Variable Neighbourhood Search for Uncapacitated Warehouse Location Problems

Kemal Alaykiran, Mehmet Hacibeyoglu

***Abstract*— Uncapacitated warehouse location problem (UWLP) is one of the basic problems of operations research and supply chain management literature. In this paper, four variations of variable neighbourhood search (VNS) algorithms are used where the first three are fundamental algorithms which are used and proposed in the literature where the fourth one is a hybrid use of iterated local search (ILS) and variable neighbourhood search (VNS), to solve the instances at a well-known and studied problem set. There are 15 problem instances with increasing number of potential warehouses and customers in this set. The results show that the proposed variation of the algorithm outperforms the traditional versions of variable neighbourhood search.**

***Index Terms*— Iterated Local Search, Optimization, Uncapacitated warehouse location problem, Variable neighbourhood search.**

I. INTRODUCTION

Uncapacitated warehouse location problem is one of the basic problems studied in supply chain management literature and also in operations research literature. The problem may be briefly defined as finding which warehouses to be opened in a set of candidates to satisfy the deterministic demand of a number of customers in order to minimize the total cost which consists of fixed opening costs and transportation costs from warehouse to the customer. The solution methods used in the literature may be divided into two broad classes: Studies using exact methods and studies using heuristic and metaheuristic methods. Since the main concentration of this paper is to investigate the performance of VNS algorithms on this problem, the studies based on exact methods are considered to be out of the scope of this paper and the studies based on heuristic and metaheuristic methods are taken into account. At [1], some location problems including UWLP, capacitated warehouse location problem, p-median and capacitated warehouse location problem with single source constraints are solved using Lagrangean heuristics. At [2], UWLP is solved using Tabu Search algorithm. Another Tabu search approach is given and applied at [3]. At [4], the problem is attacked using genetic algorithms. At [5], the problem is solved using neighbourhood search heuristics.

In this paper, UWLP is solved using four different VNS algorithms where the first three are classical algorithms of VNS. The fourth one which is the proposed method is a combination of VNS and iterated local search (ILS). The performance of the algorithms is investigated on the OR-Library instances which were first introduced at [6]. The performance of the algorithms is investigated based on the solution quality and CPU time consumed for solution. At the next chapter, the problem is defined and an example is given. At section three, the VNS algorithms used to solve this problem is given in detail with the procedures applied. At section four, the computational results are examined. At section five the paper is concluded.

Kemal Alaykiran, Department of Industrial Engineering, Faculty of Engineering and Architecture, Necmettin Erbakan University, Konya, Turkiye
Mehmet Hacibeyoglu, Department of Computer Engineering, Faculty of Engineering and Architecture, Necmettin Erbakan University, Konya, Turkiye

II. Problem Definition

Uncapacitated warehouse location problem (UWLP) which is also named in the literature as simple facility location problem or fixed charge facility location problem is one of the basic and classical problems in the supply chain literature. The problem may be briefly defined as finding which warehouses to be opened in a set of candidates to satisfy the deterministic demand of a number of customers in order to minimize the total cost which consists of fixed opening costs and transportation costs from warehouse to the customer. This problem defined in words is first modelled in [7]. The parameters, decision variables, objective function and the model structure is given below ([7], [8]):

Parameters:

I : Set of customers;
J : Set of candidate locations;
f_j : Fixed cost of opening warehouse at candidate location j where $j \in J$;
h_i : Total demand of customer i where $i \in I$;
c_{ij} : Unit cost of transportation from candidate location j to customer i.

Decision Variables:

$$X_j = \begin{cases} 1 & \text{if the warehouse at candidate location } j \text{ is opened} \\ 0 & \text{else} \end{cases}$$

Y_{ij}: The ratio of the demand of customer i satisfied by warehouse j

Objective function

$$Minimum \sum_{j\in J} f_j X_j + \sum_{j\in J}\sum_{i\in I} h_i c_{ij} Y_{ij} \qquad (2.1)$$

s.t.

$$\sum_{j \in J} Y_{ij} = 1 \qquad \forall i \in I \tag{2.2}$$

$$Y_{ij} \le X_j \qquad \forall i \in I, \forall j \in J \tag{2.3}$$

$$X_j \in \{0,1\} \qquad \forall j \in J \tag{2.4}$$

$$Y_{ij} \ge 0 \qquad \forall i \in I, \forall j \in J \tag{2.5}$$

In this model the objective function (2.1) is the total of fixed opening costs and transportation costs from these warehouses to the customers. Constraint (2.2) guarantees that all customer demand is satisfied. Constraint (2.3) assures that a transportation from a warehouse to any customer is occurred only if that candidate location is opened. With (2.4) and (2.5) the type of variables is clarified where opening decisions are binary variables and transportation variables are continuous. Under these circumstances, the solution space for a UWLP is $2^J - 1$ where there are J candidate locations. The only one solution which is infeasible is the one where none of the candidate locations are opened. Also it is clear that all Y_{ij} variables must be either 0 or 1 since there are no capacity restrictions. An example problem with 10 customers and 10 warehouses is given below (see Table I). At Table I, total demand cost for each customer and each candidate location is given. In this problem all fixed opening costs for the candidate locations is assumed to be 9000 ($f_j = 9000, \forall j \in J$).

The solution of this problem is a binary vector which denotes to which candidate location is opened. As mentioned before this problem has a solution space of $2^{10} - 1 = 1023$ feasible vectors. Two examples where the second one is the optimal solution for this problem are given.

Table I. Total demand cost of customers $(h_i * c_{ij})$

Customer / Warehouse	1	2	3	4	5	6	7	8	9	10
1	1348	2071	1530	1044	1155	1328	875	769	1286	1079
2	641	1091	769	479	526	644	368	453	624	517
3	983	5282	3924	2775	1830	2995	4370	7066	3022	4736
4	6474	5996	4205	5936	4255	4014	12859	16037	5184	13841
5	343	430	316	212	250	273	305	191	264	358
6	1284	4740	3239	2077	1497	2467	3168	5450	2489	3554
7	16394	5700	8627	13154	11761	9711	27723	31059	10635	29465
8	6678	5309	1274	3354	2714	1772	10310	11582	2197	11475
9	404	496	374	265	305	329	363	242	319	420
10	292	399	280	174	210	236	227	109	227	281

Example solution 1:

Solution Vector	1	**2**	3	**4**	**5**	**6**	7	8	**9**	10	Cost
	0	**1**	0	**1**	**1**	**1**	0	0	**1**	0	45000
1	1348	2071	1530	**1044**	1155	1328	875	769	1286	1079	1044
2	641	1091	769	**479**	526	644	368	453	624	517	479
3	983	5282	3924	2775	**1830**	2995	4370	7066	3022	4736	1830
4	6474	5996	4205	5936	4255	**4014**	12859	16037	5184	13841	4014
5	343	430	316	**212**	250	273	305	191	264	358	212
6	1284	4740	3239	2077	**1497**	2467	3168	5450	2489	3554	1497
7	16394	**5700**	8627	13154	11761	9711	27723	31059	10635	29465	5700
8	6678	5309	1274	3354	2714	**1772**	10310	11582	2197	11475	1772
9	404	496	374	**265**	305	329	363	242	319	420	265
10	292	399	280	**174**	210	236	227	109	227	281	174
										Total Cost:	**61987**

Example solution 2:

Solution Vector	1	2	3	4	5	**6**	7	8	9	10	Cost
	0	0	0	0	0	**1**	0	0	0	0	9000
1	1348	2071	1530	1044	1155	**1328**	875	769	1286	1079	1328
2	641	1091	769	479	526	**644**	368	453	624	517	644
3	983	5282	3924	2775	1830	**2995**	4370	7066	3022	4736	2995
4	6474	5996	4205	5936	4255	**4014**	12859	16037	5184	13841	4014
5	343	430	316	212	250	**273**	305	191	264	358	273
6	1284	4740	3239	2077	1497	**2467**	3168	5450	2489	3554	2467
7	16394	5700	8627	13154	11761	**9711**	27723	31059	10635	29465	9711
8	6678	5309	1274	3354	2714	**1772**	10310	11582	2197	11475	1772
9	404	496	374	265	305	**329**	363	242	319	420	329
10	292	399	280	174	210	**236**	227	109	227	281	236
										Total Cost:	**32769**

III. Methods

In this study, four different methods are used to solve UWLP. The first three are classical VNS methods which are variable neighbourhood descent (VND), the second one is basic VNS (BVNS), the third one is reduced VNS (RVNS). The fourth one is a hybrid use of ILS and VNS. The definition of the algorithms used are given step-by-step below.

A. Variable Neighbourhood Descent (VND)

VND is a VNS algorithm, it starts with an initial solution in a neighbourhood and at every iteration the best neighbour is found and the procedure continues until no improvement is achieved for a predetermined parameter (k) times. The neighbourhood structure is found using 1-opt add/remove method for all possible neighbourhoods. For more information on VND one may refer to [9].

```
procedure VND
$s_0 \leftarrow$ GenerateInitialSolution,
neighbourhood structure $N(k), k=1,.., k_{max}$
$k_{max} \leftarrow$ NumberofWarehouses
$k \leftarrow 1$
repeat
    $s^* =$ BestNeigbourhood(s)
    if $(s^*) < f(s)$ then
        $s \leftarrow s^*$
        $k \leftarrow 1$
    else
        $k = k + 1$
until $k > k_{max}$
```

B. Basic Variable Neighbourhood Search (BVNS)

BVNS is one of the core algorithms of VNS. Here, again the procedure starts with an initial solution and this time a shake mechanism is used on the initial solution and a local search procedure is applied to the neighbour solution found as a result of shake mechanism. In this algorithm 1-opt add/remove method is used for the local search until no improvement occurs in a number of iterations which is considered to be half of the number of potential warehouses. Furthermore, 4-opt add/remove method is used for the shake mechanism. For information on BVNS one may refer to [10].

```
procedure BVNS
$s_0 \leftarrow$ GenerateInitialSolution,
neighbourhood structure $N(k), k=1,.., k_{max}$
$k_{max} \leftarrow$ NumberofWarehouses
$k \leftarrow 1$
repeat
    $s' =$ Shake(s)
    $s^{*'} =$ LocalSearch(s')
    if $(s^{*'}) < f(s)$ then
        $s \leftarrow s^{*'}$
        $k \leftarrow 1$
    else
        $k = k + 1$
until $k > k_{max}$
```

C. Reduced Variable Neighbourhood Search (RVNS)

RVNS is the most basic algorithm of VNS where the solution procedure starts with an initial solution and a shake mechanism is followed to improve this solution.

```
procedure RVNS
$s_0 \leftarrow$ GenerateInitialSolution,
neighbourhood structure $N(k), k=1,.., k_{max}$
$k_{max} \leftarrow$ NumberofWarehouses
$k \leftarrow 1$
repeat
    $s' =$ Shake(s)
    if $f(s') < f(s)$ then
        $s \leftarrow s'$
        $k \leftarrow 1$
    else
        $k = k + 1$
until $k > k_{max}$
```

Since the method is easy to apply and fast to run but the solution quality is the worst amongst all other VNS algorithms since no local search is applied to the solution. Here, the shake mechanism applied is a 4-opt add/remove method. For more information one may refer to [11].

D. Hybrid ILS-VNS

In this algorithm, in order to improve the solution quality and decrease the CPU time to find reasonable solutions to UWLP, both ILS and VNS features are used together. ILS is an extension of local search where the objective is to decrease the solution space to a minor neighbourhood. For more information about ILS one may refer to [12].

```
procedure ILS-VNS
$s_0 \leftarrow GenerateInitialSolution,$
neighbourhood structure $N(k)$, $k=1,.., k_{max}$
$k_{max} \leftarrow NumberofWarehouses$
$k \leftarrow 1$
$s^{global} = s;$
repeat
    repeat
        $s^* = BestNeigbourhood(s)$
        if $f(s^*) < f(s)$ then
            $s \leftarrow s^*$
            $s^{local} = s;$
    until $f(s^*) > f(s)$
    if $f(s^{global}) > f(s^{local})$ then
        $s^{global} \leftarrow s^{local}$
        $k \leftarrow 1$
    else
        $k = k + 1$
    $s = Perturbation(s)$
until $k > k_{max}$
```

The procedure starts with an initial solution as usual and the best neighbour of this solution is found using 1-opt add/remove method for all possible neighbourhoods. If an improvement is achieved then the procedure continues with this new neighbour. If the current solution is not improved then a perturbation mechanism is applied with an uniform random number between [2,6]-opt add/remove method. The procedure continues running until the total number of runs (kmax) is met.

IV. Computational Results

In order to analyze the performance of the above described VNS algorithms a well-known benchmark problem set is used. This problem set is a common one which was first described at [6] and published at OR-library. This data set consists of 15 problem instances with increasing values of number of potential warehouses. The problems in this set are solved using VND, BVNS, RVNS and ILS+VNS algorithms. All algorithms are run 20 times for each problem instance on a computer with an I7-3840QM CPU at 2.80 GHz with 16 GB of ram working on Windows 10 operating system. The algorithms are coded using C# on Microsoft Visual Studio 2013 builder. Since the optimal values for the problem instances at this problem set are already known, the average percentage error values and CPU times are recorded. The numerical results obtained are shown at Table II. These results are summarized at Figure 1 graphically.

Table II. Properties of the instances and the numerical results for the problem set

Dataset Name	Number of		Average Percentage Error				Average CPU -TIME			
	Warehouses	Customers	VND	BVNS	RVNS	ILS-VNS	VND	BVNS	RVNS	ILS-VNS
CAP71	16	50	0,04	0,03	0,88	0,00	0,0005	0,0012	0,0009	0,0005
CAP72	16	50	0,11	0,03	1,15	0,00	0,0006	0,0013	0,0005	0,0004
CAP73	16	50	0,20	0,04	1,36	0,00	0,0006	0,0013	0,0009	0,0003
CAP73	16	50	0,00	0,05	2,88	0,00	0,0008	0,0014	0,0007	0,0004
CAP101	25	50	0,20	0,04	1,47	0,01	0,0012	0,0033	0,0014	0,0022
CAP102	25	50	0,10	0,05	1,51	0,00	0,0012	0,0027	0,0014	0,0009
CAP103	25	50	0,14	0,08	2,07	0,01	0,0012	0,0027	0,0012	0,0029
CAP104	25	50	0,45	0,03	3,59	0,00	0,0014	0,0021	0,0012	0,0008
CAP131	50	50	0,70	0,10	2,66	0,01	0,004	0,015	0,006	0,008
CAP132	50	50	0,54	0,03	4,05	0,00	0,004	0,012	0,005	0,004
CAP133	50	50	0,32	0,09	4,40	0,00	0,004	0,010	0,005	0,007
CAP134	50	50	0,67	0,03	8,01	0,00	0,005	0,008	0,005	0,003
CAPA	100	1000	4,93	1,22	38,08	0,00	0,37	1,09	0,55	0,87
CAPB	100	1000	2,57	0,75	24,78	0,15	0,34	1,21	0,43	1,41
CAPC	100	1000	3,15	0,24	19,56	0,07	0,33	1,14	0,42	1,81

Considering the average percentage errors it may found out from the values at Table 1 that RVNS shows the worst performance where ILS+VNS algorithm yields the best results for all problems. From another point of view when average CPU times are considered, instead of CAPA, CAPB and CAPC instances all algorithms finish running under 1 second. As a result of this, only the CPU times for the problems CAPA, CAPB and CAPC may be evaluated. Here, VNS and RVNS algorithms are found to be run approximately for 1 second where ILS+VNS algorithm run approximately between 1 and 2 seconds. It may be concluded from these results that the average CPU values may not be compared since the values are so close but considering the average percentage errors it may be easily figured out that BVNS and ILS+VNS outperformed RVNS and VND where the best results are obtained from the runs with ILS+VNS.

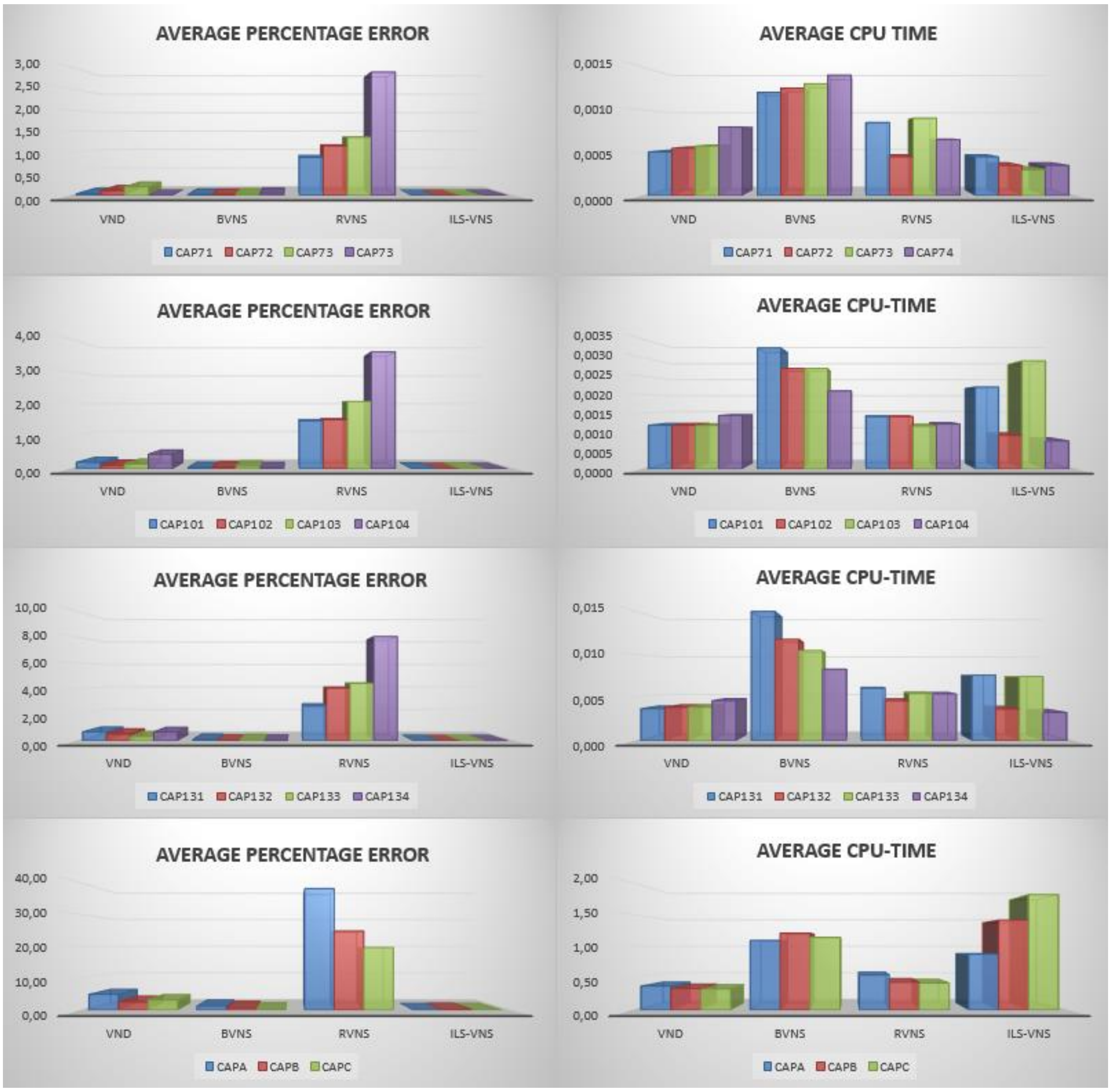

Fig. 1. Graphical representation of computational results

When the results are evaluated, it is seen that RVNS found the optimal value for 7 of the instances where this number is 3 for VND, 15 for both BVNS and ILS+VNS. But, when the average percentage errors are considered, it may be seen that the descending order of the success rates for the algorithms is ILS+VNS, BVNS, VND and RVNS. From another point of view, when the CPU times are considered it may be seen that VND algorithm works the fastest where the results yielded are not acceptable compared to BVNS and ILS+VNS.

As a result, due to the computational results, it may be seen that ILS+VNS algorithm outperformed other algorithms. However, the algorithm searches for the best neighbour at every iteration which results in an increase in the CPU time. It may be assumed that if the number of potential warehouses increase, the CPU time of ILS+VNS algorithm will increase.

V. Conclusion

In this paper, uncapacitated warehouse location problem (UWLP) which is a basic problem of supply chain management and operations research literature is studied with four algorithms of variable neighbourhood search method. The first three algorithm of VNS are fundamental algorithms of VNS which are studied and proposed in the literature which are reduced VNS (RVNS), variable neighbourhood descent (VND) and basic variable search (VNS). The fourth and the proposed VNS algorithm is named as ILS+VNS which includes features of both iterated local search and variable neighbourhood search together. In order to evaluate the performance of the mentioned algorithms, a well-known data set from OR-Library which is a common data set used in the literature by various researchers is used. All algorithms are coded and run for analyzing the average of percentage errors and total CPU times consumed for the problems in the data set. As a result of this computational study, it is seen that ILS+VNS algorithm yielded the best results for the average of percentage errors while VND is found to be the best for the

CPU times compared to other algorithms used in this study.

REFERENCES

[1] Beasley, J., "Lagrangean heuristics for location problems". Eur. J. Oper. Res., 65, 383–399, 1993.
[2] Laurent, M. And Hentenryck, P.V., "A Simple Tabu Search for the Warehouse Location", Eur. J. Oper. Res., 157, 576-591, 2004.
[3] Sun, M.H., "Solving the Uncapacitated Facility Location Problem Using Tabu Search", Computers & Operations Research, 33, 2563-2589, 2006.
[4] Kratica, J., Tosic, D., Filipovic, V., Ljubic, I. , "Solving the simple plant location problem by genetic algorithm", RAIRO Operations Research, 35, 127–142, 2001.
[5] Ghosh, D., "Neighborhood search heuristics for the uncapacitated facility location problems", Eur. J. Oper. Res., 150, 150-162, 2003
[6] Beasley, J., "OR-Library: Distributing Test Problems by Electronic Mail", Journal of the Operational Research Society, 41, 3, 1069-1072, 1990.
[7] Balinski, M., "Integer Programming: Methods, Uses, Computation", Management Science, 12, 253-313, 1965.
[8] Daskin, M.S., "Network and Discrete Location", John Wiley & Sons, New York, USA, 55-145, 1995.
[9] Hansen, P., Mladenovic, N., Perez, J., "Variable neighbourhood search: methods and applications", Ann. Oper. Res., 175, 367-407, 2010.
[10] Hansen, P., Mladenovic, N., "Variable neighbourhood search: Principles and applications", Eur. J. Oper. Res.130, 449-467, 2001.
[11] Xiao, Y., Kaku, I., Zhao, Q., Zhang, R., "A reduced variable neighbourhood search algorithm for uncapcitated multilevel lot-sizing problems", Eur. J. Oper. Res., 214, 223-231, 2011.
[12] Lourenco, HR., Martin, O., Stützle, T., "Iterated local search. In: Handbook of meta-heuristics", Berlin, Heidelberg: Springer-Verlag, 2003.

Kemal Alaykıran is currently working as an Assistant Professor at the Industrial Engineering Department of Necmettin Erbakan University (Konya, Turkey). He received the B.Sc., M.Sc., degrees in Industrial Engineering at Selcuk University and Ph.D. Degree in Industrial Engineering at Gazi University (Ankara, Turkey). His research interests are mathematical modeling, optimization and simulation of systems.

Mehmet Hacibeyoglu is currently Assistant Professor at the Computer Engineering Department of the Konya Necmettin Erbakan University (Konya, Turkey). He received the B.Sc., M.Sc., and Ph.D. Degrees in Computer Engineering from the Selcuk University (Konya, Turkey) in 2003, 2006, and 2012, respectively. His research interests are system administration in Unix and Linux, machine learning, feature selection, logic circuits, meta heuristic algorithms, information security, and data mining.