INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 27 (2020) 480–493
DOI: 10.1111/itor.12646

Less is more approach: basic variable neighborhood search for the obnoxious *p*-median problem

Nenad Mladenović[a,b], Abdulaziz Alkandari[c], Jun Pei[d,e], Raca Todosijević[f,g] and Panos M. Pardalos[h]

[a] *Emirates College of Technology, Abu Dhabi, United Arab Emirates*
[b] *Ural Federal University, Yekaterinburg, Russia*
[c] *College of Technological Studies, Public Authority for Applied Education and Training, Kuwait City, Kuwait*
[d] *School of Management, Hefei University of Technology, Anhui Sheng, China*
[e] *Key Laboratory of Process Optimization and Intelligent Decision-Making, Ministry of Education, Hefei, China*
[f] *LAMIH UMR CNRS 8201, Université Polytechnique Hauts-de-France, 59313 Valenciennes Cedex 9, Famars, France*
[g] *Mathematical Institute of the Serbian Academy of Science and Arts, Kneza Mihaila 36, 11000 Belgrade, Serbia*
[h] *Department of Industrial and Systems Engineering, Faculty of Engineering, University of Florida, Gainesville, FL 32611-6595, USA*
E-mail: nenadmladenovic12@gmail.com [Mladenović]; abdl_alkandari@hotmail.com [Alkandari]; feiyijun198612@126.com [Pei]; racatodosijevic@gmail.com [Todosijević]; pardalos@ufl.edu [Pardalos]

Received 21 December 2017; received in revised form 10 February 2019; accepted 11 February 2019

Abstract

The goal of the *less is more* approach (LIMA) for solving optimization problems that has recently been proposed in Mladenović et al. (2016) is to find the minimum number of search ingredients that make a heuristic more efficient than the currently best. In this paper, LIMA is successfully applied to solve the obnoxious *p*-median problem (OpMP). More precisely, we developed a basic variable neighborhood search for solving the OpMP, where the single search ingredient, the *interchange* neighborhood structure, is used. We also propose a new simple local search strategy for solving facility location problems, within the interchange neighborhood structure, which is in between the usual ones: *first improvement* and *best improvement* strategies. We call it *facility best improvement* local search. On the basis of experiments, it appeared to be more efficient and effective than both *first* and *best improvement*. According to the results obtained on the benchmark instances, our heuristic turns out to be highly competitive with the existing ones, establishing new state-of-the-art results. For example, four new best-known solutions and 133 ties are claimed in testing the set with 144 instances.

Keywords: heuristic; less is more; obnoxious location; heap data structure; facility best improvement

1. Introduction

Obnoxious location problems are those that take into account the so-called obnoxious or semiobnoxious effects. These effects often occur when it is desired that a facility be located as far as possible from an inhabited center (obnoxious effect) or when a facility cannot be located far enough, although its closeness can have immediate disturbing or dangerous effects (semiobnoxious effect). Location problems with obnoxious effect arise when locating nuclear power plants, chemical plants, and dump sites, while examples of location problems with semiobnoxious effect include locating airports, recycle plants, and so on.

The first obnoxious location model in the literature is given by Church and Garfinkel (1978) who considered location of a facility in a network. Since that time, researchers studied many variants of obnoxious location problems. The recent literature review on existing variants and solution techniques developed for solving obnoxious problems may be found in Colmenar et al. (2016).

In this paper, we focus on the obnoxious p-median problem (OpMP). Given a set J of possible facility locations and a set of customers I, as well as the distance d_{ij} between each customer $i \in I$ and facility location $j \in J$, the OpMP consists in choosing p facility locations from the given set so that the sum of the distances between each customer and its nearest facility is maximized. Formally, the problem may be stated as:

$$\max \sum_{i \in I} \min\{d_{ij} : j \in S\}, \tag{1}$$

subject to

$$S \subset J, |S| = p. \tag{2}$$

The OpMP may be stated as a mixed integer program using binary variables y_j, $j \in J$, which indicate whether the facility location j is chosen or not, and binary variables x_{ij}, $i \in I$, $j \in J$, which indicate whether the client i is allocated to the facility location j or not. The resulting model is as follows (Belotti et al., 2007):

$$\max \sum_{i \in I} \sum_{j \in J} d_{ij} x_{ij}, \tag{3}$$

subject to

$$\sum_{j \in J} y_j = p, \tag{4}$$

$$x_{ij} \leq y_j \, \forall i \in I, j \in J, \tag{5}$$

$$y_j + \sum_{k \in S(i,j)} x_{ik} \leq 1 \, \forall i \in I, j \in J, \tag{6}$$

$$y_j, x_{ij} \in \{0, 1\} \, \forall i \in I, j \in J, \tag{7}$$

where $S(i, j)$ is the set of facilities more distant than j from the client i, that is,

$$S(i, j) = \{k \in J | (d_{ik} > d_{ij}) \vee (d_{ik} = d_{ij} \wedge k > j)\}.$$

The meaning of constraints in the model is as follows. Constraint (4) ensures that exactly p facility locations are chosen, while constraint (5) guarantees that customers are allocated only to open facilities. Finally, constraints (6) assign each customer to the closest open facility. Note that the constraints that assign one facility to each customer, that is, $\sum_{j \in J} x_{ij} = 1 \, \forall i \in I$, are included in constraints (6), where sign $\leq$ is used instead of equality because of the non-negativity of distances d_{ij}.

Unlike classical p-median problem, which is well studied in the literature (see, e.g., Mladenović et al., 2007), the obnoxious p-median gained relatively few attention. Since its introduction in the 1990s (Eiselt and Laporte, 1995; Welch and Salhi, 1997; Cappanera, 1999), just a few methods have been proposed to tackle this NP-hard problem (Tamir, 1991). Belotti et al. (2007) developed a branch and cut (B&C) algorithm coupled with a Tabu search (TS), while recently Colmenar et al. (2016) proposed a greedy randomized adaptive search procedure (GRASP) based heuristic, which uses a filtering mechanism in order to avoid applying a local search on low-quality constructed solutions. More recently, Herrán et al. (2020) proposed a parallel heuristic based on variable neighborhood search (VNS).

In this paper, we follow the recent heuristic approach, that is, *less is more approach* (LIMA), to solve optimization problems (Mladenović et al., 2016; Brimberg et al., 2017). LIMA's main idea is to find the minimum number of search ingredients when solving a particular optimization problem, which would make some heuristic more efficient than the currently best in the literature. In other words, the goal is to make a heuristic as simple as possible, but at the same time, more effective and efficient than the current state-of-the-art heuristic. We believe that discovering the simple and user-friendly heuristic for some particular problem, which is at the same time more effective and efficient than the others, represents a contribution to the science (rather than just combining methods without having good computational results, only to claim new methodological contribution, as it is often done in some hybrid approaches). By minimizing the number of ingredients in the search, it is much easier to get the answer to the typically made question "why heuristic is working well?". Following the LIMA's idea, we propose a basic VNS heuristic for solving the OpMP. The proposed heuristic uses only one neighborhood structure in both intensification and diversification phases of VNS.

The second contribution of this paper is our new simple local search strategy for solving facility location problems, that is in between usual ones (i.e., the first improvement and the best improvement). We call it "facility best improvement local search."

The usually used data structures for solving facility location problems are proposed in Hansen and Mladenović (1997) and Resende and Werneck (2003) (for the survey, see also Mladenović et al., 2007). In this paper, we propose new simple data structure for saving and updating a solution. It is based on *heap* data structure. Its complexity is analyzed.

With our Basic VNS that contains new results and contributions mentioned above, we succeed to obtain remarkable results on the benchmark instances from the literature. The proposed approach yet quite simple outperforms existing single-thread state-of-the-art approaches, that is, TS and GRASP. In addition, it is quite competitive in comparison with the parallel VNS (P-VNS). For example, four new best-known solutions (BKS) and 133 ties are claimed in testing the set with 144 instances. Such outcome indicates that sometimes less may yield more.

The rest of the paper is organized as follows. In Section 2, we provide thorough description of the proposed approach, whereas in Section 3, we assess the merit of the proposed approach. Finally, in Section 4, we draw some concluding remarks and indicate possible future research directions.

2. Basic variable neighborhood search for the OpMP

VNS is a flexible framework for building heuristics for approximate solution of optimization problems. It was introduced by Mladenović and Hansen (1997) and its main idea is to systematically change neighborhood structures during the search for an optimal (or near-optimal) solution. This idea comes from the following properties: (i) a local optimum relative to one neighborhood structure is not necessarily a local optimum for another neighborhood structure; (ii) a global optimum is a local optimum with respect to all neighborhood structures. The first property is usually exploited by using multiple local searches in the improvement step as in variable neighborhood descent (see, e.g., Hansen et al., 2017; Mjirda et al., 2017; Duarte et al., 2018). The second property suggests using several neighborhoods, if the found local optima are of poor quality.

The basic variant of VNS (called Basic VNS) consists of executing, alternately, one local search procedure (used to improve a solution) and one so-called shaking procedure (used to hopefully resolve local minima traps), together with the neighborhood change step. The whole process is iterated until a predefined stopping condition (e.g., maximum number of iterations or maximum CPU time) is met. Many variants of VNS have been derived from the Basic VNS scheme and applied to different optimization problems (for a survey, see, e.g., Hansen et al., 2010; for recent publications, see Irawan et al., 2020; Janković et al., 2017; Pinto et al., 2020; Brimberg et al., 2019; Chagas et al., 2020; Gruler et al., 2020).

2.1. Basic VNS

For solving the OpMP, we developed a heuristic that follows the rules of Basic VNS. Before providing more details of our heuristic, we will describe the solution representation for the OpMP. The solution of the OpMP is represented as a set S of p-chosen facilities because once we know the chosen facilities, each customer is directly assigned to the closest chosen facility. The steps of our Basic VNS for OpMP are given in Algorithm 1. It starts by creating an initial solution at random (see Algorithm 2), which is also the simplest possible way to start. Namely, the initial solution is formed by choosing p facilities from the set J at random. Such a solution is improved through main VNS loop, which includes executing shaking, local search, and neighborhood change procedures, one after another (see steps 5–9). In both *shaking* and *local search* steps, the same *Interchange* neighborhood structure is used, based on closing an open facility in S and opening facility that does not belong to S. Formally, the interchange neighborhood of a given solution S is defined as:

$$Interchange(S) = \{S' \subset J || S \cap S'| = |S| - 1,\ |S'| = |S| = p\}.$$

Algorithm 1. VNS heuristic for solving the OpMP

```
   Function VNS(S, k_max, t_max);
1  S ← Initial_solution ();
2  repeat
3  |  k ← 1;
4  |  while k ≤ k_max do
5  |  |  S' ← Shake(S, k);                    /* Shaking */
6  |  |  S'' ← LS(S');                        /* Local search */
7  |  |  k ← k + 1;                           /* Next neighborhood */
8  |  |  if S'' is better then S then
9  |  |  |  S ← S''; k ← 1;                  /* Make a move */
   |  |  end
   |  end
10 |  t ← CpuTime();
   until t > t_max;
11 Return S;
```

Algorithm 2. Procedure for creating an initial solution

```
   Function Initial_solution();
1  S = ∅;
2  for i = 1 to p do
3  |  Select j in J \ S at random;
4  |  S ← S ∪ {j};
   end
```

2.2. Heap data structure

In order to efficiently evaluate the objective function value of each solution in that neighborhood, for each customer i we maintain a heap data structure $h(i)$, containing facilities in the solution S ordered according to their closeness to the client i, and a $|I| \times |J|$ matrix *index* whose each entry $index(i, j)$ $j \in S$ corresponds to the index of the facility j in the heap $h(i)$. Thanks to such data structure, evaluating each solution S' requires $O(|I|)$ operations. Namely, if the first facility in the heap $h(i)$ is replaced by a facility i', new facility closest to the customer i becomes facility i', the second or the third facility in the heap $h(i)$. Otherwise, new facility closest to the customer i is either the first facility in the heap or newly opened facility i'. After executing one interchange move, updating data structures $h(i)$ and *index* requires $O(|I|\log(p))$ operations. Such complexity is due to data structure *index*, which enables us to detect the position of each facility to be closed by a move in the constant time in each heap $h(i)$.

The data structure used in Colmenar et al. (2016) is based on maintaining two matrices. Such data structure enables move evaluation in the complexity $O(|I|\log(|J|)) + O(|I|) = O(|I|\log(|J|))$,

while our heap data structure enables move evaluation in $O(|I|\log(p))$. Therefore, our heap data structure may be considerably better than the data structure presented in Colmenar et al. (2016), since the value p may be significantly less than $|J|$.

2.3. Shaking

The shaking routine is also the simplest possible. It has two formal parameters: the solution S and the neighborhood index k that determines the number of interchange steps performed. In other words, each iteration consists in choosing a random facility to go into S, replacing the other, also taken at random, that should go out. Obviously, it could happen that the same facility is chosen two times, that is, it is first removed from S and then returned back to S. It is clear that we could easily forbid such cases. However, for the sake of simplicity, we did not do that. It means that solutions from $N_s(S)$ are generated, where $s \leq k$.

Algorithm 3. Shaking procedure

```
   Function Shake(S,k);
1  for i = 1 to k do
2  |  Select S' in Interchange(S) at random;
3  |  S ← S';
   end
```

As has already been described, our Basic VNS uses one neighborhood structure within both key steps of VNS that are iterated: improvement procedure and shaking procedure (see steps 5 and 6). Moreover, the *Move or not* step is also the simplest possible (steps 6–9): move is made only if the better solution in the local search (step 6) is found. In addition, we set k_{max} to p in all testing, thus reducing the number of formal parameters of Basic VNS to single one, that is, t_{max}, the limit on CPU time. So, our Basic VNS is quite simple and meets the main requirements of a heuristic: simplicity and user-friendliness.

3. Computational results

The proposed Basic VNS is coded in C++ language and executed on a machine with an Intel Xeon E7 4820 CPU (2.00 GHz) and 16 GB of RAM. For testing purposes, the set of 144 benchmark instances from the literature have been used. These instances were created by Belotti et al. (2007), transforming the benchmark instances for the classical p-median problem publicly available at www.people.brunel.ac.uk/~mastjjb/jeb/orlib/pmedinfo.html. In particular, 24 instances (from pmed17 to pmed40) of the p-median problem were transformed into 72 instances for the OpMP. The transformation of each instance was performed by the procedure described in Belotti et al. (2007). Given the original instance with n nodes, the procedure generates three instances for the OpMP by considering three values of p: $\lfloor n/2 \rfloor$, $\lfloor n/4 \rfloor$ and $\lfloor n/8 \rfloor$ and selecting $n/2$ nodes at random to constitute the set of clients and declaring the remaining $n/2$ nodes to be the facilities. Recently,

Table 1
Comparison of different local searches on instance pmed40-p225

	Deviation (%)			Distance			
Local search	Minimum	Average	Maximum	Minimum	Average	Maximum	Time
LS_FI	0.28	2.45	5.61	15.00	38.23	68.00	8.72
LS_BI	0.72	3.08	5.94	20.00	42.29	65.00	6.55
LS_FBI	0.26	2.08	4.77	11.00	36.45	67.00	5.94

Herrán et al. (2020) extended the benchmark set introducing 72 new test instances generated in the same manner.

In the rest of the section, we first compare different search strategies that may be used to explore interchange neighborhood structure, and then we compare Basic VNS against state-of-the-art methods.

3.1. Comparison of different local search strategies

The local search with respect to interchange neighborhood structure may be performed using different search strategies. Let us compare the two usual and the new one:

- First_Improvement (FI) search strategy: as soon as an improving solutions is detected in the neighborhood, it is set as new incumbent solution, that is, the move is made;
- Best_Improvement (BI) search strategy: the best among all improving solution is chosen to be the new incumbent solution;
- Facility_Best_Improvement (FBI) search strategy: the best among all improving moves (if any) that involves exchanging given facility $j \in S$ by another facility $j' \notin S$ is executed and the resulting solution is set to be new incumbent solution. Note that BI search strategy differs from this search strategy in the sense that BI looks for the best exchange move with respect to all pairs of facilities j and j' ($j \in S$, $j' \notin S$), while FBI looks for the best facility to replace certain facility j.

Therefore, we distinguish three different local search procedures denoted as LS_FI, LS_BI, and LS_FBI, which use FI, BI, and FBI search strategies, respectively. Note that each of them, after accepting new incumbent solution, proceeds by exploring the interchange neighborhood centered now at this new incumbent solution.

In the first series of experiments, we are interested to detect which of these three local searches exhibit the best performances. For that purpose, on the largest instance pmed40-p225.A, each local search procedure is run 1000 times, each time starting from a different random solution. Note that we tested the local search methods on entire set of instances, but in order to save the space here we present results on instance pmed40-p225.A. This instance is of the largest size and reveals the typical performance of local searches. The summarized results are reported in Table 1 and Figure 1. In Table 1, columns 2–4 give the minimum, the average, and the maximum % deviation from the BKS, respectively, over 1000 runs. Columns 5–7 report the minimum, average, and maximum

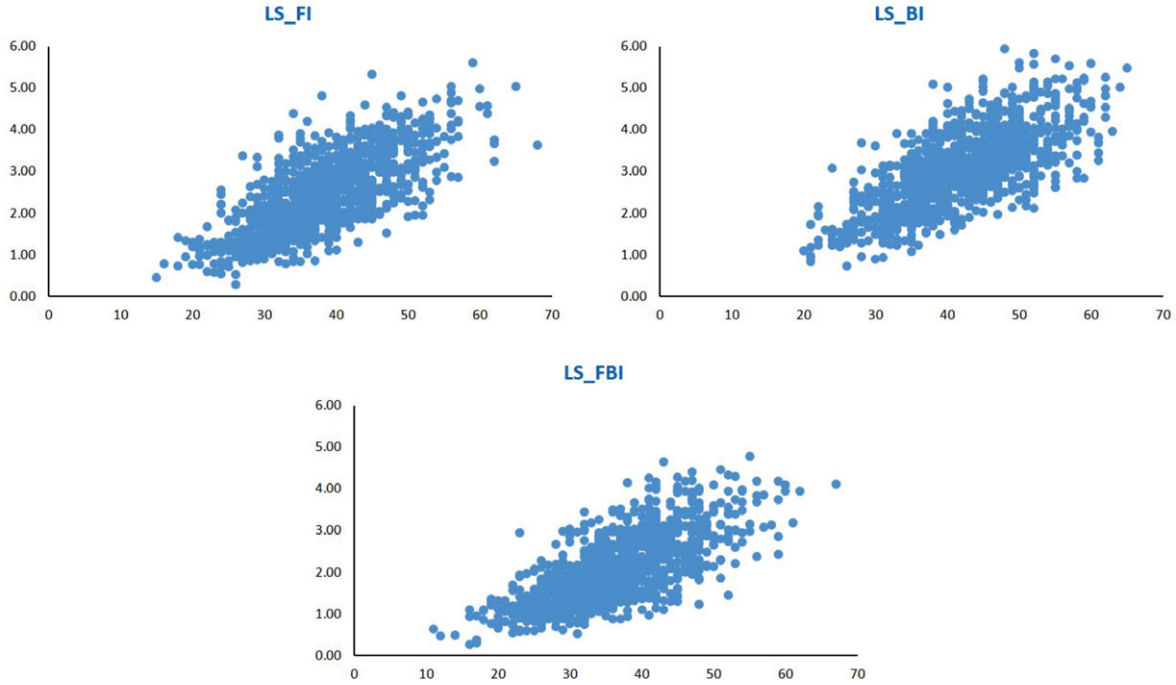

Fig. 1. Distribution of 1000 local maxima on distance-to-target diagram for different local search strategies.

distance between the generated local optima over 1000 runs and the BKS. The distance between solutions S and S' is defined as the half of the cardinality of the symmetric difference of sets S and S', that is:

$$d(S, S') = \frac{|S \triangle S'|}{2}. \tag{8}$$

The last column reports the average computing time spent to reach a local maximum (in seconds). Figure 1 depicts distributions of local optima for these three local search procedures. Distributions are shown on distance-to-target diagrams (Boese et al., 1994), where each local optimum is presented by the point (x, y). Its coordinates are:

(1) x: the distance between the local optimum and the BKS;
(2) y: the percentage deviation of value of the local optimum from the best-known value.

From the reported results, it follows that best performances are exhibited by our new strategy `LS_FBI`. Local search `LS_FI` is also able to produce high-quality solutions (of similar quality as those provided by `LS_FBI`), but it consumes much more CPU time than the other two local searches. Finally, local search `LS_BI` exhibits the worst performance regarding the solution quality provided. It confirms the observations obtained by comparative analysis on other combinatorial optimization problems (see, e.g., Hansen and Mladenović, 2006): BI is the worst choice if the initial

solution is taken at random. For the random initial solution, here we show that the semirandom or FBI strategy can be better than FI.

3.2. Comparison with state-of-the-art methods

On each test instance, Basic VNS has been executed setting the t_{max} parameter to 600 seconds. As a local search `LS_FBI` is used within Basic VNS, since it exhibits the best performances as shown in the previous section. The obtained results are compared with those reported in Herrán et al. (2020). Those results were obtained executing TS algorithm and B&C algorithm from Belotti et al. (2007) and GRASP heuristic proposed in Colmenar et al. (2016) on the benchmark instances as well as P-VNS (Herrán et al., 2020). Among the compared heuristics, TS, GRASP, and Basic VNS are single-thread heuristics, wh P-VNS executes several threads in parallel. All testings in Herrán et al. (2020) were carried on a machine with an Intel i5 660 CPU (3.3 GHz) and 8 GB of RAM, while we used an Intel Xeon E7 4820 CPU (2.00 GHz). Due to the different computing environments, we have normalized the CPU times reported in Herrán et al. (2020) using the approach described in Dongarra (2014) and data from www.cpubenchmark.net. The normalized CPU times $\overline{T}_H$ are computed using the following formula:

$$T_H^* = \frac{P(\text{Intel i5 660 CPU})}{P(\text{Intel Xeon E7 4820 CPU})} T_H,$$

where P(Intel i5 660 CPU) and P(Intel Xeon E7 4820 CPU) are the Passmark CPU scores of the computer used in Herrán et al. (2020) and our computer, respectively, and T_H is time reported in Herrán et al. (2020). We assume that the computational power is proportional to the number of cores used, which is used to estimate the Passmark CPU scores of the 2-core machine used in Herrán et al. (2020) and our 10-core machine.

In Tables 2 and 3, we report the results comparison. In both tables, instances are grouped according to the size. In Herrán et al. (2020), instances are grouped in three classes: small instances ($p = n/16$), medium ($p = n/8$), and large ($p = n/4$). Each class contains 48 instances. Our VNS was executed 30 times on each test instance. In each run, we recorded the best solution found as well as the time elapsed upon reaching this solution for the first time (so-called time-to-target). Therefore, for each instance, we stored the best solution value found, the average solution value, and average time-to-target in 30 runs. In Table 2, we report the average of these values over instances within the same class. Column "% dev." reports the average of the percentage deviations of the best solution values found by our VNS on each instance from the current BKS. On each test instance, the percentage deviation is calculated as

$$\frac{BKS - VNS_{best}}{BKS} \times 100.$$

Columns 2–6 in Table 3 provide the number of the current BKS that each method is able to reproduce, whereas the last column "#new BKS" reports the number of new BKS established by our VNS on each class of instances. In both tables, the last row reports summary results over all instances in the benchmark set.

Table 2
Comparison with the sate-of-the-art methods

		B&C		TS		GRASP		P-VNS		VNS			
										Value		Average	
Instances	BKS	Value	T_H^*	Value	T_H^*	Value	T_H^*	Value	T_H^*	Best	Average	time	% dev.
Small	7582.4	7452.0	1875.5	7463.0	283.8	7582.4	258.1	7582.4	17.5	7582.4	7582.2	48.1	0.00000
Medium	5856.7	5784.8	7205.9	5814.5	413.9	5855.5	423.0	5856.6	59.6	5856.9	5850.4	220.1	–0.00249
Large	4213.2	4090.3	7607.5	4169.8	345.6	4205.0	669.4	4213.1	226.9	4212.8	4202.3	329.6	0.00961
All	5884.1	5775.7	5562.9	5815.8	347.8	5881.0	450.1	5884.0	101.5	5884.0	5878.3	199.3	0.00237

Table 3
Comparison with the sate-of-the-art methods (number of BKS)

Instances	B&C	TS	GRASP	P-VNS	VNS	"#new BKS"
Small	9	16	48	48	48	0
Medium	6	6	35	43	45	3
Large	1	1	9	46	40	1
All	16	23	92	137	133	4

From the reported results, we may infer that the proposed approach yet quite simple clearly outperforms existing single-thread state-of-the-art approaches, that is, TS and GRASP. Our Basic VNS is able to find solutions of higher quality consuming less CPU time than GRASP and TS. In addition, it is quite competitive with the P-VNS, although P-VNS cannot be fairly compared with single-thread approaches since it uses more computational resources. If we compare the number of BKS (see Table 3), we can observe that TS, GRASP, Basic VNS, and P-VNS are able to reproduce 23, 95, 133, and 137 existing BKS, respectively. Moreover, on the medium size instances, our VNS offers more BKS than any method in comparison. In addition, our Basic VNS succeeds to establish four new BKS (three for medium instances and one for a large instance). This further implies that at the moment basic VNS holds 137 BKS, the same number P-VNS held before.

4. Conclusions

In this paper, the OpMP has been studied. For solving this NP-hard problem, we have developed a Basic VNS heuristic that follows the principle of the newly proposed "less is more" heuristic approach, that is, it searches for the minimum number of ingredients that enables the proposed heuristic to be more efficient than the current state-of-the-art. More precisely, our basic VNS uses just one neighborhood structure in both shaking and local search procedures. However, although quite simple, the proposed basic VNS outperforms the state-of-the-art results of single-thread heuristics, thus confirming, once again, that sometimes less may yield more. In addition, it is highly competitive with the P-VNS heuristic.

The future work may include developing "less is more" heuristic approaches for solving other NP-hard optimization problems. Moreover, semirandom local search strategy (FBI), which we proposed here, may be used for solving other discrete optimization problems. In addition, heap data structure, which we introduced to solve facility location problems, can be tried out for solving p-median or p-center problems and may be compared with those usually used.

Acknowledgments

This research is partially supported by Serbian Ministry of Education, Science and Technological Development under the grants nos. 174010 and 044006. In addition, this research is partially supported by the framework of the grant number BR05236839 "Development of information technologies and systems for stimulation of personality's sustainable development as one of the bases of development of digital Kazakhstan" and by the National Natural Science Foundation of China (Nos. 71871080, 71601065, 71690235, 71501058), Innovative Research Groups of the National Natural Science Foundation of China (71521001), the Humanities and Social Sciences Foundation of the Chinese Ministry of Education (No. 15YJC630097), and the Base of Introducing Talents of Discipline to Universities for Optimization and Decision-Making in the Manufacturing Process of Complex Product (111 project).

References

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

Boese, K.D., Kahng, A.B., Muddu, S., 1994. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters* 16, 2, 101–113.

Brimberg, J., Mladenović, N., Todosijević, R., Urošević, D., 2017. Less is more: solving the max-mean diversity problem with variable neighborhood search. *Information Sciences* 382, 179–200.

Brimberg, J., Mladenović, N., Todosijević, R., Urošević, D., 2019. Solving the capacitated clustering problem with variable neighborhood search. *Annals of Operations Research* 272, 1, 289–321.

Cappanera, P., 1999. *A Survey on Obnoxious Facility Location Problems*. Dipartimento di Informatica, Università di Pisa.

Chagas, J.B.C., Silveira, U.E.F., Santos, A.G., Souza, M.J.F., 2020. A variable neighborhood search heuristic algorithm for the double vehicle routing problem with multiple stacks. *International Transactions in Operational Research* 27, 1, 112–137.

Church, R.L., Garfinkel, R.S., 1978. Locating an obnoxious facility on a network. *Transportation Science* 12, 2, 107–118.

Colmenar, J.M., Greistorfer, P., Martí, R., Duarte, A., 2016. Advanced greedy randomized adaptive search procedure for the obnoxious p-median problem. *European Journal of Operational Research* 252, 2, 432–442.

Dongarra, J.J., 2014. Performance of various computers using standard linear equations software. Technical report, CS-89-85, University of Manchester, Manchester, UK.

Duarte, A., Sánchez-Oro, J., Mladenović, N., Todosijević, R., 2018. Variable neighborhood descent. In Martí, R., Pardalos, P.M., Resende, M.G.C. (eds) *Handbook of Heuristics*. Springer, Basel, Switzerland, pp. 341–367.

Eiselt, H., Laporte, G., 1995. Objectives in location problems. In Drezner, Z. (ed.) *Facility Location: a Survey of Applications and Methods*. Springer, New York, pp. 151–180.

Gruler, A., Panadero, J., de Armas, J., Prez, J.A.M., Juan, A.A., 2020. A variable neighborhood search simheuristic for the multiperiod inventory routing problem with stochastic demands. *International Transactions in Operational Research* 27, 1, 314–335.

Hansen, P., Mladenović, N., 1997. Variable neighborhood search for the p-median. *Location Science* 5, 4, 207–226.
Hansen, P., Mladenović, N., 2006. First vs. best improvement: an empirical study. *Discrete Applied Mathematics* 154, 5, 802–817.
Hansen, P., Mladenović, N., Pérez, J.A.M., 2010. Variable neighbourhood search: methods and applications. *Annals of Operations Research* 175, 1, 367–407.
Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S., 2017. Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization* 5, 3, 423–454.
Herrán, A., Colmenar, J.M., Martí, R., Duarte, A., 2020. A parallel variable neighborhood search approach for the obnoxious p-median problem. *International Transactions in Operational Research* 27, 1, 336–360.
Irawan, C.A., Imran, A., Luis, M., 2020. Solving the bi-objective capacitated p-median problem with multilevel capacities using compromise programming and VNS. *International Transactions in Operational Research* 27, 1, 361–380.
Janković, O., Mišković, S., Stanimirović, Z., Todosijević, R., 2017. Novel formulations and VNS-based heuristics for single and multiple allocation p-hub maximal covering problems. *Annals of Operations Research* 259, 1–2, 191–216.
Mjirda, A., Todosijević, R., Hanafi, S., Hansen, P., Mladenović, N., 2017. Sequential variable neighborhood descent variants: an empirical study on the traveling salesman problem. *International Transactions in Operational Research* 24, 3, 615–633.
Mladenović, N., Brimberg, J., Hansen, P., Moreno-Pérez, J.A., 2007. The p-median problem: a survey of metaheuristic approaches. *European Journal of Operational Research* 179, 3, 927–939.
Mladenović, N., Hansen, P., 1997. Variable neighborhood search. *Computers & Operations Research* 24, 11, 1097–1100.
Mladenović, N., Todosijević, R., Urošević, D., 2016. Less is more: basic variable neighborhood search for minimum differential dispersion problem. *Information Sciences* 326, 160–171.
Pinto, T., Alves, C., Valério de Carvalho, J., 2020. Variable neighborhood search algorithms for the vehicle routing problem with two-dimensional loading constraints and mixed linehauls and backhauls. *International Transactions in Operational Research* 27, 1, 549–572.
Resende, M.G., Werneck, R.F., 2003. On the implementation of a swap-based local search procedure for the p-median problem. *Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments (ALENEX03)*, pp. 119–127.
Tamir, A., 1991. Obnoxious facility location on graphs. *SIAM Journal on Discrete Mathematics* 4, 4, 550–567.
Welch, S., Salhi, S., 1997. The obnoxious p facility network location problem with facility interaction. *European Journal of Operational Research* 102, 2, 302–319.

Appendix

Tables A1 and A2 contain detailed results of our VNS on entire set of instances. For each instance, we report the best solution value found, the average solution value, and average time-to-target in 30 runs. In addition, we report the percentage deviation of the best solution value found by VNS from the current BKS for each instance. The percentage deviation is calculated as:

$$\frac{BKS - VNS_{best}}{BKS} \times 100.$$

A negative percentage deviation means that a new BKS value is established. All new BKS values are boldfaced as well as corresponding percentage deviations.

Table A1
Detailed results of VNS on set A instances

Instance	BKS	VNS value		VNS time	% dev.	Instance	BKS	VNS value		VNS average time	% dev.
		Best	Average					Best	Average		
pmed17-p100.A	4054	4054	4054.0	26.5	0.000	pmed29-p150.A	4141	4141	4132.0	356.6	0.000
pmed17-p25.A	7317	7317	7317.0	13.3	0.000	pmed29-p37.A	7404	7404	7404.0	119.6	0.000
pmed17-p50.A	5411	5411	5410.9	210.8	0.000	pmed29-p75.A	5880	5880	5774.1	232.5	0.000
pmed18-p100.A	4220	4220	4219.8	155.5	0.000	pmed30-p150.A	4385	4385	4378.9	214.1	0.000
pmed18-p25.A	7432	7432	7432.0	2.1	0.000	pmed30-p37.A	7704	7704	7704.0	11.2	0.000
pmed18-p50.A	5746	5746	5746.0	25.0	0.000	pmed30-p75.A	6189	6189	6183.0	287.9	0.000
pmed19-p100.A	4033	4033	4032.5	128.3	0.000	pmed31-p175.A	4135	4135	4120.5	416.4	0.000
pmed19-p25.A	7020	7020	7020.0	1.6	0.000	pmed31-p43.A	7424	7424	7422.0	87.9	0.000
pmed19-p50.A	5387	5387	5386.6	94.4	0.000	pmed31-p87.A	5905	5905	5905.0	182.0	0.000
pmed20-p100.A	4063	4063	4063.0	59.3	0.000	pmed32-p175.A	4242	4242	4225.6	474.8	0.000
pmed20-p25.A	7648	7648	7648.0	1.2	0.000	pmed32-p43.A	7794	7794	7793.0	90.3	0.000
pmed20-p50.A	5872	5872	5872.0	21.4	0.000	pmed32-p87.A	5925	5925	5918.9	234.5	0.000
pmed21-p125.A	4155	4155	4153.8	146.6	0.000	pmed33-p175.A	4105	4105	4096.0	439.3	0.000
pmed21-p31.A	7304	7304	7304.0	6.8	0.000	pmed33-p43.A	7598	7598	7598.0	59.1	0.000
pmed21-p62.A	5784	5784	5782.3	242.4	0.000	pmed33-p87.A	5793	5793	5789.4	335.8	0.000
pmed22-p125.A	4358	4358	4353.8	304.3	0.000	pmed34-p175.A	4287	4287	4264.9	399.8	0.000
pmed22-p31.A	7900	7900	7900.0	25.2	0.000	pmed34-p43.A	7725	7725	7725.0	44.6	0.000
pmed22-p62.A	5995	5995	5994.1	90.8	0.000	pmed34-p87.A	5849	5849	5841.2	294.4	0.000
pmed23-p125.A	4114	4114	4105.4	331.3	0.000	pmed35-p100.A	5845	5845	5842.6	304.9	0.000
pmed23-p31.A	7841	7841	7841.0	3.4	0.000	pmed35-p200.A	4007	4004	3984.6	445.2	0.075
pmed23-p62.A	5785	5785	5784.9	138.8	0.000	pmed35-p50.A	7155	7155	7154.0	164.8	0.000
pmed24-p125.A	4091	4091	4091.0	99.9	0.000	pmed36-p100.A	6461	6461	6458.3	126.5	0.000
pmed24-p31.A	7425	7425	7425.0	9.5	0.000	pmed36-p200.A	4319	4314	4299.0	471.0	0.116
pmed24-p62.A	5528	5528	5527.9	127.6	0.000	pmed36-p50.A	8179	8179	8178.4	95.3	0.000
pmed25-p125.A	4155	4155	4153.0	242.7	0.000	pmed37-p100.A	6203	6203	6195.0	366.4	0.000
pmed25-p31.A	7552	7552	7552.0	3.1	0.000	pmed37-p200.A	4593	4589	4575.5	469.4	0.087
pmed25-p62.A	5767	5767	5767.0	177.2	0.000	pmed37-p50.A	7830	7830	7829.9	260.0	0.000
pmed26-p150.A	4341	4339	4333.6	265.7	0.046	pmed38-p112.A	5913	**5915**	5907.7	395.4	**–0.034**
pmed26-p37.A	8112	8112	8112.0	2.6	0.000	pmed38-p225.A	4428	4428	4394.1	503.2	0.000
pmed26-p75.A	5789	5789	5787.6	247.4	0.000	pmed38-p56.A	7432	7432	7432.0	81.6	0.000
pmed27-p150.A	4062	4062	4052.2	349.0	0.000	pmed39-p112.A	5935	5935	5927.3	355.5	0.000
pmed27-p37.A	7556	7556	7556.0	27.9	0.000	pmed39-p225.A	4369	4369	4345.0	507.7	0.000
pmed27-p75.A	5668	5668	5664.3	199.5	0.000	pmed39-p56.A	7712	7712	7712.0	61.2	0.000
pmed28-p150.A	4099	4099	4091.9	343.6	0.000	pmed40-p112.A	6272	6272	6266.5	399.1	0.000
pmed28-p37.A	7366	7366	7366.0	28.6	0.000	pmed40-p225.A	4571	4567	4548.8	514.1	0.088
pmed28-p75.A	5681	5681	5675.6	242.8	0.000	pmed40-p56.A	8211	8211	8209.7	133.9	0.000
Average	5795.3	5795.3	5793.8	122.1	0.0013	Average	5997.8	5997.4	5987.1	276.0	0.0092

Table A2
Detailed results of VNS on set B instances

Instance	BKS	VNS value		VNS time	% dev.	Instance	BKS	VNS value		VNS average time	% dev.
		Best	Average					Best	Average		
pmed17-p100.B	3992	3992	3992.0	14.6	0.000	pmed29-p150.B	4157	4157	4152.0	326.3	0.000
pmed17-p25.B	6905	6905	6905.0	3.1	0.000	pmed29-p37.B	7529	7529	7529.0	15.1	0.000
pmed17-p50.B	5563	5563	5563.0	88.6	0.000	pmed29-p75.B	5709	5709	5707.1	264.9	0.000
pmed18-p100.B	4122	4122	4119.9	241.7	0.000	pmed30-p150.B	4313	4313	4282.4	395.2	0.000
pmed18-p25.B	7662	7662	7662.0	2.0	0.000	pmed30-p37.B	8048	8048	8048.0	6.1	0.000
pmed18-p50.B	5852	5852	5852.0	15.0	0.000	pmed30-p75.B	6041	6041	6039.6	166.6	0.000
pmed19-p100.B	4016	4016	4016.0	86.4	0.000	pmed31-p175.B	4138	4138	4119.8	430.9	0.000
pmed19-p25.B	6816	6816	6816.0	2.6	0.000	pmed31-p43.B	7320	7320	7320.0	96.4	0.000
pmed19-p50.B	5423	5423	5423.0	23.6	0.000	pmed31-p87.B	5618	**5621**	5614.6	303.5	**–0.053**
pmed20-p100.B	4067	4067	4066.9	151.1	0.000	pmed32-p175.B	4244	4242	4219.2	391.1	0.047
pmed20-p25.B	7349	7349	7349.0	1.3	0.000	pmed32-p43.B	7899	7899	7899.0	15.7	0.000
pmed20-p50.B	5665	5665	5665.0	24.7	0.000	pmed32-p87.B	5852	5852	5824.1	435.7	0.000
pmed21-p125.B	4033	4033	4026.1	216.4	0.000	pmed33-p175.B	4156	4156	4140.9	397.5	0.000
pmed21-p31.B	7331	7331	7331.0	4.8	0.000	pmed33-p43.B	7611	7611	7611.0	24.3	0.000
pmed21-p62.B	5870	5870	5870.0	57.7	0.000	pmed33-p87.B	5840	5840	5835.0	265.3	0.000
pmed22-p125.B	4338	4338	4332.5	310.7	0.000	pmed34-p175.B	4270	4270	4262.3	387.5	0.000
pmed22-p31.B	7695	7695	7695.0	3.2	0.000	pmed34-p43.B	7514	7514	7514.0	21.9	0.000
pmed22-p62.B	6259	6259	6259.0	26.1	0.000	pmed34-p87.B	5857	5857	5856.0	342.0	0.000
pmed23-p125.B	4095	4095	4093.5	261.7	0.000	pmed35-p100.B	5639	5639	5629.2	414.8	0.000
pmed23-p31.B	7137	7137	7136.9	32.5	0.000	pmed35-p200.B	4109	4108	4089.8	465.8	0.024
pmed23-p62.B	5724	5724	5724.0	94.4	0.000	pmed35-p50.B	7570	7570	7570.0	54.2	0.000
pmed24-p125.B	4072	4072	4060.2	271.5	0.000	pmed36-p100.B	6219	6219	6201.3	331.3	0.000
pmed24-p31.B	7190	7190	7190.0	10.1	0.000	pmed36-p200.B	4319	4319	4297.1	468.4	0.000
pmed24-p62.B	5752	5752	5751.1	141.2	0.000	pmed36-p50.B	8144	8144	8144.0	56.7	0.000
pmed25-p125.B	4233	4233	4230.4	258.8	0.000	pmed37-p100.B	6209	**6211**	6198.1	326.6	**–0.032**
pmed25-p31.B	7552	7552	7552.0	16.8	0.000	pmed37-p200.B	4609	4609	4596.2	474.1	0.000
pmed25-p62.B	5692	5692	5690.9	180.9	0.000	pmed37-p50.B	8379	8379	8379.0	25.2	0.000
pmed26-p150.B	4173	4173	4166.7	346.7	0.000	pmed38-p112.B	5949	5949	5930.1	447.8	0.000
pmed26-p37.B	7643	7643	7643.0	7.4	0.000	pmed38-p225.B	4446	4446	4427.7	562.2	0.000
pmed26-p75.B	5923	5923	5923.0	132.9	0.000	pmed38-p56.B	7535	7535	7534.8	218.5	0.000
pmed27-p150.B	4144	4144	4137.6	297.1	0.000	pmed39-p112.B	6198	6198	6192.1	320.0	0.000
pmed27-p37.B	7448	7448	7448.0	19.9	0.000	pmed39-p225.B	4266	4266	4248.3	541.0	0.000
pmed27-p75.B	5844	5844	5843.3	156.6	0.000	pmed39-p56.B	7625	7625	7624.8	142.3	0.000
pmed28-p150.B	4069	4069	4059.0	368.1	0.000	pmed40-p112.B	6200	6200	6181.8	410.5	0.000
pmed28-p37.B	7388	7388	7388.0	6.8	0.000	pmed40-p225.B	4524	**4525**	4506.3	489.3	**–0.022**
pmed28-p75.B	5642	5642	5637.8	261.4	0.000	pmed40-p56.B	8022	8022	8021.6	188.9	0.000
Average	5741.1	5741.1	5739.4	115.0	0.000	Average	6002.2	6002.3	5992.9	284.0	–0.001