Jurnal Teknik Industri, Vol. 15, No. 2, Desember 2013, 95-102
ISSN 1411-2485 print / ISSN 2087-7439 online
DOI: 10.9744/jti.15.2.95-192

A Variable Neighborhood Search-Based Heuristic for the Multi-Depot Vehicle Routing Problem

Arif Imran[1]

Abstract: The multi-depot vehicle routing problem (MDVRP) is addressed using an adaptation of the variable neighborhood search (VNS). The proposed VNS algorithm besides using several neighborhoods and a number of local searches has a number of additional features. These include a scheme for identifying borderline customers, a diversification procedure and a mechanism that aggregates and disaggregates routes between depots. The proposed algorithm is tested on the data instances from the literature and produces competitive results.

Keywords: Meta-heuristic, routing, multi-depot, variable neighborhood.

Introduction

The multi depot vehicle routing problem (MDVRP) can be found in many logistics companies as many logistics companies operate from more than one depot to serve their customers. In this problem, we are given a number of customers, n, and a number of depots, m. Each customer must be served by one vehicle only and each vehicle must start and finish its journey at a depot. The capacity of a vehicle and the maximum length of a route must not be exceeded. The objective is to find the least cost routes by considering several depots.

There are several published papers addressing the MDVRP. The first heuristic is developed by Tillman [29]. He used the Clarke and Wright saving criterion [6]. Tillman and Cain [30] incorporated the procedure in Tillman [29] within a partial enumerative scheme that maximizes a saving criterion. Wren and Holliday [31]; Gillett and Johnson [11] presented a two-phase algorithm that utilised the sweep procedure. Golden *et al.* [13] put forward two algorithms. The first one constitutes a modification of the saving method of Yellow [32], and the second is a two-stage approach based on an assignment first and route second. In this work, they introduce the borderline customer concept to assign customers to the depots, which we are using later in this study. Laporte *et al.* [19] proposed a branch and bound algorithm to address the MDVRP with the symmetric distance case. They also presented an algorithm for solving the asymmetric case (Laporte *et al.*, [20]).

Chao *et al.* [4] developed a multi-phase heuristic that use a search procedure based on the record-to-record travel algorithm of Dueck [10] and the 2-opt of Lin [21]. Renaud *et al.* [25] and Cordeau *et al.* [7] put forward a tabu search approach.

[1]* Department of Industrial Engineering, National Institute of Technology, Jl. P.H.H Mustafa 23 Bandung 40124, Indonesia. Email: arifimr@yahoo.com

Salhi and Sari [27] used multi-level heuristic which enhanced by two reduction tests which made it considerably faster when compared to other heuristics with no serious effect on the solution quality. Two hybrid genetic algorithms are developed by Ho *et al.* [14]. Yu *et al.* [33] put forward an ant colony optimization with weight and mutation strategy (ACO-WM) and an ant colony optimization with parallel improvement (PIACO) to solve the MDVRP.

Some Applications

The MDVRP is used in many practical problems. Cassidy and Bennet [3] developed a two phase approach to address the school meal delivery problem. A two-phase approach, a route first cluster second, is presented by Ball *et al.* [1] for the distribution of chemical product in the USA and Canada. Perl and Daskin [24] incorporate in their location-routing formulation to solve the distribution of manufacturing products in the USA. Benton [2] put forward a saving method combined with a branch and bound based algorithm to address delivery to retail outlets from a bakery in Indiana. A combination of linear programming and heuristics is presented by Klot *et al.* [15] to tackle the distribution problem of dairy products in Haifa, Israel. Min *et al.* [22] used a combination of exact methods and heuristic for backhauls to address a distribution problem of the hardware products in the USA. Tarantilis and Kiranoudis [28] used the list-based threshold accepting (LBTA) algorithm to solve an open MDVRP that was faced by the Greek industry distributing meat from depots to butchers' shops. The proposed algorithm was able to improve the operations of the company.

We propose an adaptation of the basic VNS algorithm of Mladenovic and Hansen [23] to solve the MDVRP. This implementation is similar to the one designed for the multi-depot heterogeneous vehicle routing problem (MDHFVRP) in Imran [16] and Salhi *et al.* [26] except here it is used to deal with

homogeneous vehicles and in this paper more calculations are applied.

The paper is organized as follows. The proposed VNS algorithm is presented in the methods section, followed by a brief explanation of its main steps in. Computational results are analyzed in result and discussion section. The last section summarizes the findings and highlights some research avenues that worth pursuing in the future.

Methods

A Variable Neighborhood Search Method

The basic VNS algorithm is presented in Figure 1. VNS starts by selecting a set of neighbourhood structures N_k, where N_k is the neighbourhood in k distances, and by generating an initial solution x. A random point x'in $N_k(x)$ is generated. A local search is performed to find x''. If $x''$is better than x then $x = x''$, and the search returns to $N_1(x)(k = 1)$, otherwise the search continues to $N_2(x)(k = k + 1)$. This inner loop is repeated until $k = k_{max}$. The algorithm is stopped after a certain number of stopping criteria such as the maximum total number of iterations, the maximum CPU time, or the maximum number of iterations between two successive improvements or when k_{max} is reached.

An Overview of the Proposed Algorithm

At the beginning, we split the customers into two subsets; one for those who will be served from their nearest depots and the others which we refer to as the *borderline customers*. Each depot is initially considered individually and solved as a standard VRP. In each depot, giant tours are first created using the sweep method and then refined by the 2-opt procedure. Dijkstra's algorithm is then used for each giant tour to construct its corresponding optimal partitioning (i.e., the routes). As optimality is guaranteed only on the chosen giant tour, the solution is then improved using our VNS-based heuristic for this set of routes.

All borderline customers are then inserted into their best possible place in the routes originating from their nearest or the second nearest depot. We then treat all routes from all depots together by considering them as one large set of routes identified by their first and end node which correspond to their originating depot. The implementation of the VNS is then applied over all these routes to improve the overall solution. Once this is completed this large set of routes is split back to its original subset of routes, each served from one depot only. A diversification procedure is then applied to each depot and its corresponding routes. This process of having all routes as one large set of routes and re-arranging them into single depot problems and vice versa is repeated several times until a stopping criterion is met. In addition, at the last stage we also re-arrange the routes into their corresponding depots where Dijkstra's algorithm is then used as our final post optimiser. If this produces an improved solution, the entire search is repeated by putting all routes together again where VNS is activated, otherwise the search terminates.

Initialization. Select a set of neighbourhood structures N_k, for $k = 1, \dots, k_{max}$ that will be used in the search; find an initial solution x and choose a stopping condition;
Repeat the following sequence until the stopping condition is met:
(1) Set $k \leftarrow 1$
(2) *Repeat* the following steps until $k = k_{max}$
(a)*Shaking* Generate a point x' at random from the k^{th} neighbourhood of $x(x' \in N_k(x))$
(b)*Local search* Apply some local search method with x' as an initial solution; denote with $x''$the so obtained local optimum;
(c)*Move or not* If the local optimum x'' is better than the incumbent x, move there $(x \leftarrow x'')$, and continue the search with $N_1(k \leftarrow 1)$; otherwise, set $k \leftarrow k + 1$ and go to Step 2(a).

Figure 1. VNS Algorithm

The Algorithm

Figure 2 summarizes the main steps of the VNS-based heuristic. We first provide the necessary notations.

NbDivMax	:	the maximum number of diversifications
maxDijk	:	the maximum number of times Dijkstra algorithm is used
NbDiv	:	a counter for the number of diversifications
NbDijk	:	a counter for the number of times Dijkstra's algorithm is used
maxiter1	:	the maximum number of iterations used in Step (1) of Figure 2
maxiter2	:	the maximum number of iterations used in Step (4) of Figure 2
n	:	the number of customers, $i = 1,2,\dots, n$
m	:	the number of depots, $j = n+1,,\dots, n+m$
q_i	:	the demand of customer i ($i = 1,2,\dots, n$)
d_{ij}	:	the Euclidean distance between nodes i and j ($I = 1,,\dots, n+m$; $j = 1,,\dots, n+m$)
p_i (p_i')	:	the nearest (second nearest) depot to

customer i *(i= 1,…, n)* where p_i (p_i') = *n+1, ,…, n+m*

ε : a prescribed positive value for the borderline customers, say $\varepsilon = 0.7$

B : the set of borderline customers *kmax*: the maximum number of neighborhoods. *lmax*: the maximum number of local search operators.

Explanation of the Main Step Initial Solution (Step 0)

The initial solution without borderline customer is obtained in three steps; (i) construct a giant tour for each depot using the sweep algorithm of Gillett and Miller [12], (ii) improve this tour using the 2-opt of Lin [21], (iii) construct the cost network and (iv) apply Dijkstra's algorithm [8] to find the optimal solution for the shortest path based on the corresponding cost network.

This partitioning procedure based on solving the shortest path problem was presented by several authors for the VRP and by Salhi and Sari [27] for the multi-depot HFVRP. To avoid using the largest distance between two successive customers in a given route, these 2 customers, say a and b are used as the starting and ending points, in the construction of the cost network, in the giant tour. For instance starting from 'a' anticlockwise till reaching 'b' and starting from 'b' clockwise till reaching 'a' will lead to two cost networks. For each depot, the construction of the cost network is performed and Dijkstra's algorithm implemented, see Imran *et al.* [18] for details.

Neighborhood Structures (Step 1b(i) and Step 4b(i))

Six neighborhoods are used in this study (i.e. k_{max} = 6). These include the 1-1 interchange (swap), two types of the 2-0 shift, the 2-1 interchange, and two types of the perturbation. The order of the neighborhoods is as follows; the 1-1 interchange is used as N_1, the 2-0 shift of type 1 as N_2, the 2-1 interchange as N_3, the perturbation of type 1 as N_4, the perturbation of type 2 as N_5, and finally the 2-0 shift of type 2 as N_6. The detailed description of these neighborhoods can be found in Imran *et al.* [18] for the single depot case.

Local Search (Step 1b(ii) and Step 4b(ii))

Six refinement procedures are adopted as our local searches and which also make up our multi-level heuristic. The order of the refinement procedures is as follows: the 1-insertion inter-route as the first refinement procedure R_1, the 2-opt inter-route as R_2, the 2-opt intra-route as R_3, the swap intra-route as R_4, 1- insertion intra-route as R_5, and finally the 2-insertion intra-route as R_6. The process starts by generating a random feasible solution x' from N_1, which is used as the temporary solution. The multi-level approach then starts by finding the best solution x'' using R_1. If x'' is better than x', then $x' = x''$ and the search returns to R_1, otherwise the next refinement procedure is applied. This process is repeated until R_6 cannot produce a better solution. As these six local searches are very similar to the ones developed by Imran *et al.* [18] and Imran and Okdinawati [17] and for the single depot case, these are not reported here.

Definition and Insertion of Borderline Customers (Step 2)

A borderline customer is a customer that happens to be situated approximately half a way between its nearest and its second nearest depots. Here, we determine the borderline customers as follows: For each customer $i = 1,2,\ldots, n$ find the nearest and second nearest depots (i.e., p_i, p_i' respectively) Set $B = \phi$, the set of borderline customer. For each customer $i = 1,2, \ldots, n$

- Identify its nearest and second nearest depots respectively (p_i, p_i')
- Compute $\rho = d_{ip_i}/d_{ip_i'}$ If $\rho \leq \varepsilon$ allocate customer i to its nearest depot, otherwise consider customer i as a borderline customer, which will be left temporarily unassigned, and set $B = B \cup \{i\}$.

The larger the value of ε, the smaller the number of borderline customers is. In particular if $\varepsilon = 1$, there will be no borderline customers except if some customers happen to be situated exactly half way between the two corresponding depots. If $\varepsilon = 0$, any customer is a borderline customer. According to previous experiments by Salhi and Sari [27], the value of $\varepsilon = 0.7$ was found to be appropriate in generating an enough number of borderline customers.

Insertion of the Borderline Customers (Step 2)

All borderline customers are inserted in their best possible place using the nearest or the second nearest depot. This is carried out using the 1-0 insertion procedure. For each customer $i \in B$, compute

$$Ins(i) = Min\{Ins(i, R_j), R_j \in DP1(i) \cup DP2(i)\} \quad (1)$$

where $DP1(i)$ and $DP2(i)$ denote the set of routes originating from the nearest and second nearest depots of customer i, and

$$Ins(k, R_j) = \min_{l \in R_j}(d_{i,l} + d_{i,l+1} - d_{l,l+1}) \quad (2)$$

In this step all routes are considered together to form a large set of routes irrespective of their

Step (0) *Initialization*. Define a set of neighbourhood structures N_k, for $k = 1,..., k_{\max}$ and set of local search R_l, for $l = 1,..., l_{\max}$. Set *NbDivMax=5, maxDijk=5*, *NbDiv* = 0, *NbDijk*=0. *maxiter1*=50 and *maxiter2*=250.
Allocate the customers to the depots and determine the borderline customers.
enerate an initial solution x_i for each depot i and set $x_{ibest} = x_i$,

Step (1) *Repeat* the following sequence until *maxiter1* is reached:
(a) For each depot i, set $k \leftarrow 1$
(b) *Repeat* the following steps until $k = k_{\max}$:
(i) *Shaking.* Generate a point x_i' at random from the k^{th} neighbourhood of x_i ($x_i' \in N_k(x_i)$) ;
(ii) *Local search:* Apply a multi-level approach to find the best neighbour x_i'' .
(iii) *Move or not.* If the local optimum x_i'' is better than the incumbent x_i , set $x_i \leftarrow x_i''$ and go to (a); otherwise set $k \leftarrow k+1$.

Step (2) Insert the borderline customers to its closest or second closest depot.

Step (3) Define *xt* as the solution of all depots.

Step (4) *Repeat* the following sequence until *maxiter2* is reached:
(a) Set $k \leftarrow 1$
(b) *Repeat* the following steps until $k = k_{\max}$:
(i) *Shaking.* Generate a point xt' at random from the k^{th} neighbourhood of xt ($xt' \in N_k(xt)$) ;
(ii) *Local search:* Apply a multi-level approach to find the best neighbour xt'' .
(iii) *Move or not.* If the local optimum xt'' is better than the incumbent xt , set $xt \leftarrow xt''$ and go to (5); otherwise set $k \leftarrow k+1$.

Step (5) While *NbDivMax* is not reached, re-arrange the routes according to their depots, apply the diversification procedure within each depot and go to Step (3).

Step (6) If *maxDijk* is met stop,
Else - re-arrange the routes for each depot
- for each depot construct the cost network using the partial solution of the incumbent xt and apply Dijkstra's algorithm to get the partial solution of the new $\tilde{x}t$.
- if the new solution $\tilde{x}t$ is better than xt , set $xt \leftarrow \tilde{x}t$, *NbDiv* = 0 and go to Step (4), else stop.

Figure 2. The VNS-based algorithm for the MDVRP

originating depots. As each route is defined as a string where the first and last elements represent the depot number from which such a route originates, it is therefore a simple procedure to apply the VNS to all these routes (Step 4) as some customers may now have the opportunity to shift between routes that are not necessarily originating from the same depot. An illustration in the case of two depots with four routes (two routes in the first depot and two routes in the second one) is given in Figure 3 where the j^{th} depot is represented by $n+j$ $(j=1,\ldots,m)$. Figure 2 represents a brief flow chart that describes how routes are aggregated, the VNS applied, then the routes regrouped again depot by depot which then allowed the diversification procedure and the Dijkstra's algorithm to be used on each depot. This procedure is repeated several times until the stopping criterion is fulfilled.

The Diversivication Procedure (Steps 5)

This procedure is used when there is no further improvement after all the local searches are performed. The idea is to explore other regions of the search space that may not have been visited otherwise. The incumbent best solution is used as an input for the diversification procedure to obtain the new initial solution. The idea is to construct a cost network by starting from a node which is not the first point of any route, when following clockwise direction, and also not the end point of any route, when following anticlockwise direction.

This will ensure that a route from this incumbent best solution will be split, a new cost network constructed and hence a new solution generated.

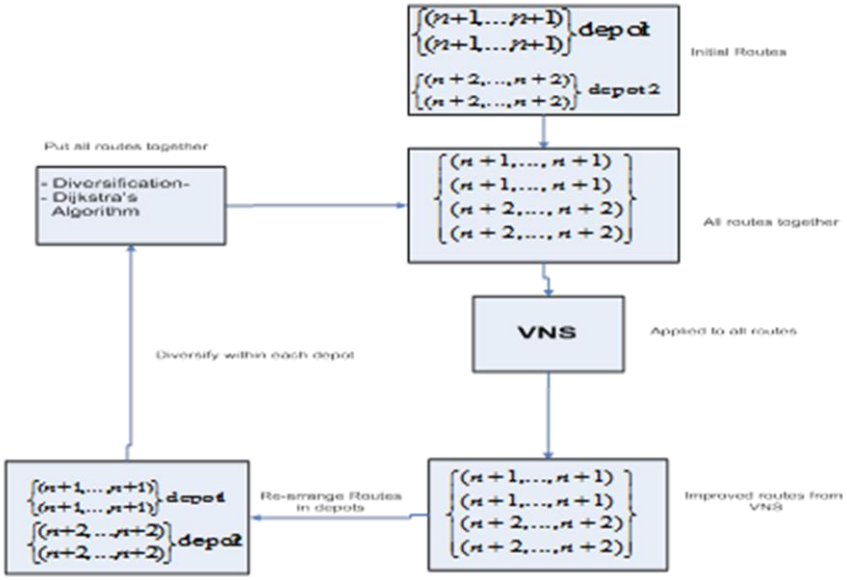

Figure 3. An outline of aggregation and disaggregation procedure (case of two depots and four routes)

Figure 3. An outline of aggregation and disaggregation procedure (case of two depots and four routes)

Step 1. Connect all points; the last point of the previous route is connected to the first point of the next route.
Step 2. Calculate all distances between two consecutive points.
Step 3. Select the largest distance between two consecutive points which are not two end points of different routes, say (e_1, e_2) as the starting point.
Step 4. Construct the cost network starting from e_2 clockwise and apply the Dijkstra's Algorithm.
Step 5. As in Step 4, but start from e_1 counter clockwise.

Figure 4. The diversification procedure

Figure 4. The diversification procedure

Table1. Comparison of VNS-based algorithm average deviation with other algorithms

No	*n*	*m*	*Q*	Best Solution	Chao *et al.* [4]	Renaud *et al.* [25]	Dorigo *et al.* [9]	Salhi & Sari [27]	Yu *et al.* [33]	Yu *et al.* [33]	VNS
1	50	4	80	**576.87**	**576.87**	**576.87**	**576.87**	587.8	**576.87**	**576.87**	**576.87**
2	50	4	160	**473.53**	**473.53**	**473.53**	484.28	476.7	**473.53**	**473.53**	**473.53**
3	75	5	140	**641.19**	**641.19**	**641.19**	645.16	644.5	**641.19**	**641.19**	**641.19**
4	100	2	100	**1001.04**	1012.0	1003.87	1020.52	1042.3	1001.49	1001.49	1011.20
5	100	2	200	**750.03**	756.5	750.26	750.26	777.2	750.26	750.26	753.63
6	100	3	100	**876.50**	879.1	**876.50**	878.34	888.6	**876.50**	**876.50**	885.72
7	100	4	100	**881.97**	893.8	892.58	898.80	911.8	887.11	885.69	890.13
8	249	2	500	**4387.38**	4511.6	4485.09	4508.14	4513.3	4500.15	4482.38	4527.73
9	249	3	500	**3873.64**	3950.9	3937.82	4083.44	4005.3	3913	3912.23	3935.00
10	249	4	500	**3650.04**	3727.1	3669.38	3747.62	3824.7	3693.4	3663.0	3706.09
11	249	5	500	**3546.06**	3670.2	3648.95	3599.93	3714.3	3564.74	3554.08	3589.36
12	80	2	60	**1318.95**	1327.1	**1318.95**	1327	1326.8	**1318.95**	**1318.95**	**1318.95**
13	80	2	60	**1318.95**	1345.9	**1318.95**	**1318.95**	**1318.9**	**1318.95**	**1318.95**	**1318.95**
14	80	2	60	**1360.12**	1372.5	1365.69	1375.22	**1360.1**	1373.18	1365.68	**1360.12**
15	160	4	60	**2505.42**	2610.3	2551.46	2588.22	2579.3	2565.67	2551.45	**2505.42**
16	160	4	60	**2572.23**	2605.3	**2572.23**	2604.9	2584.5	**2572.23**	**2572.23**	**2572.23**
17	160	4	60	**2708.99**	2816.6	2731.37	2776.99	2720.2	**2708.99**	**2708.99**	2720.23
18	240	6	60	**3702.85**	3877.4	3781.04	3907.88	3822.1	3846.05	3781.03	3724.63
19	240	6	60	**3827.06**	3863.9	**3827.06**	3863.03	3863.9	**3827.06**	**3827.06**	3839.36
20	240	6	60	**4058.07**	4272.0	4097.06	4231.28	4074.8	4142	4097.06	4074.78
21	360	9	60	**5474.74**	5791.5	5656.47	5579.86	5788.0	5495.54	**5474.74**	5567.23
22	360	9	60	**5702,16**	5857.4	5718.00	5897.64	5765.9	5832.07	5772.23	5742.60
23	360	9	60	**6078,75**	6494.6	6145.58	6341.61	6106.6	6183.13	6125.58	6113.68
Average deviation					2.32	0.86	2.11	2.08	0.85	**0.50**	0.68
Number of best solution					3	8	2	2	9	**10**	8

Q: vehicle capacity related to the classical MDVRP data sets

Table 2. CPU time of different algorithms (in minutes)

No	n	m	Q	Chao *et al.* [4]	Renaud *et al.* [25]	Salhi & Sari [27]	Yu *et al.* [33]	Yu *et al.* [33]	VNS
1	50	4	80	1.1	3.2	0.2	N.A	N.A	0.48
2	50	4	160	1.2	4.8	0.4	N.A	N.A	0.97
3	75	5	140	1.8	5.8	1.1	N.A	N.A	0.77
4	100	2	100	2.2	11.4	1.5	N.A	N.A	0.92
5	100	2	200	2.4	12.8	4.4	N.A	N.A	1.17
6	100	3	100	2.1	8.4	0.9	N.A	N.A	1.22
7	100	4	100	4.8	6.8	1.9	N.A	N.A	0.97
8	249	2	500	24.1	69.4	33.8	N.A	N.A	6.25
9	249	3	500	20.9	41.2	24.9	N.A	N.A	4.73
10	249	4	500	7.2	43.0	26.1	N.A	N.A	7.72
11	249	5	500	16.7	36.4	26.2	N.A	N.A	5.45
12	80	2	60	2.8	5.4	2.1	N.A	N.A	0.83
13	80	2	60	0.7	4.8	1.6	N.A	N.A	0.68
14	80	2	60	1.3	2.6	1.8	N.A	N.A	0.8
15	160	4	60	2.3	15.5	10.3	N.A	N.A	2.53
16	160	4	60	6.1	11.1	10.8	N.A	N.A	2.27
17	160	4	60	6.5	5.8	6.9	N.A	N.A	3.03
18	240	6	60	8.6	23.2	28.9	N.A	N.A	6.57
19	240	6	60	22.3	22.0	25.2	N.A	N.A	5.23
20	240	6	60	14.6	10.0	30.5	N.A	N.A	6.35
21	360	9	60	78.5	48.7	63.9	N.A	N.A	19.17
22	360	9	60	132.4	33.5	60.5	N.A	N.A	13.00
23	360	9	60	24.4	17.3	64.5	N.A	N.A	17.17

The steps of the diversification procedure are presented in Figure 4.

Use of the Dijkstra's Algorithm as an Extra Refinement (Step 6)

Dijkstra's algorithm, besides being used to generate an initial solution, is also applied as a post optimizer. Here, the cost network is constructed from the incumbent best solution. The aim is to see whether the optimal solution for the shortest path based on the corresponding cost network is better to the current one or not. In this procedure, the two end points of the first route of the incumbent best solution are used as the starting points and then all the other routes are combined to form the giant tour.

Results and Discussion

The implementation of the VNS-based algorithm is tested on 23 problems from Christofides and Eilon [5] (problem no.1-no.7), Gillett and Johnson [11] (problem no.8-no.11) and Chao *et al.* [4] (problem no.12-no.23) and executed using a Pentium IV-M PC with 1 GB of RAM.

The results obtained are then compared with the existing results. In Table 1 the best solutions are recorded in bold and the new best solutions are underlined. The average deviation (AD) is calculated as

$$AD = \sum_{k=1}^{NI} 100 \frac{cost_k - best_k}{best_k} \quad (3)$$

where NI, $cost_k$ and $best_k$ denote the number of instances, the cost for the k^{th} instance and the best known solution for the k^{th} instance respectively. Table 1 shows that the proposed algorithm produces competetive result. The algorithm produces 8 ties with the existing best known solutions. In terms of the average deviation it produces better average deviation than the ones of Chao *et al.* [4], Renaud *et al.* [25], Dorigo *et al.* [9], Salhi and Sari [27] and Yu *et al.* [33] (ACO-WM algorithm) but it is not as good as the ones of Yu *et al.* [33] that uses the PIACO algorithm.

The CPU time of other researchers are obtained from various machines. The comparison of computational effort discussed in this paper, as the run time not only depends on the CPU of the machines but also on the operating system, the compiler, the programming language and the precision used during the execution of the code. Table 2 shows that the CPU time used by the proposed algorithm is acceptable.

Conclusion

In this study, an efficient implementtation of the VNS-based algorithm is put forward to solve the MDVRP. The algorithm is equipped with a scheme for determining borderline customers, a multi-level based approach acting as the set of local searches, the Dijkstra's algorithm, a diversification procedure and a mechanism to aggregate the routes from

different depots and re-aggregate them into corresponding depots accordingly. The algorithm produce competitive result when compared to the results from the literature.

The result can be improved by combining VNS with other method or introducing new local searches and neighborhoods into the VNS algorithm. The proposed methodology could be developed to solve related multi-depot routing problems such as variants that include the presence of time windows, the case of pickups and deliveries, among others.

References

1. Ball, M., Golden, B., Assad, A., and Bodin, L., Planning for Truck Fleet Size in the Presence of a Common Carrier Option, *Decision Science* 14,1983, pp. 103-120.
2. Benton, W.C., Evaluating a Modified Heuristic for the Multiple Vehicles Scheduling Problem, *Working Paper* RS86-14, College Administration Science, the Ohio State University, Columbus, OH, 1986.
3. Cassidy, P.J., and Bennet, H.S., TRAMP-A Multi-Depot Vehicle Scheduling System, *Operational Research Quarterly* 23, 1972, pp. 151-162.
4. Chao, I.M., Golden, B., and Wasil, E., A New Heuristic for the Multi-Depot Vehicle Routing Problem that Improves upon Best-Known Solutions, *American Journal Mathematics Management Science* 13, 1993, pp. 371-406.
5. Christofides, N., and Eilon, S. An Algorithm for the Vehicle Dispatching Problem, *Operational Research Quarterly* 20, 1969, pp. 309-318.
6. Clarke, G., and Wright, J.W., Scheduling of Vehicle from Central Depot to a Number Number of Delivery Points, *Operations Research* 12, 1964, pp. 568-581.
7. Cordeau, J.F., Gendreau, M., and Laporte, G., A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problem, *Networks* 30, 1997, pp. 105-119.
8. Dijkstra, E.W., A Note on Two Problems in Connection with Graphs, *Numerische Mathematik* 1, 1959, pp. 269-271.
9. Dorigo, M., Maniezzo,V., and Colorni, A., Ant System: Optimization by a Colony of Cooperating Agents, IEEE Transactions on System Man and Cybernetics 26(1), 1996, pp. 29-41.
10. Dueck. G., New Optimization Heuristics: The Great Deluge Algorithm and Record to Record Travel, *Journal of Computation Physics* 104, 1993, pp. 86-92.
11. Gillett, B.E., and Johnson, J.W., Multi-Terminal Vehicle Dispatching Algorithm, *Omega* 4, 1976, pp. 711-718.
12. Gillett, B.E., and Miller, L.R., A Heuristic Algorithm for the Vehicle Dispatch Problem, *Operations Research* 22, 1974, pp. 340-344.
13. Golden, B., Magnanti, T., and Nguyen, H., Implementing Vehicle Routing Algorithms, *Networks* 7, 1977, pp. 113-148.
14. Ho, W., Ho, G.T.S., Ji, P., and Lau, H.C.W., A Hybrid Genetic Algorithm for the Multi-Depot Vehicle Routing Problem, *Engineering Applications of Artificial Intelligence* 21, 2008, pp. 548-557.
15. Klots B., Gal, S., and Harpaz, A., Multi-Depot and Multi Product Delivery Optimization Problem with Time and Service Constraints. *Technical Report*, IBM Israel, Haifa, 1992.
16. Imran, A., *An Adaptation of Metaheuristics for the Single and Multiple Depots Heterogeneous Fleet Vehicle Routing Problems*, PhD thesis, University of Kent, United Kingdom, 2008.
17. Imran, A., and Okdinawati, L., An Adaptation of the Variable Neighborhood Search Heuristic to Solve the Vehicle Routing Problem, *Jurnal Teknik Industri,* Jurusan Teknik dan Manajemen Industri, Universitas Muhammadiyah Malang, 1, 2011, pp. 11-17.
18. Imran A, Salhi, S., and Wassan N.A., A Variable Neighborhood-Based Heuristic for the Heterogeneous Fleet Vehicle Routing Problem, *European Journal of Operational Research* 197, 2009, pp. 509-518.
19. Laporte, G., Norbert, Y., and Arpin, D., Optimal Solutions to Capacitated Multi-Depot Vehicle Routing Problem, *Congressus Numerantium* 44, 1984, pp. 283-292.
20. Laporte, G., Norbert, Y., and Taillefer, S., Solving a Family of Multi-Depot Vehicle Routing and Location-Routing Problems. *Transportation Science* 22, 1988, pp. 161-172.
21. Lin, S., Computers Solutions of the Travelling Salesman Problem, *Bell System Technical Journal* 44, 1965, pp. 2245-2269.
22. Min, H., Current, J., and Schilling D., The Delivery Depot Vehicle Routing Problem with Backhauling, *Journal of Business Logistics* 13, 1992, pp. 259-288.
23. Mladenovic, N., and Hansen, P. Variable Neighborhood Search. *Computers & Operations Research* 24, 1997, pp. 1097-1100.
24. Perl, J., and Daskin, M.S., A Warehouse Location-Routing, *Transportation Research* 19B, 1985, pp. 381-396.
25. Renaud, J., Laporte, G., and Boctor, F.F., A Tabu Search Heuristic for the Multi-Depot Vehicle Routeing Problem, *Computers & Operations Research* 23, 1996, pp. 229-235.
26. Salhi, S., Imran, A., and Wassan, N.A., The Multi-Depot Vehicle Routing Problem with Heterogeneous Vehicle Fleet: Formulation and a Variable Neighborhood Search Implementation, *Computers & Operations Research*. doi/10.1016/j.cor.2013.05.011, 2013.
27. Salhi, S., and Sari, M., A Multi-level Composite Heuristic for the Multi-Depot Vehicle Fleet Mix Problem, *European Journal of Operational Research* 103, 1997, pp. 95-112.

28. Tarantilis, C.D., and Kiranoudis, C.T., Distribution of Fresh Meat, *Journal of Food Engineering* 51, 2002, pp. 85–91.
29. Tillman, F., The Multiple Terminal Delivery Problem with Probabilistic Demands, *Transportation Science* 3, 1969, 192-204.
30. Tillman, F.A., and Cain, T.M., An Upper bound Algorithm for the Single and Multiple Terminal Delivery Problem, *Management Science* 18, 1972, pp. 664-682.
31. Wren, A., and Holiday, A., Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points, *Operational Research Quarterly* 23, 1972, pp.333-344.
32. Yellow, P., A Computational Modification to the Savings Method of Vehicle Scheduling, *Operational Research Quarterly* 21, 1970, pp. 281-283.
33. Yu, B., Yang, Z. Z., and Xie, J. X., A Parallel Improved Ant Colony Optimization for Multi Depot Vehicle Routing Problem, *Journal of the Operational Research Society* 62, 2011, pp. 183–188.