$$S \subseteq J$$

$$|S| = p$$

In this paper, we propose a Multi-Objective Memetic Algorithm (MOMA) to deal with this problem. In particular, we introduce different multi-objective local search approaches, as well as specific crossover and mutations operators to assure that individuals represent feasible solutions for *Bi-OpM* . These elements are then integrated in our memetic implementation. One of the main objectives of this paper is to evaluate the effect of different multi-objective local search methods in the performance of the corresponding MOMA. We therefore compare each MOMA variant with two classical algorithms: the Non-dominated Sorting Genetic Algorithm II, NSGA-II (Deb et al., 2002); and the Strength-Pareto Evolutionary Algorithm 2, SPEA2 (Zitzler et al., 2002). In order to isolate the effect of the local search method, NSGA-II and SPEA2 are configured with the proposed specific crossover and mutation operators. Additionally, the set of parameters were tuned through the iterated race method (López-Ibáñez et al., 2011).

The rest of the paper is organized as follows. In Section 2 related works are described. In Section 3 we present three multi-objective local search methods for the *Bi-OpM* problem. Section 5 is devoted to detail the internal strategies of the memetic algorithm we propose. The experimentation and the analysis of results are detailed in Section 6. Finally, conclusions and future work are drawn in Section 7.

2 Related works

Among the wide number of facility location problems (FLP), where both location and customer strategies are considered, we can find the uncapacited facility location problem (UFLP) as the most studied model. The UFLP involves locating a number of facilities to minimize the sum of the cost of the links between the facilities and the customers. This problem is $\mathcal{NP}$-complete, as proven in (Krarup and Pruzan, 1983). The warehouse location problem (WLP), also known as (simple) facility or plant location problem, is an uncapacited optimization problem, which can be modeled as a Linear Programming problem (Balinski, 1965) or a network problem (Drezner and Hamacher, 2004; Melkote

and Daskin, 2001). It is a site-selecting location-allocation model with a min-sum objective. More precisely, given a number of potential facility or warehouse sites, the set of costumers has to be completely serviced, minimizing the total fixed site-costs (location) plus the total variable customer assignment costs (allocation). The WLP is also $\mathcal{NP}$-hard (Garey and Johnson, 1979; Papadimitriou and Yannakakis, 1991) and, as such, has been in the focus of researchers interested in developing heuristic approaches to provide high-quality solutions.

The p-median problem, pMP (Hakimi, 1964), is very similar to WLP but it presents two different aspects: (1) there are no fixed site-costs involved and (2) the number of opened sites, p, is no longer a decision variable, but it becomes included in the model. For fixed values of p, the pMP can be solved in polynomial time, but it is strongly $\mathcal{NP}$-hard for variable values of p (Current et al., 2004). Consequently, any pMP is a special case of the general class of WLPs. Therefore, many approaches to solve the later have been proposed to tackle the former. More recently, the pMP was approached by using dispersion, population and equity criteria in (Batta et al., 2014).

The obnoxious p-median problem, OpM , (Church and Garfinkel, 1978; Erkut and Neuman, 1989) deals with situations in which there exist several kinds of facilities presenting obnoxious or semi-obnoxious features. A facility is called undesirable or obnoxious if its influence in the surrounding area is negative. Such facilities may be involved when dealing with hazardous materials, waste disposal, water treatment, nuclear power, or chemical plants. Similarly, large public facilities like airports or railway stations, can also fall in this category. In these cases, facilities may cause lower quality of life, having also an adverse effect on the standard living and the environment, just to mention two associated issues (Segal, 2003). Therefore, the goal in this context is to reduce the negative effect of obnoxious facilities by selecting which ones will be opened in a given scenario. As in pMP, the number of facilities to be opened, p, is included in the model, whereas the number of clients, which correspond to towns or cities, is fixed. A deeper description of the problem can be found in (Colmenar et al., 2016), where the objective function is to maximize the sum of the minimum distances between each open facility and its nearest client.

(Tralhão et al., 2010) studied the problem of distributing containers in cities by using a multi-objective approach. The objective function consists of the total investment cost, the average distance from dwellings to the respective multi-compartment container, the number of individuals too close to any container, and the number of dwellings too far from the respective multi-compartment container. The authors proposed a Mixed Integer Linear Program (MILP) approach to obtain an interactive decision support system, where a human expert is responsible of selecting the most suitable solution. Their proposal is complex and it is not easily generalized to other problem scenarios, given the complexity of the model. The same authors published later in (Coutinho-Rodrigues et al., 2012) a simplification of their MILP approach, taking into account only two objectives: the container investment cost and a weighted average customer dissatisfaction.

Another work related to an obnoxious bi-objective location scenario is described in (Fernandes et al., 2014), where a waste transfer station siting problem is taken as an example. Again, a MILP technique is applied in order to generate the non-dominated set of solutions for their interactive decision support system. Additionally, their approach incorporates a multi-attribute analysis module (to allow the decision-maker to proceed with a more detailed analysis of a subset of solutions) and selected from the first interactive phase. Finally, the MILP is tested over case study of a real world problem

applied to waste transfer station siting.

Recently, a genetic algorithm (GA) was developed to deal with hazardous materials (Ardjmand et al., 2016). Specifically, the cost of transportation is considered to be of a stochastic nature. The objective function minimizes the total cost and risk of locating facilities and transportation of hazardous materials. The proposed GA obtains results in fast computation times, but it only considers a weighted fitness function where the risk and cost factors are combined linearly.

The aim of this paper is to formally introduce a bi-objective optimization model for the bi-objective Obnoxious p-median problem (Bi-OpM). As it is indicated in (Belotti et al., 2007), the OpM does not consider the dispersion of the opened facilities, which would result in an improved model. Then, good solutions might contain two (ore more) very close facilities. Decision makers usually advise against this kind of solutions. For example, when facilities represent nuclear plants it is important to separate them from population centers. However it is also important to separate, as much as possible, the set of nuclear plants from each other.

3 Multi-objective local search methods

The use of multi-objective local search methods has not been extensively treated in the context of multi-objective optimization problems. As far as we know, the first multi-objective memetic algorithm, known as the multi-objective genetic local search (MOGLS), was presented in (Ishibuchi and Murata, 1998). However, the MOGLS used a weighted sum function for parent selection and local search. The dominance was only used for maintaining a population archive.

Another multi-objective memetic algorithm is described in (Cheng et al., 2009). There, a local search is coupled with NSGA-II, but the local search is applied only to one solution, which is selected through a weighted aggregation function generated at random.

In (Li et al., 2013) another hybridization between NSGA-II and a local search is presented. Specifically, the proposed method tackled the environmental power unit commitment problem, where two objectives are considered, minimizing the emissions and the generation costs. As shown by the authors, the solutions have more quality than those obtained with the traditional NSGA-II.

In this work we propose three different local search algorithms for the Bi-OpM . These three methods differ in how each one explores the neighborhood, which is the same for the three of them. Specifically, it is based on the exchange of one selected facility (belonging to the current solution) with one of the non-selected facilities. We denote this operation as *exchange*, and it is defined as follows: given a feasible solution S, a facility $j_1 \in S$, and a facility $j_2 \in J \setminus S$, this operation produces a neighbor solution S' such that $S' = S \setminus \{j_1\} \cup \{j_2\}$. For the sake of clarity, we represent this operation as:

$$S' \leftarrow exchange(S, j_1, j_2)$$

All local search methods are implemented by using the first improvement strategy, which tries to reduce the computational effort to explore the whole neighborhood by performing the first improving move encountered. In this kind of scanning, the order in which the neighbors are inspected can have a significant influence on the efficiency of the search. To add a diversification component in our exploration, the selection of both facilities is performed at random.

The first local search algorithm is based on the concept of dominance since it only performs moves that lead to solutions not dominated by the current one. Let us il-

lustrate the search strategy with an example. In Figure 1, we show a solution S in a bi-dimensional space where f_1 and f_2 are the objective functions. Considering that the Bi-OpM is a maximization problem, the position of S might define four different regions in this space. Solutions in D_S are strongly dominated by S since those solutions presents lower values in both objective functions. Regions ND_1 and ND_2 contain solutions that are not dominated by S. Specifically, any solution $S_1 \in ND_1$ verifies that $f_1(S_1) \geq f_1(S)$ and $f_2(S_1) \leq f_2(S)$. Similarly, any solution $S_2 \in ND_2$ verifies that $f_2(S_2) \geq f_2(S)$ and $f_1(S_2) \leq f_2(S)$. Finally, any solution in ND_{12} dominates S in both objective functions.

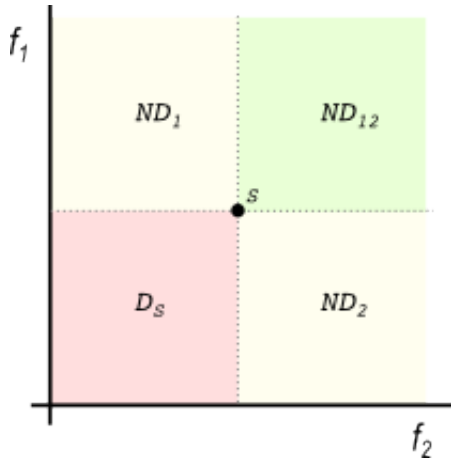


Figure 1: Dominance regions for a given solution S.

The first improvement strategy is called *Dominance-Based Local Search* (*DBLS*). It starts by considering a feasible solution S. Then, the method scans the neighborhood by exchanging facilities at random. Neighbor solutions in D_S are discarded. Once it finds an improving move, i.e., a solution S' in any region ND_1, ND_2, or ND_{12}, the search continues from S'. This strategy ends when all neighbor solutions belong to D_S which means that all of them are dominated by the current one. During the search process, *DBLS* keeps an archive with the set of non-dominated solutions. Update operations of this archive are fast when the local search finds solutions in D_S since they are directly discarded. However, the remaining operations involve the exploration of the whole archive since the inclusion of a non-dominated solution may imply to remove one or more solutions from it. In order to reduce the associated computing times, the update operation is run once the DBLS ends.

The second local search follows a different approach. In particular, it considers either f_1 or f_2 but not both objectives at the same time. We call it *Alternate Objectives Local Search* ($AOLS$). The alternation of objectives is not guided by any criteria. On the contrary, it is randomly chosen with the purpose of not biasing the way in which the solution space is explored by the local search.

Considering again the example shown in Figure 1, $AOLS$ accepts moves to a neighbor solution belonging to either the region defined by $ND_1 \cup ND_{12}$ (when the selected objective is f_1) or to the region $ND_2 \cup ND_{12}$ (when the selected objective is f_2). Therefore, the search area explored with *AOLS* at each iteration is smaller than in the case of *DBLS*. It is worth mentioning that this fact does not imply that *AOLS* obtains

worse results than *DBLS*. It is well documented in local search methods that the size of the neighborhood is not necesarily related with the quality of the solution obtained at the end of the process.

The third approach is based on a composite function of both objective values that we call *Normalized Weighted Sum Local Search* (*WSLS*). Considering that both objectives are functionally equivalents (i.e., to maximize the sum of a minimum distance), we set the same weight for both of them. Additionally, we have normalized their values to avoid the bias of their different orders of magnitude. Specifically, given a solution S its corresponding evaluation is computed as:

$$f_{WS}(S) = \frac{f_1(S)}{maxF_1} + \frac{f_2(S)}{maxF_2} \tag{3}$$

where $maxF_1$ and $maxF_2$ are, respectively, the maximum value of f_1 and f_2 of any solution visited during the search process. The main idea of *WSLS* is to guide the search toward a target or ideal solution S^* (which may or may not exist) whose coordinates (in the bidimensional space of the Bi-OpM problem) are $S^* = (maxF_1, maxF_2)$. Figure 2 illustrates this situation by considering a solution S and a target solution S^*. We have also represented the hyperplane Y that identifies those solutions S' such that $f_{WS}(S') = f_{WS}(S)$. Therefore, the third local search is able to find solutions S' verifying $f_{WS}(S') > f_{WS}(S)$.

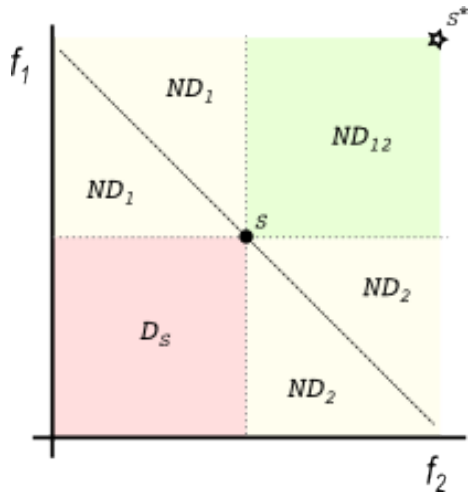


Figure 2: Dominance regions for *WSLS*.

Considering the three proposed multi-objective local search methods from a theoretical point of view, we can conclude that *DBLS* explores the largest region, which could drive to long computing times. On the other hand, *WSLS* scans solutions in a considerably smaller region, which could mean shorter computing time but also lower quality. However, as we will observe in the computational experience, these assumptions do not apply to the Bi-OpM problem.

4 Global and Local Search Balance

As it is well documented in the related literature, the balance between global (exploration) and local (exploitation) search is still an open question. We found several pro-

posals in the literature to control this balance, and we would like to highlight the discussion we found in (Sudholt, 2012). There, some questions were raised: (1) how often should be run the local search?; (2) on which solutions should be applied?; (3) how long should the local search be run?; (4) how efficient does a local search need to be?.

MOMAs mainly focus on the first and third questions by including several parameters that active or inhibit strategies to deal with the questions enumerated above. In particular, the strategy used to answer the first one is to call the corresponding local search with a fixed frequency or, alternatively, with a given probability. It Hart (1994) it is investigated the influence of this parameter, denoted as ν, on the performance of the search. The experimental results showed that evolutionary algorithms with large populations are most effective when local search is used infrequently (at least in the context of continuous optimization). Another relevant experimental conclusion revealed that a large local search frequency is preferred when the algorithm is not able to identify promising regions. Finally, the authors concluded that the inclusion of elitism favored, in general, the degree of exploitation with respect to the exploration. Therefore, in this situation the evolutionary algorithm is usually configured with a small value of the frequency.

In the context of the *Bi-OpM* problem there are two alternatives to set the value of ν. On the one hand, to tie it to individuals (i.e., by applying the local search to improve only ν individuals within the population) or to generations (i.e., by applying the local search to improve every ν generations). We have experimentally tested that in the *Bi-OpM* problem the local search procedure virtually monopolized the execution time when it is included in a MOMA template. This fact can be partially explained by the relatively low quality of the individuals in initial generations. In those cases, the local search employs a large number of iterations to improve those solutions (i.e., consuming practically the whole budget of evaluations allowed). This behavior is very well documented in some metaheuristics like GRASP (Feo et al., 1994), where the largest improvement is obtained at the beginning of the local search application. So, it is clear that a balance between the global and local search procedure has to be incorporated to our algorithm. Therefore, we have selected the second approach, considering the run of the local search on a generation basis. In particular, we will manage ν as the probability of executing the local search on the current generation of the MOMA.

The third question indicated above is related mainly with the efficiency of the local search methods. In general, they produce high quality outcomes but consuming a considerable computing time. In the context of MOMAs, this situation can either improve or deteriorate the performance of the algorithm when considering as stopping criterion the number of evaluations. There has been proposed several strategies to deal with this situation. The fist one considers to abort the search before it reaches the corresponding local optimum. The parameter that controls this strategy is called local search depth, denoted as τ. In (Ishibuchi et al., 2003) it is described an alternative strategy to reduce the computing time of the local search (i.e., by reducing the size of the neighborhood to a some fixed parameter k). This strategy is equivalent to one used in the variable-neighborhood search algorithms methodology (Mladenović and Hansen, 1997).

According to our empirical experience, the best results are obtained by adjusting the value of the local search depth according to the fitness landscape. In order to fulfill this requirement, we stop the search when it performs τ improvement steps. Therefore, if the corresponding individual within the population is located in a non-promising region, we allow the search to exhaustively scan the neighborhood to look for an improving direction in the search space. On the other hand, if almost any move within the

neighborhood produces an improvement, we select a move, according to the particular local search strategy, to save evaluations of the objective function. It is important to remark that, at the end of the process, the entire population is composed by local optima since there are not improvement moves available in the corresponding neighborhood.

It is worth mentioning that the strategies to balance global and local search described above are not exhaustive. We have only considered those ones that are directly applicable to the *Bi-OpM* problem. In particular, we only study the local search frequency (ν) and the local search depth (τ) as the most typical mechanisms. These two parameters should not be analyzed in isolation, since there is a strong dependency between both parameters. In (Sudholt, 2012) is pointed out that setting the value of a parameter ignoring the other one often does not make much sense. Indeed, there are some theoretical results to determine these parameters (see (Sudholt, 2012)). However, the existence of no free lunch theorems (Igel and Toussaint, 2005) justifies the use of an experimental setting to design an algorithm as much adapted to the optimization problem as possible.

Then, we will investigate in Section 6.2 the influence of both, ν and τ over the performance of the algorithm. Notice that, a good trade-off between global and local search clearly not only depends on the particular optimization problem but also on the evolutionary algorithm applied to it, e.g., genetic operators, population size, selection pressure, or the mutation rate (Sudholt, 2012).

5 Multi-objective Memetic Algorithm

Multi-objective programming techniques focus on finding the set of efficient solutions or Pareto Optimal Set for a given problem or, in the case of heuristic procedures, an approximation of the efficient set. In this paper, we describe a Multi-Objective Memetic Algorithm (MOMA) for the *Bi-OpM* . In particular, our approach is based on customized elements that take advantage on specific problem information. In other words, we are not applying a generic evolutionary method that, as a black box solver, can tackle any problem. On the contrary, we propose a tailored method to solve the *Bi-OpM* in an efficient way. In order to address this goal, we propose effective and customized mechanisms to create the initial population and to implement the crossover, mutation, improvement, and update strategies.

In order to have an effective search, genetic operators should be able to keep the individuals within the feasible space. Considering the nature of the *Bi-OpM* problem, each individual represents a complete solution encoded in one chromosome. More precisely, each individual will store a number of p identifiers corresponding to the opened facilities, and each gen in the chromosome stores the label that identifies a unique facility. This label is an integer value that ranges from 1 to $|J|$. It is worth mentioning that the assignation of clients to facilities does not require an specific codification since this assignation is deterministically performed (i.e., each client is assigned to the closest facility).

The initial population P is created by considering a random generator where each individual is formed by selecting (without repetition) p different facilities from J. The number of individuals in P is determined by the input parameter *size*. The rationale behind this is to construct a population as much diverse as possible. Therefore, this set is not improved with any of the local search procedures described in Section 3.

Standard crossover operators may produce infeasible solutions in the tackled optimization problem. For instance, a typical single-point crossover operator may produce individuals with repetitions of a facility. To illustrate this situation, let us assume that

we have $|J| = 10$ and $p = 4$. Figure 3 shows two parents that are crossed with the typical single point implementation. As it can be seen, facility 5 is repeated in Offspring 1, which results into an infeasible solution. Notice that in this example, Offspring 2 is a feasible solution since the four facilities are different.

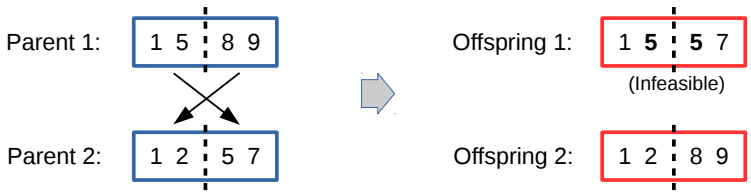


Figure 3: Typical single-point crossover. Offspring 1 is not a feasible solution for the *Bi-OpM* problem.

In order to avoid the aforementioned situation, we have implemented a variant of the single-point crossover based on the operator described in (Alp et al., 2003). There, the authors distinguish between the common genes of the parents, called *fixed genes*, and the non-common genes, called *free genes*. In their proposal, a draft individual is generated, its fitness is computed and then a greedy selection heuristic is applied to determine the genes that will be finally selected. In our case, we follow the strategy of maintaining the fixed genes, but we have applied a faster implementation that does not make use of the greedy heuristic. Algorithm 1 illustrates how our crossover operator proceeds. Given two parents S_1 and S_2, it first identifies the fixed genes (Step 1) and free genes (Step 2). Then, it initializes two offspring solutions by setting them to the set of fixed genes (Steps 3 and 4). Finally, for each offspring, the method randomly selects the remaining genes from the free genes list (Steps 6 and 9). The algorithm finishes when both offspring solutions contains exactly p facilities.

Algorithm 1: Proposed Crossover operator.

1: $FixG \leftarrow S_1 \cap S_2$
2: $FreeG \leftarrow S_1 \cup S_2 \setminus S_1 \cap S_2$
3: $S'_1 \leftarrow FixG$
4: $S'_2 \leftarrow FixG$
5: **while** $(FreeG \neq \emptyset)$ **do**
6: $\quad s_1 \leftarrow rand(FreeG)$
7: $\quad S'_1 \leftarrow S'_1 \cup \{s_1\}$
8: $\quad FreeG \leftarrow FreeG \setminus \{s_1\}$
9: $\quad s_2 \leftarrow rand(FreeG)$
10: $\quad S'_2 \leftarrow S'_2 \cup \{s_2\}$
11: $\quad FreeG \leftarrow FreeG \setminus \{s_2\}$
12: **end while**
13: **return** (S'_1, S'_2)

If the number of fixed genes is close to p, the crossover operator may produce offspring solutions similar to their parents. This fact usually accelerates the convergence of the algorithm (reducing the computing time) but reducing population diversity. On the other hand, if the number of fixed genes is close to zero, the selection of facilities is

almost random, producing the opposite effect that the aforementioned one. As we will describe in the computational experience, we have detected that one effect balances the other. Therefore, there is no need to introduce an additional strategy to increase the diversity or accelerate the convergence.

The crossover operator is applied to pairs of solutions. Each parent in the corresponding pair is selected through a binary tournament operator, which randomly takes two solutions from the entire population and returns the dominant one. In the case of two non-dominated solutions, the method returns one of them at random. Therefore, after applying the crossover operator, a new offspring population P' with the same size than P is generated.

The mutation operator also produces feasible individuals by considering the exchange move described in Section 3. In particular, it selects at random a facility $j_1 \in S$. Then, according to the probability of mutation, j_1 is removed from S and, simultaneously, a facility $j_2 \in J \setminus S$ selected also at random is incorporated in S.

The improvement strategy has a significant impact on the number of evaluations. Then, we must be selective to apply it to a set of promising solutions which, in our case, are the non-dominated ones. Our method extracts this set by merging P and P'. For the sake of clarity, we denote this set as $ND \leftarrow P \cup P'$. The proposed method improves the solutions in ND using one of the local search strategies described in Section 3, generating a new population P'' with a size that could be different than *size*.

The update operator used in our algorithm is based on the crowding distance, as described in the NSGA-II algorithm (Deb et al., 2002). In particular, this strategy considers the union of P, P', and P'', sorts those solution according to the crowding distances, and returns a number of *size* best ones. Before starting the next iteration P is updated with these solutions. These solutions are also stored in a external elite archive, E, that is processed after reaching the stopping condition of the MOMA. Finally, the algorithm only returns the non-dominated solutions from E.

Algorithm 2 shows the main steps of the proposed MOMA. It starts by creating a random population in Step 1. Then, it enters in the main loop until the stopping condition is met (i.e., a maximum number of either evaluations or generations is reached). Each generation corresponds to one iteration of the algorithm. After the initialization of the population, which will be stored in P, the crossover and mutation operators are applied in steps 3 and 4, considering the probabilities of crossover (*pCx*) and mutation (*pMut*), respectively. The resulting individuals are stored in P'. Next, we obtain the non-dominated individuals of both parents and offsprings in Step 5. The local search method receives the set of non-dominated solutions and the search parameters described in Section 4 (i.e., the frequency of improvement, ν, and the depth of the search, τ). In particular, with a probability of ν, all individuals in ND are improved with a local search depth equal to τ (see 6). As a result, we obtain a new set called P''. Then, following the strategy of the classical NSGA-II implementation, we apply the crowding distance to reduce the number of individuals to *size*, which is the size of the population introduced as a parameter in the algorithm. Then, after Step 7, the population is updated and the non-dominated individuals of this population are stored in the elite set E. Once the stopping condition is reached, the final step consists of processing the solutions in E by discarding the dominated solutions (see Step 10).

6 Computational results

In order to assess the performance of the proposed MOMA, we have conducted different experiments with two goals. On the one hand, to show the merit of the presented

Algorithm 2: Proposed Multi-Objective Memetic Algorithm (MOMA).

```
1: P ← InitPop(size, p)
2: while not StopCondition() do
3:    P' ← doCrossover(pCx, P)
4:    P' ← doMutation(pMut, P')
5:    ND ← findNonDominated(P ∪ P')
6:    P'' ← runLocalSearch(ND, ν, τ)
7:    P ← reduceByCrowding(size, P ∪ P' ∪ P'')
8:    E ← E ∪ findNonDominated(P)
9: end while
10: return findNonDominated(E)
```

strategies. On the other hand, to tune the relevant search parameters. First of all, we detail the experimental setup (Section 6.1). Then, we study the influence of the frequency and depth parameters on the performance of the local search procedures (Section 6.2). Once we have experimentally adjusted these two parameters, we compare the three proposed multi-objective local search procedures (Section 6.3). Finally, we compare the best identified MOMA variants with two well-known multi-objective algorithms (NSGA-II and SPEA2) whose parameter values were tunned using the iterated race method (Section 6.4).

6.1 Exprimental setup

All the experiments were executed on the same computer, an Intel i5 660 processor running at 3.3 GHz with 8 Gb of RAM using GNU/Linux. In addition, all the algorithms were coded in Java, and executed using the version 1.7 of the Java Runtime Environment.

We have considered 8 instances previously used in the related literature, where the number of nodes ranges from 400 to 900. In order to facilitate future comparisons, we make this set publicly available at the following URL: http://www.optsicom.es/biopm/. Table 1 summarizes the main characteristics of the set, where n indicates the number of nodes, $|I|/|J|$ represents the number of clients/facilities, and p the number of facilities required in the solution.

Instance	n	$\|I\|/\|J\|$	p
pmed17-p25	400	200	25
pmed20-p50	400	200	50
pmed22-p62	500	250	62
pmed28-p75	600	300	75
pmed33-p87	700	350	87
pmed36-p100	800	400	100
pmed39-p112	900	450	112
pmed40-p225	900	450	225

Table 1: Instances generated from the OR-Library (Beasley, 1990).

Considering that one of the main objectives of the experimental comparison is to isolate the effect of relevant parameters and/or strategies, we use the same basic configuration for all MOMA variants. Specifically, in Table 2 we show the values for

the MOMA parameters. They were set by considering the guidelines given in (Coello et al., 2006). In addition, it is suggested that the number of evaluations should be large enough to ensure a good exploration. Hence, according to these parameter values described above, it should be larger than 10^5. We then set this value to 10^6.

As we described in Section 5, we propose specific initial population, crossover, mutation and update strategies directly tailored to the *Bi-OpM* problem. Therefore, all MOMA variants use them in the basic configuration.

Parameter	Value
Population size	100
Generations	1000
Crossover probability	0.7
Mutation probability	0.1

Table 2: Parameters of the global search of the MOMA.

6.2 Balancing global and local search strategies

The first set of experiments are designed to find a balance between global and local search strategies within the proposed procedures. For the sake of brevity, we only show MOMA results that consider the dominance-based local search ($DBLS$) with several values for ν (local search frequency interpreted as a probability and τ (local search depth). See Section 4 for further details. We have experimentally tested that the results with the other two multi-objective local search methods are equivalents.

We consider 5 different values for each parameter to analyze their influence over the performance. In particular, $\nu = \{0, 0.25, 0.50, 0.75, 1\}$, where $\nu = 0$ indicates that the local search is never used, while $\nu = 1$ means that the local search is applied on each generation. Similarly, $\tau = \{\infty, 10, 5, 1\}$, where $\tau = \infty$ implies the execution of the local search until it finds a local optimum, $\tau = 10$ indicates that the local search stops after 10 improvement moves, and so on until $\tau = 1$. In order to have a fair comparison, all MOMA variants are stopped after they reach either the maximum number of generations (1000) or evaluations (10^6).

Table 3 shows the hyper-volume indicator (Zitzler and Thiele, 1999), *Hyp.*, the number of solutions in the approximate Pareto front, #*Sols.*, the computing time, *Time*, and the number of evaluations, #*Evals.*. These results are averaged over 30 independent runs and 8 instances and they are normalized with respect to the best performing algorithm in the corresponding instance. Therefore, the larger the indicator the higher the value, being 1.0 the largest possible value.

Attending to these results, and in the case of the hypervolume indicator, for any ν frequency value, the short-depth local search ($\tau = 1$) always obtains better results than the other values. Besides, given a value for the depth of the local search, the higher the frequency of the local search, the better the hypervolume results. Therefore, in terms of hypervolume, the best combination for the depth and frequency of the local search is $\nu = 1.0$ and $\tau = 1$, which means that the local search is run in all the generations with the shortest depth.

A similar result is obtained in terms of the number of solutions in the approximate Pareto front. These results show that the effect of a short local search run on each generation is able to improve the current non-dominated front increasing its width, which also may increase the hypervolume of the front.

ν	τ	*Hyp.*	*#Sols.*	*Time*	*# Evals.*
0.0	0	0.7812	0.1642	0.0409	0.0167
0.25	∞	0.4317	0.0127	0.8554	1
	10	0.8147	0.1913	0.2472	0.2466
	5	0.8485	0.227	0.1809	0.1693
	1	0.9278	0.4894	0.102	0.0781
0.5	∞	0.424	0.0094	0.8683	1
	10	0.8308	0.1685	0.41	0.4436
	5	0.8809	0.2798	0.3487	0.3811
	1	0.9725	0.6883	0.2089	0.2038
0.75	∞	0.413	0.0088	0.8434	1
	10	0.841	0.1641	0.5664	0.6482
	5	0.892	0.2632	0.484	0.5853
	1	0.9873	0.7956	0.3359	0.3515
1.0	∞	0.4196	0.008	0.8507	1
	10	0.8454	0.1623	0.6604	0.808
	5	0.8997	0.2635	0.5997	0.7561
	1	0.9937	0.8685	0.4521	0.5092

Table 3: Normalized stats for different values of local search depth parameter.

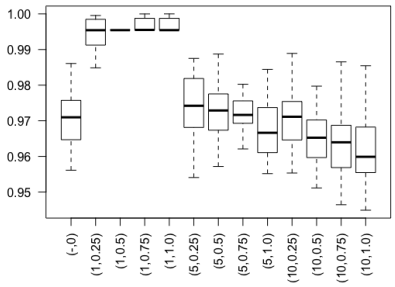
In terms of execution time and number of evaluations, the shorter the search depth, the lower the number of evaluations and the shorter the run. In this problem, we see that reducing the local search depth increases the quality of the solutions due to a better balance between the global and local explorations. However, the removal of the local search obtains the worst results in terms of hypervolume, as shown by the $\nu = 0.0$ and $\tau = 0$ indicators values.

Breaking down the hypervolume results into each one of the instances in our set, we obtain the box plots displayed in Figure 4. This representation allows us to show how the executions are distributed. Each box contains 50% of the data, where the upper and lower ends of the box correspond to the first and third quartile, respectively. The line inside the box establishes the median of the considered data. Finally, the whiskers represent the maximum and minimum value of the data. Again, it is clear that the best results are obtained by those runs with a short local search depth ($\tau = 1$) in all the instances. And, among them, the best hypervolume value is obtained in the case of $\nu = 1.0$. For the sake of a clearer display, we have removed from the figure the executions where $\tau = \infty$ because the hypervolume values were too low.

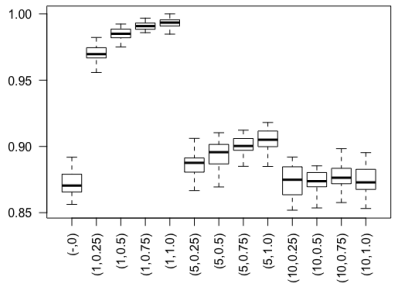
Therefore, the remaining experiments consider $\nu = 1.0$ and $\tau = 1$ for the frequency and depth in the local search, respectively. The MOMA configured with this parameters exhibits the best compromise between global and local search in terms of both quality (according to the hyper-volume) and execution time.

6.3 Multi-objective local search performance

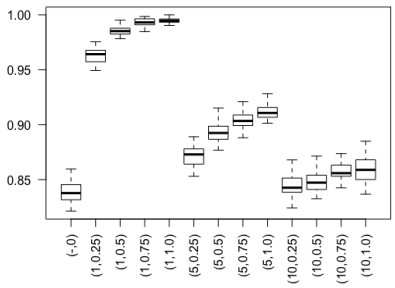
In the next experiment we compare the performance of the three multi-objective local search methods described in Section 3 when they are included in a MOMA scheme. We depict in Figure 5 the non-dominated front for the three local search procedures after 30 runs on each one of the studied instances. The represented Pareto fronts come from the union (maintaining only non-dominated solutions) of each independent execution



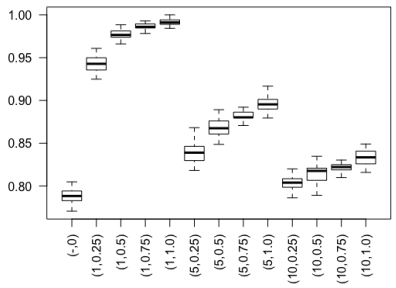
(a) pmed17.p25



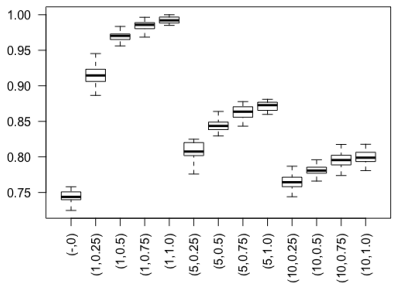
(b) pmed20.p50



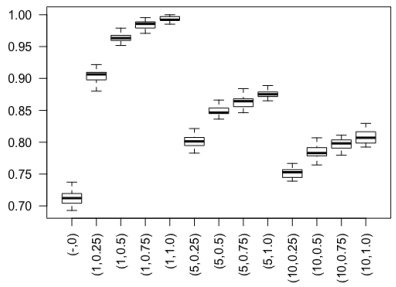
(c) pmed22.p62



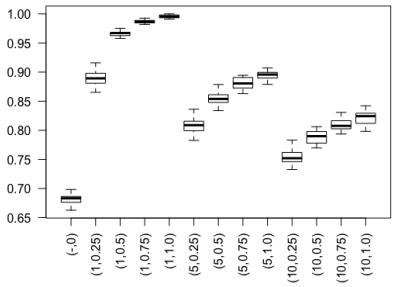
(d) pmed28.p75



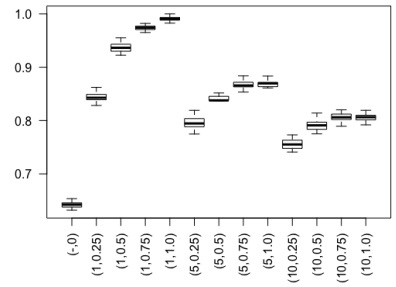
(e) pmed33.p87



(f) pmed36.p100



(g) pmed39.p112



(h) pmed40.p225

Figure 4: Hypervolume boxplots for the combination of τ (local search depth) and ν (local search frequency as a probability). X-axis shows the parameter values as (τ,ν).

of the MOMA. In a first sight, it can be seen that the MOMA with $DBLS$ and $AOLS$ obtain similar fronts in all instances. Additionally, these fronts widely cover the search space since the quantity and density of those efficient sets are large enough. Notice that we can not obtain relevant conclusions about the coverage, since none of these two fronts dominates the other one. On the other hand, MOMA with $WSLS$ presents a completely different behavior. In particular, it obtains a narrower front that seems to dominate the others in a small region close to the center of the front. This dominance can be seen more clearly in instances `pmed33.p87`, `pmed36.p100`, `pmed38.p112`, and `pmed40.p225`. It can be partially explained by the fact that this MOMA variant performs a larger number of evaluations, increasing the exploitation around a located search area. In addition, it could be also produced by the fact that $WSLS$ is directed toward the reference solution, which corresponds to the maximum objective values, as explained in Section 3.

We finish the analysis of the multi-objective local search methods by evaluating the average time of the executions by considering the same limit of function evaluations. Table 4 shows that MOMA with $DBLS$ is the fastest algorithm. Specifically, its computation time is almost 20% and 40% better than the variant with $AOLS$ and $WSLS$, respectively.

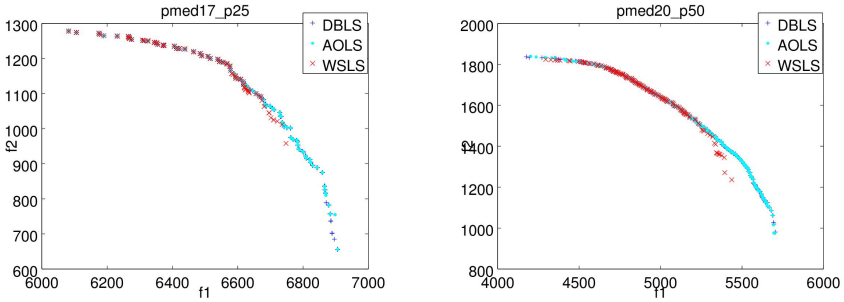
Instance	$DBLS$	$AOLS$	$WSLS$
pmed17.p25	108.13	211.82	218.80
pmed20.p50	106.10	181.04	364.16
pmed22.p62	177.33	316.21	535.63
pmed28.p75	362.02	620.92	497.59
pmed33.p87	563.38	767.19	1031.17
pmed36.p100	725.84	893.92	1676.28
pmed39.p112	1130.56	1206.04	1993.04
pmed40.p225	2152.30	2223.84	2464.50

Table 4: Time comparison.

From the results in this experiment, we cannot conclude which is the best option because each method provides alternative features that could be interesting in different decision-making scenarios. In particular, our experimentation showed that $AOLS$ obtains the best results in terms of hyper-volume. In addition, $DBLS$ obtains the best relation between quality (hyper-volume) and computing time. On the other hand, $WSLS$ is the slowest method and produces smaller hyper-volume values than the other two strategies. However, $WSLS$ produces high quality approximation to the Pareto front when we look for a compromise between both objectives. Therefore, we compare the three MOMA variants with the current state of the art in multi-objective optimization.

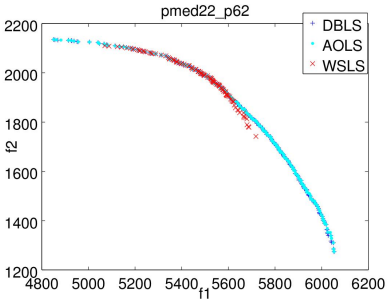
6.4 Comparison with classical MOEAs

Classical multi-objective algorithms usually perform well in the case of a small number of objectives, like in our case. Therefore, we compare the results of our memetic proposals with a baseline given by classical implementations. To this aim, we have selected NSGA-II (Deb et al., 2002) and SPEA2 (Zitzler et al., 2002) as the multi-objective optimization algorithms we compare with. According to a recent survey (Sayyad and Ammar, 2013), these two algorithms are the most used nowadays. In fact, it can be said that the current *de facto* standard evolutionary algorithm for multi-objective optimization is NSGA-II. This survey states that NSGA-II was used as a single algorithm in

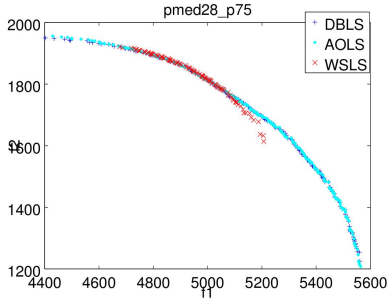


(a) pmed17.p25

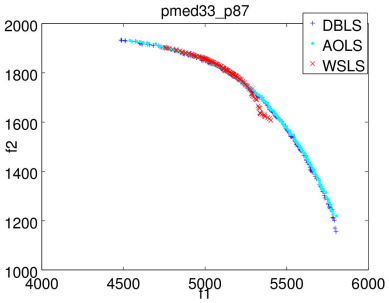
(b) pmed20.p50



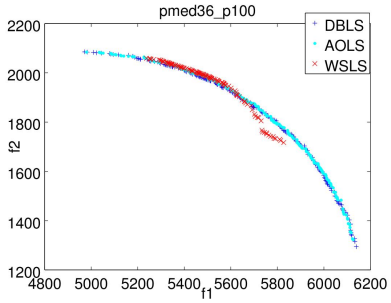
(c) pmed22.p62



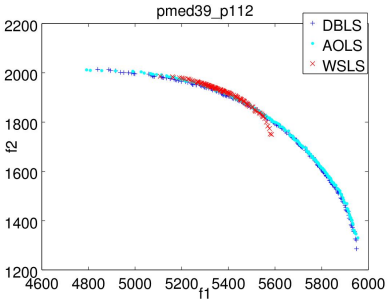
(d) pmed28.p75



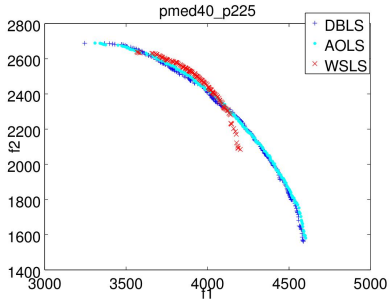
(e) pmed33.p87



(f) pmed36.p100



(g) pmed39.p112



(h) pmed40.p225

Figure 5: Non-dominanted fronts after 30 runs for the selected instances. $DBLS$, $AOLS$ and $WSLS$ are shown.

53% of the examined papers, positioning the algorithm as one of the most widely used MOEAs, and obtaining very competitive results, followed by SPEA2.

In order to find the most appropriate settings for the parameters and strategies for NSGA-II and SPEA2, we have used iRace (López-Ibáñez et al., 2011). This software implements an iterated race procedure able to find the configuration that obtains the best results. Its main objective is to automatically configure optimization algorithms by finding the most appropriate settings given a set of tuning instances of an optimization problem. In our case, we configure iRace to maximize the hypervolume.

This software package performs an off-line configuration in two phases. In the first one, it uses a set of tuning instances representative of a particular problem to set an algorithm configuration. In the second phase, the chosen algorithm configuration (i.e., parameter values and strategies) is used to solve unseen instances of the same problem. The goal is to find, during the tuning phase, an algorithm configuration that maximizes the hypervolumen over the set of instances that will be seen during the second phase. In other words, the ultimate purpose is that the high-quality configuration of the algorithm found during the tuning phase generalizes to similar but unseen instances.

Table 5 shows the configuration of iRace for both NSGA-II and SPEA2 algorithms, corresponding to the parameters of the procedures that were tunned, and to the range of values where iRace will perform the search. For instance, the value for the number of generations ranges from 100 to 5000. In the case of real values we stated a precision of two decimal digits. Besides, a number of 200 runs were executed for each algorithm using our set of 8 instances of the problem.

Parameter	Range	NSGA-II result	SPEA2 result
Generations	(100, 5000)	2966	4998
Population	(50, 250)	213	179
Mutation probability	(0.100, 0.999)	0.13	0.1
Crossover probability	(0.100, 0.999)	0.47	0.54
Estimated number of evaluations	-	631758	894642

Table 5: Parameter optimization for NSGA-II and SPEA2 with iRace.

Table 5 shows the resulting values for each parameter of NSGA-II and SPEA2. As seen, the results are similar for both of them in terms of probabilities of mutation, probability of crossover, and population size. However, the number of generations required by SPEA2 is 1.69 times larger than NSGA-II. In addition to these parameter values, we have included in the table the estimated number of evaluations considering that all the individuals are evaluated once on each generation. This value gives us a measure of the global exploration computational effort.

Once these parameter values were found, we run both NSGA-II and SPEA2 algorithms over the whole set of instances with the configuration provided by iRace. In order to have a fair comparison, we compare these two algorithms with out three MOMA proposals (see Table 2 for details) by considering an equivalent number of evaluations. In particular, we limited the number of evaluations to be 630000 in our procedures. This value is slightly lower than the estimated number of evaluations for NSGA-II, and considerable lower than the evaluations of SPEA2. Besides, we included another configuration for SPEA2 where we reduced the number of generations to 3520 (to have a SPEA2 version equivalent to NSGA-II in terms of number of evaluations). We have called SPEA2* to this limited version of SPEA2.

Figure 6 plots the non-dominated fronts after 30 executions for each algorithm. As it can be seen in the figure, and with the exception of the smallest instance, `pmed17.p25`, our MOMA approaches obtain fronts that clearly dominate the fronts obtained by NSGA-II, SPEA2 and SPEA2*. In fact, the larger the problem instance, the longer the distance between the fronts from the MOMA and the classical algorithms.

Again, $WSLS$ fronts are smaller, but they dominate the other fronts in the central region of the objective space in all the instances but `pmed40.p225`. Therefore, we can state the same conclusions about the resulting fronts compared with $DBLS$ and $AOLS$ as were discussed in Section 6.3.

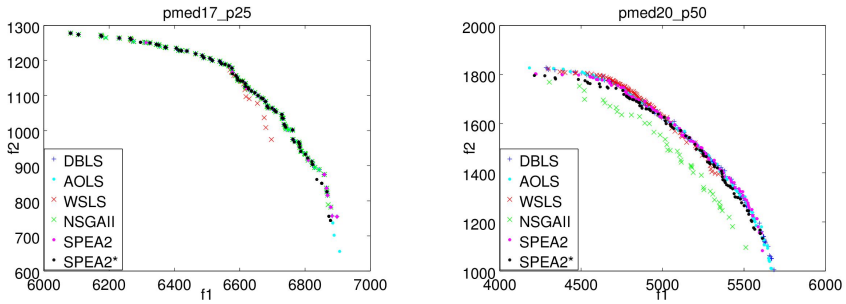
Table 6 shows a comparison of the hypervolume values for the six algorithms. In this case we do not provide normalized values (i.e., assigning 1.0 to the best method) to facilitate future comparisons. As it can be observed in this table, our methods present very competitive results in the set of tested instances. In fact, MOMA with $WSLS$, which obtains the worst values among our proposals, is better than NSGA-II in all instances but `pmed17.p25`. Similarly, it presents better results than SPEA2 in the largest instances (starting with `pmed28.p75`, which are 5 out of the 8 we are studying).

Instance	NSGA-II	SPEA2	SPEA2*	$DBLS$	$AOLS$	$WSLS$
pmed17.p25	8692436	8706070	8647530	8706765	**8710887**	8515768
pmed20.p50	9455493	9852272	9535370	**10042017**	10020130	9590186
pmed22.p62	10782181	11969641	11441200	**12565434**	12503057	11881041
pmed28.p75	8552938	9539804	9052460	10360761	**10383320**	9879425
pmed33.p87	8246628	9263670	8866950	10464880	**10496109**	9790010
pmed36.p100	9050694	10126465	9631480	**11962494**	11925733	11282702
pmed39.p112	7925756	9240195	8776820	**11301612**	11275309	10694895
pmed40.p225	7779073	8427239	8025960	**10830924**	10750521	9972719
# Best.	0	0	0	5	3	0
Avg. Dev.	0.1771	0.1029	0.1385	0.0007	0.0025	0.0534

Table 6: Hypervolume of final non-dominated fronts. Best values are depicted in bold font.

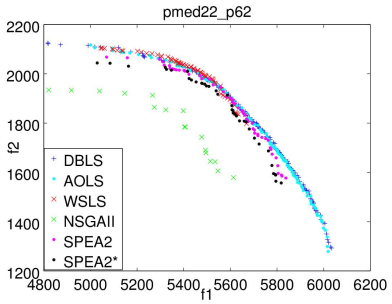
Overall, $DBLS$ and $AOLS$ obtains the best results, as indicated in the row #*Best* of Table 6. $DBLS$ obtains larger values than $AOLS$ in 5 out of the 8 instances. This result is important because it determines that $DBLS$ is a better choice than $AOLS$ under the same computational effort, limited by the maximum number of evaluations. Therefore, the prediction we stated in Section 6.3, about the better performance of $DBLS$ is now confirmed. Nevertheless, the average deviation of the hypervolume values, displayed in row *Avg. Dev.* of the table, is so small between them that we can state that $DBLS$ is better but $AOLS$ is really close to it.

Table 7 shows a comparison of the computation times for the six algorithms. Once again, the MOMA variants outperform their competitors in this regard. This result can be partially explained by the fact that NSGA-II, SPEA2, and SPEA2* executes more frequently time-consuming operations such as the computation of crowding distances or the management of elitism. As we can observe in this table, the comparison between $DBLS$ and $AOLS$ favors the first one in 5 out of the 8 instances. On the other hand, $DBLS$ obtains faster computation times in the smaller instances, whereas $AOLS$ works faster in the larger ones. Nonetheless, the three MOMA approaches perform in a similar way given the nature of their search procedures. This close computation time is

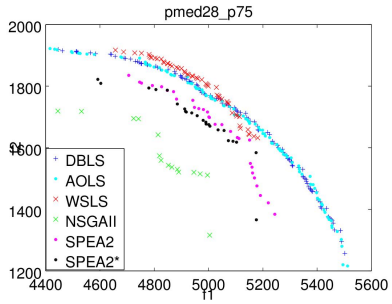


(a) pmed17.p25

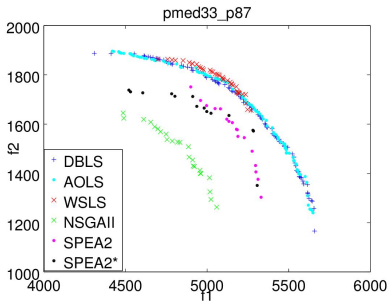
(b) pmed20.p50



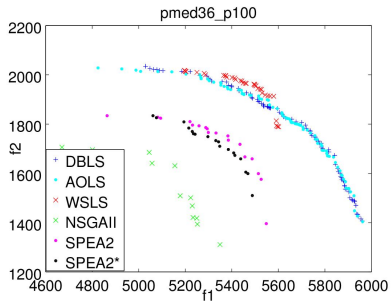
(c) pmed22.p62



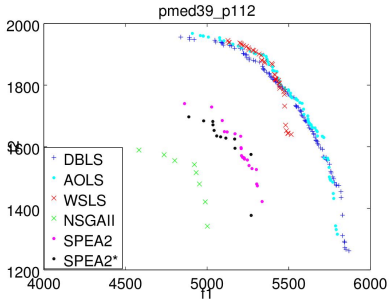
(d) pmed28.p75



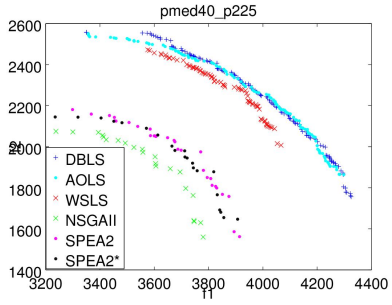
(e) pmed33.p87



(f) pmed36.p100



(g) pmed39.p112



(h) pmed40.p225

Figure 6: Non-dominanted on 30 runs. NSGA-II and SPEA2 using their iRace configurations, and the rest of them limited to aproximately the same number of evaluations.

verified by the similar value obtained in the average deviation in relation to the best computation time, as seen in the row labeled as *Avg. Dev.* in Table 7.

Instance	NSGA-II	SPEA2	SPEA2*	$DBLS$	$AOLS$	$WSLS$
pmed17.p25	58.64	239.25	122.67	**18.00**	21.95	20.14
pmed20.p50	166.51	327.2	162.24	**31.13**	40.49	33.76
pmed22.p62	266.36	507.93	210.78	**47.14**	58.94	53.65
pmed28.p75	449.72	567.24	281.33	**61.33**	78.86	64.84
pmed33.p87	654.64	821.49	367.1	**90.23**	112.05	102.23
pmed36.p100	917.75	1247.78	475.67	145.59	**123.05**	129.55
pmed39.p112	991.33	1071.55	602.49	203.32	**128.50**	204.61
pmed40.p225	3809.53	4259.45	1706.7	381.36	**282.53**	339.64
# Best.	0	0	0	5	3	0
Avg. Dev.	6.1876	9.8107	3.9686	0.1394	0.1622	0.1724
# Evals	631758	894642	633599	630000	630000	630000

Table 7: Average execution time in seconds. Best values are depicted in bold font.

7 Conclusions and Future Work

The objective of this study has been to advance the current state of knowledge about implementations of multi-objective memetic algorithms in the context of multi-objective combinatorial optimization. According to the number of publications, this field has not received as much attention as its continuous counterpart. In this paper we undertake to study the multi-objective optimization on the Obnoxious p-median problem, *Bi-OpM* , which optimizes two conflicting objectives. On the one hand, maximizing the sum of the distances to the nearest client to each obnoxious facility and, on the other hand, maximizing the dispersion of obnoxious facilities. To this aim, we propose a multi-objective memetic algorithm (MOMA) hybridized with three different improvement strategies. The first one, called dominance-based local search, $DBLS$, performs moves toward non-dominated solutions. The second one, $AOLS$, randomly selects one of the objectives to determine if the new solution improves the current one. Finally, the third approach combines the values of both objectives using a normalized weighted function, and we call it normalized weighted sum local search, $WSLS$. We additionally investigated the balance between global and local exploration. In particular, each multi-objective local search is configured with two parameters: frequency of applying the local search and depth of the corresponding local search. Finally, we introduced straightforward adaptations for crossover and mutation operators (originally designated for the OpM), that maintains the algorithm inside the space of feasible solutions, thus making the search more efficient.

In the extensive experimentation, we have studied a set of problem instances generated from the well known OR Library. Firstly, we have analyzed the balance between global and local search procedures by measuring the effect of their parameters over the performance of the search. Then, we have studied the influence of the three local search methods when considering non-dominated solutions, hypervolume and computation time. $DBLS$ and $AOLS$ obtained wider approximation of Pareto fronts, while $WSLS$ obtained the best compromise solutions.

Next, we compared the performance of the MOMA approaches with the classical multi-objective algorithms NSGA-II and SPEA2, whose parameters were tuned using

an iterated race procedure. All the classical and the MOMA algorithms shared the same basic configuration. Our experimentation disclosed that the MOMA approaches obtained better non-dominated solutions, better hypervolume values and spent less execution time, which averages an 80% of savings for the same number of evaluations.

As a conclusion, the MOMA approach seems to be a better heuristic for both quality of the solutions and execution time in this problem. The specific selection of the local search will finally depend on the decision maker. If a wide range of solutions is required, $DBLS$ and $AOLS$ are the best choices, because they provide a higher number of solutions, many of them close to higher values for the objective functions. On the contrary, if the aim is to find out solutions with a good compromise between both objectives, the $WSLS$ is the best choice.

As a future work, we will tackle related problems to take advantage of the performance of the memetic approach. In addition, we will study parallelization schemes for larger problem instances.

Acknowledgements

This research has been partially supported by the Ministerio de Economa y Competitividad of Spain (Grant Refs. TIN2015-65460-C2 and TIN2014-54806-R).

References

Alp, O., Erkut, E., and Drezner, Z. (2003). An efficient genetic algorithm for the p-median problem. *Annals of Operations Research*, 122(1-4):21–42.

Ardjmand, E., II, W. A. Y., Weckman, G. R., Bajgiran, O. S., Aminipour, B., and Park, N. (2016). Applying genetic algorithm to a new bi-objective stochastic model for transportation, location, and allocation of hazardous materials. *Expert Systems with Applications*, 51:49 – 58.

Balinski, M. (1965). Integer programming: Methods, uses, computation. *Mgt. Sci.*, 12:253–313.

Batta, R., Lejeuneb, M., and Prasad, S. (2014). Public facility location using dispersion, population, and equity criteria. *EJOR*, 234(3):819–829.

Beasley, J. E. (1990). OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072.

Belotti, P., Labbé, M., Maffioli, F., and Ndiaye, M. (2007). A branch-and-cut method for the obnoxious p-median problem. *4OR*, 5(4):299–314.

Cheng, H.-C., Chiang, T.-C., and Fu, L.-C. (2009). Multiobjective job shop scheduling using memetic algorithm and shifting bottleneck procedure. In *Computational Intelligence in Scheduling, 2009. CI-Sched '09. IEEE Symposium on*, pages 15–21.

Church, R. and Garfinkel, R. (1978). Locating an obnoxious facility on a network. *Trans. Sci.*, 12(2):107–118.

Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V. (2006). *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Colmenar, J., Greistorfer, P., Martí, R., and Duarte, A. (2016). Advanced greedy randomized adaptive search procedure for the obnoxious p-median problem. *European Journal of Operational Research*, pages –.

Coutinho-Rodrigues, J., Tralhão, L., and Alçada-Almeida, L. (2012). A bi-objective modeling approach applied to an urban semi-desirable facilitylocation problem. *EJOR*, 223(1):203213.

Current, J., Daskin, M., and Shilling, D. (2004). Discrete network location models. In Drezner, Z. and Hamacher, H. W., editors, *Facility Location: Applications and Theory*, Springer series in operations research, chapter 3, pages 83–120. Springer Science & Business Media.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Drezner, Z. and Hamacher, H. (2004). *Facility Location: Applications and Theory*. Springer.

Erkut, E. and Neuman, S. (1989). Analytical models for locating undesirable facilities. *EJOR*, 40(3):275–291.

Feo, T., Resende, M., and Smith, S. (1994). A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878.

Fernandes, S., Captivo, M., and Clmaco, J. (2014). A dss for bicriteria location problems. *Decision Support Systems*, 57(1):224–244. cited By 6.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP–Completeness*. W.H. Freeman and Co., New York.

Hakimi, S. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. *Ops Res.*, 12(3):450–459.

Hart, W. E. (1994). *Adaptive Global Optimization with Local Search*. PhD thesis, La Jolla, CA, USA. UMI Order No. GAX94-32928.

Igel, C. and Toussaint, M. (2005). A no-free-lunch theorem for non-uniform distributions of target functions. *Journal of Mathematical Modelling and Algorithms*, 3(4):313–322.

Ishibuchi, H. and Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 28(3):392–403.

Ishibuchi, H., Yoshida, T., and Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *Trans. Evol. Comp*, 7(2):204–223.

Krarup, J. and Pruzan, P. M. (1983). The simple plant location problem: Survey and synthesis. *European Journal of Operational Research*, 12(1):36 – 81.

Li, Y.-F., Pedroni, N., and Zio, E. (2013). A memetic evolutionary multi-objective optimization method for environmental power unit commitment. *Power Systems, IEEE Transactions on*, 28(3):2660–2669.

López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., and Birattari, M. (2011). The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium.

Melkote, S. and Daskin, M. S. (2001). An integrated model of facility location and transportation network design. *Transp. Res. Part A*, 35:515–538.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Comput. Oper. Res.*, 24(11):1097–1100.

Papadimitriou, C. and Yannakakis, M. (1991). Optimization, approximation and complexity classes. *J. of Computer and System Sciences*, 43:425–440.

Pareto, V. (1896). *Cours D´Economie Politique. Vols. I and II.* F. Rouge, Lausanne, Switzerland.

Sayyad, A. and Ammar, H. (2013). Pareto-optimal search-based software engineering (posbse): A literature survey. In *Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2013 2nd International Workshop on*, pages 21–27.

Segal, M. (2003). Placing an abnoxious facility in geometric networks. *Nordic J. of Computing*, 10(3):224–237.

Sudholt, D. (2012). *Handbook of Memetic Algorithms*, chapter Parametrization and Balancing Local and Global Search, pages 55–72. Springer Berlin Heidelberg, Berlin, Heidelberg.

Tralhão, L., Coutinho-Rodrigues, J., and Alçada-Almeida, L. (2010). A multiobjective modeling approach to locate multi-compartment containers for urban-sorted waste. *Waste Management*, 30(12):2418 – 2429.

Zitzler, E., Laumanns, M., and Thiele, L. (2002). SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In Giannakoglou, K. et al., editors, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pages 95–100. International Center for Numerical Methods in Engineering (CIMNE).

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.