How to Locate Disperse Obnoxious Facility Centers?

Jesús Sánchez-Oro[1], J. Manuel Colmenar[1](✉), Enrique García-Galán[1], and Ana D. López-Sánchez[2]

[1] Rey Juan Carlos University, C/ Tulipán s/n, 28933 Móstoles, Madrid, Spain
{jesus.sanchezoro,josemanuel.colmenar,enrique.garciag}@urjc.es
[2] Pablo de Olavide University, Ctra. Utrera Km 1, 41013 Sevilla, Spain
adlopsan@upo.es

Abstract. The bi-objective obnoxious p-median problem has not been extensively studied in the literature yet, even having an enormous real interest. The problem seeks to locate p facilities but maximizing two different objectives that are usually in conflict: the sum of the minimum distance between each customer and their nearest facility center, and the dispersion among facilities, i.e., the sum of the minimum distance from each facility to the rest of the selected facilities. This problem arises when the interest is focused on locating obnoxious facilities such as waste or hazardous material, nuclear power or chemical plants, noisy or polluting services like airports. To address the bi-objective obnoxious p-median problem we propose a variable neighborhood search approach. Computational experiments show promising results. Specifically, the proposed algorithm obtains high-quality efficient solutions compared to the state-of-art efficient solutions.

Keywords: Location problem · Obnoxious p-median problem · Multi-objective optimization · Variable neighborhood search

1 Introduction

The importance of locating centers no matter the nature of them is crucial to manage any company either private or public. In our field, a *location problem* can be defined as an optimization problem that seeks to place one or more centers or facilities having into account a given set of customers or demand points [9].

According to [5], location problems can be classified into four categories regarding the objective function criteria: *facility location problems*, which seek to find a place to locate a facility in order to minimize the total cost between demand points and facilities; *p-median problems*, which determine the locations of p facilities in order to minimize the total cost between demand points and facilities; *p-center problems*, which minimize the maximum distance between each demand point and its assigned facility; and *covering problems* whose objective is to find the minimum number of facilities to cover all the demand points

A. Sifaleras et al. (Eds.): ICVNS 2018, LNCS 11328, pp. 52–63, 2019.
https://doi.org/10.1007/978-3-030-15843-9_5

or to maximize the number of demand points covered by a given number of facilities. All those problems can be considered with or without a demand value in facilities and/or in demand points. In those cases they are known as *capacitated* or *uncapacitated* problems, respectively [4,15]. Furthermore, location problems can be considered on the *discrete* space, when facilities can be only placed at specific locations [12], or *continuous* space, in which facilities can be placed at any location of a given region [1]. This work deals with an uncapacitated discrete facility location problem.

The problem that is considered in this paper is known as the bi-objective obnoxious p-median problem, *Bi-OpM*, firstly introduced in [3]. It mainly consists on locating a set of obnoxious facilities on a landscape shared with customers (also known as demand points). The term obnoxious referring to a facility is used when it is desired to locate it as far as possible from the demand points. This situation appears when the interest is to locate facilities such as waste or hazardous material, nuclear power or chemical plants, noisy or polluting services like airports. Besides, the facilities should be properly distributed to avoid the situation where several obnoxious facilities are close to each other.

The *Bi-OpM* can be formally stated as follows. Let I be a set of customers, and J a set of candidate facility centers, where $|I| = n$ and $|J| = m$, and let d store the distances among all the considered nodes. The aim of the *Bi-OpM* is to locate a set P candidate facilities, having $|P| = p$ and $p < m$, while maximizing two objective functions: (f_1), the distance from each demand point to the facilities, computed as the sum of the minimum distances between each demand point and the nearest facility; and (f_2), the dispersion among the facilities, computed as the sum of the minimum distances from each facility to the rest of the selected facilities. More precisely, these objective functions can be described in the following way:

$$\begin{aligned} \max \quad & f_1 = \sum_{i \in I} \min d_{ij} : j \in P \\ \max \quad & f_2 = \sum_{j \in P} \min d_{jk} : k \in P, j \neq k \\ \text{s.t.} \quad & P \subseteq J \\ & |P| = p \end{aligned}$$

Some authors name facilities in P as *open* facilities and facilities in $J \backslash P$ as *closed* or *unopened* facilities.

On the other hand, it is important to emphasize that we are dealing with a multi-objective optimization problem. Hence, the definition of an efficient solution is the one for which no single-objective function value can be improved without deteriorating another objective function value. It is said that a solution P^* dominates another solution P if P^* is not worse than P in all the objectives, and P^* is better than P in at least one objective. Similarly, we say that P^* weakly dominates P if P^* is not worse than P in all the objectives [2]. Formally, as we are maximizing the objectives, a solution P^* dominates another solution P, if $f_i(P^*) \geq f_i(P)$ for all $i = 1, 2$ and $f_i(P^*) > f_i(P)$ for at least one $i = 1, 2$. According to this, we will say that a solution is *efficient* if there is no other solution that dominates it. The Pareto front, also known as the efficient frontier, is the set of efficient solutions. Our purpose then is to find a good approximation to the Pareto front, denoted as PF from now on.

As stated before, the *Bi-OpM* was first introduced in [3]. The authors proposed a Multi-Objective Memetic Algorithm (MOMA) defining two new variants of the crossover and mutation operators and studying three local search strategies applied in the MOMA. Furthermore, they performed a comparison using two multi-objective state-of-the-art methods, specifically the Non-dominated Sorting Genetic Algorithm II, (NSGA-II, [6]), and the Strength-Pareto Evolutionary Algorithm 2 (SPEA2, [16]), and also adding single-objective Genetic Algorithm (GA) which combines the objectives under study through a weighted sum of their values.

In this paper, a Variable Neighborhood Search algorithm (VNS) is adapted to solve the considered multi-objective optimization problem. Another contribution of this paper is to compare our algorithm against the best algorithm proposed in the literature so far, [3].

The rest of the paper is organized as follows. Section 2 describes our VNS proposal and details how the algorithm has being adapted and implemented to solve this bi-objective optimization problem. Section 3 presents the computational results where a experimentation and analysis of the results is shown. Finally, Sect. 4 summarizes the paper and discusses future work.

2 VNS Algorithm

To solve the *Bi-OpM* problem we propose a VNS approach that considers all the features of this bi-objective optimization problem. VNS is a metaheuristic framework originally introduced by [13] that relies on the idea of systematic changes in the neighborhood structures. The adaptability of the methodology has resulted in several variants in recent years (see [11] for a recent survey on the methodology), which has led to several successful applications for a variety of difficult optimization problems, such as those in [7] and [14]. In this work, VNS is adapted to solve a bi-objective optimization problem.

We propose a Basic VNS (BVNS) algorithm which combines deterministic and stochastic changes of neighborhood in order to obtain high quality solutions. Multi-objective VNS was originally proposed recently, see [8]. However, we follow a different approach in this paper, which is briefly described in Algorithm 1.

BVNS requires from two input parameters: a set of non-dominated solutions PF and the largest neighborhood to be explored, k_{max}. Starting from the first neighborhood (step 1), the method iterates until reaching the maximum predefined neighborhood $k_{\max}$ (steps 2–16). At each iteration two different phases are applied to every solution from the incumbent set of non-dominated solutions. In particular, the solution is randomly perturbed in the current neighborhood k using the *Shake* procedure (step 5). The proposed *Shake* algorithm consists in

Algorithm 1. BVNS(PF, k_{max})

```
1:  $k \leftarrow 1$
2:  while $k \neq k_{\max}$ do
3:      $PF' \leftarrow PF$
4:      for all $P \in PF'$ do
5:          $P' \leftarrow Shake(P, k)$
6:          $PF' \leftarrow Insert\&Update(P')$
7:          $P'' \leftarrow LocalSearch(P')$
8:          $PF' \leftarrow Insert\&Update(P'')$
9:          if $PF' \neq PF$ then                    ▷ Improve in the Pareto front
10:             $k \leftarrow 1$
11:             $PF \leftarrow PF'$
12:         else
13:             $k \leftarrow k + 1$
14:         end if
15:     end for
16: end while
17: return $PF$
```

randomly interchanging k assigned facilities with k candidate locations that do not belong to the current solution yet, generating solution P', which is added to the updated Pareto front, PF' (step 6). It is worth mentioning that the shake method performs random movements (whose sizes depend on the current neighborhood) which are not considered in the local search (i.e., interchange $k \leq p$ facilities simultaneously, while the local search only interchanges a single facility). Furthermore, the shake method accepts solutions of lower quality that will eventually let us explore further regions of the search space, while the local search only considers improved solutions.

A local search method is responsible of locally improving the perturbed solution P', obtaining solution P'' (step 7). Notice that every feasible solution generated during the search is a candidate solution for entering in the set of non-dominated solutions. The method *Insert & Update* (steps 6 and 8) performs this verification, inserting the solution if it is non-dominated by others already in the set, removing those solutions dominated by the new one. Regarding this behavior, any modification in the Pareto front is considered as an improvement since a new non-dominated solution has been included in it. Therefore, if the Pareto front has been modified, the search starts again from the first neighborhood (step 9), updating the incumbent Pareto front. Otherwise, the method explores the next neighborhood (step 13) until reaching the largest considered neighborhood. BVNS ends returning the set of non-dominated solutions generated in the search.

2.1 Constructive Method

The initial solution for the VNS algorithm is the set of non-dominated solutions *PF* conformed with the solutions generated by a constructive procedure inspired by the Greedy Randomized Adaptive Search Procedure (GRASP) methodology [10]. For this work, we have decided to use a semi-greedy procedure that combines greediness (intensification) and randomness (diversification) by means of a parameter α. The procedure generates a predefined number of initial solutions for both objective functions f_1 and f_2 that are evaluated for entering in the set of non-dominated solutions. Therefore, the output of the constructive phase is a set of non-dominated solutions PF, which acts as the input Pareto front for the VNS algorithm.

Algorithm 2 details the constructive method proposed for the *Bi-OpM*, which is generalized for any objective function, f_i. The input for the method is comprised of the set of candidate locations to host a facility J, the parameter α which controls the greediness/randomness of the method, and the objective function under consideration f_i. The method starts by randomly selecting a candidate location from the available ones, including it in the solution under construction (steps 1–2). Then, a Candidate List (CL) is created with the remaining candidate locations (step 3). The method iteratively selects new candidates until p locations has been selected (step 4). In each iteration, the minimum and maximum values of the objective function among all the candidates are evaluated (steps 5–6). After that, the Restricted Candidate List (RCL) is created (step 8) with the most promising candidates, i.e., those whose objective function value is larger or equal than a threshold th (step 7). On the one hand, if $\alpha = 0$, the construction is totally greedy (it only considers the facilities that produce the greatest increase in the objective function value). On the other hand, if $\alpha = 1$, then all facilities are included in the RCL so the construction is totally random. The next vertex to be added to the solution is selected at random from the RCL (step 9), updating the solution under construction and the CL (steps 10–11). The method ends when the solution has exactly p locations selected.

Algorithm 2. Construct(J, α, f_i)

```
1: $v \leftarrow Random(J)$
2: $P \leftarrow \{v\}$
3: $CL \leftarrow J \setminus \{v\}$
4: while $|P| < p$ do
5:     $g_{\min} \leftarrow \min_{v \in CL} f_i(P \cup \{v\})$
6:     $g_{\max} \leftarrow \max_{v \in CL} f_i(P \cup \{v\})$
7:     $th \leftarrow g_{\max} - \alpha \cdot (g_{\max} - g_{\min})$
8:     $RCL \leftarrow \{v \in CL \ : \ f_i(P \cup \{v\}) \geq th\}$
9:     $v' \leftarrow Random(RCL)$
10:    $P \leftarrow P \cup \{v'\}$
11:    $CL \leftarrow CL \setminus \{v'\}$
12: end while
13: return $P$
```

2.2 Local Search

The problem under consideration tries to optimize two different objective function, so a traditional approach would propose a different local search method for each objective function. Instead, we propose a single local search method that aggregates both objective functions in a unique search. Parameter β controls the influence of each objective function in the aggregated function f_a. More formally,

$$f_a \leftarrow \beta \cdot f_1 + (1 - \beta) \cdot f_2$$

Varying the value of β parameter will result in exploring different regions of the search space, potentially increasing the number of solutions included in the set of non-dominated solutions.

The local search method traverses all the selected facilities and tries to exchange it with every candidate location. The search follows a first improvement approach in order to reduce the computational effort of the method. In particular, every time an improvement move is found, it is performed and the search starts again.

The value of β must vary in order to obtain a more dense set of non-dominated solutions. In the context of the BVNS algorithm, we consider a random value of β in the range 0–1 in each local search phase, in order to explore a wider portion of the search space.

3 Computational Results

This section presents and discusses the results of the experimental experience conducted in this paper. In order to perform a fair comparison against the most competitive algorithm proposed in the literature, we have solved the same set of instances considered in [3]. Specifically, the previous paper presents eight instances where the number of nodes (indicated as $|V| = |I \cup J|$) ranges from 400 to 900, the number of demand points and facilities ($|I|$ and $|J|$) varies from 200 to 450 and the number of open facilities (p) is between 25 and 225. We show in Table 1 these features for all the instances, which are available at http://www.optsicom.es/biopm/. Our experiments were run on a computer provided with an Intel i7 2600 processor running at 3.4 GHz, 4 GB RAM and Ubuntu 16.04. The algorithms were implemented using Java 8.

After some preliminary computational experiments where several values of α and k_{max} were tested, we decided to use a random value for the α parameter and $k_{max} = 0.3 \cdot p$, since they obtained the best performance in our tests. Besides, a number of 100 iterations of the constructive method generated the initial Pareto front that our BVNS requires as input parameter.

Firstly, we show the results of our VNS proposal in terms of the hypervolume [17], which is the metric that was presented in [3]. In the previous paper, a total of seven different algorithms where compared. For the sake of the space, we will compare the results of our VNS proposal against three of the seven approaches from the state of the art. These algorithms are the NSGA-II [6],

Table 1. Description of the instances.

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

which is a classical multi-objective evolutionary algorithm, and the two variants of the memetic algorithm that obtained the best results in [3]: dominance-based local search, DBLS, and alternate objective local search, AOLS. Table 2 compares the performance of our VNS proposal showing the hypervolume values for all these algorithms and instances, the average hypervolume value normalized to the best hypervolume obtained for each instance (Avg. Norm. Hyp.), and the average normalized deviation between the hypervolume and the best hypervolume value (Avg. Dev.).

As it can be seen in the table, DBLS and AOLS obtain the best hypervolume values for two of the instances respectively, which are the four smaller ones. However, VNS reaches the best hypervolume in the four larges instances, obtaining also the best average normalized hypervolume and the best average deviation.

Notice that the results for NSGA-II, DBLS and AOLS, extracted from [3], correspond to the set of non-dominated solutions obtained after 30 runs of each algorithm. In the case of VNS, the results correspond to one single run. Therefore, it is clear that the efficiency of our VNS proposal is higher in relation to the other algorithms.

We have accounted for the number of efficient points obtained by each algorithm, which are shown in Table 3. In this case, our VNS proposal obtains the higher number of efficient points in all but the smallest instance. Therefore, VNS shows again a more efficient behavior in the optimization process, considering again that the results of VNS come from one single run.

In addition to the hypervolume and the number of efficient points, we depict in Fig. 1 the efficient points obtained in the experiments described before. From Fig. 1(a) to (h), the instances are sorted by size, from the smallest to the largest. As shown in the picture, the four fronts in the smallest instance, `pmed17.p25`, are almost completely overlapped. However, as the size of the instance grows, the front obtained with the NSGA-II algorithm is shifted downwards, while the other three algorithms maintain a good performance with similar shapes. This trend is more clear in the case of the efficient points of the largest instances,

Table 2. Hypervolume of final non-dominated fronts. Best values are depicted in bold font.

Instance	NSGA-II	DBLS	AOLS	VNS
pmed17.p25	8692436	8706765	**8710887**	8597575
pmed20.p50	9455493	**10042017**	10020130	9930426
pmed22.p62	10782181	**12565434**	12503057	12482833
pmed28.p75	8552938	10360761	**10383320**	10330591
pmed33.p87	8246628	10464880	10496109	**11011544**
pmed36.p100	9050694	11962494	11925733	**12413823**
pmed39.p112	7925756	11301612	11275309	**11707564**
pmed40.p225	7779073	10830924	10750521	**11978974**
Avg. Norm. Hyp.	0.8032	0.9726	0.9709	**0.9955**
Avg. Dev.	0.1968	0.0274	0.0291	**0.0045**

Table 3. Number of efficient points of final non-dominated fronts. Best values are depicted in bold font.

Instance	NSGA-II	DBLS	AOLS	VNS
pmed17.p25	84	85	**88**	68
pmed20.p50	41	136	121	**146**
pmed22.p62	17	140	127	**193**
pmed28.p75	15	115	102	**218**
pmed33.p87	24	103	108	**284**
pmed36.p100	12	89	82	**278**
pmed39.p112	8	104	76	**294**
pmed40.p225	22	133	123	**313**
Avg.	27.88	113.13	103.38	**224.25**

displayed in Fig. 1(e) to (h). In those instances, the front generated by our VNS proposal outperforms both the memetic and the NSGA-II approaches.

We have also obtained, for each instance, the complete set of non-dominated solutions after executing all the algorithms, that is, the final PF for each instance. This way, we have also measured the contribution of each algorithm to the final PF. Table 4 shows the ratio for each algorithm. We can see that VNS is the best contributor to the final PF but in the two smallest instances. However, it contributes in more that a 94% in the four largest instances, reaching the 100% in `pmed36.p100`. Again, it is worth mentioning that NSGA-II, DBLS and AOLS were executed 30 times, whereas VNS was run just once, and reached an average contribution of 71.27%.

Finally, we compare the execution time of the analyzed algorithms. Table 5 presents the time spent by VNS on the single run that produced the results

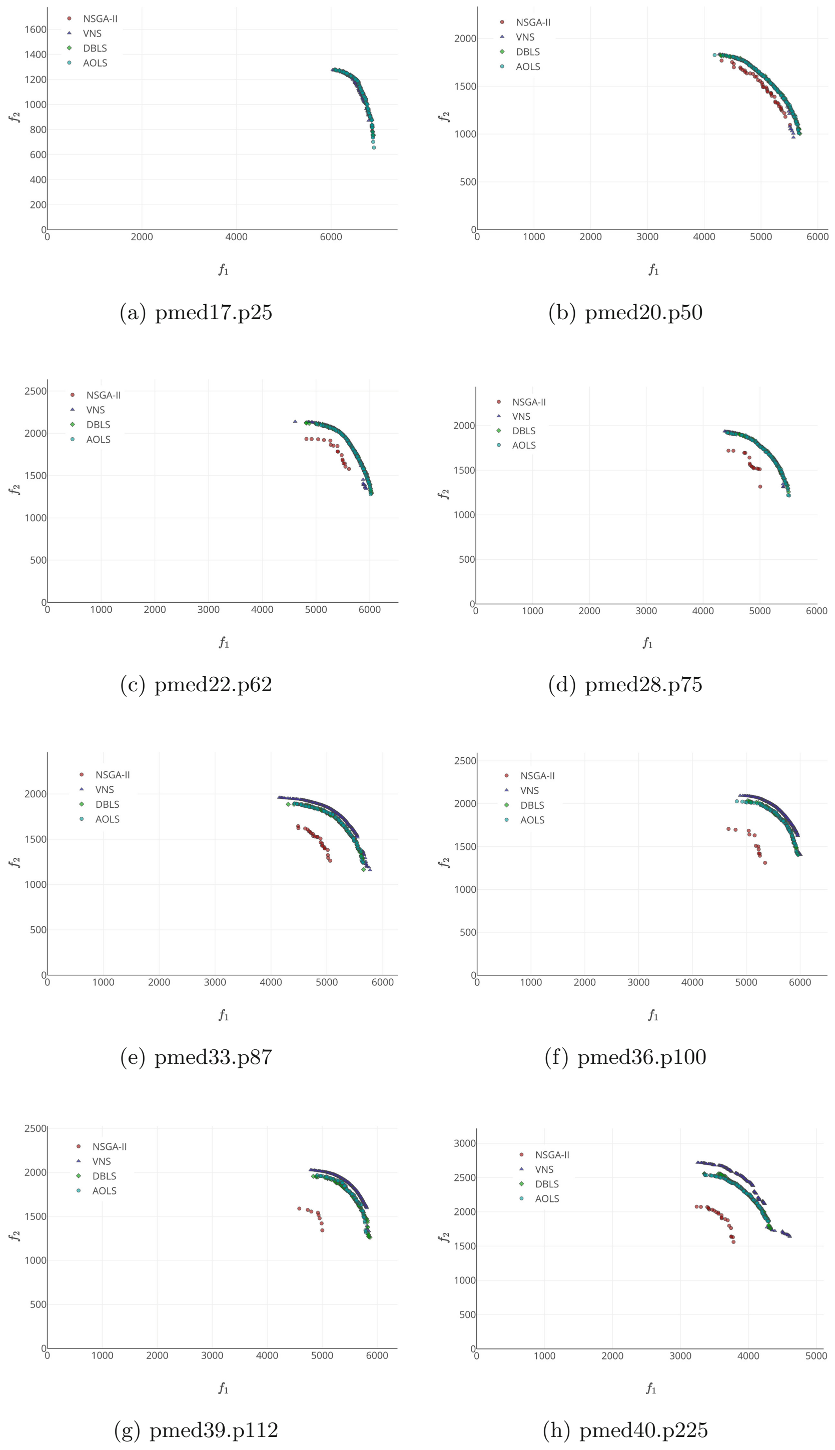

Fig. 1. Trade-off between f_1 and f_2.

Table 4. Contribution of each algorithm to the final PF. Best values are depicted in bold font.

Instance	NSGA-II	DBLS	AOLS	VNS
pmed17.p25	0.9545	0.9659	**0.9773**	0.0114
pmed20.p50	0	**0.3963**	0.25	0.3537
pmed22.p62	0	0.3108	0.0676	**0.6532**
pmed28.p75	0	0.1458	0.0833	**0.7708**
pmed33.p87	0	0.0036	0.0143	**0.9821**
pmed36.p100	0	0	0	**1**
pmed39.p112	0	0.0169	0	**0.9831**
pmed40.p225	0	0.0494	0.0031	**0.9475**
Avg.	0.1193	0.2361	0.1744	**0.7127**

previously analyzed, and the average execution time of NSGA-II, DBLS and AOLS. This average time was obtained after the 30 runs that produced the results shown before. Hence, despite that VNS is slower in one case (`pmed28-p75`), it is important to report that it obtains better results in one single run than the other algorithms after 30 runs.

Table 5. CPU time (secs). One single run of VNS versus average execution time of NSGA-II, DBLS and AOLS. Best values are depicted in bold font.

Instance	NSGA-II	DBLS	AOLS	VNS
pmed17-p25	1759.2	540.0	658.5	**21.8**
pmed20-p50	4995.3	933.9	1214.7	**410.1**
pmed22-p62	7990.8	1414.2	1768.2	**970.1**
pmed28-p75	13491.6	**1839.9**	2365.8	1856.1
pmed33-p87	19639.2	2706.9	3361.5	**1825.9**
pmed36-p100	27532.5	4367.7	3691.5	**1809.8**
pmed39-p112	29739.9	6099.6	3855.0	**2162.0**
pmed40-p225	114285.9	11440.8	8475.9	**3608.7**

4 Conclusions and Future Research

This paper generalizes the Variable Neighborhood Search algorithm (VNS) to solve a bi-objective optimization problem known as the bi-objective obnoxious p-median problem, *Bi-OpM*. To that end, the VNS approach is designed to take into account two conflicting objectives: to maximize the sum of the distances to the nearest demand point to each obnoxious facility and, to maximize the dispersion of obnoxious facilities. The interest of this problem appears because the *Bi-OpM* fits in many realistic situations where it is desired to locate facilities as far as possible from the demand points and among them.

Computational results show the superiority of the proposed algorithm over the state-of-art algorithms to solve the *Bi-OpM* so far on the same set of instances. Results obtained by the VNS algorithm outperform, in most of the instances, the three considered algorithms: NSGA-II, DBLS, and AOLS, spending less computational time.

As future work, it would be interesting to solve an extension of the *Bi-OpM* which will include an additional objective function. The new multi-objective obnoxious facility location problem that we will address, seeks to maximize the sum of the minimum distances between each demand point and its nearest facility and maximize the sum of the minimum distances between two facilities but also to minimize the number of demand points affected (or covered) by the facilities.

References

1. Carlsson, J.G., Jia, F.: Continuous facility location with backbone network costs. Transp. Sci. **49**(3), 433–451 (2014)
2. Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-Objective Problems. Genetic and Evolutionary Computation. Springer, New York (2006)
3. Colmenar, J., Martí, R., Duarte, A.: Multi-objective memetic optimization for the bi-objective obnoxious p-median problem. Knowl.-Based Syst. **144**, 88–101 (2018)
4. Cornuéjols, G., Nemhauser, G., Wolsey, L.: The uncapacitated facility location problem. In: Mirchandani, P.B., Francis, R.L. (eds.) Discrete Location Theory, pp. 119–171. Wiley-Interscience, New York (1990)
5. Dantrakul, S., Likasiri, C., Pongvuthithum, R.: Applied p-median and p-center algorithms for facility location problems. Expert Syst. Appl. **41**(8), 3596–3604 (2014)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
7. Duarte, A., Pantrigo, J., Pardo, E., Sánchez-Oro, J.: Parallel variable neighbourhood search strategies for the cutwidth minimization problem. IMA J. Manag. Math. **27**(1), 55 (2016)
8. Duarte, A., Pantrigo, J.J., Pardo, E.G., Mladenovic, N.: Multi-objective variable neighborhood search: an application to combinatorial optimization problems. J. Glob. Optim. **63**(3), 515–536 (2015)
9. Farahani, R.Z., Hekmatfar, M.: Facility Location: Concepts, Models, Algorithms and Case Studies. Springer, Heidelberg (2009)
10. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. J. Glob. Optim. **6**(2), 109–133 (1995)
11. Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S.: Variable neighborhood search: basics and variants. EURO J. Comput. Optim. **5**(3), 423–454 (2017)
12. Marín, A.: The discrete facility location problem with balanced allocation of customers. Eur. J. Oper. Res. **210**(1), 27–38 (2011)
13. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)

14. Sánchez-Oro, J., Sevaux, M., Rossi, A., Martí, R., Duarte, A.: Solving dynamic memory allocation problems in embedded systems with parallel variable neighborhood search strategies. Electron. Notes Discret Math. **47**, 85–92 (2015)
15. Wu, L.Y., Zhang, X.S., Zhang, J.L.: Capacitated facility location problem with general setup cost. Comput. Oper. Res. **33**(5), 1226–1241 (2006)
16. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K., et al. (eds.) Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), pp. 95–100. International Center for Numerical Methods in Engineering (CIMNE) (2002)
17. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)