ISSN 1748-7595 (Online)

Kent Business School

Working Paper Series

A Variable Neighborhood-Based Heuristic for the Heterogeneous Fleet Vehicle Routing Problem

Arif Imran
Kent Business School

Said Salhi
Kent Business School

Niaz Wassan
Kent Business School

Working Paper No.180
November 2008

University of Kent

A Variable Neighborhood-Based Heuristic for the Heterogeneous Fleet Vehicle Routing Problem

Arif Imran, Said Salhi, and Niaz A. Wassan
The Centre for Heuristic Optimization, Kent Business School,
University of Kent, CT2 7PE, UK.

Abstract

The heterogeneous fleet vehicle routing problem is investigated using some adaptations of the variable neighborhood search (VNS). The initial solution is obtained by Dijkstra's algorithm based on a cost network constructed by the sweep algorithm and the 2-opt. Our VNS algorithm uses several neighborhoods which are adapted for this problem. In addition, a number of local search methods together with a diversification procedure are used. Two VNS variants, which differ in the order the diversification and Dijkstra's algorithm are used, are implemented. Both variants appear to be competitive and produce new best results when tested on the data sets from the literature. We also constructed larger data sets for which benchmarking results are provided for future comparison.

Key words: metaheuristic, routing, heterogeneous fleet, variable neighborhood.

1. Introduction

The heterogeneous fleet vehicle routing problem (HFVRP) is a variant of the vehicle routing problem (VRP) where the vehicles do not necessary have the same capacity, vehicle fixed cost and unit variable cost. We are also given a set of customers, *N*, a certain number of vehicle types, *M*, each of which has a vehicle capacity Q_m, a fixed cost F_m and a unit variable cost α_m ($m = 1,\ldots,M$). As in the classical VRP, each customer must be served by one vehicle only, each vehicle must start and finish its journey at a central depot and the capacity of a vehicle and the maximum length of a route must not be exceeded. The objective of the HFVRP is to minimize the total cost which includes both the vehicle variable and fixed costs. The idea is not only to consider the routing of the vehicles, but also the composition of the vehicle fleet. According to Liu and Shen

(1999) the HFVRP can be regarded as a short-term or mid-term (or long-term) issue, depending on the planning purpose.

There are several published papers addressing the HFVRP. Golden et al. (1984) were among the first authors to tackle this problem using a constant unit variable cost. They developed algorithms based on the Clarke and Wright (1964) saving technique for the VRP as well as two implementations of the giant tour based algorithm. Desrochers and Verhoog (1991) proposed a savings based algorithm using the idea of matching. Salhi and Rand (1993) put forward an interactive route perturbation procedure (RPERT) which contains seven refinement phases, each aimed at constructing a newly constructed fleet with a lower total cost. Osman and Salhi (1996) proposed two algorithms; the first one based on a tabu search and the second is a modification of RPERT. Ochi et al. (1998) presented an evolutionary hybrid metaheuristic which combines a parallel genetic algorithm with scatter search. Gendreau et al. (1999) implemented a tabu search approach using GENIUS, initially developed by Gendreau et al. (1992) for the TSP, and some search strategies from Gendreau et al. (1994) as well as the adaptive memory procedure originally developed for the VRP by Rochat and Taillard (1995). Taillard (1999) presented a heuristic using a column generation method. Renaud and Boctor (2002) proposed a sweep-based algorithm to generate a large set of good routes, which are then used in a set partitioning algorithm. Wassan and Osman (2002) developed tabu search variants including reactive tabu search that uses special data memory structures and hashing functions. Yaman (2006) put forward six interesting formulations for the HFVRP which are enhanced by valid inequalities and lifting. Tights lower bounds and comparable upper bounds are found when tested on the Golden et al. (1984) instances. The first four formulations are based on Miller-Tucker-Zemlin constraints whereas the last two, which proved to be more successful, use flow variables. Choi and Tcha (2007) used an efficient application of column generation technique which is enhanced by dynamic programming schemes. New tight lower bounds as well as very competitive upper bounds for the Golden et al. (1984) instances are obtained. Lee et al. (2008) put forward an algorithm that uses tabu search and set partitioning. Very recently, Brandao (2008) developed two tabu search variants incorporating GENI and some neighborhood reductions. This implementation produced excellent results.

Other related HFVRPs exist but these are not widely investigated. Salhi et al. (1992) were the first who considered unit variable cost. They presented a mixed integer formulation of the problem and modified the saving based methods of Golden et al. (1984) accordingly. Salhi and Sari (1997) presented a multi-level heuristic enhanced by two powerful neighborhood reductions to address the multi-depot HFVRP. Several insertion-based saving heuristics for solving the HFVRP with time windows (HFVRPTW) were proposed by Liu and Shen (1999). Dullaert et al. (2002) developed three insertion based heuristics to tackle the HFVRPTW. The HFVRP with a fixed fleet was solved by Taillard (1999) and Tarantilis et al. (2003, 2004). Li et al. (2007) put forward record-to-record travel heuristic originally proposed by Dueck (1993) and also designed large data sets for this HFVRP. Tarantilis and Kiranoudis (2007) adapted a BoneRoute method that uses adaptive memory to solve two case studies, one from the diary and the other in the construction sector. Dondo and Cerda (2007) developed a three phase heuristic for the multi-depot HFVRPTW. The idea is to use clustering to reduce the size of the problem which is then solved optimally. Table 1 summarizes studies related to the HFVRP.

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 which also includes variants of our approach. The computational results are given in Section 4. The last section summarizes our findings.

2. Adaptation of the Variable Neighborhood Search

VNS was initially proposed by Mladenovic and Hansen (1997) for solving combinatorial and global optimization problems. 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

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. Starting from x', a local search is then performed to produce x''. The use of x' can be considered as a simple 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

Table 1: Summary of the HFVRP related papers

Authors	Journal (year)	HFVRP Type	Method Used
Golden et al.	COR (1984)	vehicle fixed cost	saving-based and giant tour
Desrochers and Verhoog	COR (1991)	vehicle fixed cost	saving-based
Salhi et al.	Omega(1992)	vehicle fixed cost and vehicle variable cost	saving-based
Salhi and Rand	EJOR (1993)	vehicle fixed cost only	perturbation/composite
Osman and Salhi	Book (1996)	" "	perturbation/composite and tabu search
Salhi and Sari	EJOR (1997)	multi-depot, vehicle fixed cost, and vehicle variable cost.	multi-level
Ochi et al.	FGCS (1998)	vehicle fixed cost	genetic algorithm and scatter search
Taillard	RAIRO (1999)	fixed fleet, vehicle fixed cost, and vehicle variable cost	tabu search
Gendreau et al.	COR (1999)	vehicle fixed cost and vehicle variable cost	tabu search
Liu and Shen	JORS (1999)	time window and vehicle fixed cost	saving-based
Dullaert et al.	JORS (2002)	" "	insertion-based
Renaud and Boctor	EJOR (2002)	vehicle fixed cost	sweep-based
Wassan and Osman	JORS (2002)	vehicle fixed cost and vehicle variable cost	reactive tabu search
Tarantilis et al.	JORS (2003)	fixed fleet, vehicle fixed cost and vehicle variable cost	threshold accepting (list based)
Tarantilis et al.	EJOR (2004)	" "	threshold accepting (non-monotonic update)
Yaman	Math. Prog (2006)	vehicle fixed cost	several formulations + valid inequalities
Choi and Tcha	COR(2007)	vehicle fixed cost and vehicle variable cost	column generation + dynamic programming
Tarantilis and Kiranoudis	EJOR(2007)	fixed fleet for dairy and construction company	boneroute + two phase construction heuristic
Li et al.	COR (2007)	fixed fleet, vehicle fixed cost and vehicle variable cost	record-to-record
Dondo and Cerda	EJOR(2007)	multi-depot, time window and vehicle fixed cost	reduction based clustering + MILP
Lee et al.	JORS(2008)	vehicle fixed cost and vehicle variable cost	tabu search + set partitioning
Brandao	EJOR(2008)	vehicle fixed cost and vehicle variable cost	tabu search

classical VNS are presented in Hansen and Mladenovic (2003a,b). An overview of heuristic search including VNS is given in Salhi (2006) and a variety of useful applications of this meta-heuristic can be found in Melian and Mladenovic (2007).

Some enhancements to the basic VNS algorithm

In this study, the basic VNS algorithm is adapted to solve the HFVRP. 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 consists of adopting a set of local search procedures including the Dijkstra's algorithm, introducing a well structured diversification scheme, and keeping one empty dummy route during the search process for added flexibility. The proposed algorithm is described in Figure 1.

Step (0) <u>*Initialization*</u>. 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 (1) Add an empty dummy route if there is none.
Step (2) Set $k \leftarrow 1$
Step (3) <u>*Repeat*</u> the following steps until $k = k_{\max}$:
(a) <u>*Shaking.*</u> Generate a point x' at random from the k^{th} neighborhood of x $(x' \in N_k(x))$;
(b) <u>*Local search:*</u> Apply a multi-level approach to find the best neighbour x''.
(c) <u>*Move or not.*</u> 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 HFVRP algorithm

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). The multi-level approach is similar to the variable neighborhood descent (VND) of Hansen and Mladenovic (2003a,b) except that the search is not necessarily linked to neighborhoods only but also to refinement procedures. 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 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.

3. Explanation of the Main Steps

The procedures used within the steps of the algorithm are described below but a detailed flow chart is provided in Appendix A.

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 corresponding optimal fleet size. Dijkstra's algorithm systematically provides an initial solution that contains routes with their appropriate types of vehicles. 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 HFVRP. Since then, this partitioning approach has been used by several authors, including Ulusoy (1985) for the fleet size and mix problem for the capacitated arc routing problem, Ryan et

al. (1993) and Renaud et al. (1996) for the VRP and Salhi and Sari (1997) for the multi-depot HFVRP. However, it is worth noting that this partitioning procedure can not be used in its original form when the number of vehicles of each type is required. 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:

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

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. Note that since the network on which the shortest path to be found is acyclic, one could obviously use the standard dynamic programming algorithm instead. In this paper we implement the former as it is available to us.

Construction of a cost network through an illustrative example

In order to apply Dijkstra's algorithm, we first construct a cost network considering customer data, capacity constraint, distance constraint, and vehicle unit variable and fixed costs. For illustration, consider 11 customers making up the following giant tour σ = (1, 2, 3, …, 11) with customer demand q = (2, 1, 2, 3, 4, 1, 2, 1, 2, 2, 3). There are two types of vehicles; one with a maximum capacity of 6 units (type 1) and the other with 9 units (type 2). Also let d_{ij} be the distance between node i and node j.

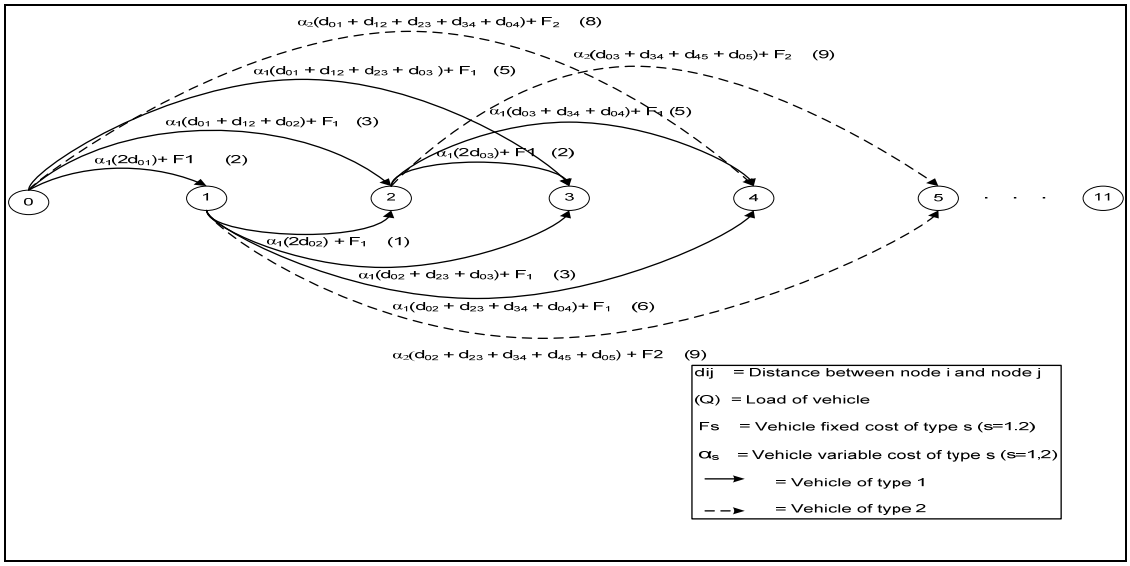


Figure 2: Network construction for Dijkstra's algorithm

We start to construct this cost network by calculating the cost from the depot, denoted by 0, to customer 1 and from this customer to the depot (return journey) as the cost of the arc 0-1. This is expressed as $C_{01} = F_1 + \alpha_1(2d_{01})$. If the total demand of both customers 1 and 2 does not violate the capacity constraint of the smallest vehicle, we calculate the cost of the arc 0-2 as $C_{02} = F_1 + \alpha_1(d_{01} + d_{12} + d_{20})$. We continue with this cost construction until the vehicle of type 1 is full, and then we start using the next large vehicle (i.e., vehicle of type 2). Figure 2 shows that we can only have customers 1, 2, and 3 together in the vehicle of type 1. Customers 1, 2, 3, and 4 can then be served by the vehicle of type 2. It is represented by the arc 0-4 which has a cost $C_{04} = F_2 + \alpha_2(d_{01} + d_{12} + d_{23} + d_{34} + d_{04})$. If the vehicle of type 2 is full we start again by using the vehicle of type 1 and calculate the distance from the depot to customer 2 and from customer 2 to the depot (arc 1-2). Customer 3 (arc 1-3) will be added to the vehicle of type 1 if the maximum capacity of this vehicle is not violated, otherwise the vehicle of type 2 is used. The process is continued until there is no more arcs connecting the last customer in the giant tour. In general, the cost of arc *ij* is defined as in Equation (2).

$$C_{ij} = F_s + \alpha_s\left(d_{0,i+1} + \sum_{k=i+1}^{j-1} d_{k,k+1} + d_{j,0}\right) \qquad (2)$$

where *s* denotes the smallest vehicle type that accommodates $\sum_{k=i}^{j} q_k$. After creating this cost network, whose origin is depot *'0'* and the destination is the last node in the giant tour, we apply Dijkstra's algorithm to produce the least cost path from the origin to the destination.

Adding flexibility via an empty dummy route

In this step, a procedure is used to create an empty route after the initialization phase and also when the diversification procedure is applied. Note that in the search we only need one empty route in our system at any time. In the case a second empty route materializes during the search, we systematically remove it. The empty route provides the search with extra flexibility as this may reduce the total cost if found worthwhile by allowing the load served from a large vehicle to be split into two smaller vehicles.

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, which is chosen after some experiments, 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, which is chosen after some experiments, 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 can not produce a better solution. This implementation of the multi-level approach is similar

to a mini VND as described in Hansen and Mladenovic (2003a,b), where the R_l represents the l^{th} neighborhood.

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 is

split, a new cost network that does not include the current solution is 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.

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

Variants arising from our VNS

In this subsection we explore the effect of using Dijkstra's algorithm as an additional local search within our multi-level heuristic and the diversification procedure as our seventh neighborhood. Five variants of our original VNS algorithm which we refer to as VNS1 are considered. The first variant uses neither Dijkstra's algorithm nor the diversification procedure. In other words, steps 4 and 5 of Figure 1 are omitted. The second and third variants both use Dijkstra's algorithm as a local search (part of Step 3b) but in the second variant the diversification procedure is not utilized (no step 5). The fourth and the fifth variants both use the diversification as the seventh neighborhood (i.e. N_7 using $k_{\max} = 7$) but Dijkstra algorithm is only used as a local search (part of Step 3b) in the latter version. For consistency, the total CPU time of VNS1 is used as a guide for the stopping criterion for all these variants. Based on our preliminary testing, it was observed that the best results are found by adopting the last variant though the other four produced reasonably good results as well. We refer to this chosen fifth variant as VNS2 in our subsequent experiments. In brief, VNS2 is similar to VNS1 except that in Figure 1, Steps 4 and 5 are removed, Dijkstra's algorithm is used as the last local search of the multi-level heuristic in Step 3b and the diversification procedure is introduced as N_7 in Step 3a.

4. Computational Experience

The algorithm is programmed in C++ which also includes a simple neighborhood reduction. The idea is to restrict the search to neighboring customers only (Salhi and Sari, 1997). This results in reducing the CPU time by approximately 13% without affecting the solution quality. Three cases of HFVRP are used as a platform to evaluate the performance of the proposed heuristics. These include the HFVRP with vehicle fixed costs only, the HFVRP with unit variable costs only and finally the HFVRP with both vehicle fixed costs and unit variable costs. For all the three cases, we carried out two sets of experiments based on 'small' and 'large' data sets which we refer to as 'Class I' and 'Class II' respectively. In Class I, the data sets from the literature (Golden et al., 1984; Taillard, 1999; Choi and Tcha, 2007) are used. Here, the largest instance has 100 customers. The lower bounds we record in this study are found by Choi and Tcha (2007) except when noted otherwise (i.e. tighter bounds by Yaman, 2006). In Class II, the large data set of Li et al. (2007) is modified using suitable vehicle capacity and unit vehicle costs. These instances range in size from 200 to 360 customers. Given that randomly generated solutions are used within our VNS heuristics (Step 3a in Figure 1), our search process is restarted a few times, 10 times for Class I and 5 times only for Class II due to their larger sizes. In the subsequent tables the best solutions are recorded in bold and the new best solutions are underlined. For each instance, say *k,* we compute the relative percentage deviation as $((\cos t_k - best_k)/best_k)\times 100,$ where $cost_k$ and $best_k$ denote, for the k^{th} instance, the cost found by our heuristic and the best known solution respectively. The average deviation is then computed over all instances in the data set.

Class I

Case of fixed cost only

Table 2 shows that our two VNS variants yield competitive results. VNS1 produces seven solutions which are equal to the best known solutions. VNS2 is even more competitive as it produces nine solutions which are equal to the best known. In terms of average percentage deviation, VNS2 is better than the recent heuristics of Choi and Tcha (2007) and Lee et al. (2008) but slightly outperformed by the very recent tabu search heuristic of Brandao (2008).

Table 2: Solution quality from the different methods

No	Size	Lower Bounds	Best Solution	Osman & Salhi (1996)	Taillard (1999)	Gendreau et al. (1999)	Renaud & Boctor (2002)	Wassan & Osman (2002)	Yaman (2006)	Choi &Tcha (2007)	Lee et al. (2008)	Brandao (2008)	VNS1	VNS2
3	20	951.61	**961.03**	**961.03**	**961.03**	**961.03**	963.61	**961.03**	**961.03**	**961.03**	**961.03**	**961.03**	**961.03**	**961.03**
4	20	6369.15	**6437.33**	6445.10	**6437.33**	**6437.33**	**6437.33**	**6437.33**	**6437.33**	**6437.33**	**6437.33**	**6437.33**	**6437.33**	**6437.33**
5	20	988.01	**1007.05**	1009.15	1008.59	**1007.05**	1007.96	**1007.05**	**1007.05**	**1007.05**	**1007.05**	**1007.05**	**1007.05**	**1007.05**
6	20	6466.94*	**6516.47**	6516.56	**6516.47**	**6516.47**	6537.74	**6516.47**	**6516.47**	**6516.47**	**6516.47**	**6516.47**	**6516.47**	**6516.47**
13	50	2392.77	**2406.36**	2471.07	2413.78	2408.41	2406.43	2422.10	2408.41	**2406.36**	2408.41	**2406.36**	**2406.36**	**2406.36**
14	50	8943.94*	**9119.03**	9125.65	**9119.03**	9119.28	9122.01	9119.86	**9119.03**	**9119.03**	9160.42	**9119.03**	**9119.03**	**9119.03**
15	50	2544.84	**2586.37**	2606.72	**2586.37**	**2586.37**	2618.03	**2586.37**	**2586.37**	**2586.37**	**2586.37**	**2586.37**	**2586.37**	**2586.37**
16	50	2685.92	**2720.43**	2745.01	2741.50	2741.50	2761.96	2730.08	2741.50	**2720.43**	2724.33	2728.14	2741.50	**2720.43**
17	75	1709.85	**1734.53**	1762.05	1747.24	1749.50	1757.21	1755.1	1747.24	1744.83	1745.45	**1734.53**	1745.33	1741.95
18	75	2342.84	**2369.65**	2412.56	2373.63	2381.43	2413.39	2385.52	2373.63	2371.49	2373.63	**2369.65**	**2369.65**	**2369.65**
19	100	8574.33*	**8659.74**	8685.71	8661.81	8675.16	8687.31	**8659.74**	8661.81	8664.29	8699.98	8661.81	8665.12	8665.05
20	100	3995.16	**4039.49**	4166.73	4047.55	4086.76	4094.54	4061.64	4047.55	**4039.49**	4043.47	4042.59	4066.94	4044.68
# Best Solutions				1	5	5	1	6	6	**9**	5	**9**	8	**9**
Average Deviation (%)				0.969	0.178	0.298	0.692	0.285	0.178	0.060	0.170	**0.032**	0.178	0.051

* Derived from Yaman (2006)

In Table 3 we report the total CPU time over ten runs for VNS1 and the corresponding number of runs for VNS2 (usually 5 to 7 runs). Our algorithm is executed on a Pentium M 1.7 GHz PC with 1GB RAM. The total CPU time of Osman and Salhi (1996) was obtained from a Vax 4500 machine (5.5 Mflop/s). The results of Taillard (1999) are the best solutions from five runs and the time reported refers to the average CPU time on a Sun Sparc 10 work station with 50MHz (10 Mflop/s). The CPU time of Gendreau et al. (1999) is the time for the best of ten runs on a Sun Sparc 10 (50 MHz). Renaud and Boctor (2002) use Pentium II 233 MHz computer and record the average CPU time over several runs. The CPU time of Wassan and Osman (2002) is the total time from five runs on a Sun Sparc 1000 with 50 MHz (10 Mflop/s). Yaman (2006) uses Sun Ultra 12×400 MHz, and Choi and Tcha (2007) found their results from five runs on a Pentium IV 2.6 GHz with 526 MB RAM. Lee et al. (2008) use a Pentium IV 1.8 PC whereas Brandao (2008) uses a Pentium M 1.4 GHz with 256 MB RAM. Note that Brandao's CPU refers to one variant only. The fastest heuristic is by Osman and Salhi (1996), but this method produced much lower solution quality due to its simple though well structured tabu search. Based on these results, though it is difficult to compare the CPU time under different machines, it is clear that our heuristic methods require a reasonable amount of CPU time.

Table 3: CPU time comparison in seconds

No	Size	Osman & Salhi	Taillard *	Gendreau et al. +	Renaud & Boctor *	Wassan & Osman	Yaman	Choi & Tcha +	Lee et al.	Brandao **	VNS1	VNS2
3	20	5	-	164	4	88	-	0	59	21	19	21
4	20	6	-	253	6	80	-	1	79	22	18	18
5	20	5	-	164	5	52	-	1	41	20	12	13
6	20	4	-	309	9	88	-	0	89	25	21	22
13	50	62	470	724	50	2084	397	10	258	145	216	252
14	50	71	570	1033	160	1660	176	51	544	220	255	274
15	50	46	334	901	45	2349	143	10	908	110	305	303
16	50	35	349	815	28	689	142	11	859	111	221	253
17	75	85	2072	1022	652	1874	1345	207	1488	322	662	745
18	75	116	2744	691	1037	2261	1923	70	2058	267	813	897
19	100	289	12528	1687	1110	8570	1721	1179	2503	438	1501	1613
20	100	306	2117	1421	307	2692	2904	264	2261	601	1626	1595

+ CPU time for the best run only.

* Average CPU time.

** CPU time of version 2 algorithm.

Case of unit variable costs only

The data set used here was created by Taillard (1999) by varying the unit variable costs of the 8 Golden et al. (1984) instances. The best solutions and the associated CPU are given in Table 4 and Table 5 respectively.

Table 4: Computational results for HFVRP with variable costs only

No	Size	Lower Bound	Best Sol.	Taillard	Gendreau et al.	Wassan & Osman	Choi & Tcha	Lee et al.	Brandao	VNS1	VNS2
13	50	1469.41	**1491.86**	1494.58	**1491.86**	1499.69	**1491.86**	**1491.86**	**1491.86**	**1491.86**	**1491.86**
14	50	582.25	**603.21**	**603.21**	**603.21**	608.57	**603.21**	**603.21**	**603.21**	**603.21**	**603.21**
15	50	978.04	**999.82**	1007.35	**999.82**	**999.82**	**999.82**	**999.82**	**999.82**	**999.82**	**999.82**
16	50	1106.12	**1131.00**	1144.39	1136.63	**1131.00**	**1131.00**	1131.90	**1131.00**	**1131.00**	**1131.00**
17	75	1021.86	**1031.00**	1044.93	**1031.00**	1047.74	1038.60	1038.60	1038.60	1038.60	1038.60
18	75	1779.41	**1800.80**	1831.24	1801.40	1814.11	1801.40	**1800.80**	**1800.80**	**1800.80**	**1800.80**
19	100	1080.68	**1100.56**	1110.96	1105.44	**1100.56**	1105.44	1105.44	1105.44	1105.44	1105.44
20	100	1501.67	**1530.16**	1550.36	1541.18	**1530.16**	1530.43	1531.61	1530.43	1533.96	1533.24
# Best Solutions				1	4	4	4	4	**5**	**5**	**5**
Average Deviation (%)				0.928	0.212	0.472	0.154	0.169	**0.150**	0.179	0.173

Table 5: CPU time comparison in seconds for HFVRP with variable costs only

No	Size	Taillard *	Gendreau et al. +	Wassan & Osman	Choi & Tcha +	Lee et al.	Brandao **	VNS1	VNS2
13	50	473	626	276	3	142	101	293	310
14	50	575	669	695	37	144	135	157	161
15	50	335	736	893	6	162	137	202	218
16	50	350	852	668	6	144	95	216	239
17	75	2245	1453	2222	103	865	312	460	509
18	75	2876	1487	2847	81	545	269	537	606
19	100	5833	1681	6009	299	331	839	972	1058
20	100	3402	1706	3174	112	314	469	999	1147

\+ CPU time for the best run only.

* Average CPU time.

** CPU time of version 2 algorithm.

VNS1 and VNS2 both produce five solutions which equal the best known. The average deviation of both variants is also found to be competitive.

Case of both vehicle fixed cost and unit variable costs

Choi and Tcha (2007) generated the data set by taking fixed costs from Golden et al. (1984) and unit variable costs from Taillard (1999). The variable costs of instances #3 to

#6 are generated to be dependent on the vehicle size. The comparison of the results is provided in Table 6. Though our methods generate fewer best results, we found better solutions for instance #17 from both VNS1 and VNS2 (but the one from VNS1 is the new best). The detailed result of this instance is provided in Appendix B. In addition, the average percentage deviation of our methods is also found to be relatively smaller when compared to the one reported by Choi and Tcha (2007).

Table 6: Computational results for HFVRP with fixed and variable costs

				Choi & Tcha		**VNS1**		**VNS2**	
No.	**Size**	**Lower Bound**	**Best Solution**	**Solution**	**Time+**	**Solution**	**Time**	**Solution**	**Time**
3	20	1138.58	**1144.22**	**1144.22**	0	**1144.22**	19	**1144.22**	21
4	20	6369.15	**6437.33**	**6437.33**	1	**6437.33**	17	**6437.33**	18
5	20	1307.74	**1322.26**	**1322.26**	0	**1322.26**	24	**1322.26**	27
6	20	6451.62	**6516.47**	**6516.47**	1	**6516.47**	21	**6516.47**	22
13	50	2959.60	**2964.65**	**2964.65**	2	**2964.65**	328	**2964.65**	361
14	50	8748.57	**9126.90**	**9126.90**	68	**9126.90**	250	**9126.90**	268
15	50	2597.22	**2634.96**	**2634.96**	5	**2634.96**	275	**2634.96**	312
16	50	3114.00	**3168.92**	**3168.92**	11	3168.95	313	3169.10	345
17	75	1979.87	**2004.48**	2023.61	100	**2004.48**	641	2008.14	631
18	75	3128.75	**3147.99**	**3147.99**	28	3153.67	835	3157.20	962
19	100	8431.87	**8664.29**	**8664.29**	1026	8666.57	1411	8665.88	1505
20	100	4082.25	**4154.49**	**4154.49**	82	4164.85	1460	4154.87	1592
# Best Solutions				**11**		7		7	
Average Deviation (%)				0.08		**0.038**		0.042	

+ CPU time for the best run only.

Class II

Two cases, based on the new large HFFVRP test problems of Li et al. (2007), are explored. In the first experiment we used the vehicle specifications as given by Li et al. (2007) whereas in the second experiment we generated new vehicle specifications. For simplicity we implement VNS2 only as this proved, in Class I data sets, to be slightly better than VNS1.

Case 1 (Fixed fleet HFVRP vs HFVRP)

This comparison is used as a guide only. As one may expect, our results would be better than the ones reported in Li et al. (2007). This is due to a prior fixation of their vehicle fleet composition. This restriction is made even more severe given that here the fixed cost

of a larger vehicle is not set to be necessarily larger than the one for a smaller vehicle. As shown in Table 7, the solution of our free fleet HFVRP has obviously the tendency to utilize the larger vehicles and hence yields a smaller total cost.

Table 7: Computational results for the fixed HFVRP vs HFVRP

No.	Size	Li et al.		VNS2	
		Time (secs)	Cost	Time (secs)	Cost
1	200	688	13222.65	1542	**12313.60**
2	240	995	36714.40	1992	**10674.25**
3	280	1438	19661.80	3698	**15513.22**
4	320	2256	23116.10	5720	**16874.80**
5	360	3277	24510.41	7745	**22463.25**

Case 2 (Benchmarks for larger instances)

To be more consistent when solving the HFVRP, though we still use the data set in terms of customers and depots as given in Li et al. (2007), we generate new vehicle specifications which are based on the concept that a larger vehicle will incur both a larger fixed cost and a larger unit running cost. This assumption may not obviously be valid if the age of the vehicle is taken into consideration. In this experiment, we create five different types of vehicles whose details are given in Table 8. We summarize in Table 9 the results for each of the three cases as presented in Class I.

Table 8: Specifications of the five new larger problems

	A	B	C	D	E
Vehicle Size	100	200	300	500	800
Unit running cost	1	1.2	1.5	2.0	2.3
Fixed cost	100	180	270	480	700

Table 9: Computational results of the large HFVRP instances

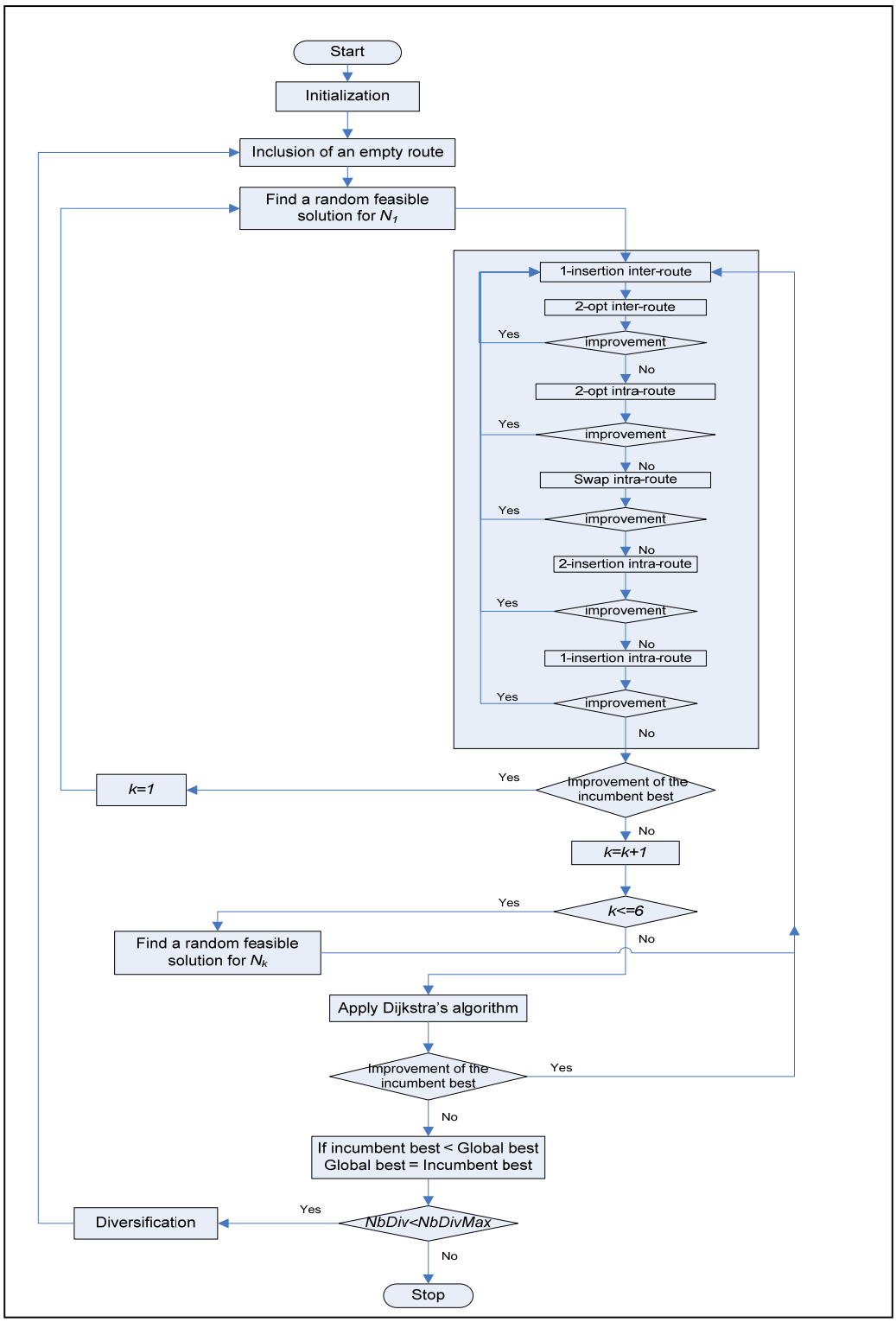
No.	Size	HFVRP with fixed cost only		HFVRP with variable cost only		HFVRP with fixed and variable cost	
		Time(secs)	Cost	Time(secs)	Cost	Time(secs)	Cost
1	200	1632	10079.85	2074	12034.83	2060	15984.42
2	240	2345	9253.52	2960	9984.64	3260	14821.36
3	280	3887	12941.42	4771	15681.77	4631	21428.13
4	320	6778	13718.38	8850	15884.69	8807	22325.24
5	360	7191	15820.29	9652	20153.96	9453	26990.80

6. Conclusions

We have put forward an adaptation of the basic VNS algorithm to tackle the HFVRP. This is enhanced by the use of additional features which include adopting a set of local search procedures including Dijkstra's algorithm, introducing a diversification scheme, and keeping a dummy empty route during the search process. Two variants are proposed differing in the way the diversification procedure and Dijkstra's algorithm are embedded into the overall framework of the algorithm. These two variants are tested on three scenarios, i.e. the case of vehicle fixed cost only, variable unit running cost only and both variable and fixed costs. It was found that our two proposed VNS heuristics yield competitive results when compared to the best known results found in the literature. In addition, our heuristics also found a few new best solutions. We have also modified existing large data instances to better suit this type of HFVRP by constructing suitable vehicle sizes, vehicle unit variable costs and vehicle fixed costs. Benchmark results for these large instances are provided which hopefully may entice other researchers to tackle this particular vehicle routing problem. Finally, this study shows that a suitable implementation of VNS can be applied successfully to solve the HFVRP and other related distribution problems.

Acknowledgement- We would like to thank the referees for their constructive comments that improved both the content as well as the presentation of the paper. We are also grateful to the Technological and Professional Skills Development Sector Project (TPSDSP) and Jurusan Teknik Industri-ITENAS Bandung, Indonesia for the sponsorship of the first author.

Appendix A: Flow chart of the VNS-based HFVRP algorithm



Appendix B: New best solutions for Instance #17(Choi and Tcha, 2007)

Route number	Demand	Vehicle type	Fixed cost	Variable cost	Distance	Route
1	49	A	50	1	20.77	34-67
2	120	B	120	1.2	69.22	52-27-54-19-35-8-46
3	199	C	200	1.5	101.89	7-53-14-59-11-66-65-38
4	118	B	120	1.2	86.91	58-10-31-9-39-72
5	116	B	120	1.2	109.61	32-50-25-55-18-24-49
6	120	B	120	1.2	77.00	51-16-23-56-63-33-6
7	118	B	120	1.2	94.63	68-22-64-42-41-43-1-73
8	120	B	120	1.2	75.08	30-74-61-28-62-2
9	120	B	120	1.2	73.55	48-47-38-37-5-29-45
10	119	B	120	1.2	108.86	13-57-15-20-70-60-71-69-21
11	115	B	120	1.2	51.32	17-3-44-40-12-26
12	50	A	50	1	15.46	75-4

Solution cost: **2004.48**

References

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

Brandao, J., 2008. A Deterministic Tabu Search Algorithm for the Fleet Size and Mix Vehicle Routing Problem. European Journal of Operational Research, doi:10.1016/j.ejor.2007.05.059.

Choi, E., Tcha D.-W., 2007. A Column Generation Approach to the Heterogeneous Fleet Vehicle Routing Problem. European Journal of Operational Research 34, 2080-2095.

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

Desrochers, M., Verhoog, T.W., 1991. A New Heuristic for the Fleet Size and Mix Vehicle Routing Problem. Computers & Operations Research 18, 263-274.

Dondo, R., Cerda, J., 2007. A Cluster-Based Optimization Approach for Multi-Depot Heterogeneous Fleet Vehicle Routing Problem with Time Windows. European Journal of Operational Research 176, 1478-1507.

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

Dueck, G., 1993. New Optimization Heuristics: The Great Deluge Algorithm and Record to Record Travel. Journal of Computation Physics 104, 86-92.

Dullaert, W., Janssens, G.K., Sorensen, K., Vernimmen, B., 2002. New Heuristic for the Fleet Size and Mix Vehicle Routing Problem with Time Windows. Journal of the Operational Research Society 53, 1232-1238.

Gendreau, M., Hertz, A., Laporte, G., 1992. New Insertion and Post Optimization Procedures for the Traveling Salesman Problem. Operations Research 40, 1086-1094.

Gendreau, M., Hertz, A., Laporte, G., 1994. A Tabu Search Algorithm for the Vehicle Routing Problem. Management Science 40, 1276-1290.

Gendreau, M., Laporte, G., Musaraganyi, C., Taillard, E.D., 1999. A Tabu Search Heuristic for the Heterogeneous Fleet Vehicle Routing Problem. Computers & Operations Research 26, 1153-1173.

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

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

Hansen, P., Mladenovic, N., 2003a. A Tutorial on Variable Neighborhood Search. Le Cashiers du GERAD G-2003-46.

Hansen, P., Mladenovic, N., 2003b. Variable Neighborhood Search. In: Glover, F., Kochenberger G.A. (Eds.), Handbook of Metaheuristic, pages 145-184. Kluwer Academic Publisher, London.

Lee,Y.H., Kim, J.I., Kang, K.H., Kim, K.H., 2008. A Heuristic for Vehicle Fleet Mix Problem Using Tabu Search and Set Partitioning. Journal of the Operational Research Society 59, 833-841.

Li, F., Golden, B., Wasil, E., 2007. A Record-to Record Travel Algorithm for Solving the Heterogeneous Fleet Vehicle Routing Problem. Computers & Operations Research 34, 2918-2930.

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

Liu, F.H., Shen, S.H., 1999. The Fleet Size and Mix Vehicle Routing Problem with Time Windows. Journal of the Operational Research Society 50, 721-732.

Melian, B., Mladenovic, N., (Eds.) 2007. Applications of Variable Neighborhood Search. IMA Journal of Management Mathematics, 18, 99-221.

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

Ochi, L.S., Viana D.S., Drummond L.M.A., Victor A.O., 1998. A parallel Evolutionary Algorithm for the Vehicle Routing Problem with Heterogeneous Fleet. Future Generation Computation System 14, 285-292.

Osman, I.H., Salhi, S., 1996. Local Search Strategies for the Mix Fleet Routing Problem. In: Rayward-Smith, V.J., Osman, I.H., Reeves, C.R., Smith, G.D. (Eds.), Modern Heuristic Search Methods, pages 132-153. John Wiley & Sons.

Renaud, J., Boctor, F.F., 2002. A Sweep-Based Algorithm for the Fleet Size and Mix Vehicle Routing Problem. European Journal of Operational Research 140, 618-628.

Renaud, J., Boctor, F.F., Laporte, G., 1996. An Improved Petal Heuristic for the Vehicle Routeing Problem. Journal of the Operational Research Society 47, 329-336.

Rochat, Y., Taillard, E.D., 1995. Probabilistic Diversification and Intensification in Local Search Vehicle Routing. Journal of Heuristic 1, 147-167.

Ryan, D.M., Hjorring, C., Glover, F., 1993. Extensions of the Petal Method for the Vehicle Routing. Journal of the Operational Research Society 44, 289-296.

Salhi, S., 2006. 'Heuristic Search in Action: The Science of Tomorrow', in Keynote Papers, OR48 (S.Salhi, Eds.), 39-58, Government Operational Research Service (GORS), London.

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

Salhi, S., Rand, G.K., (1993). Incorporating Vehicle Routing into the Vehicle Fleet Composition Problem. European Journal of Operational Research 66, 313-360.

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

Salhi, S., Sari, M., Sadi, D., Touati, N., 1992. Adaptation of Some Vehicle Fleet Mix Heuristic. OMEGA 20, 653-660.

Taillard, E.D., 1999. A Heuristic Column Generation Method for the Heterogeneous Fleet VRP. Recherche Operationnelle 33, 1-14.

Tarantilis, C.D., Kiranoudis, C.T., 2007. A Flexible Adaptive Memory-Based Algorithm for Real-Life Transportation Operations: Two Case Studies from Diary and Construction Company. European Journal of Operational Research 179, 806-822.

Tarantilis, C.D., Kiranoudis, C.T., Vassiliadis, V.S., 2003. A List Based Threshold Accepting Metaheuristic for the Heterogeneous Fixed Fleet Vehicle Routing Problem. Journal of the Operational Research Society 54, 65-71.

Tarantilis, C.D., Kiranoudis, C.T., Vassiliadis, V.S., 2004. A Threshold Accepting Metaheuristic for the Heterogeneous Fixed Fleet Vehicle Routing Problem. European Journal of Operational Research 152, 148-158.

Ulusoy, G., 1985. The Fleet Size and Mix Problem for Capacitated Arc Routing. European Journal of Operational Research 22, 329-337.

Wassan, N.A., Osman, I.H., 2002. Tabu Search Variants for the Mix Fleet Vehicle Routing Problem. Journal of the Operational Research Society 53, 768-782.

Yaman, H., 2006. Formulations and Valid Inequalities for the Heterogeneous Vehicle Routing Problem. Mathematical Programming 106, 365-390.

http://www.kent.ac.uk/kbs/research-information/index.htm