Les Cahiers du GERAD ISSN: 0711–2440

A basic variable neighborhood search heuristic for the uncapacitated multiple allocation p-hub center problem

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

G–2015–38

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

A basic variable neighborhood search heuristic for the uncapacitated multiple 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–2015–38

Abstract: The uncapacitated multiple allocation p-hub center problem (UMApHCP) consists of choosing p hub locations from a set of nodes with pairwise traffic demands in order to route the traffic between the origin-destination pairs such that the maximum cost between origin-destination pairs is minimum. It is assumed that transportation between non-hub nodes is possible only via chosen hub nodes. In this paper we propose a basic variable neighborhood search (VNS) heuristic for solving this NP hard problem. In addition we apply two mathematical formulations of the UMApHCP in order to detect limitations of the current state-of-the-art solver used for this problem. The heuristics are tested on benchmark instances for p-hub problems. The obtained results reveal the superiority of the proposed basic VNS over the state-of-the-art as well as over a multi-start local search heuristic developed by us in this paper.

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

1 Introduction

Real-world transportation concerns transferring goods, passengers, etc., from one point (origin) to another (destination). The straightforward transportation network, that may enable such transfer, is one which connects directly all origin-destination pairs. However, the cost for establishing such a network can be unreasonably high. Therefore, a better option is to build a network that uses special nodes, called hubs, to connect origin-destination pairs and therefore to enable transfer between them. Note that in this created network, direct communication between non-hub nodes is not possible. The choice of hub locations and the allocation of non-hub nodes to hubs are both parts of the optimization problem. Varying the number of hub nodes that may be assigned to a non-hub node allows us to distinguish between single allocation (each non-hub node is allocated to exactly one hub), multiple allocation (a non-hub node may use any hub to communicate with other non-hub nodes), and r - allocation (each node can be connected to at most r chosen hubs). Examples of hub networks arise in airline transportation [3], telecommunication systems [5], postal networks [6], and so on (see e.g. [2, 4]).

The most common hub network problem is the hub median problem. It consists of locating hub nodes and assigning non-hub nodes to hubs, respecting a predefined allocation strategy, such that the total transportation cost between all origin-destination pairs is minimized. For recent approaches for solving hub-median problems see e.g., [11, 14]. However, it is possible in such constituted networks that the distance (cost) between some origin-destination pairs is extremely large and therefore unacceptable. In order to resolve this issue we consider a version of the problem which seeks to minimize the maximal distance between origin-destination pairs. This hub network problem is referred in the literature as the hub center problem.

In this paper we study the uncapacitated multiple allocation p - hub center problem (UMApHCP). 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 UMApHCP, p hubs must be selected among these n nodes, where p is given. Let H be the set of hubs. A non-hub node may use any hub node from H to communicate with other nodes. 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 UMApHCP is to minimize the maximum cost between origin-destination pairs.

The UMApHCP was introduced by Ernst et al. in 2009 [8]. The authors propose two mixed integer formulations for the UMApHCP. Additionally, they prove the NP-hardness of the problem and propose a shortest path based branch-and-bound algorithm as well as a simple multi-start local search heuristic to tackle the problem. The largest test instance considered in the literature on UMApHCP has 200 nodes. However, real life transportation networks may contain significantly larger numbers of nodes. Hence, this paper also considers instances with up to 1000 nodes that have been previously used for testing purposes in several p-hub problems. To solve the UMApHCP, we develop a basic variable neighborhood search (VNS) heuristic, and verify its performance on the instances used in [8] as well as on the data set containing large size instances. We compare results obtained with our heuristic not only with those achieved by the heuristic method and exact branch-and-bound method proposed in [8], but also with the results obtained by solving the two mathematical formulations of the UMApHCP with a CPLEX 12.6 MIP solver. On the large size instances, for which there are no available results in the literature and which can not be solved by the CPLEX 12.6 mip solver, we compare the performance of our basic VNS only against a multi-start local search. All in all the contributions of the paper are two-fold: an efficient method for solving the UMApHCP is proposed, and a new larger set of test instances is introduced for the UMApHCP.

The rest of the paper is organized as follows. In the next section we review the two mathematical models for the UMApHCP, while Section 3 describes the main ingredients of our basic VNS. Section 4 contains a

comparison of the proposed approach with other existing approaches as well as a new multi-start local search. Finally, Section 5 concludes the paper.

2 Mathematical formulations

This section reviews the two mathematical formulations for the UMApHCP presented in [8]. In the first model, the following variables are defined:

- binary variable y_{ijkl} which takes the value 1, if and only if, the transfer between node i and j is accomplished through the path $i \to k \to l \to j$;
- binary variable z_k which takes the value 1, if and only if, the transfer node k is selected to serve as a hub;
- a continuous variable r for the objective function.

The so-called *four index formulation* of the UMApHCP obtained using these variables is stated as follows:

$$\min r \tag{1}$$

subject to:

$$\sum_{k\in N} z_k = p \tag{2}$$

$$\sum_{k\in N}\sum_{l\in N} y_{ijkl} = 1, \qquad i,j \in N \tag{3}$$

$$\sum_{k\in N} y_{ijkl} \le z_l, \qquad i,j,l \in N \tag{4}$$

$$\sum_{l\in N} y_{ijkl} \le z_k, \qquad i,j,k \in N \tag{5}$$

$$r \ge \sum_{k\in N}\sum_{l\in N} (\gamma c_{ik} + \alpha c_{kl} + \delta c_{lj}) y_{ijkl}, \qquad i,j \in N \tag{6}$$

$$y_{ijkl}, z_k \in \{0,1\}, \qquad i,j,k,l \in N. \tag{7}$$

Constraint 2 ensures that exactly p nodes are selected to be hubs. Constraints 3 guarantee that for each origin-destination pair exactly one path is determined. Constraints 4 and constraints 5 only allow assignment of non-hub nodes to hubs. Finally, constraints 6 impose the lower bound on the objective function r which represents the maximum transportation cost between all origin–destination pairs.

Besides variables z_k and r, the second mathematical model uses binary variables u_j^{ik} and v_{lj}^i. Namely, with respect to each origin-destination pair (i,j) variables u_j^{ik} and v_{lj}^i take value 1, if and only if, the path $i \to k \to l \to j$ is established between pair (i,j). In addition, the model uses parameter c_{max} calculated as $c_{max} = \max_{i,j\in N} c_{ij}$. So, the second mathematical model for the UMApHCP, i.e., the *three index formulation*, is stated as:

$$\min r \tag{8}$$

subject to:

$$\sum_{k\in N} z_k = p \tag{9}$$

$$u_j^{ik} \le z_k, \qquad i,j,k \in N \tag{10}$$

$$v_{lj}^i \le z_l, \qquad i,j,l \in N \tag{11}$$

$$\sum_{k\in N} u_j^{ik} = 1, \qquad i,j \in N \tag{12}$$

$$\sum_{l \in N} v_{lj}^i = 1, \qquad i, j \in N \tag{13}$$

$$r \geq \sum_{k \in N} (\gamma c_{ik} + \alpha c_{kt}) u_j^{ik} + \sum_{l \in N} \delta c_{lj} v_{lj}^i - \alpha(1 - v_{tj}^i) c_{max}, \qquad i \geq j, t \in N \tag{14}$$

$$u_j^{ik}, v_{lj}^i, z_k \in \{0, 1\}, \qquad i, j, k, l \in N \tag{15}$$

The choice of p hubs is ensured by constraint (9). Constraints (10) and (11) allow assignment of nodes only to hubs, while constraints (12) and (13) guarantee that for each origin-destination pair, both origin and destination will be allocated to exactly one hub. Finally, constraints (14) define the lower bound on the objective function r. More precisely, let us assume that the path $i \to k \to l \to j$ is established between pair (i, j). Then, if $t = l$, we have $r \geq \gamma c_{ik} + \alpha c_{kl} + \delta c_{lj}$, i.e., the right-hand side represents the real cost of the path $i \to k \to l \to j$. On the other hand, if $t \neq l$, we have inequality $r \geq \gamma c_{ik} + \alpha c_{kt} + \delta c_{lj} - \alpha c_{max}$. Obviously, for the right-hand side the following holds: $\gamma c_{ik} + \alpha c_{kl} + \delta c_{lj} \geq \gamma c_{ik} + \delta c_{lj} \geq \gamma c_{ik} + \alpha c_{kt} + \delta c_{lj} - \alpha c_{max}$.

3 Variable neighborhood search for UMApHCP

Variable neighborhood search (VNS) is a flexible framework for building heuristics for approximately solving optimization problems. It was introduced by Mladenovic and Hansen in 1997 [13]. The main idea of VNS is the systematic changing of the neighborhood structures during the search for an optimal (or near-optimal) solution. The changing of neighborhood structures is based on the following observations: (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; (iii) Empirical evidence shows that for many problems all local optima are relatively close to each other. The first property is exploited by using increasingly complex moves in order to find local optima with respect to all neighborhood structures used. The second property suggests using several neighborhoods, if local optima found are of poor quality. Finally, the third property suggests exploitation of the vicinity of the current incumbent solution.

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 the neighborhood change step. The basic VNS finishes its work when a predefined stopping condition (e.g., maximum number of iterations or maximum CPU time) is met. Many variants of VNS have been proposed until now, starting from this basic VNS scheme. Heuristic approaches based on Basic VNS and other VNS variants have been successfully applied for solving many optimization problems. For recent surveys on VNS variants and applications we refer the reader to [9, 10].

For solving the UMApHCP we now propose a heuristic that follows the rules of Basic VNS. Before providing more details of our heuristic, we describe the solution representation for UMApHCP. This is done by specifying a set H containing p hubs, i.e. $H = \{h_1, h_2, \ldots, h_p\}$, and two matrices $A = (a_{ij})$ and $B = (b_{ij})$. In those matrices, entries a_{ij} and b_{ij} correspond to the hubs that nodes i and j are allocated to in the path from node i to node j, respectively. Note that if the set H is known, the optimal node to hub allocation for each origin-destination pair (i.e., optimal matrices A and B) can be found by determining the shortest path between them via hubs from the set H. Therefore, in what follows we consider that a solution of UMApHCP is represented by the set H of chosen hubs. The straightforward calculation of all-(origin-destination)pairs shortest paths require $O(n^2p^2)$ operations. However, the calculation of all-pairs shortest paths may be performed in $O(n^2p)$ operations using a modification of the well-known Floyd-Warshall shortest path algorithm [1, 7].

The proposed heuristic starts the search from a solution built in a greedy manner. Namely, the initial p hubs are chosen as those 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.,

$$g(h) = \max\{c_{ih} | i \in N, i \neq h\}, \ h \in N; \tag{16}$$

Then the nodes with the p smallest values of function g are taken as the initial p hubs.

This obtained solution is the starting point for our Basic VNS whose steps are presented in Algorithm 1. The local search used within basic VNS is based on the exploration of the neighborhood

$$Interchange\_hub(H) = \{H' | H' \subset N,\ |H' \cap H| = p - 1\}$$

This neighborhood structure contains all solutions obtained by replacing one hub from the set H by a non-hub node. The objective function value of a resulting solution H' is calculated from scratch, i.e., in $O(n^2 p)$ operations. The search for an improving solution is performed using the first improvement strategy, i.e., as soon as a better solution than the incumbent is detected, it is accepted as the new incumbent solution and the search is resumed starting from there. Such a local search procedure will eventually get stuck at a local minimum. Therefore, to hopefully resolve the encountered local minima traps, a shaking procedure is used as depicted in Algorithm 2. For the input, the Shaking procedure requires a solution H and a parameter k. At each of k subsequent iterations, the last solution H is replaced by a randomly chosen one from its $Interchange\_hub$ neighborhood. At the end, the solution H obtained in the kth iteration is returned as the output of our shaking procedure. The maximum value of k is specified by parameter k_{max}. The stopping condition is given by a limit on CPU time, and is defined by parameter t_{max}.

Algorithm 1: Basic VNS for solving UMApHCP

```
Function VNS(H, k_max, t_max);
1  H ← greedy solution;
2  repeat
3  |  k ← 1;
4  |  while k ≤ k_max do
5  |  |  H' ← Shake(H, k) ;
6  |  |  H'' ← Interchange_hub(H') ;
7  |  |  k ← k + 1;
8  |  |  if H'' is better then H then
9  |  |  |  H ← H''; k ← 1;
   |  |  end
   |  end
10 |  t ← CpuTime();
   until t > t_max;
11 Return H;
```

Algorithm 2: Shaking procedure

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

4 Computational results

The proposed VNS algorithm is coded in C/C ++. Its performance is tested on the benchmark instances for p hub problems given in 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 benchmark instances are classified into three groups:

- **CAB data set.** The CAB data set is derived from the Civil Aeronautics Board Survey of 1970 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.** It is based on real data from the Australian postal service and was presented by Ernst and Krishnamoorthy in 1996 [6]. 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. [12], while instances with 1000 nodes are found in Ilić et al. [11]. 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 this data set is used for testing purposes for the UMApHCP problem.

Our basic VNS algorithm is tested with parameter k_{max} set to p, while parameter t_{max} depends on the data set considered. Namely, for the CAB data set t_{max} is set to 1 second; for the AP data set t_{max} is set to 10 seconds; for instances from the URAND data set with up to 400 nodes, t_{max} is set to 600 seconds; while for instances from the URAND data set with 1000 nodes t_{max} is set to 1800 seconds. The basic VNS is executed on a computer with Intel i7 2.8 GHz CPU and 16GB of RAM.

The results of our method are compared with those of the multi-start local search heuristic and exact branch-and-bound method (`B&B`) proposed in [8]. Since we did not implement these methods, we simply provide the results of these methods reported in [8]. We also compare results obtained by basic VNS with results obtained solving the two MIP formulations presented in Section 2 by a CPLEX 12.6 mip solver. The maximum CPU time allowed for the CPLEX 12.6 mip solver was set to 1 hour (3600 seconds) on each test instance. Additionally, in order to evaluate performance of our basic VNS on instances where the optimal solution values can neither be determined by the CPLEX 12.6 mip solver nor were previously known, we implement and test our version of a multi-start local search (MLS) heuristic. The MLS heuristic works iteratively, generating at each iteration an initial solution as a random set of p hubs, and executing the same local search used within our basic VNS on that solution. The multi-start local search heuristic finishes its work when the imposed time limit is reached. For the output MLS returns the best solution obtained during its execution. In order to have a fair comparison on each test instance we set the same time limit for MLS and basic VNS.

The comparative results are presented in Tables 1–5. In each of these tables the number of nodes (n) and the number of required hubs (p) for each test instance is presented as n.p in column `n.p`, while the value of cost parameter α for each instance from the CAB data set is given in column α. In Tables 1–3 we report results obtained on instances where optimal solutions were obtained by `B&B`. The optimal solution values are provided in the column headed 'Optimal value', while CPU times consumed by `B&B` to reach optimal solutions are reported in column 'B&B time'. The results obtained solving the four index formulation and the three index formulation of the UMApHCP by CPLEX are given in columns '4index form.' and '3index form.', respectively, while results obtained by the simple heuristic [8] and Basic VNS are given in columns 'Heuristic' and 'BVNS', respectively. For each of these approaches we report the percentage deviation of the obtained solution from the optimal one (column 'dev(%)') as well as CPU time in seconds (column 'time') used to reach that solution value. The percentage deviation is computed as:

$$100 \times \frac{value - optimal\_value}{optimal\_value}$$

where $value$ and $optimal\_value$ represent the solution value found by the considered method on a certain test instance and the optimal solution value for that test instance, respectively.

Tables 4 and 5 provide the comparison on test instances without known optimal solution values. For these instances we report results provided by CPLEX on the three index formulation of UMApHCP, Basic VNS and multi-start local search. For each of these approaches we report the solution values found by each (column 'value') and CPU times in seconds needed to reach these solution values (column 'time'). Finally, in each column 'dev(%)' we report the percentage deviation of the solution value found by MLS on a certain test instance from the corresponding solution value found by Basic VNS. The percentage deviation is computed as:

$$100 \times \frac{MLS\_value - VNS\_value}{VNS\_value}$$

where MLS_value and VNS_value represent corresponding solution values found by MLS and basic VNS, respectively.

Table 1: Computational results on CAB data set

				4index form.		3index form.		Heuristic		BVNS	
n.p	α	Optimal value	B&B time	dev(%)	time(s)	dev(%)	time(s)	dev(%)	time(s)	dev(%)	time(s)
10.2	0.2	1421.88	0.00	0.00	0.64	0.00	1.03	0.00	0.00	0.00	0.00
10.2	0.4	1548.37	0.00	0.00	0.52	0.00	0.86	0.00	0.00	0.00	0.00
10.2	0.6	1749.04	0.00	0.00	0.44	0.00	1.12	0.13	0.00	0.00	0.00
10.2	0.8	1749.04	0.00	0.00	0.41	0.00	1.06	0.00	0.00	0.00	0.00
10.2	1.0	1764.79	0.00	0.00	0.20	0.00	1.17	0.00	0.00	0.00	0.00
10.3	0.2	1119.54	0.00	0.00	0.50	0.00	1.31	0.00	0.00	0.00	0.00
10.3	0.4	1181.37	0.00	0.00	0.61	0.00	1.25	0.00	0.00	0.00	0.00
10.3	0.6	1308.85	0.00	0.00	0.39	0.00	1.62	0.00	0.00	0.00	0.00
10.3	0.8	1502.14	0.00	0.00	0.41	0.00	1.62	4.60	0.00	0.00	0.00
10.3	1.0	1764.79	0.00	0.00	0.17	0.00	1.45	0.00	0.00	0.00	0.00
10.4	0.2	809.36	0.00	0.00	0.41	0.00	0.81	37.41	0.00	0.00	0.00
10.4	0.4	968.20	0.00	0.00	0.59	0.00	1.28	0.00	0.00	0.00	0.00
10.4	0.6	1146.19	0.00	0.00	0.44	0.00	1.11	0.00	0.00	0.00	0.00
10.4	0.8	1411.83	0.00	0.00	0.28	0.00	1.12	3.02	0.00	0.00	0.00
10.4	1.0	1764.79	0.00	0.00	0.17	0.00	0.95	0.00	0.00	0.00	0.00
15.2	0.2	2005.02	0.00	0.00	3.03	0.00	4.99	0.00	0.00	0.00	0.00
15.2	0.4	2027.69	0.00	0.00	2.04	0.00	6.77	0.00	0.00	0.00	0.00
15.2	0.6	2081.04	0.00	0.00	1.36	0.00	6.79	0.00	0.00	0.00	0.00
15.2	0.8	2335.82	0.00	0.00	1.56	0.00	6.83	0.00	0.00	0.00	0.00
15.2	1.0	2600.08	0.00	0.00	0.70	0.00	7.42	0.00	0.00	0.00	0.00
15.3	0.2	1716.14	0.00	0.00	5.71	0.00	11.73	1.29	0.00	0.00	0.00
15.3	0.4	1738.32	0.00	0.00	2.68	0.00	8.56	0.00	0.00	0.00	0.00
15.3	0.6	1823.10	0.00	0.00	2.14	0.00	12.81	0.00	0.00	0.00	0.00
15.3	0.8	2141.83	0.00	0.00	1.42	0.00	12.68	0.00	0.00	0.00	0.00
15.3	1.0	2600.08	0.00	0.00	0.62	0.00	8.11	0.00	0.00	0.00	0.00
15.4	0.2	1287.78	0.00	0.00	6.43	0.00	6.32	0.00	0.00	0.00	0.00
15.4	0.4	1395.88	0.00	0.00	3.35	0.00	12.68	0.00	0.00	0.00	0.00
15.4	0.6	1751.45	0.01	0.00	2.28	0.00	14.12	0.00	0.00	0.00	0.00
15.4	0.8	2080.06	0.00	0.00	1.16	0.00	11.81	0.00	0.00	0.00	0.00
15.4	1.0	2600.08	0.00	0.00	0.62	0.00	7.10	0.00	0.00	0.00	0.00
Average:		1713.15	0.00	0.00	1.38	0.00	5.22	1.55	0.00	0.00	0.00

Table 2: Computational results on CAB data set-continued

				4index form.		3index form.		Heuristic		BVNS	
n.p	α	Optimal value	B&B time	dev(%)	time(s)	dev(%)	time(s)	dev(%)	time(s)	dev(%)	time(s)
20.2	0.2	1892.99	0.00	0.00	22.89	0.00	23.57	0.00	0.00	0.00	0.00
20.2	0.4	2027.69	0.00	0.00	10.86	0.00	23.65	0.00	0.00	0.00	0.00
20.2	0.6	2248.13	0.00	0.00	8.33	0.00	38.75	0.00	0.00	0.00	0.00
20.2	0.8	2335.99	0.00	0.00	9.38	0.00	33.15	8.32	0.00	0.00	0.00
20.2	1.0	2600.08	0.00	0.00	3.24	0.00	32.48	0.00	0.00	0.00	0.00
20.3	0.2	1551.25	0.01	0.00	31.00	0.00	43.01	10.65	0.00	0.00	0.00
20.3	0.4	1738.32	0.01	0.00	17.61	0.00	63.70	0.00	0.00	0.00	0.00
20.3	0.6	1916.16	0.01	0.00	13.51	0.00	55.72	0.00	0.00	0.00	0.00
20.3	0.8	2195.22	0.01	0.00	7.24	0.00	37.46	0.00	0.00	0.00	0.00
20.3	1.0	2600.08	0.01	0.00	3.10	0.00	93.27	0.00	0.00	0.00	0.00
20.4	0.2	1287.78	0.01	0.00	32.78	0.00	28.16	0.00	0.01	0.00	0.00
20.4	0.4	1472.71	0.02	0.00	15.69	0.00	115.89	0.00	0.01	0.00	0.00
20.4	0.6	1808.70	0.06	0.00	14.59	0.00	107.47	0.80	0.01	0.00	0.00
20.4	0.8	2128.11	0.02	0.00	17.30	0.00	103.44	1.17	0.01	0.00	0.00
20.4	1.0	2600.08	0.01	0.00	2.93	0.00	168.67	0.00	0.01	0.00	0.00
25.2	0.2	2049.48	0.01	0.00	105.18	0.00	82.68	0.00	0.00	0.00	0.00
25.2	0.4	2402.55	0.01	0.00	240.76	0.00	162.29	0.00	0.00	0.00	0.00
25.2	0.6	2558.74	0.01	0.00	61.59	0.00	149.45	0.00	0.00	0.00	0.00
25.2	0.8	2714.93	0.01	0.00	29.67	0.00	217.46	0.00	0.00	0.00	0.00
25.2	1.0	2739.22	0.01	0.00	42.79	0.00	210.36	0.00	0.00	0.00	0.00
25.3	0.2	1911.60	0.02	0.00	159.12	0.00	290.35	0.17	0.01	0.00	0.00
25.3	0.4	2064.67	0.02	0.00	78.94	0.00	337.10	1.28	0.01	0.00	0.00
25.3	0.6	2243.77	0.02	0.00	47.33	0.00	276.43	0.00	0.01	0.00	0.00
25.3	0.8	2515.58	0.02	0.00	32.60	0.00	687.41	0.00	0.01	0.00	0.00
25.3	1.0	2725.79	0.01	0.00	17.78	0.00	892.15	0.00	0.01	0.00	0.00
25.4	0.2	1619.48	0.04	0.00	262.52	0.00	1023.14	7.05	0.02	0.00	0.01
25.4	0.4	1774.45	0.05	0.00	178.92	0.00	800.00	0.00	0.02	0.00	0.00
25.4	0.6	2127.13	0.08	0.00	59.75	0.00	1111.89	0.00	0.02	0.00	0.00
25.4	0.8	2437.71	0.08	0.00	32.92	0.00	660.29	0.37	0.02	0.00	0.01
25.4	1.0	2725.79	0.02	0.00	17.83	0.00	375.80	0.00	0.02	0.00	0.00
Average:		2167.14	0.02	0.00	52.61	0.00	274.84	0.99	0.01	0.00	0.00

Table 3: Computational results on AP data set

			4index form.		3index form.		Heuristic		BVNS	
n.p	Optimal value	B&B time	dev(%)	time(s)	dev(%)	time(s)	dev(%)	time(s)	dev(%)	time(s)
10.2	39922.11	0.00	0.00	0.62	0.00	1.45	1.15	0.00	0.00	0.00
10.3	32713.94	0.00	0.00	0.47	0.00	1.45	0.00	0.00	0.00	0.00
10.4	31577.96	0.00	0.00	0.55	0.00	1.76	0.00	0.00	0.00	0.01
10.5	30371.32	0.00	0.00	0.44	0.00	1.42	0.00	0.00	0.00	0.00
20.2	45954.15	0.00	0.00	5.97	0.00	27.69	0.00	0.00	0.00	0.00
20.3	40909.59	0.01	0.00	7.14	0.00	95.24	6.09	0.00	0.00	0.01
20.4	38320.25	0.01	0.00	9.83	0.00	91.01	0.00	0.01	0.00	0.03
20.5	37868.15	0.01	0.00	7.91	0.00	70.81	0.00	0.01	0.00	0.02
20.10	37868.15	0.05	0.00	6.21	0.00	11.17	0.00	0.05	0.00	0.04
25.2	51533.30	0.00	0.00	34.18	0.00	124.21	0.00	0.00	0.00	0.01
25.3	45552.50	0.01	0.00	31.86	0.00	474.46	8.67	0.01	0.00	0.01
25.4	45552.50	0.02	0.00	44.87	0.00	426.63	0.00	0.02	0.00	0.01
25.5	45552.50	0.03	0.00	35.77	0.00	272.39	0.00	0.03	0.00	0.01
25.10	45552.50	0.14	0.00	23.98	0.00	30.90	0.00	0.13	0.00	0.01
40.2	61140.80	0.03	0.00	1804.41	0.26	3600.94	0.00	0.02	0.00	0.02
40.3	56309.88	0.08	0.00	999.95	10.75	3600.69	0.37	0.05	0.00	0.02
40.4	51279.14	0.14	17.38	3718.25	98.52	3600.70	5.06	0.11	0.00	0.02
40.5	49741.20	0.18	0.00	3603.42	65.42	3600.95	0.00	0.17	0.00	0.01
40.10	49741.20	1.00	0.00	1236.94	19.87	3600.53	0.00	0.98	0.00	0.02
50.2	61179.03	0.05	74.54	3603.25	56.60	3600.75	0.00	0.04	0.00	0.03
50.3	56729.94	0.16	88.23	3603.33	88.23	3600.64	0.00	0.11	0.00	0.02
50.4	52905.77	0.28	101.83	3603.65	100.89	3600.41	0.00	0.20	0.00	0.04
50.5	50707.87	0.52	110.58	3604.25	110.58	3600.66	0.00	0.50	0.00	0.03
50.10	50707.87	2.17	110.39	3602.65	108.68	3600.45	0.00	2.15	0.00	0.05
100.2	63197.10	0.56	–	–	17821.34	3615.59	0.95	0.44	0.00	0.08
100.3	57925.66	1.67	–	–	19452.24	3611.89	0.00	1.48	0.00	0.06
100.5	53949.33	22.52	–	–	20893.34	3609.63	0.76	5.91	0.00	0.50
100.10	51860.03	39.04	–	–	21739.10	3613.28	0.00	38.07	0.00	0.49
200.3	62945.55	35.37	–	–	–	–	0.00	25.77	0.00	1.92
Average:	48261.01	3.59	–	–	–	–	0.79	2.63	0.00	0.12

From the reported results, it follows that the proposed Basic VNS outperforms the previous simple heuristic for the UMApHCP. Our Basic VNS succeeded to reach the optimal solution for each test instance with known optimal solution, while the simple heuristic failed to do that on several instances. Additionally, our Basic VNS turned out to be very fast in solving test instances with known optimal solution values (i.e., all instances in the CAB data set and some instances in AP data set). The maximum CPU time consumed by Basic VNS to solve such an instance is about 2 seconds. Moreover, Basic VNS solved a large number of instances in negligible CPU time (i.e., not more than 0.1 second). Only for three of these instances Basic VNS needed more than 0.40 seconds to provide the optimal solution. On the remaining test instances, i.e., AP instances without known optimal solution values and instances from the URAND data set, Basic VNS performs better than multi-start local search regarding both average CPU time consumed (e.g., compare average CPU times on URAND data set: 224.93 of Basic VNS and 270.98 of MLS) and quality of provided solutions. Namely, on one test instance in Table 4 and eleven instances (out of 35), in Table 5 Basic VNS provides better solutions than MLS. The percentage deviation of the solution value offered by MLS from the corresponding solution value found by Basic VNS on these instances varies from 0.03% up to 1.76%. Additionally, it should be emphasized that there is no test instance where MLS succeeds to find a better solution than Basic VNS.

Regarding solutions obtained by the CPLEX 12.6 mip solver when applied to solve three and four index formulations of the UMApHCP, we may draw the following conclusions. The four index formulation cannot be used for solving instances with more than 50 nodes, while the three index formulation cannot be used for solving instances with more than 100 nodes due to insufficient memory to accommodate the corresponding models. Regarding test instances with up to 40 nodes, it turns out that CPLEX did not yield an optimal solution for just one test instance solved with the four index formulation. On the other hand, using the three index formulation, CPLEX failed to provide optimal solutions for 5 instances with 40 nodes. Moreover, on almost all test instances the CPU time consumed by CPLEX to solve the four index formulation is significantly less than that needed to solve the three index formulation.

Table 4: Computational results on AP data set-continued

	3index form.		BVNS		MLS		
n.p	value	time	value	time	value	time	dev(%)
100.4	11325764.05	3602.59	54704.51	0.16	54704.51	0.22	0.00
100.15	11325764.05	3601.89	51860.03	0.56	51860.03	0.52	0.00
100.20	11325764.05	3603.63	51860.03	1.04	51860.03	0.86	0.00
200.2	–	–	67083.28	0.97	67083.28	1.54	0.00
200.4	–	–	59420.93	2.54	59607.01	10.37	0.31
200.5	–	–	57419.32	2.15	57419.32	2.58	0.00
200.10	–	–	55958.75	2.32	55958.75	1.93	0.00
200.15	–	–	55958.75	4.24	55958.75	4.32	0.00
200.20	–	–	55958.75	7.35	55958.75	8.88	0.00
Average:	–	–	56691.59	2.37	56712.27	3.47	0.03

Table 5: Computational results on URAND data set

	3index form.		BVNS		MLS		
n.p	value	time	value	time	value	time	dev(%)
100.02	20402298.30	3607.64	124922.34	0.19	124922.34	0.01	0.00
100.03	20402298.30	3604.28	114692.05	0.08	116714.15	0.83	1.76
100.04	20402298.30	3606.40	107478.64	0.07	107478.64	0.28	0.00
100.05	20402298.30	3605.21	105545.51	0.88	105545.51	1.38	0.00
100.10	20402298.30	3601.25	98558.78	1.13	98803.55	2.21	0.25
100.15	20402298.30	3602.54	97238.22	0.94	97238.22	1.63	0.00
100.20	20402298.30	3601.78	97238.22	1.02	97238.22	1.37	0.00
200.02	–	–	130466.92	0.22	130466.92	0.17	0.00
200.03	–	–	117735.45	0.38	117735.45	1.10	0.00
200.04	–	–	111377.52	3.36	111377.52	2.09	0.00
200.05	–	–	106820.05	15.65	106820.05	4.31	0.00
200.10	–	–	102182.66	592.23	102217.40	262.01	0.03
200.15	–	–	98558.78	368.04	98951.72	56.38	0.40
200.20	–	–	98488.51	14.04	98488.51	18.88	0.00
300.02	–	–	131341.06	0.31	131341.06	1.04	0.00
300.03	–	–	120490.01	8.47	120490.01	12.33	0.00
300.04	–	–	111880.71	11.47	111880.71	8.25	0.00
300.05	–	–	108520.46	9.22	108520.46	43.07	0.00
300.10	–	–	102920.26	248.62	103699.41	366.02	0.76
300.15	–	–	100635.29	410.53	101008.71	605.28	0.37
300.20	–	–	98827.20	466.84	99680.47	64.75	0.86
400.02	–	–	131141.47	6.78	131141.47	8.94	0.00
400.03	–	–	120041.88	70.11	120041.88	26.65	0.00
400.04	–	–	111880.71	4.88	111880.71	12.78	0.00
400.05	–	–	108614.45	20.21	108614.45	17.89	0.00
400.10	–	–	103625.32	419.14	103625.32	562.39	0.00
400.15	–	–	101444.36	349.62	101660.54	459.03	0.21
400.20	–	–	99680.47	397.56	99870.34	180.39	0.19
1000.02	–	–	135325.95	52.43	135325.95	236.78	0.00
1000.03	–	–	124223.78	115.65	124223.78	479.74	0.00
1000.04	–	–	116108.88	746.09	116108.88	139.67	0.00
1000.05	–	–	112199.72	125.97	112199.72	603.79	0.00
1000.10	–	–	107333.29	1062.91	108099.05	1805.53	0.71
1000.15	–	–	104662.73	1453.04	105471.38	1800.28	0.77
1000.20	–	–	104386.36	894.36	104386.36	1696.92	0.00
Average:	–	–	110473.94	224.93	110664.82	270.98	0.18

5 Conclusion

This paper addresses the uncapacitated multiple allocation p-hub center problem (UMApHCP). For solving this problem we propose a basic variable neighborhood search (VNS) heuristic. Additionally, we use a CPLEX 12.6 mip solver in order to solve two existing mathematical formulations of the UMApHCP. Benchmark instances for p-hub problems are used for testing purposes. Some of these instances are used for the first time to test methods for solving the UMApHCP. In order to verify the performance of our basic VNS on these new instances we also implemented and tested a new multi-start local search heuristic. The obtained results confirm that our basic VNS is able to reach optimal solutions on all instances with a known optimal

solution in under 2 seconds. Additionally, it consistently outperforms the previous simple heuristic as well as the multi-start local search heuristic developed by us.

Future work may include adapting the proposed heuristic for other p-hub problems related to the UMApHCP (e.g., the capacitated multiple allocation p-hub center problem) as well as studying other p-hub center problems with different allocation strategies.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. Network flows: Theory, applications and algorithms. Prentice-Hall, Englewood Cliffs, New Jersey, USA Arrow, KJ (1963). Social Choice and Individual Values, Wiley, New York. Gibbard, A.(1973). Manipulation of Voting Schemes: A general result, Econometrica, 41:587–602, 1993.

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

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

[4] J.F. Campbell, A.T. Ernst, and M. Krishnamoorthy. Hub location problems. Facility Location: Applications and Theory, pages 373–407, 2002.

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

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

[7] A.T. Ernst and M. Krishnamoorthy. Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem. European Journal of Operational Research, 104(1):100–112, 1998.

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

[9] P. Hansen, N. Mladenović, and J.A.M. Pérez. Variable neighbourhood search: methods and applications. 4OR, 6(4):319–360, 2008.

[10] P. Hansen, N. Mladenović, and J.A.M. Pérez. Variable neighbourhood search: methods and applications. Annals of Operations Research, 175(1):367–407, 2010.

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

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

[13] N. Mladenović and P. Hansen. Variable neighborhood search. Computers & Operations Research, 24(11):1097–1100, 1997.

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