ADAPTATION OF THE VARIABLE NEIGHBORHOOD SEARCH HEURISTIC TO SOLVE THE VEHICLE ROUTING PROBLEM

ARIF IMRAN,[1] DAN **LIANE OKDINAWATI**[2]
[1]Industrial Engineering Department Institut Teknologi Nasional
[2]Business Logistics Department Politeknik Pos Indonesia

E-mail: arifimr@yahoo.com

ABSTRACT

The vehicle routing problem is investigated by using some adaptations of the variable neighborhood search (VNS). The initial solution was obtained by Dijkstra's algorithm based on cost network constructed by the sweep algorithm and the 2-opt. Our VNS algorithm use several neighborhoods which were adapted for this problem. In addition, a number of local search methods together with a diversification procedure were used. The algorithm was then tested on the data sets from the literature and it produced competitive results if compared to the solutions published.

Key words: *metaheuristic, routing, variable neighborhood*

INTRODUCTION

The Vehicle Routing Problem (VRP) has an important role in distribution management and it is one of the most widely studied problems in combinatorial optimization. The VRP is a problem where a number of customers need to be served by a number of homogeneous vehicles based at a single depot. In this problem, each customer is visited exactly once; the maximum capacity of the vehicle and the maximum length of the route must not be exceeded. The objective here is to find a set of routes which fulfil all requirements mentioned above with the least cost. There are two types of methods used to solve the VRP, namely exact methods and heuristic methods. For small instances (say $n<30$), the VRP could be solved in a reasonable computing time by using exact methods such as Integer Linear Programming. However, these methods become ineffective for large problems as the computation becomes too time consuming. The increase in computing time is due to the fact that the VRP is an NP-hard problem (Lenstra and Rinnooy Kan, 1975). In other words, it is unlikely that a polynomial time algorithm can be found for such a problem.

There are many papers addressing the VRP. The following are some VRP paper that we divide into two categories, exact and heuristic methods. *Exact Methods*, exact methods for solving the VRP were developed starting from the late 1950s by Dantzig and Ramser (1959) and Garvin *et al.*(1975). Dantzig and Ramser (1959) modified the algorithm, originally proposed for the TSP by Dantzig *et al*. (1954) to address the VRP. Eilon *et al*. (1971) developed a Dynamic Programming approach to address the VRP. Laporte and Norbert (1987) and Laporte *et al*. (1992) proposed a Branch and Bound approach. *Heuristic Methods, t*he saving algorithm was proposed by Clarke and Wright (1964). It then became a basis of many algorithms developed to solve the VRP. Christofides and Eilon (1969) proposed an improvement method for the VRP which uses the 2-opt and the 3-opt initially developed by Lin (1965) to solve the TSP. Salhi and Rand (1987) developed a heuristic that considers several refinement procedures including the perturb procedure which consider three routes simultaneously to improve the initial solution. Osman (1993) applied Simulated Annealing and Tabu Search metaheuristics with his λ- interchange method.

Taillard (1993) proposed a procedure that partitioned large problems into several sub-problems before applying Tabu Search. Xu and Kelly (1996) introduced a heuristic search using a network flow-based Tabu Search. Neural Network algorithm was also put forward and this was carried out by Torki *et al*. (1997). A variant of the threshold accepting algorithm called Backtracking Adaptive Threshold Accepting algorithm (BATA) was developed by Tarantilis *et al*. (2002). Prins (2004) developed a method based on Genetic Algorithm (GA). Pisinger and Ropke (2007)introduced a general heuristic that can solve five different variants of the VRP. An Improved Ant Colony Optimization (IACO) was proposed by Yu *et al*. (2009). The remaining parts of the paper are organized as follows. The proposed VNS algorithm is presented in Section 2. The explanation of its main steps is provided in Section

3. The computational results are given in Section 4. The last section summarizes our findings.

Adaptation of the Variable Neighborhood Search

Variable Neighbourhood Search (VNS) is a metaheuristic method first proposed by Mladenovic (1995) and later formally formulated by Mladenovic and Hansen (1997). This heuristic has been applied to several NP-hard problems with excellent success. The main reasoning of this metaheuristic is based on the idea of a systematic change of neighborhoods within a local search method. The basic VNS algorithm is presented in Figure 1 and follows closely the notation of the authors.

Initialization. Select a set of neighbourhood structures N_k, for $k = 1,..., k_{max}$ 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).

The basic VNS algorithm

The basic VNS algorithm starts by selecting a set of neighborhood structures N_k $(k = 1,..., k_{max})$, where N_k is the k^{th} neighborhood. Given an initial solution x, a random point x' in $N_k(x)$ is generated using a neighborhood structure N_k and then a local search, starting from x' is performed to producex''. The use of x' can be considered as a way of maintaining diversification through the search. If x'' is better than the incumbent best solution x, then $x = x''$, and the search returns to N_1, otherwise the search explores the next neighborhood N_{k+1}. This is repeated until $k = k_{max}$. Interesting new variants of this classical VNS are presented in Hansen and Mladenovic (2003).

Some enhancements to the basic VNS algorithm

In this study, the basic VNS algorithm is adapted to solve the VRP. To our knowledge, this is the first VNS implementation to this particular routing problem. The basic VNS algorithm is enhanced by the use of additional features which include adopting a set of local search procedures including Dijkstra, and introducing a diversification scheme. The proposed algorithm is described in Figure 1.

An overview of the proposed algorithm

An initial solution x is first generated and it is used as the initial global best, x_{best}. We have a set of neighborhood structures N_k, $(k=1,..., k_{max})$ and a set of refinement procedures which will be described later. The search begins by generating a random feasible solution x' from $N_1(x)$, which is taken as the temporary solution. x' is then improved by the set of local searches (refinement procedures) which are implemented within a multi-level framework (Salhi and Sari 1997). If the solution obtained by the multi-level approach, x'', is better than the incumbent best solution x, then $x = x''$ and the search reverts back to N_1. But if x'' is found to be worse or the same as x, we generate x' from the next neighborhood say $N_k(x)$ and apply the

```
Step (0) Initialization. Define a set of neighborhood structures $N_k$, for $k = 1, ..., k_{max}$ and a set of local
         searches $R_l$, for $l = 1, ..., l_{max}$. Set the maximum number of diversifications, $NbDivMax$ and the
         number of diversifications, $NbDiv = 0$. Generate an initial solution $x$ and set $x_{best} = x$.
Step (2) Set $k \leftarrow 1$
Step (3) Repeat the following steps until $k = k_{max}$:
         (a) Shaking. Generate a point $x'$ at random from the $k^{th}$ neighborhood of $x(x' \in N_k(x))$;
         (b) Local search: Apply a multi-level approach to find the best neighbour $x''$.
         (c) Move or not. If the local optimum $x''$ is better than the incumbent $x$,
         set $x \leftarrow x''$ and go to (2); otherwise set $k \leftarrow k+1$.
Step (4) Construct the cost network using the incumbent $x$ and apply Dijkstra's algorithm to get $\tilde{x}$. If the
         new solution $\tilde{x}$ is better than $x$, set $x \leftarrow \tilde{x}$ and go to (2).
Step (5) If the solution $x$ is better than $x_{best}$, set $x_{best} \leftarrow x$;
         If $NbDiv > NbDivMax$ then stop, else set $NbDiv \leftarrow NbDiv + 1$, apply the diversification procedure
         and go to (1).
```

Figure 1. *VNS-based VRP algorithm*

multi-level approach again. The process is repeated until the search reaches $N_{k_{max}}$ If the solution obtained from Step 3 is worse than the incumbent x_{best}, a cost network, as described in Section 3, is constructed based on x and then Dijkstra's algorithm is utilized on this cost network to generate $\tilde{x}$. If $\tilde{x}$ is better than x the search reverts back to N_1 with $x = \tilde{x}$, otherwise a diversification procedure is introduced to produce a new initial solution, *x,* and the process is repeated starting from Step 2. The search terminates after a maximum number of diversifications (*NbDivMax*) is reached.

METHOD

The procedures used within the steps of the algorithm are as follow.

Initial solution (Step 0)

The initial solution is obtained in three steps; (a) construct a giant tour using the sweep algorithm of Gillett and Miller (1974), (b) improve this tour using the 2-opt of Lin (1965), and (c) construct the cost network and then apply Dijkstra's algorithm (1959) 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 Beasley (1983) for solving the VRP and by Golden *et al.* (1984) for the Heterogeneous Fleet VRP. To avoid using the largest distance between two successive customers in a given route, the starting points, in the construction of the cost network, are used as those that generate the highest largest distances between two successive customers (i.e. gaps) in the giant tour. The number of gaps (*NG*) generated is defined as follows:

$$NG = Min\{\max(8, \frac{NR}{2}), ((i,i+1: g_i > \min(\bar{g}, \frac{g^+}{2}))\}$$

where, *NR* is the number of routes found by Dijkstra's algorithm, $(i,i+1)$ the ordered sequence of customers, g_i the i^{th} gap (i.e. the distance between customer i and $i+1$), $\bar{g}$ the average gap, and $g^+$the largest gap. The reasoning of using (1) is based on the idea of linking the value of *NG* to the number of routes and also to the number of gaps that relate to the average as well as the largest gap. For each of the *NG* selected gaps, say (i_1,i_1+1), two cost networks are then generated starting from i_1 anticlockwise and from i_1+1 clockwise. Dijkstra's algorithm is then applied to each of these $2 \times NG$ cost networks.

Neighborhood Structures (Step 3a)

Six neighborhoods, which are briefly described in this subsection, 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 1-1 interchange (the swap procedure)

This neighborhood is aimed at generating a feasible solution by swapping a pair of customers from two routes. This procedure starts by taking a random customer from a randomly chosen route and tries to swap it systematically with other customers by taking into consideration all other routes. This procedure is repeated until a feasible move is found.

The 2-0 shift

In the 2-0 shift, two consecutive random customers from a randomly chosen route are selected. These two customers are considered together for possible insertion in other routes in a systematic manner. This procedure is repeated until a feasible move is found. We name this procedure the 2-0 shift of type 1. Another 2-0 shift, which we refer to as the 2-0 shift of type 2, is similar to the above shift except that the two customers are allowed to be inserted into two different routes.

The 2-1 interchange

This type of insertion attempts to shift two consecutive random customers from a randomly chosen route to another route selected systematically while getting one customer from the receiver route until a feasible move is obtained.

A new perturbation mechanism

This scheme was initially developed by Salhi and Rand (1987) for the VRP by considering three routes simultaneously. Here, it starts by taking a random customer from a randomly chosen route and tries to relocate that customer into another route without considering capacity and time constraints in the receiver route. A customer from the receiver route is then shifted to the third route if both capacity and time constraints for the second and the third route are not violated. We refer to this as the perturbation of type 1. An extension of such a perturbation is the one that shifts two consecutive customers from a route. In this procedure, instead of removing one customer at the beginning we remove two customers. We name this procedure as the perturbation of type 2.

Local Search (Step 3b)

Six refinement procedures are adopted to make up our local search. 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.

The 1-insertion procedures (inter-route and intra-route)

Two types of the 1-insertion procedures are used. The first is the 1-insertion intra-route and the second is the 1-insertion inter-route. In the 1-insertion intra-route we remove a customer from its position in a route and try to insert it elsewhere within that route in order to have a better solution. Meanwhile, in the 1-insertion inter-route, each customer from a route is shifted from its position and tried to be inserted elsewhere into another route. If this shifting does not violate any constraints and improves the solution, the selected customer is then permanently removed.

The 2-insertion (intra-route)

The 2-insertion intra-route allows us to remove two consecutive customers and insert them elsewhere within a route to produce a cheaper route.

The 2-opt (inter-route and intra-route)

The 2-opt intra-route, usually refer to as the 2-opt (Lin, 1965), is an old but a simple and an effective improvement procedure that works by removing two non adjacent arcs and adding two new arcs while maintaining the tour structure. A given exchange is accepted if the resulting total cost is lower than the previous total cost. The exchange process is continued until no further improvement can be found. The 2-opt inter-route is similar to the 2-opt intra-route except that it considers two routes where each of the two arcs belong to a different route and reverse directions of the corresponding affected path of each route.

The swap (intra-route)

The swap intra-route is aimed at reducing the total cost of a route by swapping positions of a pair of customers within the route.

Use of Dijkstra's Algorithm as an Extra Refinement (Step 4)

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 different 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. The steps of this procedure, when the first point of the first route is used to construct a network, are presented in Figure 3.

Step 1.	Use the first node of the first route as the starting point.
Step 2.	Connect the nearest end points of other routes with the last node of the first route. Select the route which has the nearest end point as the next route. If the nearest end point is the last point in that route, reverse the route order.
Step 3.	Apply Step 2 to the remaining routes by starting from the selected route in Step 2.

Figure 3. Construction of the cost network

When we start from the other end point (i.e., the last node) of the first route, the order of that route is reversed but step 2 and step 3 of Figure 3 are similar. This construction obviously ensures that the current solution is feasible and hence Dijkstra's algorithm might discover a better one. Note that this construction can obviously be started from the end points of any route, not necessarily the first one.

The Diversification Procedure (Step 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. The steps of the diversification procedure are presented in Figure 4. In this study, the number of diversifications (ND) is set as $ND = MIN\ (100, 2N)$, where N represents the number of customers in a given instance.

RESULT

Computational Experience

The algorithm is programmed in C++ and tested to solve VRP instances of Christofides (1979). The results and CPU time are given in Table 1 and Table 2 respectively. Table 1 show that our algorithm produces good results. Three solutions for instance #1, instance #6 and instance #7 are similar to the best known solution. The solution for instance #2 and instance #3 are close to their corresponding best known solution. VNS produces better results when compared to the results of Osman (1993), Barbazoglu and Ozgur (1999), and Tarantilis *et al.* (2002). This is shown by the number of best solutions obtained and the average deviation of the 524.61solutions. In terms of CPU time, our algorithm consumes more CPU time than Xu and Kelly (1996), Prins (2004) and Yu *et al.* (2009).

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

CONCLUSIONS

We have put forward an adaptation of the basic VNS algorithm to tackle the VRP. This is enhanced by the use of additional features which include adopting a set of local search procedures including Dijkstra's algorithm and introducing a diversification scheme. It was found that our proposed VNS heuristics yield competitive results when compared to the best known results found in the literature. Finally, this study shows that a suitable implementation of VNS can be applied successfully to solve the VRP and it can be developed other related distribution problems such as Multi-depot VRP. Improving the solution quality and CPU time will be our concern in the future.

BIBLIOGRAPHY

Barbazoglu, G. and Ozgur, D. 1999. A Tabu Search for the Vehicle Routing Problem. Computers & Operations Research 26, 255–270.

Beasley, J. 1983. Route First – Cluster Second Methods for Vehicle Routing. Omega 11, 403–408.

Table 1. Results for the VRP Instances

No	Size	Best Solution	Osman (1993)	Taillard (1993)	Xu & Kelly (1996)	Barbazoglu & ozgur (1999)	Tarantilis et al. (2002)	Prins (2004)	Yu et al. (2009)	VNS8
1	50	524.61	524.61	524.61	524.61	524.61	524.61	524.61	524.61	524.61
2	75	835.26	844	835.26	835.26	836.71	838.18	835.26	835.26	836.41
3	100	826.14	838	826.14	826.14	828.72	830.69	826.14	830.00	829.44
4	150	1028.42	1044.4	1028.42	1029.14	1043.89	1036.28	1030.46	1028.42	1037.17
5	199	1291.45	1334.6	1291.45	1298.58	1306.16	1317.81	1296.39	1305.50	1315.59
11	120	1042.11	1043	1042.11	1042.11	1051.18	1042.11	1042.11	1042.11	1042.11
12	100	819.56	819.59	819.56	819.56	819.56	819.56	819.56	819.56	819.56
# Best Solutions			1	7	5	2	2	5	5	3
Average Deviation (%)			1.065	0.000	0.089	0.571	0.530	0.083	0.222	0.465

Table 2. CPU time comparison in seconds

No	Size	Osman	Taillard	Xu and Kelly	Barbazoglu and Ozgur	Tarantilis et al.	Prins	Yu et al.	VNS
1	50	114	49	30	455	122	1	2	31
2	75	50	53	49	2401	132	46	11	97
3	100	1543	580	72	2040	189	28	30	220
4	150	3560	3800	150	4604	385	330	211	891
5	199	3246	3000	273	7595	1039	1147	677	1489
11	120	1445	4600	57	4214	247	18	61	329
12	100	892	340	91	2277	65	3	31	217

Christofides, N. 1976. The Vehicle Routing Problem. Recherche Operationnelle 10, 55–70.

Christofides, N. and Eilon, S. 1969. An Algorithm for the Vehicle Dispatching Problem. Operational Research Quarterly 20, 309–318.

Christofides, N., Mingozzi, A. and Toth, P. 1979. The Vehicle Routing problem. in: Combinatorial Optimization. John Wiley & Sons.

Christofides, N., Mingozzi, A. and Toth, P. 1981. Exact Algorithm for the Vehicle Routing Problem, based on Spanning Tree and Shortest Path Relaxations. Mathematical Programming 20, 255–282.

Clarke, G. and Wright, J.W. 1964. Scheduling of Vehicle from Central Depot to a Number of Delivery Points. Operations Research 12, 568–581.

Dantzig, G., Fulkerson, D. and Johnson, S. 1954. Solution of Large Scale Travelling Salesman Problem. Operations Research 2, 393–410.

Dantzig, G. and Ramser, J. 1959. The Truck Dispatching Problem. Management Science 6, 80–91.

Dijkstra, E.W. 1959. A Note on Two Problems in Connection with Graphs. Numerische Mathematik 1, 269–271.

Eilon, S., Watson-Gandy, C. and Christofides, N. 1971. Distribution Management: Mathematical Modelling & Practical Analysis. Griffin, London.

Garvin, W., Crandall, H., John, J. and Spellman, R. 1975. Application of Linear programming in the Oil Industry. Management Science 3, 407–430.

Gillet, B.E. and Miller, L.R. 1974. A Heuristic Algorithm for the Vehicle Dispatch Problem. Operations Research 22, 340–344.

Golden, B., Assad, A., Levy, L. and Gheysens, E. 1984. The Fleet Size and Mix Vehicle Routing. Computers & Operations Research 11, 49–66.

Hansen, P. and Mladenovic, N. 2003. A Tutorial on Variable Neighbourhood Search. Le Cahiers du GERAD G-2003-46.

Laporte, G. and Nobert, Y. 1987. Exact Algorithm for the Vehicle Routing Problem. Annals of Discrete Mathematics 31, 147–184.

Laporte, G., Mercure, H. and Nobert, Y. 1992. A Branch-and-Bound Algorithm for a Class of Asymmetrical Vehicle Routeing Problems. Journal of Operational Research Society 43, 469–481.

Lenstra, J.K. and Rinnooy Kan, A.H.G. 1975. Some Simple Applications of the Travelling Salesman Problem. Operational Research Quarterly 26, 717–734.

Lin, S. 1965. Computers Solutions of the Travelling Salesman Problem. Bell System Technical Journal 44, 2245–2269.

Mladenovic, N. 1995. Variable Neighbourhood Algorithm- A New Metaheuristic for Combinatorial Optimisation. Presented at Optimisation Days, Montreal.

Mladenovic, N. and Hansen, P. 1997. Variable Neighborhood Search. Computers & Operations Research 24, 1097–1100.

Osman, I.H. 1993. Metastrategy Simulated Annealing and Tabu Search Algorithms for Vehicle Routing Problem. Annals of Operations Research 41, 421–451.

Pisinger, D. and Ropke, S. 2007. A General Heuristic for Vehicle Routing Problem. Computers & Operations Research 34, 2403–2435.

Prins, C. 2004. A Simple and Effective Evolutionary Algorithm for the Vehicle Routing Problem. Computers & Operations Research 31, 1985–2002. Boston.

Salhi, S. and Rand, G.K. 1987. Improvements to Vehicle Routing Heuristics. Journal of the Operational Research Society 38, 293–295.

Salhi, S. and Sari, M. 1997. A Multi-Level Composite Heuristic for the Multi-Depot Vehicle Fleet Mix Problem. European Journal of Operational Research 103, 95–112.

Taillard, E.D. 1993. Parallel Iterative Search Methods for Vehicle Routing Problems. Networks 23, 661–676.

Tarantilis, C.D., Kiranoudis, C.T., and Vassiliadis, V.S., 2002. A Backtracking Adaptive Threshold Accepting Metaheuristic Method for the Vehicle Routing Problem. System Analysis Modelling Simulation 42, 631–664.

Torki, A., Somhon, S. and Enkawa, T. 1997. A Competitive Neural Network Algorithm for Solving Vehicle Routing Problems. Computers & Industrial Engineering 31, 473–476.

Xu, J. and Kelly, J.P. 1996. A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem. Transportation Science 30, 379–393.

Yu, B., Yang, Z. and Yao, B. 2009. An Improved Ant Colony Optimization for Vehicle Routing Problem. European Journal of Operational Research 196, 171–176.