Noname manuscript No.
(will be inserted by the editor)

Scatter Search for the Bi-criteria *p*-median *p*-dispersion problem

J. Manuel Colmenar · Arild Hoff · Rafael Martí · Abraham Duarte

Received: date / Accepted: date

Abstract The bi-criteria p-median p-dispersion is a challenging optimization problem that belongs to the family of location problems. Up to our knowledge, no metaheuristic has been proposed to this problem, where a multi-objective approach has to be considered. In this paper we propose a multi-objective Scatter Search implementation in which three different improvement methods have been analyzed. Our results have been compared with the state of the art obtaining better hypervolume values and promising results in terms of dominance of solutions.

Keywords Facility problems · p-median · p-dispersion · Scatter Search

1 Introduction

Facility location problems have gained much attention by the research community the last few years, and many variants of the problem have been described. The variants appear in different fields of real-life problems such as telecommunication, healthcare, transportation planning, districting among others [10]. This paper focuses on a location problem considering two objectives simultaneously. The problem is defined on a network $G = (V, E)$ where V is the

J. Manuel Colmenar, Abraham Duarte
Universidad Rey Juan Carlos,
Móstoles, 28933 (Madrid), Spain
E-mail: josemanuel.colmenar@urjc.es, abraham.duarte@urjc.es

Arild Hoff
Molde University College
2110, 6402 Molde, Norway
E-mail: arild.hoff@himolde.no

Rafael Martí
Universidad de Valencia,
Burjassot, 46100 (Valencia), Spain
E-mail: rafael.marti@uv.es

set of locations with cardinality $|V| = n$, and E is the set of edges, defined with a given distance d_{ij} between any pair of locations $i, j \in V$. The distances are assumed to be symmetric, i.e., $d_{ij} = d_{ji}$. The problem is to select a given number p out of the total n locations as hubs, and then the remaining $(n-p)$ locations, named as terminals, are each assigned to its closest hub. A hub is considered to be assigned to itself. Thus, any subset of p locations (hubs) defines a feasible solution to the problem.

The first objective is to minimize the sum of the distances between the terminals and the hubs. This is the well-known p-median problem, extensively studied in the scientific literature [2]. The second objective is described as the p-dispersion problem, and it is also called the max-min diversity problem [6]. In this problem, the p hubs should be chosen in a way that maximizes the minimum distance between the selected hubs.

This bi-objective problem is described as the bi-criteria p-median p-dispersion (*BpMD*) problem. It is introduced in [15] with a practical problem of locating traffic sensors in a highway network where both objectives need to be considered simultaneously. The authors describe the problem with the mathematical model below, where x_{ij} is a variable with the value 1 if terminal i is assigned to hub j, and 0 otherwise; and the variable y_i takes the value 1 if terminal i is chosen as a hub and 0 otherwise. The constant M is a large value, typically larger than the longest distance between any pair of locations.

$$\text{Minimize } z = \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} \qquad (p\text{-median})$$

$$\text{Maximize } w \qquad (p\text{-dispersion})$$

Subject to

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \qquad (1)$$

$$x_{ij} \leq y_j \qquad \forall i, j \in V \qquad (2)$$

$$\sum_{j \in V} y_j = p \qquad (3)$$

$$w \leq d_{ij} + M(2 - y_i - y_j) \qquad \forall i, j \in V \qquad (4)$$

$$x_{ij}, y_i \in \{0, 1\} \qquad \forall i, j \in V \qquad (5)$$

The p-median objective function models the minimization of the sum of the distances between hubs and their assigned terminals. Similarly, the p-dispersion objective function describes the maximization of the parameter w, defined in Constraint (4) to be the minimum distance between hubs. Constraint (1) states that all locations must be assigned to exactly one hub. Constraint (2) makes sure that locations are only assigned to hubs ($y_j = 1$), and Constraint (3) specifies that exactly p locations play the role of hubs. In Constraint (4) the value of w in the p-dispersion problem is defined to be smaller than or equal to any distance between two hub locations, while Constraint (5) states that x and y are binary variables.

The two objective functions might be conflicting, and there will rarely be one single solution classified as the best for both objectives. Thus, the solution to the bi-objective problem will consist of a set of solutions defined as the

Pareto front where none of the solutions is dominated by any of the others. In a multi-objective problem, a solution dominates another one if it is better in at least one objective and not worse in any of the others. An efficient solution is a solution not dominated by any other solution, while a weakly efficient solution will not have any other solution with a strictly better value for any objective functions.

For the sake of clarity we will name f_1 to the p-median objective function and f_2 to the p-dispersion objective function. With this mathematical notation, we can state the main contribution of this paper as proposing a method to obtain a set S of non-dominated solutions where

$$s_i \in S \text{ iff } \nexists s_j \,/\, f_1(s_j) < f_1(s_i) \wedge f_2(s_j) > f_2(s_i)$$

The rest of the paper is organized as follows. First, Section 2 presents a review of the literature about facility location problems. Section 3 describes the state of the art algorithm for this specific problem. Section 4 details the implementation of the Scatter Search algorithm that we propose to obtain an approximation to the Pareto front. Section 5 presents our experimental experience to evaluate the quality of this front, and Section 6 draws the conclusions and the future work.

2 Literature Review

Facility location problems are shortly described as the selection of a number of locations from a set of different alternatives. These problems are NP-hard, and only small instances are solvable to optimality within reasonable time. These problems come in many different variants due to practical examples from real life, typically aiming to choose the best locations according to some diversity or equity measure. In [10] an extensive overview of location problems is given, and in [4] mathematical models and heuristic algorithms for solving a collection of such problems are shown.

The current paper considers a bi-objective problem where each of the objectives are earlier studied individually in the research literature. The p-median problem is one of the most common problems in discrete location theory. The objective is to choose a given number of locations in a way that minimizes the total distance between the remaining locations to the chosen ones. The total distance can be weighted due to different demand at different locations, or as in our problem, the weights can be considered equal and omitted in the objective function. In [11], an extensive survey of metaheuristic approaches to the problem up to that date is given, and it concludes that the introduction of metaheuristics has led to substantial improvement in solution quality on large-scale instances within reasonable computing time. In [2] the most recent overview of the p-median problem is presented. The authors show both constructive and improvement heuristics in addition to the metaheuristics, and outline a Lagrangian relaxation, which can be embedded in a branch-and-bound algorithm to obtain optimal solutions to the problem. They also present

two bi-objective extensions to the problem. First, they examine the trade-off between the p-median and the maximum covering objective, and then explore the trade-off between the p-median and the p-center objective. For small instances of these combinations, they state that extensions of the Lagrangian algorithm could be used, while on larger instances they suggest to use genetic algorithms. The combination of the p-median and the p-dispersion objectives is not mentioned in their article.

The p-dispersion problem is also called the max-min diversity problem and aims to maximize the minimum distance between two hubs. The problem is closely related to the maximum diversity problem where the objective is to maximize the sum of distances among the hubs. These problems are also extensively studied earlier and solution methods vary between exact and heuristic methods. In [3], it is developed an exact algorithm for the p-dispersion problem based on branch-and-bound, while in [12] it is considered both the maximum diversity (named as the p-dispersion-sum) and the p-dispersion problem in the paper. The authors presented a technique to develop upper bounds through Lagrangian relaxation, semidefinite programming and reformulation, and derived a branch-and-bound algorithm computing an upper bound in each node. They also presented a solution algorithm based on transforming the p-dispersion to the maximum diversity problem and used the same technique for solving it. Several attempts have been tried to solve the p-dispersion problem by metaheuristics. For example, in [8] it is shown how to solve the problem with both Simulated Annealing and Tabu Search. The GRASP and path relinking method of [14] is so far the one producing best results for the problem. They used dynamic GRASP for creating a set of elite solutions, and then tested four different variants of path relinking for exploring the paths combining the solutions. The best results were obtained when using a Tabu Search method [13]. In this variant, they propose strong tabu rules of the drop strategy that ensure a high diversification among the solution space, which allows them to find better solutions.

3 State of the Art

Up to our knowledge, the only paper dealing with the p-median and the p-dispersion objective simultaneously is [15]. The authors first made use of the ε-constraint model [1] and solved the problem to optimality for different values of ε. In this method, they defined p-median as the primary objective function and treated the p-dispersion objective as a constraint specified with a value larger than the parameter ε. By running the algorithm iteratively with different values of ε, optimal solutions for the single objective problem were found and could be included in the set constituting the Pareto front of non-dominated solutions.

A modification of the ε-constraint model turned out to be more effective and required less computational resources. Similar to the original version, the algorithm would run iteratively, solving the p-median objective and treating

the p-dispersion objective as a constraint with different threshold values in each iteration. In this model, the original constraints (4) were replaced by a group of constraints (6) ensuring that no pair of nodes (i, j) are both selected as nodes if the distance d_{ij} is smaller than the threshold value ℓ.

$$y_i + y_j \leq 1 \qquad \forall i, j \in V \text{ such that } d_{ij} < \ell \qquad (6)$$

When carefully selecting the alternative values of ℓ, this algorithm would give us the full set of non-dominated solutions to the *BpMD*problem. Let, ℓ_{min} be the initial threshold value found by measuring the p-dispersion value after solving the p-median problem. Similarly, ℓ_{max} will be the objective value when solving the problem solely with the p-dispersion objective. The value of δ_{min} defines the increase of ℓ between two successive iterations. It is found by comparing the difference between all pair of distinct d_{ij} values and choosing the smallest one. To execute the algorithm, the values of ℓ_{min}, ℓ_{max} and δ_{min} need to be determined. The Incremental Algorithm proceeds as follows to determine them:

Initialization step: Find the values of ℓ_{min}, ℓ_{max} and δ_{min}. Store the solution for ℓ_{min} in the Pareto optimal set of solutions to the combined problem. Set the value $\beta_0 = \ell_{min}$.

Iterative step: Start the iteration process with counter $t = 1, 2, 3, \ldots$. Set the value of $\ell_t = \beta_{t-1} + \delta_{min}$, where β_{t-1} is the value of the minimum dispersion objective obtained in the previous iteration. Solve the problem with ℓ_t and store the solution in the Pareto optimal set of solutions to the combined problem.

Termination: Terminate the algorithm when the parameter ℓ_t reaches the maximum value of ℓ_{max}. Store the solution for ℓ_{max} in the Pareto optimal set of solutions to the combined problem.

In [15], it is proven that this algorithm is guaranteed to obtain all efficient solutions for the *BpMD*problem. The algorithm terminates after at most $n(n-1)/2$ iterations since this number is the total number of node pairs in the problem, and in the worst case, only one pair will be excluded in every iteration. However, experiments show that the actual number of iterations usually will be significantly smaller. Still, the execution time might be too large with limited computing resources, and in [15] it is suggested a way to reduce the computational time by using a δ-value larger than δ_{min}. In this case, there is no longer a guarantee for the algorithm to obtain all efficient solutions, but it will only produce a subset of them. The strategy recommended to use with a limited execution time, is to calculate the expected execution time υ for one iteration using a fixed value for ℓ. Then one can easily find the expected number of iterations to perform within the given time frame, and the δ-value can be set to $\delta' = \frac{\ell_{max} - \ell_{min}}{\upsilon}$.

To achieve the value of ℓ_{max}, the single p-dispersion problem needs to be solved for the instance data. In [15] it is suggested a strategy where the maximum number of possible hubs are calculated for all alternative values of ℓ. Then the integer programming model would look like this:

$\upsilon(\ell) = \text{Maximize} \sum_{i \in V} y_i$

Subject to

$y_i + y_j \leq 1$	$\forall i, j \in V$ such that $d_{ij} < \ell$
$y_i \in \{0, 1\}$	$\forall i \in V$

By solving this model for increasing values of ℓ, the objective value will gradually be reduced until it is smaller than the p locations that should be chosen. Then ℓ_{max} could be identified by the value of ℓ_1 where $\upsilon(\ell_1) \geq p$ and $\upsilon(\ell_2) \leq p$ where $\ell_1 < \ell_2$. In [15] it is suggested to divide the interval between the smallest and the largest d_{ij} into 2^q smaller intervals, and carry out a binary search among those to identify the interval containing the value of ℓ_{max}.

Obviously, the method described above for solving the *BpMD* problem depends on a problem instance small enough to be solvable to optimality for the p-median and the p-dispersion problem separately within reasonable time. Experiments reported in [15] show how an instance with $n = 50$ and $p = 5$ is solved within 14 seconds and 18 efficient solutions are identified. By using an adjusted δ' value the CPU time was reduced to 5 seconds and the procedure found 6 efficient solutions. Further experiments by larger instances defining a maximum of 10 iterations showed that an instance with $n = 350$ and $p = 35$ used on average 4.5 hours and, in the worst case, almost 10 hours to complete.

In [15] it is also suggested a Lagrangian heuristic to deal with larger instances. This heuristic relaxes constraint set (1) in the original model, defining that all locations need to be assigned to exactly one hub. The objective function would then look like (7) with constraints (2), (3), (5) and (6) remaining.

$$\text{Minimize } L(\lambda) = \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} + \sum_{i \in V} \lambda_i (1 - \sum_{j \in V} x_{ij}) \qquad (7)$$

The λs are initially given the value equal to the largest d_{ij} value for location i and the upper and lower bounds are initially given values as infinity and minus infinity. Then the iterative step consist of three operations:

1. Solve the integer programming problem defined in (7). The value of $L(\lambda)$ will constitute a lower bound for the solution of the problem with the value of ℓ given in constraints (6).

2. An upper bound is calculated by assigning each location to its closest hub and calculating the objective value.

3. The λs for iteration $(k+1)$ are updated according to the formula $\lambda_i^{(k+1)} = \lambda_i^{(k)} + t^k + g_i^k$. Here $g_i^k = 1 - \sum_j x_{ij}$ and $t^k = \pi(1.05UB - LB)/\sum_{i \in N} g_i * k$ and π is a scaling parameter.

The algorithm is terminated when the relative difference between the upper and the lower bounds is smaller than a predefined value, or if the number of iterations reaches a pre-specified maximum value.

The Lagrangian heuristic in the Incremental Algorithm is shown to solve instances up to $n = 650$ and $p = 65$ with an average execution time of 12 minutes and a maximum of 71 minutes on the largest instances.

4 Scatter Search Proposal

Scatter Search (SS) is an evolutionary algorithm that explores the solution space by evolving a population of solutions called reference set [9]. It was introduced in [7] as a heuristic for integer programming. The basic design to implement SS is based on the "five-method template".

- A *Diversification Generation Method* to generate a collection of diverse trial solutions within the search space.
- An *Improvement Method* to transform a trial solution into one or more enhanced trial solutions.
- A *Reference Set Update Method* to build and maintain a reference set consisting of the b "best" solutions found (where the value of b is typically small, e.g. no more than 20). Solutions gain membership to the reference set according to their quality or their diversity.
- A *Subset Generation Method* to operate on the reference set, to produce several subsets of its solutions as a basis for creating combined solutions.
- A *Solution Combination Method* to transform a given subset of solutions produced by the *Subset Generation Method* into one or more combined solution vectors.

Several implementations have been proposed in the literature for these methods, and some of them are considered straightforward or standard proposals. The overall SS method operates over the reference set, *RefSet*, which basically can be considered as a collection of both high quality solutions and diverse solutions that are used to generate new solutions by applying the combination method. Taking into account the literature and the features of the problem under study, we have considered the implementations of those methods that we next describe. Of the five methods in SS methodology, only four are strictly required. The improvement method is usually needed if high-quality outcomes are desired, but an SS procedure can be implemented without it. On the other hand, hybrid SS designs could incorporate a short-term Tabu Search or another complex metaheuristic such as the improvement method (usually demanding more running time).

After several attempts of using a greedy strategy to create solutions for the initial population, we found in our preliminary experiments that a method which randomly selects p nodes to construct one solution is as good as the greedy alternatives that we were exploring. Moreover, the advantage of the random method is the diversity of the generated solutions, which provides very different starting points for the improvement method. Hence, although it is unusual in the SS methodology, we decided to use a random method to generate the initial population.

According to the literature, in the standard design, the *RefSet* is usually updated taking into account only the quality of the solutions. Advanced designs [5], however, update the *RefSet* taking into account both the quality and the diversity of the solutions in the population. These strategies are very clear in the case of one objective function, because it determines the quality of a

solution. However, we deal with a bi-objective problem where the concept of quality depends on two objectives. In this context, we defined the quality of a solution as the distance to an ideal solution that is defined by the best known values for the objectives. Notice that this ideal solution is not a real solution because its values come from the two extremes of the Pareto front. Expression (1) shows the calculation of the Euclidean distance between a given solution s and the reference solution s_{ref}. In our approach, therefore, the lower the distance, the better the quality. A solution enters into the *RefSet* when its distance to the ideal solutions is lower than the distance of the worst solution in the *RefSet* to this ideal solution. This is an adaptation of the standard implementation of the *RefSet Update Method* to multi-objective optimization. Note that the solutions in the *RefSet* are ordered with respect to their quality, evaluated according to this distance value.

$$dist(s) = \sqrt{(f_1(s) - f_1(s_{ref}))^2 + (f_2(s) - f_2(s_{ref}))^2} \quad (1)$$

As described above, the SS methodology basically consists of five elements (and their associated strategies). Three of them, the *Diversification Generation*, the *Improvement* and the *Combination Methods*, are problem dependent and should be designed specifically for the problem at hand. We implemented the standard *Subset Generation Method* where all the possible pairs of solutions in the *RefSet* are considered. Then, the pairs of solutions are combined in order to obtain new solutions to be improved. In our proposal, we have defined a combination method based on the distance among the locations of the solutions to be combined, trying to maximize the minimum distance among the selected locations. This idea bias the resulting solution towards the p-dispersion objective, which we have found to be more difficult to improve, according to the results of our preliminary experiments. The different difficulty in the objectives could be caused by a sparsely populated space of solutions in the p-dispersion objective in relation to the space of solutions of the p-median objective. However, we have not performed any experiment devoted to prove this idea, and we will consider it for future work. Algorithm 1 describes the combination method. Steps 1 to 3 initialize the combined solution s_c and store the locations of the solutions s_1 and s_2 into the set of locations L. Then, solution s_c is constructed iterating up to its size reaches p. On each iteration of the loop, $j^\star$ is selected as the location whose minimum distance to the rest of locations in L is the maximum one, as shown in Step 5. Finally, Steps 6 and 7 update both the combined solution and the remaining locations.

It is well-known in heuristic optimization that the local search method is a key component of any method that seeks to find the global optimum of a difficult problem. We therefore study the adaptation of three alternatives. They correspond to three local search methods that follow the first improvement strategy where a move randomly exchange a location from the current solution with a non-selected location.

The difference among the three implementations is the way they determine that a new solution improves the current one. Basically, we have followed

Algorithm 1 Combination between solutions s_1 and s_2.

1: $s_c \leftarrow \emptyset$
2: $L \leftarrow s_1$
3: $L \leftarrow L \cup s_2$
4: **while** $|s_c| < p$ **do**
5: $\quad j^\star = \arg\max_{j \in L} \arg\min_{i \in L \wedge i \neq j} \{d_{ij}\}$
6: $\quad s_c \leftarrow s_c \cup \{j^\star\}$
7: $\quad L \leftarrow L \setminus \{j^\star\}$
8: **end while**
9: **return** s_c

three different ideas that perform the exploitation of the search space very differently.

The first method is a local search based on dominance, LS_{Dom}. This method considers that a new solution is *better* if it dominates the current one. That is, when both objective values are better than those of the current solution. It is important to notice that if the method finds a non-dominated solution, which is a solution with one objective value that is worse than the current one, but the other one being better or equal, the non-dominated solution is stored in an external archive.

Notice that a local search over one objective will return a single solution. However, in this bi-objective problem, it is unlikely that one single solution will dominate all the others. In fact, the local search travels along different non-dominated solutions. Therefore, we use an external archive to store the non-dominated solutions are visited by the local search and, at the end of the execution, the algorithm will return the non-dominated solutions from the archive.

The second method is a local search that alternates the objective that is evaluated, LS_{Alt}. In this case, the method compares the value of one of the objectives in order to decide if the new solution is better. However, the objective taken into account is changed each time a better solution is found. As in the case of LS_{Dom}, if a new solution is worse but is non-dominated (the value for the other objective is better), it is stored in an external archive.

The third method is a local search that tries to minimize the normalized distance to an ideal point, LS_{Dist}. This approach follows the strategy of the quality we have defined in Equation (1). A new solution is better if the distance to the reference solution is smaller than the distance from the current solution. As in the update of the reference set, this local search method calculates the distance with Equation 1. Besides, as in the other local search methods, if a new solution is considered worse but is a non-dominated solution, it is stored in the external archive.

Finally, we have also included a loop in the algorithm that performs a restart of the algorithm if the time limit is not reached, but keeping the same archive. To this aim, we have included the condition of a time limit in all the methods of the SS. Therefore, any method is interrupted if the time limit is reached. The resulting SS implementation is described in Algorithm 2.

Algorithm 2 Scatter Search algorithm.

```
1: archive ← ∅
2: while true do
3:    Construction of the initial population P.
4:    Application of the improvement method (local search) to all solutions in P.
5:    Construction of the RefSet from the solutions in P considering the normalized distance to the reference solution.
6:    while New solution enters the RefSet do
7:       Generation of subsets from the RefSet creating all the possible pairs of solutions.
8:       Application of the combination method to all the generated subsets considering the maximization of the minimum distance among locations of the combined solutions.
9:       Application of the improvement method (local search) to all solutions generated by the combination method.
10:      Update of the RefSet using improved solutions considering the normalized distance to the reference solution.
11:   end while
12: end while
13: return RefSet ∪ archive
```

In the next section we analyze the performance of our SS proposal under the three different alternatives we have described for the improvement method.

5 Experimental experience

We have performed our experiments over a set of medium-sized instances kindly provided by the authors of the best previous method [15]. More precisely, we worked with a set of 9 instances with $n = 250$ and $p = 25$, named as *D_250*, and a set of 10 instances with $n = 350$ and $p = 35$, named *D_350*. The authors of [15] also gave us their solution (approximation of the efficient frontiers).

The aim of these experiments is to show the potential benefits of the SS approach versus the Incremental Algorithm from [15], also described in Section 3. In addition, we determine which metric will provide a better comparison between the different SS improvement methods proposed.

We have run our SS algorithm 10 times on each one of the instances using the three improvement methods described in Section 4. Therefore, 30 runs of the SS have been performed for each instance. After the preliminary experiments, the initial population size and *RefSet* size parameters were set to $P = 10$ and $b = 4$ respectively.

Given that we want to compare the performance of the Scatter Search with the Incremental Algorithm, we limit the execution of the Scatter Search to the lowest execution time given in [15] for the target instances. In the case of *D_250*, this corresponds to 75 seconds, and for *D_350* the limit is 165 seconds. In addition, for each one of the instances, we took as ideal solution s_{ref} the one with objective values equal to the best values of the objective functions given by the frontiers provided in [15]. Notice that this is not a solution of the instance, but a point in the objective values space.

Once the runs were finished, we examined the hypervolume values of the non-dominated solutions coming from each run, which form the approximation to the Pareto front. The hypervolume [16] measures the area dominated by the solutions in the given frontier taking into account a reference point. In our case, given that the p-median is a minimizing objective and the p-dispersion is a maximizing objective, the fronts are oriented as shown in Figure 1, pointing to the top left of the plot. However, we decided to use the origin as the reference point to calculate the hypervolume. Hence, we have transposed the median objective to the difference between a high upper bound and the objective value. Figure 2 shows the transposed front. We made it to consider the usual case of maximization problems where the higher the hypervolume the better the frontier. Notice that, in terms of dominance, there is no modification in relation to the relative position of each solution and its dominated area. The unique change is the orientation of those dominated areas.

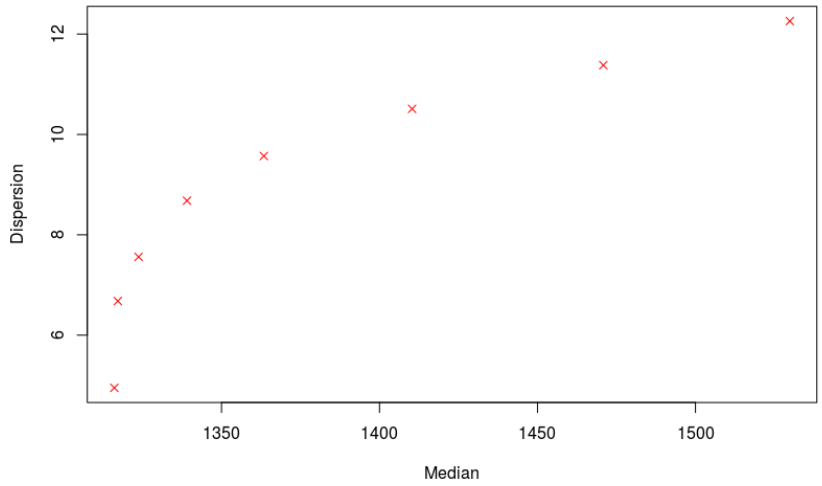

Fig. 1 Front obtained with the Incremental Algorithm for D_250_1 instance.

Table 1 shows the values of the average hypervolume values for the instances under study. We present the results for the Incremental Algorithm, as well as the values for our three local search methods in the SS algorithm. It is clear that the best hypervolume results are given by the LS_{Dist} method in all the instances but two of the *D_250* instance. Moreover, it can be seen that all the SS approaches obtain better hypervolume values than the reference case. These results occur because, running the same execution time, the SS approaches find more non-dominated solutions than the Incremental Algo-

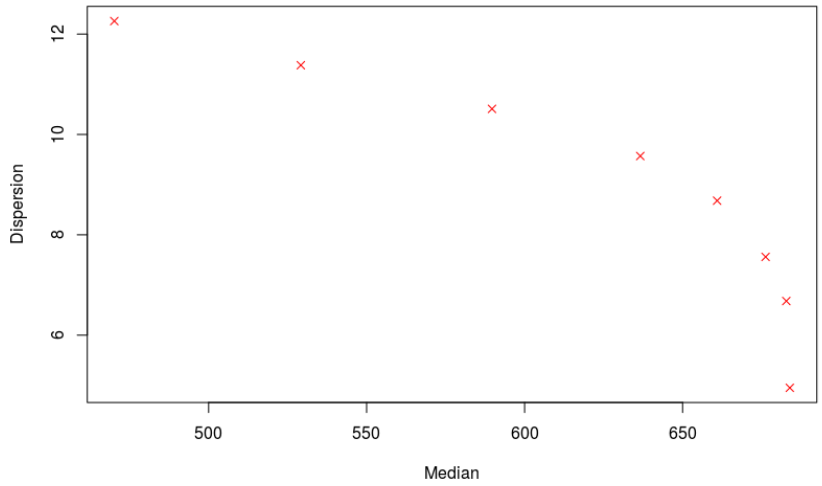

Fig. 2 Transposed front obtained with the Incremental Algorithm for D_250_1 instance.

rithm. Therefore, the fronts are both wider and more crowded, which results in higher hypervolume values.

Instance	Increm. Alg.	LS_{Dom}	LS_{Alt}	LS_{Dist}
D_250_1	789.71	5279.24	6511.86	**6593.79**
D_250_2	820.32	5368.33	**6650.29**	6473.96
D_250_3	822.88	5391.89	6463.97	**6661.65**
D_250_5	771.02	5222.3	6157.56	**6414.26**
D_250_6	827.17	5505.66	6573.13	**6604.87**
D_250_7	783.28	5174.31	6235.3	**6349.73**
D_250_8	813.23	5404.21	6241.61	**6578.55**
D_250_9	798.65	5373.02	**6576.82**	6565.28
D_250_10	791.04	5099.36	6175.9	**6452.44**
D_350_1	255.41	844.7	1723.72	**2205.68**
D_350_2	267.98	1064.89	1876.36	**2249.04**
D_350_3	225.6	637.21	1435.45	**1943.7**
D_350_4	232.61	816.57	1364.08	**2017.48**
D_350_5	234.63	605.37	1304.74	**1995.29**
D_350_6	279.64	914.61	1477.92	**2289.94**
D_350_7	337.45	1501.21	2167.32	**2829.54**
D_350_8	259.39	882.85	1610.3	**2181.73**
D_350_9	305.41	1251.2	2037.91	**2525.41**
D_350_10	272.53	1420.71	2039.94	**2478.42**

Table 1 Average hypervolume values of the instances under study. The higher the better.

However, despite that the Incremental Algorithm fronts have less number of solutions and lower hypervolume values, the SS frontiers are not better than

the Incremental Algorithm ones in terms of dominance. In fact, we found that none of the SS frontiers completely dominates the Incremental Algorithm front in any instance.

Besides, many of the SS fronts with higher hypervolume values are not good in terms of dominance. To illustrate these issues, Figure 3 shows three fronts for the *D_250_1* instance: the reference one, given by the Incremental Algorithm and labeled as "Ref."; the front with the best hypervolume value, labelled as "SS best Hv"; and a third front, labeled as "SS best Non-Dom." which is the front with the best ratio of non-dominated solutions in relation to the reference. This ratio is the number of solutions non-dominated by the reference divided by the total number of solutions in the front. As seen in this figure, 6 out of the 11 solutions (54.54% of the solutions) are not under the lines that determine the dominated area of the reference front. However, this best frontier in terms of dominance ratio does not present a high hypervolume value because the front is not wide enough compared to other frontiers. Therefore, we need another metric to better evaluate the results of these experiments.

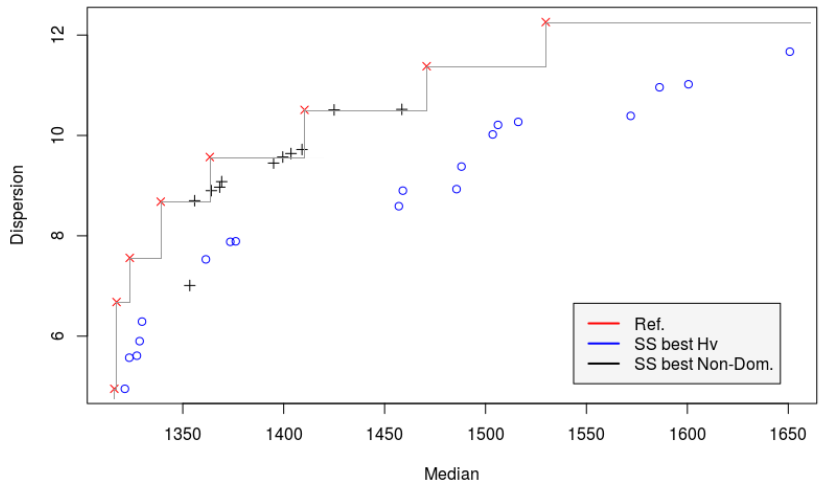

Fig. 3 Comparison of frontiers for *D_250_1* instance.

Hence, we have also calculated the ratio of non-dominated solutions for the runs of the instances under study considering the reference fronts. These ratios are displayed in Table 2, showing the average value for the runs over each instance. Again, we see that LS_{Dom} presents the worst performance, with no solutions at all being non-dominated. In addition, LS_{Norm} obtains better average values in 12 out of the 19 instances.

Instance	LS_{Dom}	LS_{Alt}	LS_{Dist}
D_250_1	0	0.011	**0.073**
D_250_2	0	**0.066**	0.052
D_250_3	0	**0.094**	0.088
D_250_5	0	0.068	**0.088**
D_250_6	0	0.015	**0.042**
D_250_7	0	0.058	**0.069**
D_250_8	0	0.015	**0.056**
D_250_9	0	0	**0.009**
D_250_10	0	**0.104**	0.1
D_350_1	0	0.033	**0.181**
D_350_2	0	**0.109**	0.054
D_350_3	0	0.042	**0.09**
D_350_4	0	0.009	**0.076**
D_350_5	0	**0.124**	0.107
D_350_6	0	0.007	**0.105**
D_350_7	0	0.096	**0.222**
D_350_8	0	**0.062**	0.053
D_350_9	0	**0.071**	0.032
D_350_10	0	0.219	**0.609**

Table 2 Average ratio of non-dominated solutions in relation to the Incremental Algorithm frontier. The higher the better.

To finally illustrate the behavior of the SS approaches, we have analyzed the plots of the solutions obtained by the different algorithms. For the sake of space we will not show all the plots, but a couple of representative cases from two of the instances under study.

Figure 4 shows the solutions obtained for the *D_250_3* instance. Here we can see that some solutions obtained with SS running LS_{Alt} and LS_{Dist} are not dominated by the reference front, mainly in the lower part of the front. It is clear that all the solutions from LS_{Dom} are dominated. This is an interesting behavior taking into account that LS_{Dom} looks for non-dominated solutions. However, we believe that this improvement condition is too hard, and the local search gets stuck in the search process, obtaining low quality solutions. On the other hand, the progression of the other local search approaches is longer because it is easier to fulfill their improvement conditions.

Figure 5 plots the solutions obtained for the *D_350_10* instance. Despite the fact that the behavior of the SS implementations is similar than in the previous case, we would like to notice that both LS_{Alt} and LS_{Dist} SS approaches obtain solutions that dominate the reference case in the bottom-left region of the plot. This result proves that the SS is a good alternative to the Incremental Algorithm, and seems to have potential to beat the state of the art in larger instances.

Therefore, we have confirmed that the SS approach obtains a higher number of non-dominated solutions than the Incremental Algorithm running the same execution time. Besides, for some of the instances, the SS approaches have obtained better solutions in terms of dominance than the state of the art algorithm. Hence, this result motivates the research on more efficient methods

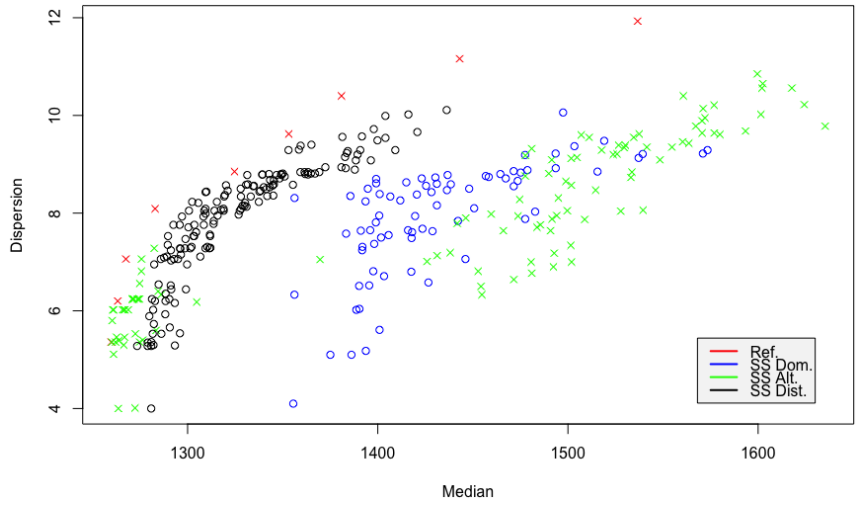

Fig. 4 Solutions from Incremental Algorithm (Ref.) and SS algorithms for *D_250_3* instance.

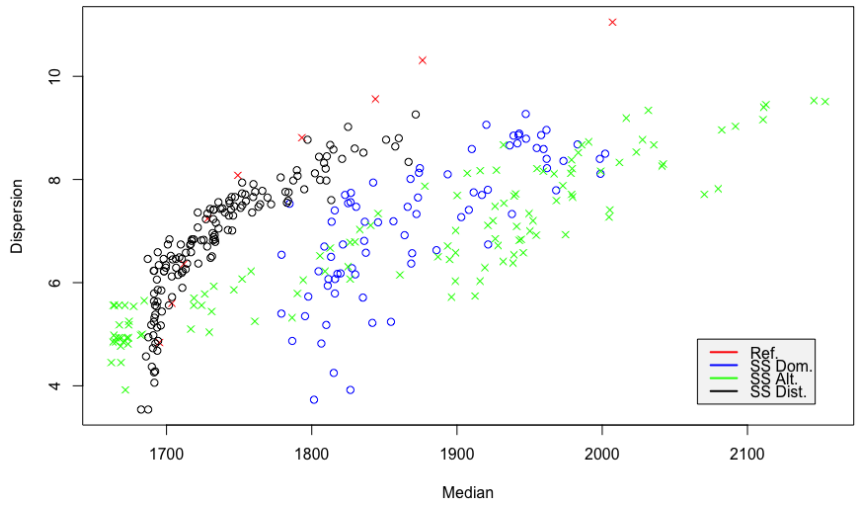

Fig. 5 Solutions from Incremental Algorithm (Ref.) and SS algorithms for *D_350_10* instance.

with the aim of obtaining solutions that dominate the Incremental Algorithm frontiers.

6 Conclusions and Future Work

In this paper we have presented a SS approach to tackle the *BpMD* problem. An initial SS implementation has been described where we apply a random

constructive method for the generation of the initial population and a combination method that tries to maximize the minimum distance among the locations of the solutions to combine. We have implemented three different improvement methods based on a first improvement local search strategy.

Our SS proposal with the three improvement methods have been tested against the state of the art algorithm for a set of medium-sized instances. We have found that, running the same execution time, our SS implementations obtain a higher number of solutions in wider Pareto fronts, which is translated into higher hypervolume values. Besides, we have analyzed the behavior of the local search alternatives, and we have concluded that, despite the solutions obtained by the SS do not completely dominate the reference results, they are able to improve a significant number of reference solutions in many of the instances. Therefore, further study on more efficient methods for the SS implementation can be the basis of more advanced search methods to obtain improved outcomes. In addition, we will study different variants of the distance measure, considering normalization of values, and different weights for each objective function.

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). The authors want to thank Emilio Alarcón for his seminal work on this problem, which triggered their interest.

References

1. Brub, J.F., Gendreau, M., Potvin, J.Y.: An exact -constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. European Journal of Operational Research **194**(1), 39 – 50 (2009). DOI http://dx.doi.org/10.1016/j.ejor.2007.12.014. URL http://www.sciencedirect.com/science/article/pii/S0377221707012106
2. Daskin, M.S., Maass, K.L.: The p-Median Problem, pp. 21–45. Springer International Publishing, Cham (2015). DOI 10.1007/978-3-319-13111-5_2. URL http://dx.doi.org/10.1007/978-3-319-13111-5_2
3. Erkut, E.: The discrete p-dispersion problem. European Journal of Operational Research **46**(1), 48 – 60 (1990). DOI http://dx.doi.org/10.1016/0377-2217(90)90297-O. URL http://www.sciencedirect.com/science/article/pii/037722179090297O
4. F. Sandoya R. Aceves, A.M.G.A.D.R.M.: Diversity and equity models. Springer (2017 (In press.))
5. Gallego, M., Duarte, A., Laguna, M., Martí, R.: Hybrid heuristics for the maximum diversity problem. Computational Optimization and Applications **44**(3), 411 (2007). DOI 10.1007/s10589-007-9161-6. URL http://dx.doi.org/10.1007/s10589-007-9161-6
6. Ghosh, J.B.: Computational aspects of the maximum diversity problem. Operations Research Letters **19**(4), 175 – 181 (1996). DOI http://dx.doi.org/10.1016/0167-6377(96)00025-9. URL http://www.sciencedirect.com/science/article/pii/0167637796000259
7. Glover, F.: Heuristics for Integer Programming Using Surrogate Constraints. Report. Business Research Division, University of Colorado (1974)
8. Kincaid, R.K.: Good solutions to discrete noxious location problems via metaheuristics. Annals of Operations Research **40**(1), 265–281 (1992). DOI 10.1007/BF02060482. URL http://dx.doi.org/10.1007/BF02060482
9. Laguna, M., Marti, R.: Scatter search: methodology and implementations in C (2012)

10. Laporte G., S.N., da Gama, F.S.: Location Science. Springer (2015)
11. Mladenovi, N., Brimberg, J., Hansen, P., Moreno-Prez, J.A.: The p-median problem: A survey of metaheuristic approaches. European Journal of Operational Research **179**(3), 927 – 939 (2007). DOI http://dx.doi.org/10.1016/j.ejor.2005.05.034. URL http://www.sciencedirect.com/science/article/pii/S0377221706000750
12. Pisinger, D.: Upper bounds and exact algorithms for -dispersion problems. Computers & Operations Research **33**(5), 1380 – 1398 (2006). DOI http://dx.doi.org/10.1016/j.cor.2004.09.033. URL http://www.sciencedirect.com/science/article/pii/S0305054804002552
13. Porumbel, D.C., Hao, J.K., Glover, F.: A simple and effective algorithm for the maxmin diversity problem. Annals of Operations Research **186**(1), 275 (2011)
14. Resende, M., Mart, R., Gallego, M., Duarte, A.: {GRASP} and path relinking for the maxmin diversity problem. Computers & Operations Research **37**(3), 498 – 508 (2010). DOI http://dx.doi.org/10.1016/j.cor.2008.05.011. URL http://www.sciencedirect.com/science/article/pii/S0305054808001032. Hybrid Metaheuristics
15. Sayyady, F., Tutunchi, G.K., Fathi, Y.: p-median and p-dispersion problems: A bi-criteria analysis. Computers & Operations Research **61**, 46 – 55 (2015). DOI http://dx.doi.org/10.1016/j.cor.2015.02.007. URL http://www.sciencedirect.com/science/article/pii/S0305054815000386
16. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - a comparative case study. In: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, PPSN V, pp. 292–304. Springer-Verlag, London, UK, UK (1998). URL http://dl.acm.org/citation.cfm?id=645824.668610