Serdica J. Computing **7** (2013), No 4, 343–354

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

VARIABLE NEIGHBORHOOD SEARCH FOR SOLVING THE CAPACITATED SINGLE ALLOCATION HUB LOCATION PROBLEM*

Miroslav Marić

Abstract. In this paper a Variable Neighborhood Search (VNS) algorithm for solving the Capacitated Single Allocation Hub Location Problem (CSAHLP) is presented. CSAHLP consists of two subproblems; the first is choosing a set of hubs from all nodes in a network, while the other comprises finding the optimal allocation of non-hubs to hubs when a set of hubs is already known. The VNS algorithm was used for the first subproblem, while the CPLEX solver was used for the second. Computational results demonstrate that the proposed algorithm has reached optimal solutions on all 20 test instances for which optimal solutions are known, and this in short computational time.

1. Introduction.

The capacitated Single Allocation Hub Location Problem (CSAHLP) is a well known flow network optimization problem, widely studied in the literature. It consists of two parts: in a given network, hubs must be established at certain nodes. Which nodes become hubs is the location part

ACM Computing Classification System (1998): G.1.6.
Key words: CSAHLP, VNS, CPLEX, Metaheuristic, Mathematical optimization.
*The paper is partly supported by the Serbian Ministry of Science within Project No. 174010.

of the problem. The second part is the allocation part: each non-hub node nh must be allocated to one of the established hubs h; all flow from and to the node nh will be routed through the hub h. Single allocation means that each node is allocated to precisely one hub. Hubs are allocated to themselves. Each hub has a predetermined capacity, meaning that the total flow of all non-hubs allocated to a hub must not surpass the hub's capacity. The reason for establishing hubs is that the cost of transferring flow between hubs is lower than if the flow went directly between non-hubs. However, establishing a hub incurs a certain cost. The objective of this optimization problem is to minimize the cost of establishing hubs and the cost of total flow in the network.

A good literature review on CSAHLP is given in [1]. The test problem instances used herein are the same as in [9].

2. Mathematical Model.

2.1. Full model. The model used herein was first introduced in this form in [2]. A fully connected network I, $|I| = n$ and two associated matrices W and C are given. Each element of the flow matrix W, W_{ij} represents the amount of flow from the node $i, i \in I$ (origin) to $j, j \in I$ (destination) and $W_{ij} \geq 0$ is assumed. There is no assumption about the symmetry of this matrix. The transportation costs per unit of flow are given by the matrix C. In general, the value C_{ij}, $i, j \in I$ is an abstract value and $C_{ij} \geq 0$ holds. However, in practical applications this value is often a function of the physical distance between nodes i and j. Having this in mind, for a given node l, if node $k, k \neq l$ is such that the value C_{lk} is lower than C_{lm} for any other node m, we will say that k is *closest* to l.

For each node $k \in I$ two values are given. G_k represents the capacity of node k, i.e., the total amount of flow that can go through this node if a hub is established at this location. On the other hand, f_k is the fixed cost of establishing a hub at node k. These are all input parameters of the $CSAHLP$ model.

$CSAHLP$ uses a formulation in which direct flow between non-hubs is not allowed. Each non-hub is allocated to one hub and hubs are allocated to themselves. Therefore, for $i, j \in I$, the total flow between i and j consists of three parts: collection from i to its hub $k, k \in I$, transfer from the hub k to the hub $l, l \in I$ to which j is allocated, and finally the delivery from l to j. Each of these segments incurs a different weighted cost, which is given by the parameters χ, α and δ respectively. Finally, the total cost per unit of flow between i and j via hubs k and l is equal to $\chi C_{ik} + \alpha C_{kl} + \delta C_{lj}$. Establishing hubs is meaningful only if the cost of transfer is lower than the cost of collection and delivery. Therefore, χ, α and δ is assumed.

Since all collection and delivery from and to the node i goes via the same hub k (single allocation), it is useful to define two new values for each node i. O_i defines the total out flow from i: $O_i = \sum_{j \in I} W_{ij}$. Similarly, D_j represents the total flow destined for j: $D_j = \sum_{i \in I} W_{ij}$.

To build a mathematical model, we need binary decision variables which will indicate which nodes are hubs and also to which hub each non-hub, allocated. To accomplish this, a set of binary variables $Z_{ij}, i, j \in I$ is introduced. First, $Z_{kk} = 1 \Leftrightarrow k \in I$ is a hub. Second, $Z_{ij} = 1 \Leftrightarrow i$ is allocated to j. Finally, a set of continuous variables $Y^i_{kl}, i, k, l \in I$ represents the total amount of flow from node i, collected by the hub k and transferred to the hub l.

Using this notation, $CSAHLP$ is formulated as follows:

$$\min \sum_{i \in I} \sum_{k \in I} C_{ik} Z_{ik} (\chi O_i + \delta D_i) + \sum_{i \in I} \sum_{k \in I} \sum_{l \in I} \alpha C_{kl} Y^i_{kl} + \sum_{k \in I} f_k Z_{kk} \tag{1}$$

subject to:

$$\sum_{k \in I} Z_{ik} = 1 \quad \text{for every } i \in I \tag{2}$$

$$Z_{ik} \le Z_{kk} \quad \text{for every } i, k \in I \tag{3}$$

$$\sum_{l \in I} Y^i_{kl} - \sum_{l \in I} Y^i_{lk} = O_i Z_{ik} - \sum_{j \in I} W_{ij} Z_{jk} \quad \text{for every } i, k \in I \tag{4}$$

$$\sum_{i \in I} O_i Z_{ik} \le G_k Z_{kk} \quad \text{for every } k \in I \tag{5}$$

$$\sum_{l \in I, l \ne k} Y^i_{kl} \le O_i Z_{ik} \quad \text{for every } i, k \in I \tag{6}$$

$$Y^i_{kl} \ge 0 \quad \text{for every } i, k, l \in I \tag{7}$$

$$Z_{ik} \in \{0, 1\} \quad \text{for every } i, k \in I. \tag{8}$$

The objective function (1) minimizes the sum of the fixed cost of establishing hubs and the collection, transfer and delivery costs between all origin and destination nodes via their corresponding hubs. The constraint set (2) ensures single allocation of each non-hub. Constraints (3) ensure that each hub must be allocated to itself and non-hub nodes can be allocated only to hub nodes. The flow conservation in the network is enforced by the constraint set (4). The constraint set (5) limits the collection at each hub to its capacity. The constraint set (6) was introduced in [2] and it ensures that the flow from node i goes only through the hub k to which it is allocated. The variables Y_{kl}^i are continuous and non-negative (7), while the variables Z_{ij} are binary (8).

2.2. Model for a fixed set of hubs. If the location part of the $CSAHLP$ is finished, i.e., the set of hubs $H \subseteq I$ is predetermined, the allocation of each non-hub node to a hub still needs to be done. In order to do this, for some $H, H \subseteq I$, we build a sub model $CSAHLP(H)$ which will find the optimal allocation and obtain the corresponding solution for this fixed set of hubs H.

For solving $CSAHLP(H)$ the formulation from the previous subsection can be used as a base and then simplified in an appropriate manner. The main change is that the Z_{ik} and Y_{kl}^i variables take indexes from different sets: for Z_{ik}, $i \in I \backslash H, k \in H$ and for Y_{kl}^i, $i \in I \backslash H, k, l \in H$. All other changes in the model are a consequence of this change.

We use the same definitions and notation as in the previous subsection and formulate $CSAHLP(H)$:

$$\begin{aligned} \min \sum_{k\in H} C_{kk}(\chi O_k + \delta D_k) &+ \sum_{i\in I\backslash H}\sum_{k\in H} C_{ik} Z_{ik}(\chi O_i + \delta D_i) + \alpha \sum_{k\in H}\sum_{j\in H} C_{kj} W_{kj} \\ &+ \alpha \sum_{i\in I\backslash H}\sum_{k\in H} Z_{ik} \sum_{l\in H} C_{kl} W_{li} + \sum_{i\in I\backslash H}\sum_{k\in H}\sum_{l\in H} \alpha C_{kl} Y_{kl}^i + \sum_{k\in H} f_k \end{aligned} \tag{9}$$

subject to:

$$\sum_{k\in H} Z_{ik} = 1 \quad \text{for every } i \in I\backslash H \tag{10}$$

$$\sum_{l\in H} Y_{kl}^i - \sum_{l\in H} Y_{lk}^i = (O_i - W_{ii}) Z_{ik} - W_{ik} - \sum_{j\in I\backslash H, j\neq i} W_{ij} Z_{jk} \quad \text{for every } i \in I\backslash H, k \in H \tag{11}$$

$$O_k + \sum_{i\in I\backslash H} O_i Z_{ik} \le G_k \quad \text{for every } k \in H \tag{12}$$

$$\sum_{l\in H, l\neq k} Y_{kl}^i \leq O_i Z_{ik} \quad \text{for every } i \in I\backslash H, k \in H \tag{13}$$

$$Y_{kl}^i \geq 0 \quad \text{for every } i \in I\backslash H, k, l \in H \tag{14}$$

$$Z_{ik} \in \{0,1\} \quad \text{for every } i \in I\backslash H, k \in H. \tag{15}$$

The objective function (9) contains 6 terms. The first term of the objective function (1), which represents collection and distribution, is split into two terms, one for hub nodes and one for non-hubs. The second term from (1) represents the transfer costs and in the $CSAHLP(H)$ model it is split into three terms: one for the transfer of the outflow W_{kj} from hub k to another hub j, one for the transfer of the outflow W_{li} from hub l to non-hub i and one for all other transfer costs, as defined by the variables Y_{kl}^i. Finally, the fixed costs are calculated only for the elements of the set H. The constraint set (3) becomes obsolete and all other constraint sets from $CSAHLP$ model are translated into constraints of the $CSAHLP(H)$ model by modifying the indexes of Z and Y variables.

It is obvious that the number of variables in $CSAHLP(H)$ model is $n \cdot h^2 + n \cdot h$, compared to $n^3 + n^2$ variables of the $CSAHLP$, where $n = |I|$ and $h = |H|$. If $h \ll n$ this results in a huge change in the number of variables and also in the number of constraints.

3. Variable Neighborhood Search.

Variable Neighborhood Search (VNS) is a metaheuristic for solving mathematical optimization problems introduced by Mladenović and Hansen in [8]. The basic idea is a systematic change of neighborhoods in two phases:

- Local search phase—when a local minimum is determined,
- Shaking phase—when escaping local minimum.

The VNS algorithm is applied to many classes of problems: location theory, cluster analysis, scheduling, vehicle routing, network design, lot-sizing, artificial intelligence, engineering, pooling problems, biology, phylogeny, reliability, geometry, telecommunication design, etc. ([6], [7], [5], [3]).

Let us denote by:

- S, the solution space;

- $f : S \to R$, the objective function;
- $X \subset S$, the feasible set given by the set of constraints over the solution space.

The problem of mathematical optimization is defined as finding:

$$\min\{f(x),\ x \in X\}.$$

Let us denote by:

- $N_k, k = 1, \ldots, k_{max}$ the sequence of a of pre-selected neighborhood structures;
- $N_k(x)$ the set of solutions in the k th neighborhood of x.

In shaking phase the current solution is moving to random solution in the k-th neighborhood of the current solution. The shake function is represented by Algorithm 1.

Algorithm 1 Shake

```
  Input: $initial_{sol}, k$
  Output: $new_{sol}$
  $new_{sol} \leftarrow$ random solution in $N_k(initial_{sol})$
```

In the local search phase a local minimum with respect to the neighborhood structure is determined. Two strategies, the first improvement and the best improvement strategy, are used. In the first improvement strategy the first determined solution in the given neighborhood which is better than the initial one is accepted, while in the best improvement strategy the best solution in the given neighborhood, better than the initial one is accepted. The procedure is repeated while there are improvements, i.e., until the local minimum is determined.

An example of a local search with the best improvement strategy, for $k_{max} = 1$, is illustrated in Algorithm 2.

The basic VNS algorithm consists of the repetition of two main phases: shake phase and local search phase, for as long as the criteria for stopping the algorithm are fulfilled. In Algorithm 3 the basic VNS is illustrated.

As an initial solution for the basic VNS algorithm, a variant of VNS named Reduced VNS (RVNS) is mainly used. The reduced VNS method is obtained if a random point is selected from $N_k(x)$ and no local search descent is made. If

Algorithm 2 Local Search

```
Input: $initial_{sol}$
Output: $new_{sol}$
$new_{sol} \leftarrow initial_{sol}$
$improvement \leftarrow$ true
while $improvement$ do
  $improvement \leftarrow$ false
  $temp_{sol} \leftarrow$ best solution in $N_1(new_{sol})$
  if $temp_{sol}$ is better than $new_{sol}$ then
    $new_{sol} \leftarrow temp_{sol}$
    $improvement \leftarrow$ true
  end if
end while
```

Algorithm 3 Basic VNS

```
Input: $initial_{sol}, k_{max}$
Output: $new_{sol}$
$new_{sol} \leftarrow initial_{sol}$
while not stopping condition is met do
  for $k \leftarrow 1$ to $k_{max}$ do
    $temp_{sol} \leftarrow new_{sol}$
    $temp_{sol} \leftarrow Shake(temp_{sol}, k)$
    $temp_{sol} \leftarrow LocalSearch(temp_{sol})$
    if $temp_{sol}$ is better than $new_{sol}$ then
      $new_{sol} \leftarrow temp_{sol}$
      $k \leftarrow 1$
    end if
  end for
end while
```

Algorithm 4 Reduced VNS

```
Input: $initial_{sol}, k_{max}$
Output: $new_{sol}$
$new_{sol} \leftarrow initial_{sol}$
while not stopping condition is met do
  for $k \leftarrow 1$ to $k_{max}$ do
    $temp_{sol} \leftarrow new_{sol}$
    $temp_{sol} \leftarrow Shake(temp_{sol}, k)$
    if $temp_{sol}$ is better than $new_{sol}$ then
      $new_{sol} \leftarrow temp_{sol}$
      $k \leftarrow 1$
    end if
  end for
end while
```

the new solution is better than the initial one, the new solution is accepted. In Algorithm 4 the reduced VNS is illustrated.

4. VNS for CSAHLP.

In this paper a combination of RVNS and the basic VNS algorithm, described in Section 3, is used.

4.1. Solution encoding and neighborhood structures. The solution of the problem is encoded as an array H of values {0, 1}. The value 1 in the i th place in the array means that the corresponding hub $i \in H$ is allocated. The value 0 means that the corresponding hub is not allocated. Solution x' is in the k th neighborhood of solution x, $x' \in N_k(x)$, if x' can be obtained from x with at most k inversions in array H. One inversion is defined as changing the value of exactly one bit in solution H, from 0 to 1 or 1 to 0.

4.2. Objective function calculation. When an array of allocated hubs H is obtained by VNS algorithm, the objective function is calculated by solving the sub-model $CSAHLP(H)$, described in Section 2.2. This will find the optimal allocation and obtain the corresponding solution for fixed set of hubs. For solving the sub-model the CPLEX solver is used.

4.3. Initial solution. The initial solution for RVNS is a feasible solution of the problem obtained by the Monte Carlo algorithm. In the Monte Carlo algorithm random solutions are generated until a feasible one is found. The solution obtained by the RVNS algorithm is used as an initial solution for VNS algorithm.

5. Computational Results. The proposed algorithm was tested on 20-well known instances from the literature, which include from 10 to 50 potential hubs. All optimal solutions from the literature are obtained in reasonable time. VNS is implemented in the C# programming language, while the sub model solver is implemented in CPLEX, version 12.6. All the tests were carried out under the Windows 7 operating system, with an Intel core i7 860 processor and 8 GB ram memory.

Value $k_{\max}$ in the RVNS algorithm is set to 2, while $k_{\max}$ in the VNS algorithm is set to $k_{\max} = \min(n/5, 10)$. The stopping condition for the initial Monte Carlo algorithm is finding a feasible solution or 100 successive iterations without improvement. The stopping condition for the RVNS algorithm is 100 successive iterations without improvement, while the stopping condition for the basic VNS algorithm is 50 successive iterations without improvement. If at any time the program execution time exceeds $20 \cdot n$ seconds, execution is interrupted and the best found solution is taken as the result. For each instance the program runs 20 times on 20 different random seeds.

Table 1 presents the results of this algorithm on 20 test problems. The columns represent respectively the test instance name, the optimal solution from the literature, the solution obtained by the VNS algorithm, the time in seconds needed for the VNS to obtain the solution, the average gap, as percentage, of the obtained solution from the optimal solution and the standard deviation, also in percents, of the obtained solution from the optimal solution. The mark *opt* in the third column means that the same solution as the optimal has been obtained.

In order to solve larger instances, particularly those with 50 nodes, more CPU time is needed. For this reason, additional tests were performed by limiting the execution time to 60 seconds (VNS (60s)). A comparison of these time-limited tests with both the Simulated Annealing algorithm (SA) from [4] and the VNS without the 60 seconds limit is presented in Table 2. The comparison is organized as follows: the first column represents the instance name and it is followed by both minimal upper gap and average gap in percentages for all three methods VNS, VNS (60s) and SA, respectively. For every solution *sol* the upper gap is defined as

$$gap = \left(\frac{sol}{sol_{optimal}} - 1 \right) \cdot 100.$$

The results in Table 2 demonstrate that only the VNS algorithm has found optimal solutions for all 4 test instances. The SA algorithm from [4] reached optimal solution on 3 test instances, while the VNS (60s) algorithm obtained solutions close to the optimal ones, but not optimal. The smallest average gaps

Table 1. Computational results

Instance	Optimal Solution	Solution	Time [s]	agap [%]	sigma [%]
hub10LL	224250.05	opt	0.16	0.00	0.00
hub10LT	250992.26	opt	0.42	0.00	0.00
hub10TL	263399.94	opt	0.07	0.00	0.00
hub10TT	263399.94	opt	0.03	0.18	0.76
hub20LL	234690.96	opt	0.91	0.00	0.00
hub20LT	253517.40	opt	0.79	0.00	0.00
hub20TL	271128.18	opt	1.82	0.26	0.78
hub20TT	296035.40	opt	3.16	0.00	0.00
hub25LL	238977.95	opt	6.06	0.00	0.00
hub25LT	276372.50	opt	6.31	0.20	0.27
hub25TL	310317.64	opt	7.62	0.00	0.00
hub25TT	348369.15	opt	10.84	0.21	0.42
hub40LL	241955.71	opt	32.85	0.00	0.00
hub40LT	272218.32	opt	68.13	1.67	2.37
hub40TL	298919.01	opt	38.86	0.00	0.00
hub40TT	354874.10	opt	94.86	2.37	5.45
hub50LL	238520.59	opt	89.79	0.36	0.71
hub50LT	272897.49	opt	171.54	1.83	2.36
hub50TL	319015.77	opt	82.40	0.48	0.75
hub50TT	417440.99	opt	637.74	5.63	5.09

were achieved with the SA algorithm, slightly greater average gaps appeared using thr VNS algorithm, while reported values of average gap for the VNS algorithm with 60 seconds limitation were not satisfactory.

6. Conclusion. The VNS algorithm for solving the CSAHLP is presented herein. In the location part of the problem local search and shaking are used to explore the neighborhoods of a given solution and find promising new hub configurations. An exact optimizer (CPLEX) is used for the allocation part

Table 2. Comparison—VNS, VNS (60s), SA

Instance	VNS		VNS (60s)		SA	
	min gap	agap	min gap	agap	min gap	agap
hub50LL	0.00	0.36	4.27	50.54	0.00	0.00
hub50LT	0.00	1.83	10.16	48.06	0.00	0.40
hub50TL	0.00	0.48	22.46	82.89	0.00	0.08
hub50TT	0.00	5.63	36.83	83.67	0.71	1.32

of non-hub nodes to hubs. The results demonstrate that the algorithm is very efficient in terms of computational time, while reaching optimal solutions on all test instances. Further research is aimed towards applying the same approach to other hub location problems, namely, the Uncapacitated Single Allocation Hub Location problem (USAHLP) and others.

REFERENCES

[1] Alumur S., B. Y. Kara Network hub location problems: The state of the art. *European Journal of Operational Research*, **190** (2008), No 1, 1–21.

[2] Correia I., S. Nickel, F. S. da Gama. The capacitated single-allocation hub location problem revisited: A note on a classical formulation. *European Journal of Operational Research*, **207** (2010), No 1, 92–96.

[3] Davydov I., Y. Kochetov, E. Carrizosa. VNS heuristic for the centroid problem on the plane. *Electronic Notes in Discrete Mathematics,* **39** (2012), 5–12.

[4] Ernst A., M. Krishnamoorthy. Solution algorithms for the capacitated single allocation hub location problem. *Annals of Operations Research*, **86** (1999), 141–159.

[5] García-López F., B. Melián-Batista, J. A. Moreno-Pérez, J. Moreno-Vega. The Parallel Variable Neighborhood Search for the p-Median Problem. *Journal of Heuristics,* **8** (2002), No 3, 375–388.

[6] Hansen P., N. Mladenovic, J. Brimberg, J. Prez. Variable Neighborhood Search. Handbook of Metaheuristics (Eds M. Gendreau, J.-Y. Potvin), International Series in Operations Research & Management Science, Vol. **146**, Springer US, 2010, 61–86.

[7] Kratica J., A. Savić, V. Filipović, M. Milanović. Solving the task assignment problem with a variable neighborhood search. *Serdica Journal of Computing*, **4** (2010), No 4, 435-446.

[8] Mladenovic N., P. Hansen. Variable neighborhood search. *Computers & Operations Research*, **24** (1997), No 11, 1097–1100.

[9] STANIMIROVIC Z. Solving the Capacitated Single Allocation Hub Location Problem Using Genetic Algorithm. *Recent Advances in Stochastic Modelling and Data Analysis*, 2007, 464–471.

Miroslav Marić
Department of Computer Science and Informatics
Faculty of Mathematics, University of Belgrade
Studentski trg 16, 11 000 Belgrade, Serbia
e-mail: `maricm@matf.bg.ac.rs`

Received May 12 11, 2014
Final Accepted July 24, 2014