Optimization Letters (2022) 16:281–300
https://doi.org/10.1007/s11590-020-01675-z

ORIGINAL PAPER

An efficient heuristic for a hub location routing problem

Mustapha Ratli[1] · Dragan Urošević[2] · Abdessamad Ait El Cadi[1] · Jack Brimberg[3] · Nenad Mladenović[4] · Raca Todosijević[1]

Received: 31 May 2020 / Accepted: 19 November 2020 / Published online: 28 November 2020

Abstract
This paper examines a new model for hub location known as the hub location routing problem. The problem shares similarities with the well studied uncapacitated single allocation p-hub median problem except that the hubs are now connected to each other by a cyclical path (or tour) known as the global route, each cluster of non-hub nodes and assigned hub is also connected by a single tour known as a local route, and the length of the local routes is constrained to a maximum number of nodes. Thus, aside from the normal tasks of hub selection and allocation of non-hub nodes, up to p + 1 travelling salesman problems need to be solved. A heuristic based on the general variable neighborhood search framework is proposed here to solve this very complicated problem. The improvement phase of the algorithm uses a sequential variable neighborhood descent with multiple neighborhoods required to suit the complex nature of the problem. A best sequencing of the neighborhoods is established through empirical testing. The perturbation phase known as the shaking procedure also uses a well-structured selection of neighborhoods in order to effectively diversify the search to different regions of the solution space. Extensive computational testing shows that the new heuristic significantly outperforms the state-of-the-art. Out of 912 test instances from the literature, we are able to obtain 691 new best known solutions. Not only are the improvements in objective values quite impressive, but also these new solutions are obtained in a small fraction of the time required by the competing algorithms.

Keywords p-hub · Hub location-routing problem · Heuristics · General variable neighborhood search

1 Introduction

The aim of a transportation/communication network is to facilitate transport/communication between all origin–destination (O–D) pairs. Establishing direct transport/communication between all O–D pairs may be unreasonable for many reasons espe-

Extended author information available on the last page of the article

cially from the economic (or cost) perspective. As an alternative approach, the idea of hub networks has been proposed. The main feature of a hub network that differentiates it from other types is the existence of special nodes (hubs) designated to enable the connection between O–D pairs. Namely, instead of having a direct path between each pair of nodes, nodes communicate via hubs. For example, if we have an origin node O and a destination D, a valid path between them may have the form $O - hub_1 - hub_2 - D$. More precisely, the flow from O to D is first collected at hub_1 then transferred to hub_2 and finally delivered to node D. So, the main principle of hub networks is that direct connection between non-hub nodes is not possible, and instead, all transport/communication is performed via hub nodes. Hub nodes have a role to collect, consolidate and transfer, and finally distribute the flow in a hub network. Note that on a path $O - hub_1 - hub_2 - D$, hub nodes hub_1 and hub_2 may be the same. In this case the same hub is used for collection, consolidation and transfer, and distribution purposes,.

During the past years, many variants of the hub location problem, including those with and without capacity constraints, or with different allocation strategies, topologies and objective functions, have been studied. For more details on the classification, solution approaches and applications of hub location problems, we refer the reader to [1,3–6,8–11,14–17,21–23,29–31,34–36].

In this paper, we study a relatively new type of hub location problem known as the hub location routing problem. The main feature that differentiates this variant is the existence of local and global routes. A local route starts and ends at a hub that serves one or more non-hub nodes, while the global route is used to interconnect hub nodes, and its main purpose is to transfer the flow between hubs (see Fig. 1).

The problem studied in this paper consists of designating p nodes on a given network to serve as hubs, assigning each non-hub node to exactly one hub, connecting non-hub

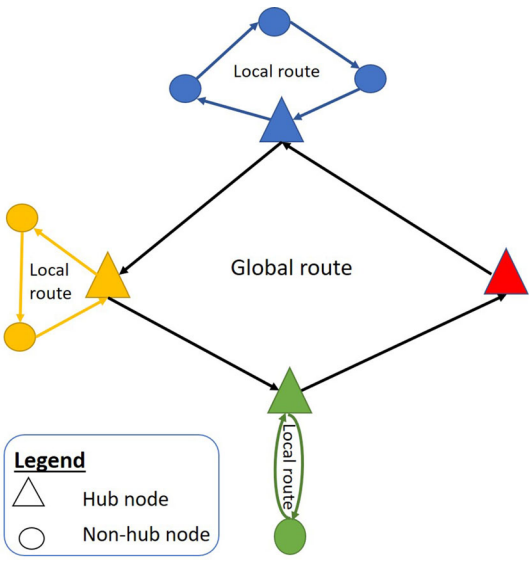

Fig. 1 A solution of a hub location routing problem

nodes assigned to the same hub by a route and linking hub nodes by a route so that the total routing cost is minimized. A constraint is placed on the maximum length of the local routes by specifying the maximum number of non-hub nodes allowed on a route. There is neither a capacity limitation imposed on the hubs nor a cost for hub installation. However, as in other hub location problems, a discount factor, $0 \leq \alpha \leq 1$, is applied when calculating the cost of the global route. The problem studied in this paper is referred to as the p-hub location routing problem (pHLRP).

The problem is formally defined on a graph $G = (V, E)$, where V is a given set of nodes and $E = \{(i, j)|i, j \in V\}$ is the set of edges that connect them. A non-negative cost c_e is associated with each edge $e \in E$ representing the routing cost along that edge. A feasible solution of the problem provides p nodes from set V to serve as hubs. Each of these hubs is the root point of a local route starting and ending at it. Each local route is composed of non-hub nodes, and may contain at most C nodes including the hub. If we denote the chosen hubs as $hub_1, hub_2, \ldots, hub_p$ and corresponding local routes as $r_1, r_2, \ldots, r_p$, where r_i is an ordered list of nodes, then $r_1 \cup r_2 \cup \cdots \cup r_p = V$, and $r_i \cap r_j = \emptyset, i \neq j$. In other words, each non-hub node is assigned to exactly one hub. If we suppose that $r_i = \{hub_i, n_1^i, \ldots, n_k^i, hub_i, \}$ then the cost of the route is calculated as

$$cost(r_i) = c_{(hub_i, n_1^i)} + \sum_{m=1}^{k-1} c_{(n_m^i, n_{m+1}^i)} + c_{(n_k^i, hub_i)}.$$

If we suppose that the cost of a tour linking the hubs is $cost(hub\_tour)$ then the overall cost of the solution defined by hubs $hub_1, hub_2, \ldots, hub_p$ and, corresponding local routes $r_1, r_2, \ldots, r_p$, is calculated as: $\sum_{i=1}^{p} cost(r_i) + \alpha cost(hub\_tour)$. Note that the length of route r_i, with the above notation is, $length_i = k + 1$. Note also that a route r_i may contain no non-hub nodes at all. In this case the length of the local route is 1, i.e., it contains the trip from hub hub_i to itself.

Applications of the pHLRP may be found, for example, in public and urban transportation systems such as subway and train lines. For example, in subway and train transportation, the inter-hub route consists of platforms that are used to consolidate passengers from many origins and allow their delivery to many destinations. In addition, some of these platforms are served by bus lines which are actually local routes that connect the passengers to the platforms. Another application of this problem arises when designing distributed data networks.

Regarding solution approaches for the problem, there are three heuristic methods available so far [20]: (i) a multi-start variable neighborhood descent, (ii) a Biased Random-Key Genetic Algorithm (BRKGA), and (iii) Local Solver version 6.0. According to the documentation, Local Solver combines different search techniques, constraint propagation, and linear mixed-integer as well as nonlinear programming. The multi-start variable neighborhood descent, in each iteration, applies a variable neighborhood descent (VND) on a random initial solution. Two different variable neighborhood descents (VND) are proposed in [20]. They both explore the same neighborhood structures defined by inserting non-hub nodes from one tour to another; exchanging two non-hub nodes; and changing a hub node. In addition both use the

Lin-Kernighan heuristic to improve tours. As search strategies, both VNDs use neither a first nor best improvement search strategy, but execute a random improvement move. However, they differ in the way the exploration of neighborhoods is organized. The first one uses a nested VND scheme, while the other performs consecutive nested Neighborhood Search (CNS). The multi-start VND based on nested VND is denoted by M-VND, while the one using CNS is denoted as M-CNS. Both M-VND and M-CNS exhibit better performance than BRKGA.

In this paper we propose a general variable neighborhood search (GVNS) based approach. GVNS also employs VND as a local search but does not use a completely random re-start to resolve local optimum traps as is the case in M-VND and M-CNS. Instead, it modifies the current solution to varying degrees in order to resolve a local optimum trap. In addition, our GVNS differs from M-VND and M-CNS in the VND procedure. The difference comes from the number of neighborhood structures, the way the neighborhoods are organized in VND, and the search strategy. More precisely, our VND explores seven neighborhood structures in a sequential way using the first improvement strategy. This is a completely different approach from that used in [20]. In addition, our VND does not use the Lin-Kernighan heuristic, but adopts instead other neighborhood structures applicable to the traveling salesman problem (TSP).

The proposed GVNS approach is validated on benchmark instances, and its performance demonstrates both a superior efficiency and effectiveness. Namely, our GVNS succeeds to establish a large number of new best-known solutions. In addition, it turns out to be very efficient in solving large problem instances in a short time. Namely, in almost all large instances it provides significantly better solutions than previous ones in a fraction of the time. Hence, the contributions of the paper may be summarized as:

- A new heuristic for pHLRP is proposed and tested. It is based on exploring seven neighborhoods within a GVNS framework.
- The proposed approach outperforms the previous ones from the literature to a great extent, particularly on large scale instances.
- 691 out of 912 best-known solutions are improved by the proposed GVNS heuristic;
- The algorithm is also very fast compared to the previous methods.

The rest of the paper is organized as follows. The next section details the different components of the new heuristic, and puts them all together within the GVNS framework. This is followed by extensive computational work, which includes a validation of the selected VND structure and parameter specifications, a comparison with the state-of-the-art, and a statistical analysis that confirms the superiority of the new heuristic. Lastly, we provide some conclusions and suggestions for future research.

2 Proposed variable neighborhood search approach

In this section, we provide a detailed description of our solution approach. The section starts by describing the solution representation and the procedure for generating an initial solution. After that, details are provided on the various neighborhood struc-

tures used to refine the solution. Finally, the main ingredients of the proposed general variable neighborhood search are described.

2.1 Solution representation

Any solution of the pHLRP may be represented by the three following components: set $H = \{hub_1, hub_2, \ldots, hub_p\}$ containing the p selected hub nodes, set $R = \{r_1, r_2, \ldots, r_p\}$ containing the local routes assigned to the hubs, and the global route denoted as hub_tour which connects the hubs from set H. In the established notation each local route r_i originates at hub node hub_i. In other words, each route r_i has the form $hub_i - n_1^i - \cdots - n_k^i - hub_i$ where $n_1^i - n_2^i \cdots - n_k^i$ is the ordered sequence of non-hub nodes visited along the route starting and ending at hub_i. Hence, the solution is represented as the triplet $S = (H, hub\_tour, R)$.

2.2 Initial solution

To construct an initial solution we proceed in the following way. From the set V we choose p nodes at random to constitute set H. After that, we assign nodes to hubs and form routes as follows. For each non-hub node the corresponding hub is chosen at random from the set H, and the node is added at the end of the route of the chosen hub. The assignment is made while respecting the imposed constraint on the route length. The hub_tour is constructed as a tour that connects hubs $hub_1, hub_2, \ldots, hub_p$ in this order. The outline of the procedure is given by Algorithm 1. Note that a feasible solution is generated in this way.

Algorithm 1: Generating an initial solution

```
   Function Initial_Solution(S);
 1 Choose randomly p nodes from V to constitute set
   H = {hub_1, hub_2, ..., hub_p} ;
 2 hub_tour ← hub_1 − hub_2 − ··· − hub_p − hub_1;   /*global tour*/;
 3 for i = 1 to p do
 4 |  r_i ← hub_i − hub_i;  /*creating local tours with just root hubs*/;
 5 |  length_i = 1;
   end
 6 for each n ∈ V − H do
 7 |  Select at random i ∈ [1, p] such that length_i < C;
 8 |  r_i ← hub_i − ··· − n − hub_i;  /*adding node at the end of current tour*/;
 9 |  length_i ← length_i + 1;
   end
10 return (H, hub_tour, R);
```

2.3 Neighborhood structures

The solution generated by procedure Algorithm 1 will likely be far from the optimal one. However, it may be improved by replacing a hub by a node which is not a hub, or moving a non-hub node from one route to another, or changing the order of visiting

nodes in the local routes and in the global route. Accordingly, we define the following neighborhoods:

- *Insert neighborhood* (N_{insert})- This neighborhood is based on a move that takes a single non-hub node from one local route and inserts it in another local route, while respecting the capacity of the local routes.
- *Exchange* ($N_{exhange}$) - This neighborhood is based on a move that exchanges two non-hub nodes from two different local routes. Note that these moves preserve the number of non-hub nodes in each local route.
- *Change hub* (N_{change_hub}) - This neighborhood is based on moves that replace a single hub node by a non-hub node from the local route originating at this hub.
- *LocalRoute neighborhoods* - The purpose of LocalRoute neighborhoods is to optimize the order of visiting non-hub nodes within local routes as well as hub nodes within the global route. Here we adopt some k-opt neighborhoods used for solving the Traveling Salesman Problem (TSP) and its variants (see e.g. [12,24,28]). They are: 1-opt, 2-opt and OR-opt-1. The 2-opt neighborhood is based on a move that inverts a part of the route. A simplified variant of the 2-opt move is the 1-opt move, which inverts a part of a tour consisting of 2 consecutive nodes. The OR-opt-1 neighborhood is based on an insertion move that relocates a node from one position to another. Two types of insertion move may be distinguished: forward and backward insertion moves depending on whether a node is moved forward or backward in the tour, respectively. 1-opt, 2-opt, OR-opt-1 forward and OR-opt-1 backward will be denoted as N_{1opt}, N_{2opt}, $N_{Oropt1for}$ and $N_{Oropt1back}$, respectively.

Note that all of these neighborhood allow feasible moves only, i.e., moves that respect the capacity constraints (i.e., do not violate the maximum length of a local route). The change in value of the objective function caused by performing any of these moves may be calculated in constant time O(1).

2.4 Variable Neighborhood descent

The above neighborhood structures do not necessarily yield the same local optima. That is, a local optimum with respect to one neighborhood structure is not necessarily a local optimum with respect to another. In order to efficiently exploit the defined neighborhood structures we embed them in a sequential variable neighborhood descent (VND) framework [7,13,24]. In this way, the VND takes a feasible solution at the input and as the output returns a solution which is a local optimum with respect to all employed neighborhood structures. The steps of the VND procedure are provided in Algorithm 2. As observed in previous works on VND (see e.g. [24]), the ordering of the neighborhood structures in the search as well as the search strategy may have a significant impact on the final performance of the VND procedure. Due to the large size of the neighborhood structures we selected a first improvement search strategy, i.e., as soon as an improving solution is detected in some neighborhood structure it replaces the incumbent solution, and the search is re-started from it. Regarding the order of neighborhoods within the VND, we then performed an empirical study to identify a most suitable one (see Sect. 3). This empirical study provided the final selected

order shown in line 1 of the algorithm. The statement `LocalSearch`(S, N_k) in line 3 denotes the local search with respect to the kth neighborhood in the ordered list N of neighborhoods.

Algorithm 2: Basic Sequential Variable Neighborhood Descent

```
   Function BVND(S);
1  N = {N_insert, N_exchange, N_change_hub, N_1opt, N_2opt, N_Oropt1back, N_Oropt1for};
2  k ← 1;
   while k ≤ 7 do
3  |  S' ← LocalSearch(S, N_k);
4  |  k ← k + 1;
   |  if S' better than S then
5  |  |  S ← S';
6  |  |  k ← 1;
   |  end
   end
7  return S
```

2.5 General variable neighborhood search

In this section we describe our *general variable neighborhood search* (GVNS) heuristic. Recall that GVNS is a variant of variable neighborhood search [13,26] that uses a *variable neighborhood descent* (VND) in the intensification phase (instead of a basic local search with a single neighborhood (see e.g., [2,25])) and the usual *shaking procedure* in the diversification phase. Our implementation employs the VND presented in Algorithm 2 for the intensification (or improvement) phase, while at the diversification phase we employ the shaking procedure presented in Algorithm 3. The shaking procedure moves the search to new regions of the solution space by relocating non-hub nodes to different routes and changing the order of visitation of a single local route. The improvement phase and the shaking procedure together with the neighborhood change step are executed alternately until a predefined stopping criterion is met. Here the stopping criterion is given by a limit on CPU time (T_{max}) (see line 10 in Algorithm 4). GVNS has an additional parameter k_{max} which denotes the maximum number of iterations (moves) that may be performed within a single shake (Algorithm 3) and which affects the level of diversification obtained by the shaking procedure. The parameter k_{max} is determined empirically in our study (see Sect. 3).

Algorithm 3: Shaking procedure

```
   Function shaking(S, k);
   N = {N_1, N_2, N_3} = {N_insert, N_Oropt1for, N_Oropt1back};
1  for i = 1 to k do
2  |  Select S'' in N_1(S) at random;
3  |  S ← S'';
   end
4  for i = 1 to k do
5  |  Select S'' in N_2(S) ∪ N_3(S) at random;
6  |  S ← S'';
   end
7  return S;
```

Algorithm 4: General VNS

```
Function GVNS(S, k_max, T_max);
1  S ← Initial_Solution(S);
2  repeat
3      k ← 1;
4      while k ≤ k_max do
5          S' ← shaking (S, k) ;
6          S'' ← BVND (S') ;
7          k ← k + 1;
8          if S'' is better then S then
9              S ← S''; k ← 1;
           end
       end
10     T ← CpuTime();
   until T > T_max;
11 Return S;
```

3 Computational results

In this section, we present the results obtained by our GVNS heuristic on benchmark instances from the literature. In addition, we present the results of preliminary testing used to determine the order of neighborhoods within the VND, as well as to determine an appropriate value of the parameter k_{max}. Also note that a CPU time limit T_{max} of 60 s is imposed on all runs of the GVNS heuristic.

The experiments are conducted on a machine with an Intel(R) Core(TM) i5-3470 CPU @ 3.20 GHz and 16 GB of RAM.

3.1 Benchmark instances

For testing purposes, the benchmark instances originally proposed in [20] are used. The benchmark set is derived from 76 TSPLIB instances by varying the value of parameter α and considering different scenarios on the tour length C, and the requested number of hubs p. More precisely, the value of the parameter α is taken from the set $\{0.2, 0.4, 0.6, 0.8\}$, while three scenarios are considered for values of C and p:

- *Scenario ST* $p = \lceil 0.2|V| \rceil$ and $C = \lceil \frac{|V|}{p} \rceil$. In this case the local routes are forced to be non-empty and tight.
- *Scenario SL* $p = \lceil 0.2|V| \rceil$ and $C = \lfloor 1.8 \lceil \frac{|V|}{p} \rceil \rfloor$. Here some local routes may be very large and some may be empty.
- *Scenario SQ* $p = C = \lceil \sqrt{|V|} \rceil$. Here local routes may be large or empty, but not as loose as scenario SL .

Hence, in total we have $76 \times 4 \times 3 = 912$ test instances. The instances with 100 nodes or less are classified as small, while the remaining ones are large.

3.2 Determining the best order of neighborhoods

The aim of this section is to determine the best order of neighborhoods inside our VND procedure. Due to the large number of neighborhood structures considered here, examining all possible orders would be far too time consuming. For this reason, we divide the neighborhood structures into two groups: hub-location neighborhoods and LocalRoute neighborhoods. Hub-location neighborhoods are those that concern the selection of hubs and node-hub assignments, while LocalRoute neighborhoods are those that "optimize" local tours. Having such a division of neighborhoods, we are now interested to determine the best order of neighborhood groups. The order of neighborhoods within the set of hub-location neighborhoods may be determined so that those based on changing node-hub allocations are examined first followed by those that change the hubs. Such a way of exploring neighborhoods is suggested for example in [16]. Hence, respecting the rule that the neighborhoods of smaller cardinality are examined first, we obtain the following order of neighborhoods within the hub-location group: $N_{hub} := \{N_{insert}, N_{exchange}, N_{change\_hub}\}$. In similar fashion, we order neighborhood structures within the LocalRoute neighborhood group as $N_{LocalRoute} := \{N_{1opt}, N_{2opt}, N_{Oropt1back}, N_{Oropt1for}\}$. To validate this order we note the following observations: the lowest cardinality neighborhood is examined first; the 2-opt neighborhood should be examined before the OR-opt neighborhoods (e.g., as noted in [24]); ORopt_backward should be examined before ORopt_forward (see e.g., [12,27,32]).

Now that the order of neighborhoods within each group is specified, we examine the best order of groups within the VND paradigm. Hence, we consider the following orders: 1) hub-location neighborhoods, LocalRoute neighborhoods; 2) LocalRoute neighborhoods; hub-location neighborhoods, and 3) hybrid order, where after each neighborhood in the hub-location group we apply the LocalRoute neighborhoods. To differentiate these three VNDs we designate them as:

- VND_1-order of neighborhoods: $N_{LocalRoute}, N_{hub}$;
- VND_2-order of neighborhoods: $N_{hub}, N_{LocalRoute}$;
- VND_3-order of neighborhoods: $N_{insert}, N_{LocalRoute}, N_{exchange}, N_{LocalRoute}, N_{change\_hub}, N_{LocalRoute}$;

To assess the performance of the above three VNDs, each one is executed 50000 times on a selected subset of instances, each time starting from a different initial solution generated at random as described in Algorithm 1. For testing purposes, we selected a subset of 240 instances with different topologies. This subset is based on the following 20 TSPlib instances: a280, li535, att532, bier127, brg180, ch130, d198, d493, fl417, gil262, gr137, gr202, gr229, gr431, kroA150, kroA200, lin318, pa561, pcb442, pr144. Each of these instances is considered under 3 different scenarios and 4 different values of the parameter α. For each VND, the average results taken over 20 instances, all using the same scenario and the same value of parameter α, are provided in Table 1. More precisely, we report the averages of the best, the average and the worst solution values (Columns '`Best`', '`Avg.`' and '`Worst`', respectively), as well as the average time-to-target (Columns '`time`') in seconds. The time-to-target represents

the CPU time consumed by a method to reach a final solution value reported at the output.

From the reported results we observe, that average times-to-target of all three VNDs are almost the same. Regarding solution quality, VND_1 exhibits the poorest performance. Comparing VND_2 and VND_3, we observe that in all cases but one (see the line corresponding to the scenario SL and $\alpha = 0.8$), VND_2 provides better (best) solutions. In addition, in 9 out of 12 cases, we see that the averages of best, average and worst solution values provided by VND_2 are better than the corresponding ones of VND_3. Consequently, VND_2 may be identified as the best VND. This further implies that the best order of neighborhoods is:
$N_{insert}, N_{exchange}, N_{change\_hub}, N_{1opt}, N_{2opt}, N_{Oropt1back}, N_{Oropt1for}$.

3.3 Assessing the value of parameter k_{max}

After detecting the best order, we needed to also tune parameter k_{max}, which determines the level of diversification that will be used within GVNS to resolve local optimum traps. It is known that for larger values of parameter k_{max}, the corresponding VNS heuristic tends to behave as a random multi-start heuristic (i.e., a heuristic that generates a completely random solution at each re-start that does not retain any features of the current best solution). In order to avoid this worst-case scenario, we want to keep the k_{max} value as small as possible while having at the same time high quality results. Since the number of local routes in a solution is expressed by the parameter p, we try to keep the number of local routes affected by the shaking procedure to be as small as possible by attempting the following settings: $k_{max} = \min(5, p)$, $k_{max} = \min(10, p)$ and $k_{max} = \min(20, p)$. In all testing, the time limit parameter T_{max} is set to 60 s as noted above, and for each choice of k_{max} GVNS is executed 20 times, each time starting from a different initial solution generated by Algorithm 1. For testing purposes, we used the same subset of 240 instances as described above. The average results taken over 20 instances, all using the same scenario and the same value of parameter α, are provided in Table 2. The column headings are defined in the same way as previously described.

From the reported results, it follows that for scenarios ST and SL the best choice is $k_{max} = \min(10, p)$, regardless of the value of α. This choice of the value of parameter k_{max}, yields better best and average solution values than the other choices. On the other hand for the SQ scenario the best choice turns out to be $k_{max} = \min(5, p)$. Hence, in all subsequent testing, we will use $k_{max} = \min(10, p)$ for the ST and SL scenarios, and $k_{max} = \min(5, p)$ for the scenario SQ. It should also be emphasized that regardless of the value of k_{max}, our GVNS exhibits a good stability with respect to time-to-target.

An interesting observation may be deduced after comparing the results reported in Table 1 (multi-start VND) and Table 2 (GVNS). Namely, in all test cases, the solutions obtained by the proposed GVNS are much better than the ones reached by the multi-start VND. Hence, it follows that the proposed shaking procedure resolves local optimum traps more effectively than the random re-start used within the multi-start VND.

Table 1 Examining the order of neighborhoods

Scenario	α	VND_1				VND_2				VND_3			
		Best	Avg	Worst	Time	Best	Avg	Worst	Time	Best	Avg	Worst	Time
ST	0.20	717,77.74	901, 67.84	121,935.08	0.02	712,26.03	869, 69.66	115,418.17	0.03	71,596.91	90, 075.72	120,590.16	0.02
	0.40	761,84.18	94, 547.77	126,247.10	0.02	75,677.18	91, 383.44	119,671.61	0.03	75,986.17	94, 452.92	124,801.37	0.02
	0.60	80,574.50	98, 927.70	130,559.35	0.02	80,083.37	95, 797.22	123,943.04	0.03	80,366.33	98, 830.98	129,046.59	0.02
	0.80	84,927.63	103, 307.63	134,954.50	0.02	84,465.91	100, 211.00	128,230.27	0.03	84,711.24	103, 210.72	133,378.87	0.02
SL	0.20	60,422.63	78, 516.64	111,671.84	0.02	59,315.53	77, 713.53	110,931.06	0.03	59,984.30	77, 446.11	108,335.95	0.02
	0.40	65,568.32	83, 159.29	115,908.96	0.02	64,614.65	82, 404.74	115,140.66	0.03	64,980.45	82, 101.35	112,598.39	0.02
	0.60	69,870.82	86, 755.15	116,882.66	0.02	69,815.15	87, 095.95	119,544.63	0.03	70,469.45	87, 801.93	120,190.97	0.02
	0.80	75,269.11	92, 444.57	124,510.93	0.02	74,752.71	91, 787.16	124,174.51	0.03	74,475.16	91, 410.92	121,290.36	0.02
SQ	0.20	79,918.11	110, 713.92	152,816.48	0.02	74,942.70	100, 979.95	137,259.14	0.02	76558.87	104, 853.10	141,014.93	0.02
	0.40	82,031.57	112, 570.34	154,382.57	0.02	77,059.41	102, 651.91	138,,688.81	0.02	78,713.62	106, 768.35	142,821.72	0.02
	0.60	84,072.92	114, 426.76	156,062.59	0.02	79,109.89	104, 323.87	140,214.11	0.02	80,863.38	108, 684.24	144,937.46	0.02
	0.80	86,086.51	116, 283.17	157,859.40	0.02	81,149.12	105, 995.84	141,931.68	0.02	82,973.38	110, 599.89	147,089.56	0.02

Table 2 Examining the best value of parameter k_{max}

Scenario	α	$k_{max} = 5$				$k_{max} = 10$				$k_{max} = 20$			
		Best	Avg	Worst	Time	Best	Avg	Worst	Time	Best	Avg	Worst	Time
ST	0.20	63,444.90	64,516.38	65,898.29	41.42	63,422.17	64,354.05	65,521.38	39.36	63,574.80	64,810.37	67,128.20	42.75
	0.40	67,973.36	69,001.99	70,575.70	39.47	67,943.81	68,914.67	71,243.84	38.44	68,028.15	69091.72	70,778.23	42.39
	0.60	72,440.34	73,519.15	75,503.06	40.30	72,015.45	73,103.02	75,230.09	39.70	72,136.86	73,351.96	75,507.71	41.11
	0.80	76,712.72	77,885.95	79,419.65	40.38	76,075.13	77,244.80	78,799.40	39.20	76,374.06	77,503.15	79,770.73	41.88
SL	0.20	47284.34	49,443.28	52,862.24	47.63	46,265.19	48,309.87	50,928.14	50.77	46,986.67	49,377.72	52,,330.98	51.88
	0.40	52,954.10	54,894.23	56,726.35	47.58	52,067.85	53,997.06	56,575.05	49.34	52,732.18	54,662.34	56,685.87	50.99
	0.60	58,391.70	59,947.73	62,288.71	47.17	57,673.53	59,135.34	61,871.95	48.97	58,287.43	59,865.25	62,368.40	50.62
	0.80	62,998.77	64,620.66	67,177.38	45.68	62,163.45	63,817.90	65,807.28	47.71	63,043.72	64,540.63	66,923.07	50.18
SQ	0.20	59,,605.44	61,784.58	65,560.35	37.00	60,132.44	63,724.50	68394.79	40.54	60,104.31	63721.53	70,226.27	42.34
	0.40	61,436.86	63,965.32	68,382.95	37.41	61,926.98	65,749.88	70,523.27	37.91	62,090.51	65,803.11	72,761.95	41.72
	0.60	63,641.54	66,262.42	70,644.02	38.42	64,927.75	67,914.76	71,865.33	39.42	64,542.76	68,476.99	72,641.26	41.10
	0.80	65,291.53	68,563.55	73,636.96	37.69	66,676.31	70,068.34	74,933.92	37.15	67,530.69	70,917.50	77,291.06	40.96

3.4 Comparison with the state-of-the-art

In this section we compare our GVNS against the state-of-the-art heuristics presented in [20]: M-VND, M-CNS, and LocalSolver. In all tests the parameters of our GVNS are set as previously described, i.e., $T_{max} = 60\,\text{s}$, $k_{max} = \min(10, p)$ on scenarios ST and SL, and $k_{max} = \min(5, p)$ on scenario SQ. As an improvement routine, VND_2 is used. On each of the 912 instances tested GVNS is executed 20 times, each time starting from a different initial solution generated by Algorithm 1.

The detailed comparison may be found at the following webpage: http://www.mi.sanu.ac.rs/~nenad/p-hubLR/, while here we present the summary results. All instances of the same size type (small or large), tested under the same scenario (ST, SL or SQ) and the same value of the parameter $\alpha \in \{0.2, 0.4, 0.6, 0.8\}$ are considered as a test case. Hence, for each test case, we report the average results over all instances in that test case. Table 3 reports the results on small instances, while Table 4 provides results on large instances. The column headings are defined as follows. Columns '`Scenario`' and 'α' provide information on the scenario considered and the value of parameter α, respectively. The next column '`Best Known`' provides the average of best-known solution values obtained from the literature for each test case. After that, in the next columns, for each of methods M-VND, M-CNS, and LocalSolver (LS), we report the average percentage deviations of their solution values from the best-known solution values and the average time-to-target. For our GVNS, we report the averages of the best and average solution values on each test case, the average time-to-target and the average percentage deviations of the best found solution values from the current best known values. For each method, the percentage deviation is calculated as:

$$\frac{value - best\_known}{best\_known} \times 100,$$

where *value* stands for the solution value found by the considered method and *best_known* is the best-known solution value reported in the literature [20]. Note that the percentage deviations for M-VND, M-CNS, and LocalSolver are taken directly from [20] as well as all other data related to these methods.

Table 5 provides the following information for each test case: number of instances where the proposed GVNS establishes a new best-known solution (columns '`Better`'); number of instances where it reaches the previous best-known solution (columns '`Same`'); and number of instances where it fails to improve or reach the previous best-known solution (columns '`Worse`').

From the results reported on small size instances in Table 3, we observe that our GVNS provides very good solution values in short time. On all 12 test cases, the averages of best and average solution values are less than the averages of best-known values. The previous best-known solution values are improved up to 0.73%. Just on two test cases (SL, 0.20 and SL, 0.80), our GVNS provides higher percentage deviation from the best known solution values than LS. In the remaining 10 cases our GVNS not only provides less percentage deviation than LS but also improves previous best-known solution values. The total number of new best-known solutions established by our GVNS is 135 (see Table 5). On 138 instances, our GVNS matches the previous

Table 3 Comparison on small instances

Scenario	α	Best	M-VND		M-CNS		LS		GVNS			
		known	% dev.	Time	% dev.	Time	% dev.	Time	Best	Avg.	Time	% dev.
ST	0.2	15,424.81	1.56	149.11	1.67	192.61	0.00	734.82	15,342.96	15,354.13	2.76	−0.73
	0.4	16,486.64	1.46	220.79	1.77	207.43	0.00	748.96	16,451.74	16,467.91	3.15	−0.17
	0.6	17,597.70	1.55	194.46	1.31	134.57	0.00	563.75	17,503.14	17,516.94	3.42	−0.69
	0.8	18,646.88	1.21	155.46	1.32	127.04	0.00	723.64	18,518.44	18,537.35	3.10	−0.81
SL	0.2	12343.49	2.46	462.86	4.08	225.18	0.00	1515.07	12,049.44	12,191.94	14.58	0.24
	0.4	13,785.21	1.99	560.75	2.33	273.18	0.01	1305.79	13,429.93	13,574.40	12.45	−0.15
	0.6	14,961.04	2.01	437.29	1.54	289.28	0.02	1052.25	14,665.06	14,803.51	12.78	−0.18
	0.8	15,842.44	2.53	507.70	1.93	287.74	0.04	1365.44	15,696.35	15,821.68	12.13	0.11
SQ	0.2	15253.10	1.44	236.89	1.63	317.18	0.00	772.61	15129.86	15,183.70	6.21	−0.48
	0.4	16,090.39	1.23	299.04	1.49	310.82	0.00	1037.86	15,928.77	15,998.14	6.02	−0.58
	0.6	16,899.24	0.99	253.39	1.14	276.96	0.99	253.39	16,704.71	16,772.26	5.76	−0.63
	0.8	17,631.69	0.94	234.07	1.14	220.25	0.00	467.57	17,433.16	17,489.68	5.58	−0.59

Table 4 Comparison on large instances

Scenario	α	Best known	M-VND		M-CNS		LS		GVNS			
			% dev.	Time	% dev.	Time	% dev.	Time	Best	Avg.	Time	% dev.
ST	0.2	66,282.29	10.91	1324.58	0.00	2571.19	0.00	2571.19	61,865.73	62717.09	39.56	−5.32
	0.4	72,087.65	8.83	1419.98	0.07	2183.58	0.07	2183.58	66,343.12	67,237.63	35.78	−6.26
	0.6	77,574.11	7.21	1399.13	0.34	2412.44	0.34	2412.44	70,557.37	71,585.04	36.98	−7.02
	0.8	82,293.78	6.32	1160.09	0.91	2325.85	0.91	2325.85	74,928.83	75,810.64	36.52	−7.23
SL	0.2	50,604.37	9.06	1724.75	0.09	2331.94	0.09	2331.94	46,400.44	47,844.43	47.60	−7.93
	0.4	58,373.36	4.55	1637.38	2.57	2403.27	2.57	2403.27	51,687.82	53,027.42	46.72	−9.49
	0.6	61,881.48	3.25	1804.33	4.51	2669.48	4.51	2669.48	56,609.40	57,855.23	45.84	−6.52
	0.8	68,767.48	3.14	1783.71	5.15	2780.52	5.15	2780.52	61,074.15	62,364.64	45.19	−9.52
SQ	0.2	63877.36	14.51	1544.54	0.01	2328.21	0.01	2328.21	57,694.33	60,465.82	37.04	−6.92
	0.4	69,004.53	9.67	1657.96	0.00	2130.04	0.00	2130.04	59,623.69	62,730.97	36.18	−9.71
	0.6	73,234.89	8.21	1768.04	0.18	2279.96	0.18	2279.96	62,408.70	65,183.88	37.13	−10.42
	0.8	75,046.66	5.96	1535.13	0.30	2538.75	0.30	2538.75	64,741.78	67,600.53	36.40	−11.02

Table 5 Number of instances where GVNS provides a better, same or worse solution than best-known from the literature

Scenario	α	Small instances			Large instances		
		Better	Same	Worse	Better	Same	Worse
ST	0.2	11	16	1	47	0	1
	0.4	9	19	0	47	0	1
	0.6	10	17	1	47	0	1
	0.8	12	16	0	46	0	2
SL	0.2	12	0	16	47	0	1
	0.4	13	1	14	47	0	1
	0.6	12	3	13	43	0	5
	0.8	12	2	14	47	0	1
SQ	0.2	11	14	3	45	0	3
	0.4	13	15	0	46	0	2
	0.6	10	17	1	47	0	1
	0.8	10	18	0	47	0	1
Total		135	138	63	556	0	20

best-known solutions, while on 63 instances it fails to improve or match the previous best-known solutions.

Comparing the results on large scale instances, it follows that our GVNS exhibits better performance than any other solution approach. On every instance it either improves or matches the previous best-known solution. The average percentage improvement is in the range of 5.32 and 11%. It should be emphasized that our GVNS consumes a lot less CPU time-to-target than any other method. M-VND has very high percentage deviation from the previous best-known solution values, reaching as high as 14.51%. M-CNS and LS exhibit better performance than M-VND, but still worse than GVNS. For example, on test case SL 0.60, where percentage deviations of M-CNS and LS are both 4.51%, our GVNS improves best-known solutions by 6.52%. Moreover, our GVNS improves the previous best-known solutions on 556 out of 576 test instances (see Table 5). It should be emphasized that, as reported in [20], M-CNS and LS exhibit the same performance on all large instances.

Overall, we see that GVNS improves best known solution values on 691 out of 912 instances; matches the previous best-known solution values on 138 instances, while on only 83 instances it neither improves nor reaches the previous best-known solution values. Thus as of now, 829 out of 912 best-known solutions are attributed to the proposed GVNS, i.e., around 90% of the current best-known solutions. In addition, it follows that the proposed GVNS is highly suitable for solving large scale instances in a short time by providing high quality solutions within 60 s. This further implies that our GVNS successfully responds to the main requirement of a heuristic approach, that is, to provide high quality solutions in short time.

Table 6 p-values of the Wilcoxon signed-rank test

Scenario	α	Small instances			Large instances		
		M-VND	M-CNS	LS	M-VND	M-CNS	LS
ST	0.2	5.35E-04	5.35E-04	1.22E-02	1.63E-09	3.06E-09	3.06E-09
	0.4	1.22E-04	2.93E-04	3.90E-03	1.63E-09	5.03E-09	5.03E-09
	0.6	6.10E-05	2.92E-04	2.00E-03	1.63E-09	4.18E-09	4.18E-09
	0.8	2.93E-04	6.10E-05	4.88E-04	2.40E-09	7.56E-09	7.56E-09
SL	0.2	3.97E-02	2.50E-03	9.23E-01	1.63E-09	2.56E-08	2.56E-08
	0.4	3.06E-02	7.40E-03	7.19E-01	1.63E-09	1.41E-08	1.41E-08
	0.6	9.40E-03	1.28E-02	7.37E-01	1.97E-09	4.05E-07	4.05E-07
	0.8	7.40E-03	1.99E-02	5.01E-01	1.63E-09	9.26E-09	9.26E-09
SQ	0.2	1.60E-03	1.00E-03	2.45E-02	1.63E-09	4.08E-08	4.08E-08
	0.4	1.22E-04	1.32E-04	2.44E-04	1.63E-09	1.41E-08	1.41E-08
	0.6	3.05E-04	3.05E-04	3.05E-04	1.63E-09	8.72E-09	8.72E-09
	0.8	4.38E-04	1.22E-04	2.00E-03	1.63E-09	1.25E-08	1.25E-08

3.5 Statistical tests

The aim of this section is to determine if the difference in performance between our GVNS and each other method in the comparison is statistically significant or not. In this regard, the Wilcoxon signed-rank test is performed with a significance level of 0.05. The outcome of the tests is provided in Table 6. For each test case, the corresponding row provides p-values obtained after applying the Wilcoxon signed-rank test on the results found by the proposed GVNS and the methods shown in the column headings. A p-value less than 0.05 indicates that there is a significant difference between the two methods, otherwise there is insufficient evidence. From the table we see that on large instances all p-values are orders of magnitude less than 0.05, which implies that there is a significant difference between the proposed GVNS and each of the previously proposed methods. Since, on large instances, the proposed GVNS provides better solution values than any existing method (see Table 4), it may be inferred that the GVNS is the superior method on large size instances, in terms of the solution quality. Similar conclusions may be drawn when comparing GVNS with M-VND and M-CNS on small instances. When comparing the GVNS and LS on small instances, the previous conclusions are applicable on all test cases except for small instances and scenario SL, where no significant differences between GVNS and LS are observed.

4 Conclusions

In this paper an established hub location routing problem is studied, and an efficient solution approach is proposed. The new solution approach follows the general variable neighborhood search framework where in our case seven different neighborhood structures are identified to exploit the nature of the problem. These seven neighborhoods

are embedded in a sequential variable neighborhood descent scheme in an efficient and effective ordering determined from a detailed empirical study. In addition, all other parameters and components of our approach are carefully tailored with the aim to obtain high-quality solutions in a short time. Comparing our heuristic with the state-of-the-art approaches reveals that we succeeded to do so to a great extent. Our heuristic established 691 new best-known solutions on a benchmark set of 912 instances while consuming orders of magnitude less CPU time than all competitors. In addition, 829 out of 912 current best-known solutions can be attributed to the proposed GVNS. The statistical tests we conducted strongly confirm the superiority of our approach over the previous ones.

The SL small-sized instances with poor GVNS performance should be the subject of future work in order to detect possible anomalies in the solution space of these instances. As additional future research directions, we suggest the hybridization of the proposed solution method with some exact approaches to hopefully obtain guaranteed optimal solutions, as well as extending the study to other hub location routing problems. Some variants to be studied include for example, capacitated variants of the current problem, with capacities on the edges or at the hubs, allowing the assignment of non-hub nodes to more than one hub, and allowing for different configurations such as a tree structure for inter-hub connections.

In addition, some recent works use an adaptive search strategy, which combines first and best improvement strategies. More specifically, the best improvement is selected for small- and medium-sized instances, and the first improvement for the solution of large problem instances [18,19]. Therefore, this approach may be also a promising future work direction. Moreover, in [19] new adaptive shaking approaches can be found, which may further enhance the efficiency of GVNS. Hence, this is one more possible research direction. Another possible research direction may be to examine the use of an adaptive neighborhoods' ordering mechanism of neighborhood structures as was done in [33], for example.

Acknowledgements This research is supported by ELSAT2020 (Eco-mobilité, Logistique, Scurité and Adaptabilité des Transports l'horizon 2020) Project co-financed by European Union, France and Région Hauts-de-France.

References

1. Alumur, S., Kara, B.Y.: Network hub location problems: the state of the art. Eur. J. Oper. Res. **190**(1), 1–21 (2008)
2. Amaldass, N.I.L., Lucas, C., Mladenovic, N.: Variable neighbourhood search for financial derivative problem. Yugosl. J. Oper. Res. **29**(3), 359–373 (2019)
3. Brimberg, J., Mišković, S., Todosijević, R., Urošević, D.: The uncapacitated r-allocation p-hub center problem. Int. Trans. Oper. Res. (2020). https://doi.org/10.1111/itor.12801
4. Brimberg, J., Mladenović, N., Todosijević, R., Urošević, D.: A basic variable neighborhood search heuristic for the uncapacitated multiple allocation p-hub center problem. Optim. Lett. **11**(2), 313–327 (2017)
5. Brimberg, J., Mladenović, N., Todosijević, R., Urošević, D.: General variable neighborhood search for the uncapacitated single allocation p-hub center problem. Optim. Lett. **11**(2), 377–388 (2017)
6. Brimberg, J., Mladenović, N., Todosijević, R., Urošević, D.: A non-triangular hub location problem. Optim. Lett. (2019). https://doi.org/10.1007/s11590-019-01392-2

7. Duarte, A., Sánchez-Oro, J., Mladenović, N., Todosijević, R.: Variable neighborhood descent. In: Martí, R., Pardalos, P.M., Resende, M.G.C. (eds.) Handbook of Heuristics, pp. 341–367. Springer, Cham (2018)
8. Ernst, A.T., Hamacher, H., Jiang, H., Krishnamoorthy, M., Woeginger, G.: Uncapacitated single and multiple allocation p-hub center problems. Comput. Oper. Res. **36**(7), 2230–2241 (2009)
9. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Baatar, H.: Reformulations and computational results for uncapacitated single and multiple allocation hub covering problems. Work. Pap. Ser. **1**, 1–18 (2011)
10. Farahani, R.Z., Hekmatfar, M., Arabani, A.B., Nikbakhsh, E.: Hub location problems: a review of models, classification, solution techniques, and applications. Comput. Ind. Eng. **64**(4), 1096–1109 (2013)
11. Gavriliouk, E.O.: Aggregation in hub location problems. Comput. Oper. Res. **36**(12), 3136–3142 (2009)
12. Gelareh, S., Gendron, B., Hanafi, S., Monemi, R.N., Todosijević, R.: The selective traveling salesman problem with draft limits. J. Heuristics **26**, 339–352 (2020)
13. Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S.: Variable neighborhood search: basics and variants. EURO J. Comput. Optim. **5**(3), 423–454 (2017)
14. Hoff, A., Peiró, J., Corberán, Á., Martí, R.: Heuristics for the capacitated modular hub location problem. Comput. Oper. Res. **86**, 94–109 (2017)
15. Hwang, Y.H., Lee, Y.H.: Uncapacitated single allocation p-hub maximal covering problem. Comput. Ind. Eng. **63**(2), 382–389 (2012)
16. Ilić, A., Urošević, D., Brimberg, J., Mladenović, N.: A general variable neighborhood search for solving the uncapacitated single allocation p-hub median problem. Eur. J. Oper. Res. **206**(2), 289–300 (2010)
17. Janković, O., Mišković, S., Stanimirović, Z., Todosijević, R.: Novel formulations and vns-based heuristics for single and multiple allocation p-hub maximal covering problems. Ann. Oper. Res. **259**(1–2), 191–216 (2017)
18. Karakostas, P., Sifaleras, A., Georgiadis, M.C.: A general variable neighborhood search-based solution approach for the location-inventory-routing problem with distribution outsourcing. Comput. Chem. Eng. **126**, 263–279 (2019)
19. Karakostas, P., Sifaleras, A., Georgiadis, M.C.: Adaptive variable neighborhood search solution methods for the fleet size and mix pollution location-inventory-routing problem. Expert Syst. Appl. **153**, 113444 (2020)
20. Lopes, M.C., de Andrade, C.E., de Queiroz, T.A., Resende, M.G., Miyazawa, F.K.: Heuristics for a hub location-routing problem. Networks **68**(1), 54–90 (2016)
21. Martí, R., Corberán, Á., Peiró, J.: Scatter search for an uncapacitated p-hub median problem. Comput. Oper. Res. **58**, 53–66 (2015)
22. Meyer, T., Ernst, A.T., Krishnamoorthy, M.: A 2-phase algorithm for solving the single allocation p-hub center problem. Comput. Oper. Res. **36**(12), 3143–3151 (2009)
23. Mikić, M., Todosijević, R., Urošević, D.: Less is more: general variable neighborhood search for the capacitated modular hub location problem. Comput. Oper. Res. **110**, 101–115 (2019)
24. Mjirda, A., Todosijević, R., Hanafi, S., Hansen, P., Mladenović, N.: Sequential variable neighborhood descent variants: an empirical study on the traveling salesman problem. Int. Trans. Oper. Res. **24**(3), 615–633 (2017)
25. Mladenović, N., Alkandari, A., Pei, J., Todosijević, R., Pardalos, P.M.: Less is more approach: basic variable neighborhood search for the obnoxious p-median problem. Int. Trans. Oper. Res. **27**(1), 480–493 (2020)
26. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
27. Mladenović, N., Todosijević, R., Urošević, D.: An efficient gvns for solving traveling salesman problem with time windows. Electron. Notes Discrete Math. **39**, 83–90 (2012)
28. Mladenović, N., Todosijević, R., Urošević, D.: An efficient general variable neighborhood search for large travelling salesman problem with time windows. Yugosl. J. Oper. Res. **23**(1), 19–30 (2013)
29. Peiró, J., Corberán, A., Martí, R.: Grasp for the uncapacitated r-allocation p-hub median problem. Comput. Oper. Res. **43**(1), 50–60 (2014)
30. Peker, M., Kara, B.Y.: The p-hub maximal covering problem and extensions for gradual decay functions. Omega **54**, 158–172 (2015)
31. Talbi, E.G., Todosijević, R.: The robust uncapacitated multiple allocation p-hub median problem. Comput. Ind. Eng. **110**, 322–332 (2017)

32. Todosijević, R., Mjirda, A., Mladenović, M., Hanafi, S., Gendron, B.: A general variable neighborhood search variants for the travelling salesman problem with draft limits. Optim. Lett. **11**(6), 1047–1056 (2017)
33. Todosijević, R., Mladenović, M., Hanafi, S., Mladenović, N., Crévits, I.: Adaptive general variable neighborhood search heuristics for solving the unit commitment problem. Int. J. Electr. Power Energy Syst. **78**, 873–883 (2016)
34. Todosijević, R., Urošević, D., Mladenović, N., Hanafi, S.: A geneal vaiable neighbohood seach fo solving the uncapacitated r-allocation p-hub median poblem. Optim. Lett. **11**(6), 1109–1121 (2017)
35. Weng, K., Yang, C., Ma, Y.: Two artificial intelligence heuristics in solving multiple allocation hub maximal covering problem. In: Huang, D.S., Li, K., Irwin, G.W. (eds.) Intelligent Computing, pp. 737–744. Springer, Berlin, Heidelberg (2006)
36. Yaman, H.: Allocation strategies in hub networks. Eur. J. Oper. Res. **211**(3), 442–451 (2011)

Affiliations

Mustapha Ratli[1] · Dragan Urošević[2] · Abdessamad Ait El Cadi[1] · Jack Brimberg[3] · Nenad Mladenović[4] · Raca Todosijević[1]

✉ Raca Todosijević
raca.todosijevic@uphf.fr

Mustapha Ratli
mustapha.ratli@uphf.fr

Dragan Urošević
draganu@mi.sanu.ac.rs

Abdessamad Ait El Cadi
abdessamad.aitelcadi@uphf.fr

Jack Brimberg
Jack.Brimberg@rmc.ca

Nenad Mladenović
nenadmladenovic12@gmail.com

1 LAMIH UMR CNRS 8201-Université Polytechnique Hauts-de-France, 59313 Valenciennes Cedex 9, France

2 Mathematical Institute, Serbian Academy of Sciences and Arts, Belgrade, Serbia

3 Royal Military College of Canada, Kingston, ON, Canada

4 Khalifa University, Abu Dhabi, UAE