Les Cahiers du GERAD ISSN: 0711–2440

General variable neighborhood search for the uncapacitated single allocation p-hub center problem

J. Brimberg, N. Mladenović,
R. Todosijević, D. Urošević

G–2015–42

April 2015

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

General variable neighborhood search for the uncapacitated single allocation p-hub center problem

Jack Brimberg [a,c]

Nenad Mladenović [b,c]

Raca Todosijević [b]

Dragan Urošević [d]

[a] *Department of Mathematics and Computer Science, The Royal Military College of Canada, Kingston (Ontario) Canada*

[b] *LAMIH – Université de Valenciennes et du Hainaut-Cambrésis, 59313 Valenciennes Cedex 9, France*

[c] *GERAD, HEC Montréal, Montréal (Québec) Canada, H3T 2A7*

[d] *Mathematical Institute, Serbian Academy of Sciences and Arts, 11000 Belgrade, Serbia*

jack.brimberg@rmc.ca
nenad.mladenovic@gerad.ca
racatodosijevic@gmail.com
draganu@turing.mi.sanu.ac.rs

April 2015

Les Cahiers du GERAD

G–2015–42

Abstract: In this paper we propose a general variable neighborhood search heuristic for solving the uncapacitated single allocation p-hub center problem (USApHCP). For the local search step we develop a nested variable neighborhood descent strategy. The proposed approach is tested on benchmark instances from the literature and found to outperform the state-of-the-art heuristic based on ant colony optimization. We also test our heuristic on large scale instances that were not previously considered as test instances for the USApHCP. Moreover, exact solutions were reached by our GVNS for all instances where optimal solutions are known.

Acknowledgments: This work was conducted at the National Research University Higher School of Economics, Nizhni Novgorod, Russia and supported by RSF.

1 Introduction

In many airline, transportation, postal and telecommunication networks, establishing direct transport between origin and destination is too expensive. For that reason, special nodes, called hubs, are used to enable flow from the origin to the destination. The main role of hubs is to consolidate, switch and sort flows. In such networks any two non-hub nodes are connected using hub nodes. Hence, transportation among non-hub nodes is possible only via the hub nodes assigned to them. The application of hub networks include airline passenger travel [2], telecommunication systems [5], postal networks [7] and so on (see e.g. [1, 3]).

The most common optimization problems of this type are the hub median and hub center problems. The hub median problem consists of locating a set of hubs, assigning non-hub nodes to hubs and finding routing of flow between origin-destination pairs such that the total transportation cost is minimized. This problem is typical in airline transportation and in telecommunication systems. However, the main drawback is that the worst-case origin-destination distance (cost) is neglected and sometimes it could be unacceptably large (especially for the delivery of perishable or time sensitive items). The objective of a hub center problem is to minimize the maximum distance (cost) between origin-destination pairs. Therefore, in some cases, the hub center problem is a better option than the hub median problem.

In this paper we study the uncapacitated single allocation p-hub center problem (USApHCP). The problem is defined on a complete symmetric graph $G = (N, E)$, where $N = \{1, 2, ..., n\}$ is a set of nodes, while $E = \{(i, j)|i, j \in N\}$ is a set of arcs. Each arc (i, j) has infinite capacity and cost c_{ij} satisfying the triangle inequality. In the USApHCP, p hubs must be selected among these n locations, where p is given. Let H be the set of hubs. Every non-hub node $i \in N \setminus H$ is assigned to a single hub $h_i \in H$. Direct transportation between non-hub nodes is not allowed. The transportation cost from node $i \in N$, assigned to hub h_i, to node $j \in N$, assigned to hub h_j, is calculated as:

$$d_{ij} = \gamma c_{ih_i} + \alpha c_{h_i h_j} + \delta c_{h_j j}$$

where parameters γ, α and δ are unit rates for collection (origin?-hub), transfer (hub?-hub) and distribution (hub?-destination), respectively. Generally, α is used as a discount factor to provide reduced unit costs on arcs between hubs, so $\alpha < \gamma$ and $\alpha < \delta$. The objective of the USApHCP is to minimize the maximum cost between origin-destination pairs.

Mathematically, the USApHCP may be modeled in terms of the following variables [6]. Let x_{ik} be a binary variable such that $x_{ik} = 1$ if node i is allocated to hub k, and $x_{kk} = 1$ for each hub node k that is selected. Additionally, let r_k be a non-negative variable representing the maximum distance (cost) between hub k and the nodes that are allocated to it. Then, the so-called radius formulation of USApHCP is given as [6]:

$$\min \ z \tag{1}$$

subject to:

$$\sum_{k \in N} x_{ik} = 1, \qquad i \in N \tag{2}$$

$$x_{ik} \leq x_{kk}, \qquad i, k \in N \tag{3}$$

$$\sum_{k \in N} x_{kk} = p \tag{4}$$

$$r_k \geq c_{ik} x_{ik}, \qquad i, k \in N \tag{5}$$

$$z \geq \gamma r_k + \alpha c_{k\ell} + \delta r_\ell, \qquad k, l \in N, k \geq \ell \tag{6}$$

$$x_{ik} \in \{0, 1\}, \qquad i, k \in N \tag{7}$$

Constraints (2) ensure that each node is assigned exactly to one hub node, while constraints (3) guarantee that each non-hub node is assigned only to hub nodes. Constraint (4) ensures that exactly p hubs are located.

Constraint (5) defines the value of the radius of a hub, i.e., the greatest distance (cost) between this hub and any node allocated to it. Constraint (6) ensures that the objective function value is not less than the travel cost between any pair of nodes allocated to hubs k and ℓ, since r_k and r_ℓ correspond to the longest distance between a node allocated to k and ℓ, respectively.

The USApHCP is NP-hard [6, 15]. Even in the case that hub locations are known, its subproblem of allocating nodes to hubs, called the hub center single allocation problem (HCSAP), turns out to be NP-hard [6]. The USApHCP was introduced by O'Kelly and Miller [17] in 1991. In 1994, Campbell [4] formulated the problem as a quadratic integer program. Kara and Tansel [15] linearized the formulation in 2000. Hamacher and Meyer [9] proposed an approach that combines algorithms for solving hub covering problems and a binary search procedure in 2006. In 2009 Ernst et al. [6] proposed several very efficient integer programming formulations of USApHCP, using the concept of a hub radius. Gavriliouk [8] developed a heuristic based on aggregation for the USApHCP. Meyer et al. [16] proposed an exact 2-phase algorithm. In the first phase a set of potential optimal hub combinations is computed using a shortest path based branch and bound, while in the second phase the allocation is determined exactly using a reduced formulation. Additionally, they developed an ant colony based heuristic for the USApHCP.

In this paper we propose a heuristic based on Variable Neighborhood Search (VNS) [12] that uses Nested Variable Neighborhood Descent (Nest-VND) [14, 18] as a local search. Additionally, we propose several neighborhood structures for the USApHCP as well as a way to efficiently explore them. The merit of this approach is disclosed on benchmark instances from the literature where we obtain better results than the state-of-the-art heuristic based on ant colony optimization. Additionally, we test our heuristic on large scale instances that were not previously considered as test instances for the USApHCP. Results obtained on those instances are significantly better than those obtained by a CPLEX 12.6 mip solver, regarding both solution quality and consumed CPU time.

The rest of the paper is organized as follows. In the next section we give details of our heuristic for solving the USApHCP. We first describe a procedure for creating an initial solution; then we present the neighborhood structures and how we combined them into a Nest-VND. At the end of Section 2 we present the steps of the proposed approach. In Section 3 we present computational results on benchmark instances from the literature, while Section 4 concludes the paper.

2 Solution approach

Solution representation. Solving the USApHCP concerns finding an optimal set of p hubs as well as an optimal allocation of nodes to hubs. Therefore, a solution of the USApHCP may be represented as $S = (H, A)$, where H is a set containing p hubs, i.e. $H = \{h_1, h_2, \ldots, h_p\}$, while $A = (a_k),\ k = 1, \ldots, p$, is a matrix where each row k contains the nodes allocated to the hub k. The set of such nodes is denoted as A_k. In addition, we use an array r to represent the maximum distance (cost) between hub h_k and the nodes that are allocated to it, i.e.,

$$r_k = \max_{i \in A_k} \{c_{ih_k}\},\ k = 1, \ldots, p. \tag{8}$$

As will be shown in the remainder of this section, the purpose of this array is to simplify the updating of the objective function if a hub node is replaced by another non-hub node or some node is re-allocated to another hub node. Our data structure yields the following proposition.

Property 2.1 *The objective function value of a solution $S = (H, A)$ may be calculated in $O(p^2)$ using the aforementioned data structure.*

Proof. The objective value of solution $S = (H, A)$ is given by $f(S) = \max_{h_k, h_l \in H, i \in A_k, j \in A_l} (\gamma c_{ih_k} + \alpha c_{h_k h_l} + \delta c_{h_l j})$. However, this equation may be rewritten as $f(S) = \max_{h_k, h_l \in H} (\max_{i \in A_k} \gamma c_{ih_k} + \alpha c_{h_k h_l} + \max_{j \in A_l} \delta c_{h_l j})$. Taking into account equation (8), we further obtain $f(S) = \max_{h_k, h_l \in H} (\gamma r_k + \alpha c_{h_k h_l} + \delta r_l)$. Now, it is obvious that the last equation requires $O(p^2)$ operations to calculate the objective value of solution S. □

Initial solution. The initial solution of the USApHCP problem is built in a greedy way as follows. Firstly, the set H of p hubs is built. Its elements are found as p nodes whose maximum distances from any other node are the p smallest. More precisely, let $g(h)$ be the maximum distance of a node h from any other node, i.e.,

$$g(h) = \max\{c_{ih} | i \in N, i \neq h\},\ h \in N; \tag{9}$$

Then the nodes with the p smallest values of function g are taken as the initial p hubs. Matrix A is determined by assigning each node to its closest hub.

Neighborhood structures. For a given solution $S = (H, A)$, we define the following neighborhood structures:

- $Interchange\_hub(S) = \{S' = (H', A) | H' \subset N,\ |H' \cap H| = p - 1\}$: This neighborhood structure contains all solutions obtained by replacing one hub from the set H by a non-hub node. So, a search in this neighborhood for an improving solution of S will be performed by replacing any single hub node in S by any non-hub node. In the resulting solution, to the new hub will be assigned the nodes that were assigned to the old hub (i.e., matrix A remains unchanged). Hence, in order to calculate the value of some solution generated by such a replacement, it suffices firstly to update values in the array r and after that to calculate its objective value as outlined in Property 2.1.
- $Re - allocate\_node(S)$ contains all solutions that may be obtained by changing one node-to-hub allocation, i.e., moving a node from one set A_i to another set A_j. If a movement of a node from a set A_i to a set A_j occurs, the objective value of the resulting solution is calculated using Property 2.1 with updated values of r_i and r_j. Note that if the updated value of r_i is not less than the initial one, the resulting solution can not be better than the initial solution.

Nested variable neighborhood descent. The previously defined neighborhoods are explored using a nested search strategy (Nested Variable Neighborhood Descent (Nest-VND) [10, 14]). Namely, at each point of the $Interchange\_hub$ neighborhood we apply a local search with respect to the $Re - allocate\_node$ neighborhood (Algorithm 1). The proposed Nest-VND uses a first improvement search strategy, i.e., as soon as Nest-VND detects a solution better than the current one, it resumes the search starting from the better solution.

Algorithm 1: Nested VND.

```
   Function Nest-VND(S);
1  Improve ← True;
2  while Improve do
3  |  Improve ← False;
4  |  for each S'' ∈ Interchange_hub(S) do
5  |  |  Select S' as a best solution in Re − allocate_node(S'');
6  |  |  if S' is better than S then
7  |  |  |  S ← S';
8  |  |  |  Improve ← True;
   |  |  end
   |  end
   end
9  return S.
```

The reason we use this nested strategy is that the number of solutions visited by Nest-VND is very large. It is equal to the product of the cardinalities of the two neighborhood structures used. The large cardinality of the set of points visited obviously increases the chances to find an improvement in the nested neighborhoods. We also note that a Nest-VND approach has been successfully applied to some other variants of the p-hub problem ([14, 18]).

General variable neighborhood search. Variable neighborhood search (VNS) is a metaheuristic proposed by Mladenovic and Hansen in 1997 [12]. The basic variant of Variable neighborhood search (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 neighborhood change. The basic VNS finishes its work when a pre-defined stopping condition (e.g., maximum number of iterations or maximum CPU time) is met. From this basic VNS scheme many variants of VNS have been derived (see e.g. [11, 10] for recent surveys) and successfully applied for solving many optimization problems (see e.g. [11, 13]). The best known and widely-used VNS variant is the so-called General VNS (GVNS). It is derived from the basic VNS scheme by replacing a simple local search by some more advanced improvement procedure that examines several neighborhood structures. The most common improvement procedures used within GVNS, are based on Variable neighborhood descent (VND), such as sequential VND, nested VND, cyclic VND and so on.

The steps of the GVNS that we propose for solving the USApHCP are presented in Algorithm 3. For the local search step our GVNS uses the Nest-VND given in Algorithm 1. The Shaking procedure (see Algorithm 2) requires a solution S and a parameter k as the input. At each of k iterations of this procedure, the solution S is replaced by a randomly chosen point from its $Interchange\_hub$ neighborhood. At the end, the solution S obtained in the kth iteration is returned as the output of the imposed shaking.

Our GVNS procedure requires two parameters as shown in Algorithm 3. The first one denoted by t_max represents the CPU time allowed before terminating a run, while the second one, named k_max, represents the maximum number of iterations that can be executed within the inner loop of the Shaking procedure.

Algorithm 2: Shaking procedure

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

Algorithm 3: GVNS for solving USApHCP

```
   Function GVNS(S, k_max, t_max);
1  S ← greedy solution;
2  repeat
3  |  k ← 1;
4  |  while k ≤ k_max do
5  |  |  S' ← Shake(S, k) ;
6  |  |  S'' ← Nest-VND(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

The proposed GVNS algorithm, coded in C/C++, is compared with a CPLEX 12.6 mip solver when used to solve the radius formulation for the USApHCP described in the introduction, as well as a 2-phase exact algorithm and ant colony optimization (ACO) based approach developed in [16]. All experiments described in this section were carried out on a computer with Intel i7 2.8 GHz CPU and 16GB of RAM, using benchmark instances from the literature. For all instances the cost parameters γ and δ are set to 1; i.e., the non-unit

collection and distribution cost factors traditionally used for the AP data are ignored. The instances examined are classified in three groups:

- **CAB data set.** The CAB data set is derived from the Civil Aeronautics Board Survey of 1970 of passenger data in the United States. For instances in this data set, the cost parameter α ranges from 0.2 to 1.0.
- **The AP (Australian Post) data set.** This data set is based on real data from the Australian postal service and was presented by Ernst and Krishnamoorthy in 1996 [7]. The cost parameter α is set to 0.75.
- **URAND data set** The URAND data set consists of instances with up to 1000 nodes. The instances with up to 400 are found in Meyer et al. [16], while instances with 1000 nodes are found in Ilić et al. [14]. The x and y coordinates of the nodes are randomly generated from U[0,100000]. The cost parameter α is set to 0.75. Note that this is the first time that instances as large as 1000 nodes are used for testing purposes for the USApHCP problem.

Comparisons of results on the test instances are summarized in Tables 1 to 5. The column headings are defined as follows:

- n.p – means that there are n nodes in the test instance and p hubs to be located
- α – the value of cost parameter α
- value – value of a solution found by a certain method.
- dev(%) – percentage deviation of the value returned by the considered method from the optimal value. It is calculated as $100 \cdot (value - optimal\_value)/optimal\_value$.
- impr.(%) – the percentage improvement achieved by GVNS over the CPLEX 12.6 mip solver regarding solution values returned by both methods. It is calculated as $100 \cdot (CPLEX\_value - GVNS\_value)/CPLEX\_value$
- time – CPU time in seconds consumed by a solution approach to find the best solution to a problem

All results reported in Tables 1 to 5 are obtained on the same computer except for the 2-phase exact algorithm and ACO method which were executed on a dual Intel Xeon 3.2 GHz machine with 4 GB RAM. The reported results for these two methods are taken from [16].

Our GVNS algorithm was run with parameter k_max set to p on all test instances. The time limit for GVNS was set to 10 seconds for all instances, except those in Table 5. Namely, for instances with 1000 nodes in Table 5 the time limit was 600 seconds, while for all others here the time limit was 300 seconds. The time limit for the CPLEX 12.6 mip solver was set to 3600 seconds.

From the presented results the following conclusions may be drawn:

- On each test instance for which the optimal solution was known before or found by the CPLEX solver, our GVNS succeeded to reach that solution in much shorter time than any considered method.
- Referring to Tables 1 to 4, we see that the maximum CPU time used by GVNS to solve an instance exactly is around nine seconds.
- On the CAB data set which contains the smallest test instances, our GVNS required 0.01 second or less to find an optimal solution.
- The CPLEX 12.6 mip solver was able to find an optimal solution within one hour of computation, for all instances in the CAB data set and all instances, but one, in the AP data set. On 11 out of 35 instances in the URAND data set, CPLEX 12.6 succeeded to find an optimal solution.
- For those instances where the ACO heuristic was applied, our GVNS performs much better when comparing both the solution quality and consumed CPU time. More precisely, the ACO heuristic failed to reach optimal solutions on several test instances although it spent significantly more CPU time than the GVNS (e.g., compare average CPU times on the URAND data set: 33.68 seconds (ACO heurstic) and 0.95 seconds (GVNS)).

Table 1: Computational results on AP data set

	2-phase algorithm		ACO-heuristic		CPLEX		GVNS	
n.p	Optimal value	time	dev(%)	time	dev(%)	time	dev(%)	time
25.2	53207.46	0.39	0.00	0.39	0.00	0.50	0.00	0.00
25.3	46608.31	0.48	0.00	0.47	0.00	0.77	0.00	0.00
25.4	45552.50	0.50	0.00	0.50	0.00	0.23	0.00	0.01
25.5	45552.50	0.56	0.00	0.56	0.00	0.20	0.00	0.01
40.2	61682.50	0.68	0.00	0.68	0.00	1.67	0.00	0.01
40.3	58192.76	0.85	0.00	0.85	0.00	2.39	0.00	0.00
40.4	52265.27	0.96	0.00	0.96	0.00	1.48	0.00	0.00
40.5	49741.20	1.05	0.00	1.05	0.00	0.83	0.00	0.01
50.2	65523.37	0.94	0.00	0.93	0.00	3.12	0.00	0.01
50.3	60132.14	1.16	0.00	1.15	0.00	4.65	0.00	0.03
50.4	52905.77	1.37	0.00	1.37	0.00	3.51	0.00	0.01
50.5	50707.87	1.50	0.00	1.49	0.00	4.96	0.00	0.01
100.2	65914.82	2.79	0.00	2.72	0.00	41.22	0.00	0.01
100.3	60658.88	3.37	0.00	3.23	0.00	79.50	0.00	0.01
100.4	56124.74	3.76	0.00	3.66	0.00	101.65	0.00	0.05
100.5	54243.50	4.49	0.22	3.98	0.00	72.81	0.00	1.40
200.2	68231.97	10.13	0.00	9.43	0.00	919.40	0.00	0.01
200.3	64237.35	12.79	0.00	11.32	0.00	1665.34	0.00	0.01
200.4	59999.08	14.20	0.00	12.82	**0.28**	3600.38	0.00	0.01
200.5	58561.76	25.27	0.00	14.79	0.00	2016.8	0.00	0.67
Average:	56502.19	4.36	0.01	3.62	0.01	426.07	0.00	0.11

Table 2: Computational results on AP data set – continued

	CPLEX		GVNS	
n.p	Optimal value	time	dev(%)	time
10.2	40382.65	0.33	0.00	0.00
10.3	34772.38	0.23	0.00	0.00
10.4	32574.20	0.28	0.00	0.02
10.5	32531.21	0.30	0.00	0.00
20.2	45954.15	0.37	0.00	0.01
20.3	43400.45	0.20	0.00	0.01
20.4	38607.29	0.34	0.00	0.01
20.5	37868.15	0.14	0.00	0.01
20.10	37868.15	0.14	0.00	0.00
25.10	45552.50	0.23	0.00	0.01
40.10	49741.20	0.47	0.00	0.04
50.10	50707.87	1.53	0.00	0.04
100.10	51860.03	31.28	0.00	0.13
100.15	51860.03	25.77	0.00	0.19
100.20	51860.03	11.09	0.00	0.66
200.10	55958.75	681.91	0.00	0.11
200.15	55958.75	302.39	0.00	0.60
200.20	55958.75	353.70	0.00	1.75
Average:	45189.81	78.37	0.00	0.20

Table 3: Computational results on CAB data set

		CPLEX		GVNS				CPLEX		GVNS	
n.p	α	Optimal value	time	dev(%)	time	n.p	α	Optimal value	time	dev(%)	time
10.2	0.2	1425.58	0.17	0.00	0.00	20.2	0.2	1892.99	0.34	0.00	0.00
10.2	0.4	1627.52	0.33	0.00	0.00	20.2	0.4	2160.75	0.28	0.00	0.00
10.2	0.6	1759.13	0.16	0.00	0.00	20.2	0.6	2274.67	0.38	0.00	0.00
10.2	0.8	1759.13	0.14	0.00	0.01	20.2	0.8	2501.92	0.53	0.00	0.00
10.2	1.0	1839.65	0.27	0.00	0.01	20.2	1.0	2609.18	0.25	0.00	0.00
10.3	0.2	1119.53	0.19	0.00	0.00	20.3	0.2	1551.25	0.30	0.00	0.00
10.3	0.4	1185.06	0.11	0.00	0.01	20.3	0.4	1760.15	0.61	0.00	0.00
10.3	0.6	1387.00	0.12	0.00	0.01	20.3	0.6	1997.79	0.66	0.00	0.00
10.3	0.8	1588.94	0.17	0.00	0.01	20.3	0.8	2263.54	0.61	0.00	0.00
10.3	1.0	1790.55	0.16	0.00	0.01	20.3	1.0	2600.27	0.20	0.00	0.00
10.4	0.2	830.25	0.14	0.00	0.00	20.4	0.2	1355.41	0.45	0.00	0.00
10.4	0.4	968.20	0.25	0.00	0.01	20.4	0.4	1472.71	0.61	0.00	0.00
10.4	0.6	1146.19	0.19	0.00	0.01	20.4	0.6	1834.83	0.58	0.00	0.00
10.4	0.8	1454.44	0.16	0.00	0.01	20.4	0.8	2153.00	0.53	0.00	0.00
10.4	1.0	1764.79	0.11	0.00	0.00	20.4	1.0	2600.08	0.30	0.00	0.01
15.2	0.2	2005.02	0.27	0.00	0.00	25.2	0.2	2131.20	0.38	0.00	0.00
15.2	0.4	2160.75	0.22	0.00	0.00	25.2	0.4	2402.55	0.50	0.00	0.00
15.2	0.6	2214.09	0.27	0.00	0.00	25.2	0.6	2558.74	0.59	0.00	0.00
15.2	0.8	2423.80	0.17	0.00	0.00	25.2	0.8	2714.93	0.36	0.00	0.00
15.2	1.0	2609.18	0.28	0.00	0.00	25.2	1.0	2827.16	0.42	0.00	0.00
15.3	0.2	1749.04	0.19	0.00	0.00	25.3	0.2	1923.12	0.75	0.00	0.00
15.3	0.4	1760.15	0.27	0.00	0.00	25.3	0.4	2100.47	0.83	0.00	0.00
15.3	0.6	1844.92	0.19	0.00	0.00	25.3	0.6	2340.25	0.42	0.00	0.00
15.3	0.8	2166.54	0.17	0.00	0.00	25.3	0.8	2554.13	0.55	0.00	0.00
15.3	1.0	2600.08	0.17	0.00	0.00	25.3	1.0	2758.39	0.39	0.00	0.00
15.4	0.2	1340.96	0.25	0.00	0.00	25.4	0.2	1619.48	0.84	0.00	0.01
15.4	0.4	1434.38	0.22	0.00	0.00	25.4	0.4	1884.84	0.94	0.00	0.01
15.4	0.6	1754.51	0.16	0.00	0.00	25.4	0.6	2182.49	0.78	0.00	0.01
15.4	0.8	2080.06	0.17	0.00	0.00	25.4	0.8	2454.35	0.73	0.00	0.01
15.4	1.0	2600.08	0.14	0.00	0.00	25.4	1.0	2726.28	0.53	0.00	0.00
Average:		1746.32	0.19	0.00	0.00	Average:		2206.90	0.52	0.00	0.00

Table 4: Computational results on URAND data set

	2-phase algorithm		ACO-heuristic		CPLEX		GVNS	
n.p	Optimal value	time	dev(%)	time	dev(%)	time	dev(%)	time
100.2	127987.22	3.43	0.00	3.35	0.00	39.56	0.00	0.01
100.3	119919.87	4.33	0.00	4.08	0.00	101.70	0.00	0.01
100.4	111452.93	5.39	0.00	4.81	0.00	117.20	0.00	0.06
100.5	106174.56	5.84	0.00	5.54	0.00	159.06	0.00	0.03
200.2	132146.71	12.90	0.00	12.14	0.00	1120.07	0.00	0.00
200.3	121782.28	16.97	0.00	15.31	0.00	2465.87	0.00	0.13
200.4	116242.74	32.77	1.09	17.57	1.93	3600.24	0.00	0.31
200.5	108511.29	26.46	0.00	20.46	3.02	3600.33	0.00	0.29
300.2	132146.71	32.41	0.00	27.73	4.98	3600.34	0.00	0.55
300.3	122270.07	69.72	0.38	34.91	58.67	3600.39	0.00	0.13
300.4	116573.58	247.48	0.41	41.20	21.54	3600.14	0.00	1.23
300.5	109522.22	89.05	0.00	48.19	21.89	3601.14	0.00	0.23
400.2	131721.57	70.70	0.32	54.75	75.33	3600.86	0.00	0.13
400.3	122270.07	161.67	1.54	68.17	97.99	3600.55	0.00	1.33
400.4	115843.77	982.85	2.06	88.36	211.73	3600.64	0.00	9.26
400.5	109522.22	254.53	0.00	92.30	229.73	3602.76	0.00	1.51
Average:	119005.49	126.03	0.36	33.68	45.43	2500.68	0.00	0.95

Table 5: Computational results on URAND data set - continued

	CPLEX		GVNS		
n.p	value	time	value	time	impr.(%)
100.10	98558.78	20.34	98558.78	3.31	0.00
100.15	97238.22	5.29	97238.22	1.01	0.00
100.20	97238.22	4.73	97238.22	1.28	0.00
200.10	103239.25	3600.34	103239.25	24.85	0.00
200.15	99203.00	514.82	99203.00	149.64	0.00
200.20	98488.51	248.70	98488.51	8.10	0.00
300.10	113743.75	3600.83	103793.14	90.28	8.75
300.15	102611.43	3600.55	100944.92	41.76	1.62
300.20	100635.29	3600.67	99634.50	278.58	0.99
400.10	133945.25	3600.42	103793.14	53.17	22.51
400.15	341306.00	3602.59	101444.36	185.63	70.28
400.20	112200.76	3600.84	99870.34	233.74	10.99
1000.02	205288.42	3604.92	135931.84	1.32	33.78
1000.03	382750.00	3602.34	124361.62	8.45	67.51
1000.04	382750.00	3603.07	120184.81	42.97	68.60
1000.05	382750.00	3601.81	113229.99	56.91	70.42
1000.10	382750.00	3602.68	107522.15	460.23	71.91
1000.15	382750.00	3601.77	104662.73	453.48	72.66
1000.20	382750.00	3602.72	104386.36	234.85	72.73
Average:	210536.68	2695.76	105985.57	122.61	30.14

- Regarding results presented in Table 5, obtained on instances that may be considered as the hardest, it follows that our GVNS outperforms significantly the CPLEX 12.6 mip solver. The average CPU time consumed by GVNS is more than 20 times less than the average time spent by the CPLEX mip 12.6 solver, while the solution returned by GVNS is either the same or better than that returned by CPLEX. Further, the average improvement on solution values achieved by GVNS over CPLEX on the considered data set is 30.14%. Also, it should be emphasized that on instances with 1000 nodes and with $p > 2$, the solution value reported by GVNS is more than 3 times less than the one reported by CPLEX.

4 Concluding remarks

This paper examines the uncapacitated single allocation p-hub center problem (USApHCP). A heuristic method to solve this problem is proposed based on the general variable neighborhood search (GVNS) framework. A powerful local search is applied within the heuristic that uses a nested structure with two local search neighborhoods. The data structure developed allows for an efficient implementation of the procedure. We test the heuristic on benchmark instances from the literature as well as on several new instances of larger size than previously reported. Obtained results demonstrate the superiority of the GVNS heuristic over the state-of-the-art method based on ant colony optimization in terms of both solution quality and CPU run time. Furthermore, the new heuristic obtained the optimal solution in all known cases in a small fraction of the time of the corresponding exact method.

Future research will apply the GVNS framework to other types of hub location problems.

References

[1] Alumur, S., & Kara, B.Y. (2008). Network hub location problems: The state of the art. European Journal of Operational Research, 190(1), 1–21.

[2] Bryan, D.L., & O'Kelly, M.E. (1999). Hub-and-spoke networks in air transportation: An analytical review. Journal of Regional Science, 39(2), 275–295.

[3] Campbell, J.F., Ernst, A., & Krishnamoorthy, M. (2001) Hub location problems. In: Hamacher H., Drezner Z., editors. Location theory: Applications and theory. Berlin: Springer, 373–406.

[4] Campbell, J.F. (1994). Integer programming formulations of discrete hub location problems. European Journal of Operational Research, 72(2), 387–405.

[5] Chung, S.H., Myung, Y.S., & Tcha, D.W. (1992). Optimal design of a distributed network with a two-level hierarchical structure. European Journal of Operational Research, 62(1), 105–115.

[6] Ernst, A.T., Hamacher, H., Jiang, H., Krishnamoorthy, M., & Woeginger, G. (2009). Uncapacitated single and multiple allocation p-hub center problems. Computers & Operations Research, 36(7), 2230–2241.

[7] Ernst, A.T., & Krishnamoorthy, M. (1996). Efficient algorithms for the uncapacitated single allocation p-hub median problem. Location Science, 4(3), 139–154.

[8] Gavriliouk, E.O. (2009). Aggregation in hub location problems. Computers & Operations Research, 36(12), 3136–3142.

[9] Hamacher, H.W., & Meyer, T. (2006). Hub cover and hub center problems. Technical Report 98, FB Mathematik TU Kaiserslautern.

[10] Hansen, P., & Mladenović, N. (2002). Developments of variable neighborhood search. In: Ribeiro C., Hansen P., editors. Essays, surveys in metaheuristics. Amsterdam: Kluwer, 415–440.

[11] Hansen, P., Mladenović, N., & Moreno-Pérez, J.A. (2010). Variable neighbourhood search: Methods and applications. Annals of Operation Research, 175, 367–407.

[12] Mladenović, N., & Hansen, P. (1997). Variable neighborhood search, Computers & Operations Research, 24, 1097–1100.

[13] Mladenović, N., Todosijević, R., & Urošević, D. (2012). An efficient General variable neighborhood search for large TSP problem with time windows, Yugoslav Journal of Operations Research, 22, 141–151.

[14] Ilić, A., Urošević, D., Brimberg, J., & Mladenović, N. (2010). A general variable neighborhood search for solving the uncapacitated single allocation p-hub median problem. European Journal of Operational Research, 206(2), 289–300.

[15] Kara, B.Y., & Tansel, B.C. (2000). On the single-assignment p-hub center problem. European Journal of Operational Research, 125(3), 648–655.

[16] Meyer, T., Ernst, A.T., & Krishnamoorthy, M. (2009). A 2-phase algorithm for solving the single allocation p-hub center problem. Computers & Operations Research, 36(12), 3143–3151.

[17] O'Kelly, M.E., & Miller, H.J. (1991). Solution strategies for the single facility minimax hub location problem. Papers in Regional Science, 70(4), 367–380.

[18] Todosijević, R., Urošević, D., Mladenović, N., & Hanafi, S. (2015). A general variable neighborhood search for solving the uncapacitated r-allocation p-hub median problem. Optimization Letters, doi:10.1007/s11590-015-0867-6.