Computers & Operations Research 52 (2014) 315–325

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation

Said Salhi [a], Arif Imran [b], Niaz A. Wassan [a,*]

[a] The Centre for Logistics & Heuristic Optimization, Kent Business School, University of Kent, Canterbury CT2 7PE, UK
[b] Department of Industrial Engineering, National Institute of Technology (ITENAS), Bandung 40124, Indonesia

ARTICLE INFO

Available online 25 May 2013

Keywords:
Multi-depot vehicle routing
Heterogeneous vehicles
Distribution network
ILP formulation
Variable neighborhood search

ABSTRACT

The multi-depot fleet size and mix vehicle routing problem, also known as the multi-depot routing with heterogeneous vehicles, is investigated. A mathematical formulation is given and lower as well as upper bounds are produced using a three hour execution time of CPLEX. An efficient implementation of variable neighborhood search that incorporates new features in addition to the adaptation of several existing neighborhoods and local search operators is proposed. These features include a preprocessing scheme for identifying borderline customers, a mechanism that aggregates and disaggregates routes between depots, and a neighborhood reduction test that saves nearly 80% of the CPU time, especially on the large instances. The proposed algorithm is highly competitive as it produces 23 new best results when tested on the 26 data instances published in the literature.

1. Introduction

Logistics based companies often use various types of vehicles and operate from more than one distribution center, usually referred to as warehouse or depot. In this situation, the customers are not necessarily assigned to their nearest depots. In this study, we investigate this practical logistical problem where the aim is to minimize the total distribution cost while satisfying the necessary constraints. This problem is known as the multi-depot fleet size and mix vehicle routing problem. This is also referred to as the multi-depot heterogeneous vehicle routing problem (MDHFVRP). This problem seems to suffer from a shortage of published papers when compared to its counterpart namely the standard multi-depot vehicle routing problem (MDVRP) which uses a homogeneous vehicle fleet (i.e., one single type of vehicles only).

Our aim is to revisit and explore further this dormant but practical logistical problem and to provide new best results for future benchmarking purposes. We believe that most companies operate from many depots and use various vehicle types and therefore studying this variant is as useful as other classical variants of the vehicle routing problem (VRP).

In the MDHFVRP, we are given a number of customers, n, a number of depots, m, and a number of vehicle types, K, each of which has a capacity Q_k, a fixed cost F_k and a unit running cost α_k ($k=1,\ldots,K$). In this study, the number of vehicles per type is considered unlimited, each customer must be served by one vehicle only, and each vehicle must start and finish its journey at the same depot. The capacity of any used vehicle and the maximum length of a route must not be exceeded. The objective is to find the least total cost that includes the sum of the vehicle fixed and running costs. As a by-product of solving this problem, the vehicle fleet composition (i.e., the number of vehicles per type) will be found for all depots as well as for each depot.

Though the solution approaches for the single depot and the multi-depot problems have some similarities, it can also be said that it is naive to just decompose the multi-depot problem into several one depot sub-problems by simply assigning the customers to their nearest depots and then apply a suitable approach to each of the single depot problems. Though this simple approach is easy to use, it generally leads to poor suboptimal solutions. Obviously this solution could be considered as a starting solution in other heuristics or meta-heuristics whose search procedure would consider the entire multi-depot problem as one large problem. A brief summary of the MDVRP and its variants is provided in Table 1.

Our aim is achieved through an efficient implementation of a powerful a variable neighborhood search (VNS)-based meta-heuristic. This problem is still a niche and we aim to address which will hopefully attract other researchers to consider studying it. In that paper, the authors put forward a multi-level type heuristic which is similar in principle to the variable neighborhood descent as described in Hansen et al. [14] except that instead of using different neighborhoods, a series of local search operators

* Corresponding author. Tel.: +44 1227823821.
E-mail addresses: S.Salhi@kent.ac.uk (S. Salhi), imran@itenas.ac.id (A. Imran), N.A.Wassan@kent.ac.uk (N.A. Wassan).

0305-0548/$ - see front matter Crown Copyright © 2013 Published by Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.05.011

Table 1
Summary of the MDVRP and MDHFVRP related papers.

Authors	Journal (year)	MDVRP/MDHFVRP	Method Used
[42]	Transp Sci (1969)	MDVRP	Clarke and Wright saving-based as above
[43]	Man Sci (1971)	MDVRP	
[46]	ORQuart (1972)	MDVRP	Saving based & refinements
[4]	ORQuart (1972)	MDVRP (School meal delivery problem)	Saving based & refinements
[10]	Omega (1976)	MDVRP	Clustering & sweep heuristic
[13]	Networks (1977)	MDVRP	Borderline customer & saving-based
[1]	Dec Sci (1983)	MDVRP (distribution of chemical product in the USA and Canada)	Saving based & route first, cluster second
[23,32]	Cong Num (1984)	MDVRP	Branch and bound
[3]	Transp Res (1984, 1985)	MDVRP	Incorporate (P) in location routing & formulation
[33]	Working Paper (1986)	MDVRP (delivery to retail outlets from a bakery in Indiana)	Saving method & branch and bound
[23]	AJMMS(1987)	MDVRP	(T–C) modified distance formula & a saving variant
[18]	Transp Sci (1988)	MDVRP	Branch and bound
[27]	IBM Report (1992)	MDVRP (distribution problem of dairy products)	LP & heuristics
[5]	J Bus Log (1992)	MDVRP (distribution problem of the hardware products in the USA)	Exact methods & heuristic
[35]	AJMMS (1993)	MDVRP	Record to record
[39]	COR (1996)	MDVRP	Tabu Search
[6]	EJOR (1997)	MDVRP and MDHFVRP	Multi-level heuristic
[22]	Networks (1997)	MDVRP	Tabu Search
[41]	EJOR (2000)	MDHFVRP with pickups and deliveries	Set covering, & analytic results
[40]	Appl Art Int (2001)	MDVRP	Genetic algorithm & clustering
[45]	J Food Eng (2002)	Open MDVRP (the distribution of meat in Greece)	List-based threshold accepting
[12]	EJOR (2002)	MDVRPTW	One & two stage methods
[31]	J of Heur (2004)	MDVRPTW	VNS
[25]	IEEETASE (2005)	MDVRP with fixed vehicle fleet	Several heuristics
[29]	EJOR (2005)	MDVRP with pickups and deliveries	Combination of a number of heuristics
[34]	COR (2007)	MDVRP	Adaptive large neighborhood search
[9]	EJOR (2007)	MDHFVRP with time windows	Exact methods
[15]	EAAI (2008)	MDVRP	Genetic algorithm
[30]	J of Sched (2009)	MDVSP (vehicle scheduling)	Five heuristics & formulations
[49]	APJOR (2011)	MDVRP with weight related cost	Formulation & scatter search
[48]	JORS (2011)	MDVRP	Ant Systems
[19]	Exp Sys &Appl (2012)	MDVRP with loading cost	VNS

and perturbation schemes are used to reduce the risk of local optimality. The only closest study to the current study is the work by Dondo and Cerda [9] who proposed an algorithm for the multi-depot vehicle fleet mix (MDVFM) with time windows. Their approach is based on a clustering mechanism to reduce the problem size and a mixed integer linear program (MILP) branch and bound solver. For the case of one depot vehicle fleet mix, see the recent papers by Tutuncu [44] and the one by Imran et al. [21] and the references therein. For the case of the classical MDVRP with homogeneous fleet, see Cordeau et al. [7] when time windows are imposed and Pisinger and Ropke [34] and Yu et al. [48] when no time windows are present. For vehicle routing problems in general including contemporary topics and recent challenges in the field of vehicle routing, the reader will find the edited book by Golden et al. [13] to be useful and informative.

The paper is organized as follows. In Section 2, the mathematical formulation is provided followed by an overview of the proposed VNS algorithm, including its main steps in Section 3. A brief explanation of the main steps of the proposed algorithm is covered in Section 4 and a neighborhood reduction scheme, which is incorporated into the search, is described in Section 5. The computational results are reported and analyzed in Section 6. A summary of our findings and highlights of some research avenues that we believe to be worth pursuing in the future are given in the last section.

2. A mixed integer linear formulation for the MDHFVRP

A mixed integer linear formulation of the MDHFVRP which is an extension of the one originally presented in Salhi et al. [38] for the single depot problem is given. Recently, Lee et al. [24] also reproduced a similar formulation for the single depot problem. The proposed formulation is a flow-based formulation instead of a node-based as this type is shown to be promising in solving other vehicle routing related problems, see for instance the work of Yaman [47].

A four index binary variable which identifies the type of vehicle chosen to travel along a given arc and the depot it originates from is used. This guarantees a vehicle to return from the depot it originated from. If this restriction is relaxed, a three index variable can be used instead.

2.1. Parameters

Let

n and m be the number of customers and depots respectively. These are also denoted by nodes.
$(1,..,n+m)$ where the m depots are represented by $n+1,\ldots,n+m$.
q_i is the demand of the ith node $(i=1,..,n+m)$ with $q_i = 0 \quad (i = n+1, \ldots, n+m)$.
K is the number of vehicle types.
Q_k is the capacity of the vehicle of type k $(k=1,\ldots,K)$.
F_k is the fixed cost of the vehicle of type k $(k=1,\ldots,K)$.
α_k is the unit running cost of the vehicle of type k $(k=1,\ldots,K)$
D_{ij} is the distance between nodes i and j $(i, j=1,\ldots, n+m)$.

2.2. Decision variables

Let
$X_{ijkd} = 1$ if a vehicle of type k $(k=1,\ldots,K)$ traveling along arc (i,j) $(i=1,\ldots,n+m;\ j=1,\ldots,n+m)$ and originating from depot d $(d=n+1, \ldots,n+m)$ is selected, and $X_{ijkd} = 0$ otherwise.

Y_{ij} is a non-negative continuous variable denoting the total load remaining in the vehicle before reaching node j while traveling along arc (i,j) $(i=1,\ldots,n+m;\ j=1,\ldots,n+m)$.

2.3. Formulation

$$\text{Minimise } Z = \sum_{d=n+1}^{n+m}\sum_{k=1}^{K} F_k \sum_{i=n+1}^{n+m}\sum_{j=1}^{n} X_{ijkd} + \sum_{d=n+1}^{n+m}\sum_{k=1}^{K}\sum_{i=1}^{n+m}\sum_{j=1}^{n+m} \alpha_k D_{ij} X_{ijkd} \quad (1)$$

$$\text{s.t.} \quad \sum_{d=n+1}^{n+m}\sum_{k=1}^{K}\sum_{i=1}^{n+m} X_{ijkd} = 1, \quad j=1,\ldots,n \quad (2)$$

$$\sum_{d=n+1}^{n+m}\sum_{k=1}^{K}\sum_{j=1}^{n+m} X_{ijkd} = 1, \quad i=1,\ldots,n \quad (3)$$

$$\sum_{i=1}^{n+m} X_{ijkd} = \sum_{i=1}^{n+m} X_{jikd}, \quad k=1,\ldots,K;\ j=1,\ldots n+m; \quad d=n+1,\ldots,n+m \quad (4)$$

$$\sum_{i=n+1}^{n+m}\sum_{j=1}^{n} Y_{ij} = \sum_{j=1}^{n} q_j \quad (5)$$

$$\sum_{i=1}^{n+m} Y_{ij} - \sum_{i=1}^{n+m} Y_{ji} = q_j, \quad j=1,\ldots,n \quad (6)$$

$$Y_{ij} \le \sum_{d=n+1}^{n+m}\sum_{k=1}^{K} Q_k X_{ijkd}, \quad i=1,\ldots,n+m; \quad j=1,\ldots,n \quad (7)$$

$$X_{d_1 ikd_2} = 0, \quad i=1,\ldots,n;\ k=1,\ldots,K; \quad d_1 \neq d_2 = n+1,\ldots,n+m \quad (8)$$

$$X_{id_1 kd_2} = 0, \quad i=1,\ldots,n;\ k=1,\ldots,K; \quad d_1 \neq d_2 = n+1,\ldots,n+m \quad (9)$$

$$X_{ijkd} \in \{0,1\} \quad i,\ j=1,\ldots,n+m;\ k=1,\ldots,K;\ d=n+1,\ldots,n+m \quad (10)$$

$$Y_{ij} \ge 0, \quad i,\ j=1,\ldots,n+m \quad (11)$$

The objective function (1) represents the total cost that includes both the vehicle fixed cost and the traveling cost. Constraints (2) show that each customer is visited only once, (3) ensure the flow conservation (a vehicle that enters a customer site must leave) and (4) guarantee that at most one vehicle type originating from a given depot will cover an arc i–j. Constraints (5) show that the total quantity leaving all depots is exactly the total customers demand and (6) guarantee that the quantity remaining after visiting customer j is exactly the load before visiting this customer minus the its demand. Constraints (7) guarantee that the vehicle capacity of any vehicle type is not violated. Constraints (8) and (9) guarantee that a customer leaving a depot (or returning to a depot) cannot be linked to a different depot respectively. Constraints (10) and (11) refer to the binary and the non-negativity of the decision variables respectively.

2.4. Tightening of the formulation

Constraints (12)–(18) are added to tighten the above formulation by reducing the search space. This is achieved by introducing valid inequalities (i.e., (12) and (13) instead of (7)) and by setting the optimal values of some of the decision variables to zero in equalities (14)–(18), as these do not need to be found due to the definition of the problem

$$Y_{ij} \le \sum_{d=n+1}^{n+m}\sum_{k=1}^{K} (Q_k - q_i) X_{ijkd}, \quad i=1,..,n+m; \quad j=1,...,n \quad (12)$$

$$Y_{ij} \ge \sum_{d=n+1}^{n+m}\sum_{k=1}^{K} q_j X_{ijkd}, \quad i, \quad j=1,...,n \quad (13)$$

$$Y_{ij} = 0, \quad i=1,...,n; \quad j=n+1,...,n+m \quad (14)$$

$$Y_{ij} = 0, \quad i=n+1,...,n+m; \quad j=n+1,...,n+m \quad (15)$$

$$Y_{ii} = 0, \quad i=1,...,n \quad (16)$$

$$X_{ijkd} = 0, \quad i, \quad j=n+1,...,n+m;\ k=1,...,K; \quad d=n+1,...,n+m \quad (17)$$

$$X_{iikd} = 0, \quad i=1,...,n+m;\ k=1,...,K; \quad d=n+1,...,n+m \quad (18)$$

Constraints (12) state that the load on any arc cannot be larger than the remaining load on that vehicle after delivering to customer i on arc (i,j) and originating from any depot. Note that for simplicity of the formulation the demand of each depot is set to zero. Constraints (13) guarantee that there is at least enough amount left to serve node j when traveling toward node j on arc (i,j). Constraints (14) are introduced to guarantee that all vehicles must travel empty to the depots and constraints (15) and (16) impose that there is no load carried between depots or from a customer to itself respectively. Constraints (17) and (18) are similar to the previous two sets of constraints as these impose that there is no arc between depots or from a customer to itself respectively using any type of vehicles. Note that the role of (16) and (18) are also part of constraints (6) but these will eliminate those combinations from the outset. The above formulation has $mK(n+m)^2$ binary variables, including $nK(3m-2)+m^3K+m^2K$ variables set to zero, $(n+m)^2$ integer variables with $n(m+1)+m^2+m$ variables set to zero, and $2n^2+n[3+m(1+K)]+Km^2+1$ constraints.

Note that alternative node-based formulations that use an integer variable (u_i) to denote the total demand left after leaving node i can also be adopted for our problem. These are used to satisfy both the subtour elimination constraints and the capacity constraints (see [17]).

2.5. Other practical considerations

The formulation can be modified to cater for some other practical scenarios such as:

(a) If the number of vehicles of a given type is known, say N_k, the following set of constraints can be added:

$$\sum_{d=n+1}^{n+m}\sum_{i=n+1}^{n+m}\sum_{j=1}^{n} X_{ijkd} \le N_k, \quad k=1,...,K \quad (18)$$

(b) If the total number of vehicles at a given depot is known, say, M_i, the following set of constraints can be added:

$$\sum_{k=1}^{K}\sum_{j=1}^{n} X_{ijki} \le M_i, \quad i=n+1,...,n+m \quad (19)$$

In addition, if the number of vehicles of type k at depot i is also limited, say to M_{ik}, the following constraints similar to (13) can

be added:

$$\sum_{j=1}^{n} X_{ijki} \le M_{ik}, \quad i = n+1, ..., n+m; \quad k = 1, ..., K \tag{20}$$

(c) If at a given depot, say d_1, denoted by $d = n + d_1$, some type of vehicles cannot be accommodated, say $V(d)$ be such a set, then

$$X_{djkd} = 0, \quad k \in V(d) \subset \{1, .., K\}; \quad j = 1, ..., n \tag{21}$$

(d) If a vehicle is not required to return to the same depot from where it originated, the problem becomes relatively easier to solve. This is achieved by using a similar formulation (1–17) except that a 3 index variables X_{ijk} is used instead.

(e) If a route length constraint is imposed (TD), the following sets of constraints are added with the new non-negative continuous variable T_{ijd} denoting the shortest length traveled from depot d to customer j with i being the predecessor of j. These are adapted from Kara [16] flow or arcs-based constraints proposed for the distance constrained VRP

$$\sum_{j \ne i}^{n+m} T_{ijd} - \sum_{j \ne i}^{n+m} T_{jid} = \sum_{j=1}^{n+m} D_{ij} \sum_{k=1}^{K} X_{ijkd} \quad \forall i = 1, ..., n; \quad d = n + 1, ..., n+m \tag{22}$$

$$T_{ijd} \le (TD - D_{jd}) \sum_{k=1}^{K} X_{ijkd}, \quad i \ne j = 1, ..., n; \quad d = n+1, ..., n+m \tag{23}$$

$$T_{idd} \le TD \sum_{k=1}^{K} X_{idkd}, \quad i = 1, ..., n; \quad d = n+1, ..., n+m \tag{24}$$

$$T_{ijd} \ge (D_{ij} + D_{di}) \sum_{k=1}^{K} X_{ijkd}, \quad i \ne j = 1, ..., n; \quad d = n+1, ..., n+m \tag{25}$$

$$T_{did} = D_{di} \sum_{k=1}^{K} X_{dikd}, \quad i = 1, ..., n; \quad d = n+1, ..., n+m \tag{26}$$

$$T_{ijd} = 0, \quad i, j = n+1, ..., n+m; \quad d = n+1, ..., n+m \tag{27}$$

$$T_{ijd} \ge 0, \quad i, j = 1, ..., n+m; \quad d = n+1, ..., n+m \tag{28}$$

Alternative node-based formulations that use a non-negative variable v_i, defining the distance (time) a vehicle travels up to node I, can also be explored here (see [2]).

3. The variable neighborhood search (VNS) algorithm

3.1. Overview

The MDHFVRP is NP hard given it can be reduced to the classical VRP, which is known to be NP hard, by setting $K=1$ and $m=1$. It is therefore difficult, if not impossible, to obtain an optimal solution for large size problems within a reasonable amount of computing time even with the use of the most powerful MIP solvers such as CPLEX. This shortcoming will be shown in the computational results section (see Section 6). The best way forward is therefore to tackle this complex combinatorial problem by means of an appropriate design and an efficient implementation of one of the modern heuristics which are also commonly known as meta-heuristics. For an overview on heuristic search in general, see Salhi [36].

In this study, we opt for a VNS-based algorithm as this meta-heuristic has so far proved to be successful in solving a variety of hard combinatorial problems. To our knowledge, this is the first time a VNS heuristic is implemented for this particular routing problem. For basic information as well as new developments of VNS related work including successful applications, the reader would find the original paper by Mladenovic and Hansen [28] and the recent review paper by Hansen et al. [14] to be useful and informative.

The overview of our approach is briefly given as follows. We first split the customers set into two subsets; one for those customers that are served from their nearest depots and the others, which we refer to as the *borderline customers*, are inserted afterward. We define a borderline customer as a customer that happens to be situated approximately halfway between its nearest and its second nearest depots. An illustration of the borderline customers for the case of three depots is shown in Fig. 1 but a formal definition is provided in Section 4.4.

First each depot with its associated customers, excluding the borderline customers, is initially considered individually and solved as a standard single depot HFVRP which we briefly describe here. For each depot, starting from a customer location, a tour which includes the depot and which passes through all customer locations is first constructed using the Sweep method (see [11]). This tour construction is repeated several times starting from different customer points to construct giant tours. These large tours are then refined using the 2-opt procedure [26]. For each giant tour, the optimal partitioning (i.e., the routes and the different types of vehicles) is obtained. We use Dijkstra's algorithm [8] but dynamic programming can also be used as the cost network is a directed network. As the tour partitioning optimality is guaranteed only on each giant tour, the solution made up of this set of routes (without the borderline customers) is then improved using our VNS-based heuristic as follows.

All borderline customers are then inserted into their best feasible insertion places based on the routes originating from their nearest and second nearest depots only. Once all customers are included into their respective routes, all the routes from all depots, including an empty route associated with each depot, are put together as one large set of routes. These routes are identified by their end nodes which correspond to their originating depot. The implementation of our VNS is then applied using all these routes regardless from which depot they originated from. Once this improvement phase is completed, this large set of routes is then split back to its original subset of routes. Note that each subset is identified by one depot only and contains only those routes that are served from that particular depot. For each depot with its corresponding routes, a diversification procedure is then applied to dismantle the existing cost network and hence perturb the existing solution. A new set of routes is then determined from each depot using Dijkstra's algorithm.

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 of the algorithm, we disaggregate the

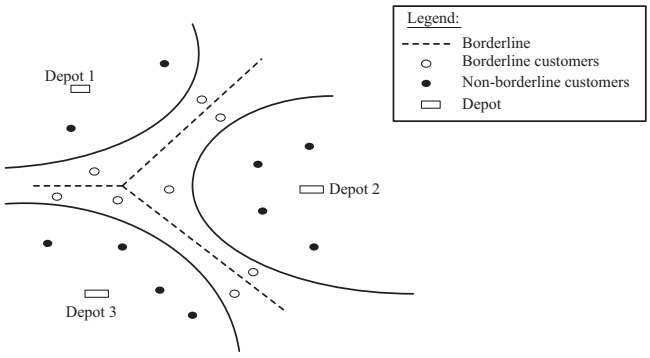

Fig. 1. An illustration of the borderline customers.

Step 0: *Initialisation*

(a) Define the set of neighborhood structures N_k, for $k = 1,\ldots,k_{\max}$ and the set of local search operators R_l, for $l = 1,\ldots,l_{\max}$. Set the values for *MaxDiv, MaxDijkstra*, *maxiter1* and *maxiter2*. Initialise *NbDiv* and *NbDijkstra* to 0.

(b) Define the set of borderline customers, see subsection 4.4, and allocate the other customers to their nearest depots.

(c) For each depot *j (j=n+1,...,n+m)* and its corresponding customers generate an initial partial solution x_j (as shown in Subsection 4.1) and set $x_{jbest} = x_j$ as the incumbent best solution for depot *j* and define $X_{best} = \bigcup\{x_j; j = n+1,\ldots,m+n\}$

Step 1: *Application of VNS within each depot j (j=n+1 to n+m)*
Repeat the following sequence until *maxiter1* is reached:

(a) set $k = 1$

(b) Repeat the following steps until $k = k_{\max}$:

(i) *Shaking*: Generate a solution x'_j at random from the k^{th} neighbourhood of x_j $(x'_j \in N_k(x_j))$;

(ii) *Local search:* Apply the multi-level approach (see Subsection 4.3) to find the best neighbouring solution x''_j.

(iii) *Move or not:* If x''_j yields a better quality solution than the incumbent x_j, set $x_j = x''_j$ and go to Step 1(a); otherwise set $k = k+1$ and go to Step 1(b).

Step 2: *Insertion of the borderline customers*
Insert these borderline customers into routes belonging to their closest or second closest depots using the least insertion cost procedure (see Subsection 4.4).

Step 3: *Application of VNS on multiple depots*

(a) Put all routes originating from all depots together as a large set of routes and define *X* as the full solution of the multi-depot problem.

(b) Apply the VNS procedure of Step 1 using the solution X for *maxiter2* iterations (here, the solution is represented by X instead of the x_j).

Step 4: *Diversification phase*
While *MaxDiv* is not reached, disaggregate the routes according to their depots, apply the diversification procedure within each depot (see Subsection 4.6), set *NbDiv* = *NbDiv*+1 and go to Step 3.

Step 5: *Termination criterion*

(a) If *NbDijkstra* > *MaxDijkstra,* use the solution *X* as the final solution and stop,

(b) Else

(i) For each depot *j=n+1 to n+m,*

- construct the cost network using the partial solution of the incumbent X.
- apply Dijkstra's algorithm to get the new partial solution x_j for the j^{th} depot (see Subsection 4.7).

(ii) Compute the new solution made up of the combined partial solutions, $X' = \bigcup\{x_j; j = n+1,\ldots,m+n\}$. If X' yields a better solution than the one generated by X, set $X = X'$, *NbDijkstra= NbDijkstra+1* and *NbDiv=0* and go to Step 3(b), else use the solution *X* as the final solution and stop.

Fig. 2. The VNS-based algorithm for the MDHFVRP.

routes again into their corresponding depots where Dijkstra's algorithm is used as our final post optimiser. If this last step of the process produces an improved solution, the entire search is repeated by putting all routes together again where our VNS is activated, otherwise the search terminates.

We would like to emphasize that the difference of considering multiple depots simultaneously when compared to solving multiple single depot problems separately is that the local search operators are carried out differently. In this case, some customers can be too far away from each other that they will obviously be served from different depots and hence need not be considered together as potential neighbors in the local search. On the other hand those customers that are close to each other such as borderline customers, even if they belong to different depots may be re-assigned to routes originating from other depots. In addition, it is worth mentioning that each route should start and finish at the same depot and therefore this added restriction needs to be checked as part of the implementation. This guarantees that the search, to maintain feasibility, would not explore a better quality solution by attempting to connect the two end points of a route to two different depots.

3.2. The VNS-MDHVRP algorithm

Fig. 2 summarizes the main steps of the VNS-based heuristic with the necessary additional notations provided here.

x_j: a solution based on depot *j* only, $j = n+1,\ldots, n+m$.
X: a solution based on all depots (i.e., $X = \cup\{x_j; j = n+1, \ldots, n+m\}$).
$k_{\max}$: the maximum number of neighborhoods.
$N_k(x_j)$: the *k*th neighborhood using solution x_j ($k = 1,\ldots k_{\max},$).
$l_{\max}$: the maximum number of local search operators.
R_l: the *l*th local search operator ($l = 1,\ldots l_{\max},$).
ND: the number of times diversification is carried out within each depot.
MaxDivL: the maximum number of diversifications.
MaxDijkstra: the maximum number of times Dijkstra algorithm is used.
*maxiter*1: the maximum number of iterations VNS is applied in step 1 (within each depot).
*maxiter*2: the maximum number of iterations VNS is applied in step 4 (between depots).

NbDiv: a counter for the number of diversification steps.
NbDijkstra: a counter for the number of times Dijkstra's algorithm is used.

4. Explanation of the main steps

4.1. Initial solution (step 0)

The initial solution, without the borderline customers, is obtained in four phases: (i) for each depot form a giant tour using the sweep algorithm of Gillett and Miller [11], (ii) improve this tour using the 2-opt of Lin [26], (iii) construct the cost network and (iv) apply Dijkstra's algorithm (1959) to find the optimal routes with their corresponding vehicle types.

To avoid using large distances between two successive customers in a given route (i.e., large gaps), the two customers that make up each of the large gaps are used as the starting and ending points in the giant tours and hence in the construction of the cost networks using clockwise and anticlockwise. The number of gaps (*NG*) is defined as follows:

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

where *NR* represents 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 customers *i* and *i*+1), $\overline{g}$ is the average gap, and g^+ is the largest gap.

For each of the *NG* selected gaps, two cost networks are then generated one anticlockwise and one clockwise, and then Dijkstra's algorithm is then applied to each of these 2*NG*cost networks. The choice of having at most eight gaps is based on identifying two gaps in each quadrant approximately while it is also shown empirically that eight gaps (16 cost networks) do not require an excessive amount of computational burden.

4.2. A brief on the construction of the cost network

Consider a giant tour originating from the depot *d* and having *k* customers put for simplicity in the following sequence (*d*–1–2–3–⋯–*k*–*d*).

We first calculate the cost from the depot *d* to customer 1 and from this customer to the depot as the cost of the arc (*d*,1). This is expressed as $C_{d1} = F_1 + \alpha_1(2D_{d1})$. If the sum of the demands of customers 1 and 2 is less than the smallest vehicle, the arc (*d*,2) is added to the network with a cost of $C_{d2} = F_1 + \alpha_1(D_{d1} + D_{12} + D_{2d})$. We continue with this construction until the smallest vehicle is not big enough to accommodate the new customer, then the next large vehicle is used. Assume we can only have customers 1, 2, and 3 together in the smallest vehicle then customers 1, 2, 3, and 4 can be served by the second largest vehicle represented by the arc (*d*,4) which has a cost $C_{d4} = F_2 + \alpha_2(D_{d1} + D_{12} + D_{23} + D_{34} + D_{d4})$. The process is continued until there are no more arcs connecting the last customer in the giant tour.

In general, the cost of arc (*i*,*j*) is defined as $C_{ij} = F_s + \alpha_s(D_{d,i+1} + \Sigma_{k=i+1}^{j-1} D_{k,k+1} + D_{jd})$ where *s* denotes the smallest vehicle that accommodates the total demand $\Sigma_{k=i}^{j} q_k$. This giant tour is then optimally partitioned using Dijkstra's algorithm.

4.3. Neighborhood structures (steps 1 and 3)

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. These are briefly described as follows:

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 of all other routes. This procedure is repeated until a feasible move is found.
The 2–0 *shift* (*type* 1 *and type* 2): In the 2–0 shift of type 1, two consecutive random customers from a randomly chosen route are selected. This pair of customers is then checked for possible insertion in other routes. This procedure is repeated until a feasible move is found. The 2–0 shift of type 2 is similar to the one of type 1 except that the two customers are considered for insertion into two different routes. These insertion moves are carried out in a systematic manner.
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.
Perturbation mechanisms (*type* 1 *and type* 2): This scheme was initially developed by Salhi and Rand [37] for the VRP by considering three routes simultaneously. The idea is to systematically take a customer from a route and relocate it into another route without considering capacity and time constraints in the receiver route. A customer from this receiver route is then shifted to the third route if both capacity and time constraints for the second and the third routes are not violated. This is the perturbation of type 1. An extension of this perturbation is to shift two consecutive customers from a route instead of removing one customer only. We refer to this as the perturbation of type 2. These moves are evaluated systematically.

4.4. Local search operators (steps 1 and 3)

Our multi-level local search heuristic uses six neighborhood search operators which are briefly described here.

The 1-*insertion procedures* (*intra-route and inter-route*): In the 1-insertion intra-route a customer is removed from its position in a route and inserted elsewhere within that route. This move is repeated and the one that yields the largest improvement is chosen. On the other hand, in the 1-insertion inter-route, the removed customer is checked for possible insertion into other routes systematically while maintaining feasibility.
The 2-*insertion* (*intra-route*): This is similar to the 1-insertion intra-route except that here two consecutive customers are taken into consideration.
The 2-*opt* (*intra-route and inter-route*): The 2-opt intra-route, which is usually known as the 2-opt [26], is an old but a simple and an effective improvement procedure that works by removing two non-adjacent arcs and adding two new arcs by reversing the direction of one of the two resulting paths while maintaining the tour structure. The 2-opt inter-route is similar except that it is applied to two routes instead of one.
The swap (*intra-route*): The swap intra-route aims at reducing the total cost of a route by swapping positions of a pair of customers within a route.

In our implementation, these neighborhood search operators are applied in the following order: the 1-insertion inter-route as R_1, the 2-opt inter-route as R_2, the 2-opt intra-route as R_3, the swap

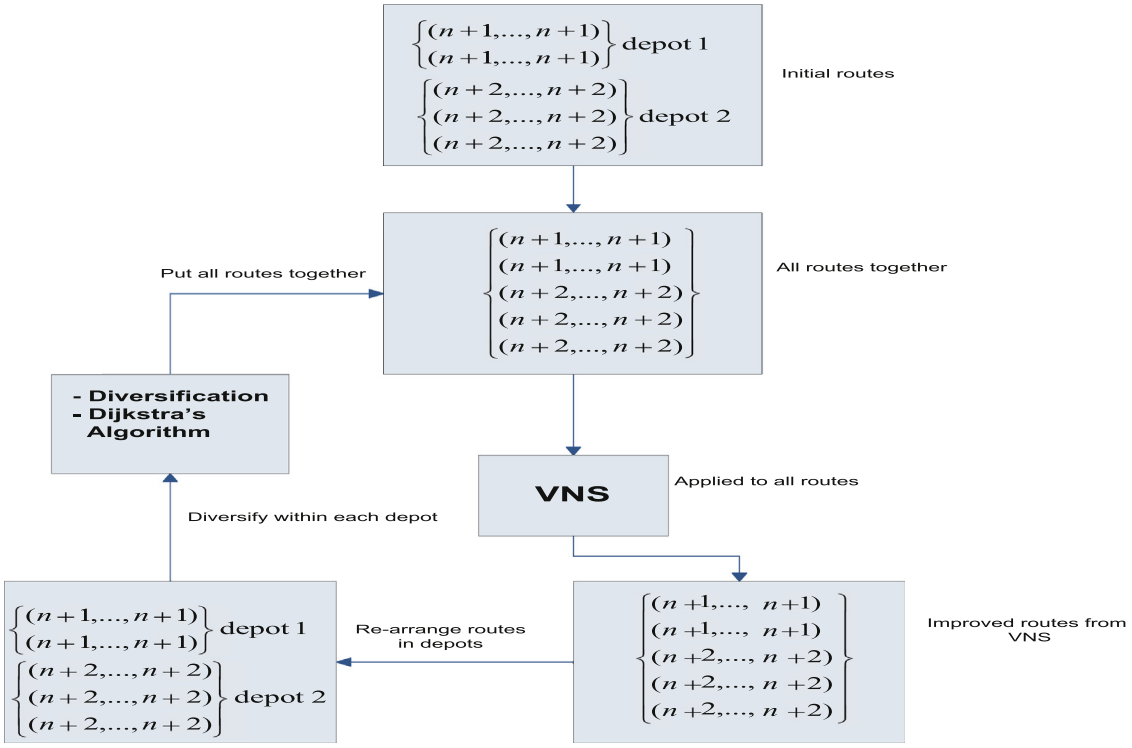

Fig. 3. The aggregation and disaggregation procedure: the case of two depots and five routes.

intra-route as R_4, the 1-insertion intra-route as R_5, and finally the 2-insertion intra-route as R_6. This order is chosen based on the level of complexity of the local search operators. This choice was shown to be effective in a related study by Salhi and Sari [39]. For instance, the process starts by generating a random feasible solution x' from N_1, which is used as the temporary solution. The multi-level local search approach is used to find 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. Note that the same process is also applied in step 3b of the algorithm except here all routes from all depots are considered together.

4.5. Definition and insertion of the borderline customers (step 2)

We first provide the following additional notations:

p_i: the nearest depot to customer i ($i=1,\ldots,n$) where $p_i \in \{n+1,\ldots,n+m\}$ with $p_i = ArgMin(D_{ij}, j=n+1,\ldots,n+m)$.

p_i': depot to customer i ($i=1,\ldots,n$) where $p_i' \in \{n+1,\ldots,n+m\}$ with $p_i' = ArgMin(D_{ij}, j=n+1,\ldots,n+m; j \neq p_i)$.

B: the set of borderline customers.

ε: a positive value for the borderline customers ($0<\varepsilon<1$) but close to 1.

Here, we determine the borderline customers as follows:

- For each customer $i=1,\ldots,n$, find p_i and p'_i and set $B=\phi$.
- For each customer $i=1,\ldots,n$.
  - Compute $\rho = D_{ip_i}/D_{ip_i'}$.
  - If $\rho \leq \varepsilon$ (i.e., customer i is not a borderline), allocate customer i to its nearest depot p_i.

Otherwise consider customer i as a borderline customer which will be left temporarily unassigned at this stage (i.e., this may be assigned to either p_i or p'_i) and set $B = B \cup \{i\}$.

The larger the value of ε, the smaller the number of borderline customers is. The two extreme values $\varepsilon=0$ or 1 are not considered as explained next. If $\varepsilon=1$, the only borderline customers will be those customers that happen to lie exactly halfway between their two nearest depots (this is usually an empty subset). In this case the problem reduces to the classical 'allocate all customers first then route second' strategy. If $\varepsilon=0$, every customer is a borderline customer (this is the full customer set) and the problem becomes one single large problem without any consideration of the customer locations. According to previous experiments by Salhi and Sari [39], the value of $\varepsilon=0.7$ was found to generate an appropriate number of borderline customers.

4.6. The insertion procedure of the borderline customers

All borderline customers are inserted in their best possible place in the routes originated from the nearest and the second nearest depots. This is carried out using the 1–0 insertion procedure as described earlier.

For each customer $i \in B$, compute

$$Ins(i) = Min\{Ins(i,\sigma_r), \sigma_r \in DP1(i) \cup DP2(i)\}$$

with $$Ins(i,\sigma_r) = \underset{s \in \sigma_r}{Min}(D_{i,s} + D_{i,s+1} - D_{s,s+1}).$$

where

σ_r denotes the rth route and $s \in \sigma_r$ refers to the sth customer in this route.

$DP1(i)$ and $DP2(i)$ are the set of routes originating from the nearest and the second nearest depots of customer i respectively ($i=1,\ldots,n$) (i.e., those routes originating from p_i and p'_i).

4.7. Aggregation of all routes from all depots (steps 3 and 4)

In both steps, all routes are considered together to form one large set of routes irrespective of their originating depots. As each route is defined as a string where the first and the 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 (see step 3(b) of the algorithm). In this case, some customers may now have the opportunity to shift between routes that are not necessarily originating from the same depot. This shift operator is more effective for those borderline customers.

Fig. 3 illustrates the case of two depots ($m=2$) with five routes (two routes in the first depot and three routes in the second one) where the jth depot is represented by $n+j$ ($j=1,\ldots,m$). Fig. 3 also represents a brief flow chart that describes how routes are aggregated, the VNS applied, the routes regrouped again on a depot by depot basis which allow the diversification procedure and the Dijkstra's algorithm to be used. These last two components are described briefly in Sections 4.6 and 4.7 respectively. The aggregation and disaggregation procedures are applied several times until the stopping criterion is fulfilled.

4.8. The diversification procedure (step 4)

The diversification procedure is used when there is no further improvement after step 3 is performed for *maxiter*2 times. The idea is to explore those regions of the search space that may not have been visited otherwise. Here we 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 way will ensure that a route from this incumbent best solution will be split and therefore the new cost network, which does not include the current solution, is constructed, and hence a new solution will be generated. The main steps of this procedure are given in Fig. 4.

In this study, the number of diversifications (ND) used within each depot is set as $ND = Min(100, 2n)$, where n represents the number of customers in a given instance.

4.9. Use of Dijkstra's algorithm as an extra refinement (step 5)

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. A cost network is then constructed and Dijkstra's algorithm is applied, acting as a post optimizer in this case. The aim here is to see whether the new combined solution (made up of those optimal partial solutions for the shortest path based on their corresponding cost networks) is better than the current one or not.

5. Incorporation of the neighborhood reduction test

One way to reduce the CPU time is to integrate a reduction test into the search. The aim of using a neighborhood reduction scheme is to remove a large amount of unnecessary calculations (useless checks) that would have been carried out otherwise, usually at the expense of a small (or negligible) amount of risk in missing a better solution. The ideal scheme would obviously be the one that eliminates all unnecessary evaluations without affecting the solution quality. In this research, we adopt a scheme originally designed by Salhi and Sari [39] and which is based on the relationship between borderline customers and their neighboring customers. In other words, for each borderline customer, we determine those customers which are situated within a given radius. We introduce a logical matrix $pos(i,j)$ defining whether or not customers i and j are worth considering together for our neighborhood moves. For instance, if two customers happen to be too far from each other, we can assume with a small risk that these two customers are unlikely to be inserted next to each other. Based on this observation, we set $pos(i,j)=true$ if these two customers are potential candidates which can then be evaluated for any possible move within the search, and $pos(i,j)=false$ otherwise. It is worth noting that in our case, in several instances, this reduction test not only cut down in the computational time, but it also provided a basis for the search to find new best solutions. The steps of this neighborhood reduction test, known as *The Between Depot Reduction Test*, are given in Fig. 5. We refer to our VNS implementation with reduction test as VNS2 and the one without it as VNS1.

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

Fig. 4. The perturbation-based diversification mechanism.

Step 1- Set $pos(i,j) = \textbf{\textit{false}},\ \forall i,j = 1,\ldots,n+m$ and define the set of borderline customers, say B, using $\varepsilon = 0.7$.

Step 2- For each depot, j, define S_j as the set of customers whose nearest depot is depot j (i.e., $S_j = \{i = 1,\ldots,n / p_i = j\}$) and set $pos(i_1,i_2) = \textbf{\textit{pos}}(i_2,i_1) = \text{true},\ \forall i_1,i_2 \in S_j$ and $pos(i_1,p_j) = \textbf{\textit{pos}}(p_j,i_1) = \text{true}\ \forall i_1 \in S_j$

Step 3- For all $i_1 \in B$ do
- Compute the radius $RAD(i_1) = (D_{i_1,p_{i_1}})/2$
- Set $pos(i_1,p'_{i_1}) = \textbf{\textit{pos}}(p'_{i_1},i_1) = \textbf{\textit{true}}$,
- for all $i_2 \in \{1,\ldots,n\} - S_j$ with $j = p_{i_1}$ do
if $D_{i_1,i_2} \le RAD(i_1)$ set $\textbf{\textit{pos}}(i_1,i_2) = \textbf{\textit{pos}}(i_2,i_1) = \textbf{\textit{true}}$.

Fig. 5. The neighborhood reduction test.

6. Computational results

Our VNS-based algorithm is coded in C++ and executed on a Pentium IV-M with 1 GB of RAM. To our knowledge, the only data set that exists in the literature and which can be used in our testing consists of the 26 instances generated by Salhi and Sari [39], and whic h are derived from the commonly used MDVRP data set. These instances vary in size from 50 to 360 customers and 2 to 9 depots.

Hence, our proposed heuristic will be tested against the lower and upper bounds which we denote by LB and UB respectively found by using CPLEX for 3 h CPU. The results are also compared against those obtained in Salhi and Sari [39] which we refer to SS.

6.1. Data instances

For convenience, we briefly describe how these instances were generated. The vehicle specifications which include the number of vehicle types (K), vehicle capacity (Q_k), vehicle variable cost (α_k) and vehicle fixed cost (F_k) are derived using the following formula.

$Q_k = (0.4 + 0.2k)\widehat{Q}$; $\alpha_k = 0.7 + 0.1k = (0.8,\ 0.9,\ 1,\ 1.1,\ 1.2)$ and $F_k = 70 + 10k = (80,\ 90,\ 100,\ 110,\ 120)$ with $k=1,\ldots,K$ ($K=5$) and $\widehat{Q}$ denoting the original vehicle capacity used for the classical MDVRP instances.

6.2. VNS algorithm parameters

The parameter values used in our implementation include:

$maxiter1 = 50$, $maxiter2 = 100$, $MaxDiv = 5$ and $MaxDijkstra = 5$.

These values were empirically found using some preliminary testing aimed at balancing, in terms of computing effort, all the main steps of the algorithm (see [20]).

6.3. Summary of the results

Table 2 displays the characteristics of the problems, the lower and upper bounds found by CPLEX 12.3, the solution costs and the corresponding CPU time (in minutes) found by SS, VNS1 and VNS2. The best results are shown in bold. In summary, we also give the overall average deviation (in %) over all the 26 instances found by VNS1, VNS2, and SS where the deviation for each instance is computed as follows:

$$\text{Deviation } (\%) = (\text{cost (heuristic)} - \text{cost (best)})/\text{cost(best)}) \times 100.$$

6.4. VNS without reduction test (VNS1)

If we consider the solution values of SS and VNS1 in Table 2, our algorithm produces 10 new best solutions out of 26. The worst deviation that VNS1 produced from SS is just below 1.3% which is found for instance #26.

The overall average deviation from the best VNS1 is much smaller than SS (0.384 vs 1.581). In terms of CPU time, the SS method consumes less than VNS1, especially in the largest instances where SS is nearly 10 times faster.

6.5. VNS with reduction test (VNS2)

According to Table 2, VNS2 produces a better average deviation than SS but it is slightly larger than VNS1 as one might expect. However, it yields more new best solutions (13 new ones). This type of behavior would not happen if exact methods were used instead. This is due to the fact that the optimal solution over a larger feasible region cannot obviously be worse than the one found over a subset of the larger region. This claim is not true for the case of heuristics as there is always the risk that the search would not be exhaustive and hence optimality would not be guaranteed. The CPU time of VNS2, as expected, is found to be much smaller than VNS1. On average VNS2 uses 3.71 min whereas VNS1 requires 14.04 min which is nearly four times slower (an average saving of about 75%). This saving in computer time becomes even more significant for the larger instances where VNS2 requires about 20% of the time of VNS1. However, when compared to SS, VNS2 uses approximately a similar amount of computational time.

We would like to note that SS uses a VAX type machine 15 years ago whereas VNS implementations are implemented on a Pentium IV.

6.6. Lower and upper bound information

For completeness, we have also provided the lower and upper bounds using 3 h CPU of CPLEX 12.3. It was found that the new

Table 2
Results for the VNS-MDHFVRP algorithm ($K=5$).

Problem details					CPLEX with tightening/(3 h CPU)		Total cost (solution value)			CPU time (in min)		
#	***n***	***m***	$\widehat{Q}$	**TD**	**LB**	**UB**	**SS**	**VNS1**	**VNS2**	**SS**	**VNS1**	**VNS2**
1	55	4	100	∞	1299.8	1621.9	1428.4	**1419.7**	1427.6	0.1	1.7	0.3
2	85	3	100	∞	1996.0	2677.8	**2131.9**	2132.3	2132.5	0.6	1.4	0.7
3	85	3	160	∞	1326.1	2516.0	1499.5	1494.2	**1489.6**	0.6	1.4	0.7
4	50	4	80	∞	1318.5	1725.2	1526.7	1503.3	**1499.3**	0.1	0.6	0.2
5	50	4	160	∞	799.7	1378.9	992.8	**974.9**	984.5	< 0.1	0.7	0.2
6	75	5	140	∞	1341.4	2561.1	1611.1	**1571.2**	1588.4	0.2	1.5	0.5
7	100	2	100	∞	2079.3	3039.7	2361.9	2321.9	**2313.7**	0.6	1.8	1.4
8	100	2	200	∞	1228.3	2265.6	1498.4	**1459.3**	1466.9	2.0	3.0	1.8
9	100	3	100	∞	1970.2	3390.3	2277.5	**2224.0**	2246.9	0.5	2.1	1.0
10	100	4	100	∞	1971.3	3609.9	2297.1	**2249.2**	2256.4	0.6	2.1	1.1
11	249	2	500	310	5428.7	23284.4	6718.6	**6610.1**	6696.0	27.1	21.5	13.7
12	249	3	500	310	NF	NF	6211.4	6094.7	**6068.8**	13.6	16.3	7.3
13	249	4	500	310	//	//	6018.7	**5943.3**	6043.0	7.2	19.0	6.2
14	249	5	500	310	//	//	6030.8	5915.7	**5882.8**	7.4	19.3	6.4
15	80	2	60	∞	1607.6	2846.4	2108.2	2085.8	**2076.2**	0.5	2.3	1.2
16	80	2	60	200	//	//	2126.8	2118.6	**2096.4**	0.5	3.3	1.1
17	80	2	60	180	//	//	2160.1	2188.6	**2139.3**	0.4	3.2	1.1
18	160	4	60	∞	3032.6	NF	4116.2	**4001.8**	4024.9	2.3	7.8	2.7
19	160	4	60	200	//	//	4178.9	4167.7	**4148.7**	2.1	7.8	2.7
20	160	4	60	180	//	//	4344.1	4358.3	**4338.2**	1.2	10.8	3.3
21	240	6	60	∞	NF	NF	6217.0	5994.6	**5970.5**	4.2	22.1	4.8
22	240	6	60	200	//	//	6233.6	6216.3	**6196.3**	3.8	21.6	5.1
23	240	6	60	180	//	//	**6493.1**	6539.4	6567.1	2.0	25.2	5.3
24	360	9	60	∞	//	//	9184.6	8935.6	**8883.1**	13.4	58.5	10.9
25	360	9	60	200	//	//	9332.0	**9263.0**	9294.8	8.7	55.4	7.8
26	360	9	60	180	//	//	**9706.6**	9831.5	9887.7	4.9	54.7	9.0
Average deviation (%)					//	//	1.581	**0.384**	0.440			
# Best solutions							3	10	**13**			

TD: the maximum distance allowed.
$\widehat{Q}$: vehicle capacity used in the classical MDVRP data set.
LB, UB: lower and upper bounds using 3 h of CPU with CPLEX 12.3 on PC Intel ® Core™ 2 Duo with 8 GB.
SS: Salhi and Sari [39] on VAX4000-500.
VNS1, VNS2: VNS without and with the reduction test respectively on Pentium IV-M with 1 GB.
NF: no solution (upper or lower bound) was found during the allowed CPU time.

formulation produced tighter bounds on all the instances where CPLEX found the bounds. In brief, the formulation with tightening yielded an average improvement of nearly 2% and over 60% for LB and UB respectively. Moreover, the largest deviation is found to be nearly 4% and over 110% for LB and UB respectively. Though the upper bounds are improved these are still inferior to the heuristic solutions found by VNS. The average percentage gaps between the heuristic solutions and the tighter lower bound, which are computed over those instances where the LB is found, are 19.1%, 17.1% and 17.6% for SS, VNS1 and VNS2 respectively. The gap is defined as

$$\text{Gap}(\%) = (\text{cost(heuristic)}-\text{LB}/\text{LB}) \times 100.$$

These large gaps may not be that informative and useful. This could be due to the limited CPU time allowed for CPLEX to obtain these lower bounds.

In summary it is worth noting that if we consider both implementations (i.e., VNS1 and VNS2) together, we can claim that a total of 23 new best solutions out of the 26 are now discovered. These new results can serve as a useful guide for benchmarking purposes.

7. Conclusion and suggestions

In this study, the multi-depot routing with heterogeneous fleet which seems to suffer from a shortage of published work is examined. An MILP formulation with new valid inequalities and some variable settings are proposed to tighten the original formulation. An efficient VNS-based algorithm to solve the MDHFVRP is designed. The algorithm is equipped with a scheme for determining borderline customers, a multi-level heuristic acting as the local search engine, Dijkstra's algorithm for determining the optimal partitioning, a diversification procedure and a mechanism to aggregate the routes from different depots and disaggregate them into corresponding depots accordingly. A neighborhood reduction test that aims to eliminate unnecessary computations is also introduced to speed up the search by cutting nearly 80% of the original amount of CPU time, especially on the large instances. For benchmarking purposes the proposed VNS-based implementations obtained 23 new best solutions in total out of the 26 instances. This is a remarkable result which shows clearly the superiority of our VNS-based heuristics.

The proposed methodology could be adapted to solve related multi-depot routing problems where for instance the vehicle does not need to return to its original depot. From a strategic view point, given that the customer demand may change over time, the vehicle fleet composition problem, which is a medium term decision problem and which is found as a part of our solution in this study, could become even harder to determine. An investigation on how to obtain robust solutions could be worthwhile as it is of interest to both academics as well as practitioners.

Acknowledgments

The authors would like to thank both referees for their constructive suggestions which improved the content as well as the presentation of the paper. This research has been in part financed by the Algerian Ministry of Education (Sciences Fondamentales) under the research project PNR 8/U160/64 and the Indonesian Directorate General of Higher Education (DIKTI).

References

[1] Ball M, Golden B, Assad A, Bodin L. Planning for truck fleet size in the presence of a common carrier option. Decision Science 1983;14:103–20.
[2] Bektas T, Elmastas S. Solving school bus routing problems through integer programming. Journal of the Operational Research Society 2007;58:1599–604.
[3] Benton WC. Evaluating a modified heuristic for the multiple vehicle scheduling problem. Working paper RS86-14, College Administration Science, The Ohio State University, Columbus, OH; 1986.
[4] Cassidy PJ, Bennet HS. TRAMP—a multi-depot vehicle scheduling system. Operational Research Quarterly 1972;23:151–62.
[5] Chao IM, Golden B, Wasil E. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. American Journal Mathematics Management Science 1993;13:371–406.
[6] Cordeau JF, Gendreau M, Laporte G. A tabu search heuristic for periodic and multi-depot vehicle routing problem. Networks 1997;30:105–19.
[7] Cordeau J-F, Laporte G, Mercier A. An improved tabu search algorithm for handling of route duration constraints in vehicle routing problems with time windows. Journal of the Operational Research Society 2004;55:542–6.
[8] Dijkstra EW. A note on two problems in connection with graphs. Numerische Mathematik 1959;1:269–71.
[9] Dondo R, Cerda J. A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. European Journal of Operational Research 2007;176:1478–507.
[10] Gillett BE, Johnson JW. Multi-terminal vehicle dispatching algorithm. Omega 1976;4:711–8.
[11] Gillett BE, Miller LR. A heuristic algorithm for the vehicle dispatch problem. Operations Research 1974;22:340–4.
[12] Giosa I, Tansini I, Viera I. New assignment algorithms for the multi-depot vehicle routing problem. European Journal of Operational Research 2002;53 997–984.
[13] Golden B, Magnanti T, Nguyen H. Implementing vehicle routing algorithms. Networks 1977;7:113–48.
[14] Hansen P, Mladenovic N, Moreno Perez JA. Variable neighborhood search: methods and applications. Annals of Operations Research 2010;175:367–407.
[15] Ho W, Ho GTS, Ji P, Lau HCW. A hybrid genetic algorithm for the multi-depot vehicle routing problem. Engineering Applications of Artificial Intelligence 2008;21:548–57.
[16] Kara I. Arc based integer programming formulations for the distance constrained vehicle routing problem. In: Third IEEE international symposium on logistics and industrial informatics (LINDI); 2011. p. 33–8, 10.1109/LINDI.2011.6031159.
[17] Kara I, Laporte G, Bektas T. A note on the lifted Miller–Tucker–Zemlin subtour elimination constraints for the capacitated vehicle routing problem. European Journal of Operational Research 2004;158:793–5.
[18] Klots B, Gal S, Harpaz A. Multi-depot and multi product delivery optimization problem with time and service constraints. IBM Israel; 1992 Haifa Technical Report.
[19] Kuo Y, Wang CC. A variable neighborhood search for the multi-depot vehicle routing problem with loading. Expert Systems with Applications 2012;39:6949–54.
[20] Imran A. An adaptation of metaheuristics for the single and multiple depots heterogeneous fleet vehicle routing problems. PhD thesis. University of Kent, Canterbury, UK; 2008.
[21] Imran A, Salhi S, Wassan NA. A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem. European Journal of Operational Research 2009;197:509–18.
[22] Irnich S. A multi-depot pickup and delivery problem with a single hub and heterogeneous vehicles. European Journal of Operational Research 2000;122:310–28.
[23] Laporte G, Norbert Y, Taillefer S. Solving a family of multi-depot vehicle routing and location-routing problems. Transportation Science 1988;22:161–72.
[24] Lee YH, Kim JI, Kang KH, Kim KH. A heuristic for vehicle fleet mix problem using tabu search and set partitioning. Journal of the Operational Research Society 2008;59:833–41.
[25] Lim A, Wang F. Multi-depot vehicle routing problem: a one-stage approach. IEEE Transactions on Automation Science and Engineering 2004;4:397–402.
[26] Lin S. Computers solutions of the travelling salesman problem. Bell System Technical Journal 1965;44:2245–69.
[27] Min H, Current J, Schilling D. The delivery depot vehicle routing problem with backhauling. Journal of Business Logistics 1992;13:259–88.
[28] Mladenovic N, Hansen P. Variable neighborhood search. Computers & Operations Research 1997;24:1097–100.
[29] Nagy G, Salhi S. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. European Journal of Operational Research 2005;162:126–41.
[30] Pepin A-S, Desaulniers G, Hertz A, Huisman D. A comparison of five heuristics for the multiple depot vehicle scheduling problem. Journal of Scheduling 2009;12:17–30.
[31] Polack M, Hartl RF, Doerner K. A variable neigbourhood search for the multi depot vehicle routing problem with time windows. Journal of Heuristics 2004;10 627–627.
[32] Perl J, Daskin MS. A warehouse location-routing. Transportation Research 1985;19B:381–96.
[33] Perl J. The multi-depot routing allocation problem. American Journal of Mathematical & Management Sciences 1987;7:8–34.
[34] Pisinger D, Ropke S. A general heuristic for vehicle routing problem. Computers & Operations Research 2007;34:2403–35.
[35] Renaud J, Laporte G, Boctor FF. A tabu search heuristic for the multi-depot vehicle routeing problem. Computers & Operations Research 1996;23:229–35.

[36] Salhi S. Heuristic search in action: the science of tomorrow. In: Salhi S, editor, OR48 keynote papers. London: GORS; 2006. p. 39–58.
[37] Salhi S, Rand GK. Improvements to vehicle routing heuristics. Journal of the Operational Research Society 1987;38:293–5.
[38] Salhi S, Sari M, Saidi D, Touati NAC. Adaptations of some vehicle fleet mix heuristics. Omega 1992;20:653–60.
[39] Salhi S, Sari M. A multi-level composite heuristic for the multi-depot vehicle fleet mix problem. European Journal of Operational Research 1997;103:95–112.
[40] Tarantilis CD, Kiranoudis CT. Distribution of fresh meat. Journal of Food Engineering 2002;51:85–91.
[41] Thangiah SR, Salhi S. Genetic clustering: an adaptive heuristic for the multi-depot vehicle routing problem. Applied Artificial Intelligence 2001;15:361–83.
[42] Tillman FA. The multiple terminal delivery problem with probabilistic demands. Transportation Science 1969;3:192–204.
[43] Tillman FA, Cain TM. An upper bound algorithm for the single and multiple terminal delivery problem. Management Science 1972;18:664–82.
[44] Tutuncu GY. An interactive GRAMPS algorithm for the heteregenous fixed fleet vehicle routing problem with and without backhauls. European Journal of Operational Research 2010;201:593–600.
[45] Wassan NA, Osman IH. Tabu Search variants for the mix fleet vehicle routing problem. Journal of the Operational Research Society 2002;53:768–82.
[46] Wren A, Holiday A. Computer scheduling of vehicles from one or more depots to a number of delivery points. Operational Research Quarterly 1972;23:333–44.
[47] Yaman H. Formulations and valid inequalities for the heterogeneous vehicle routing problem. Mathematical Programming Series A 2006;106:365–90.
[48] Yu BY, Z.Z, Xie JX. A parallel improved ant colony optimization for multi depot vehicle routing problem. Journal of the Operational Research Society 2011;62:183–8.
[49] Zhang H, Tang J, Fung RYK. A scatter search for multi-depot vehicle routing problem with weight-related cost. Asian Pacific Journal of Operations Research 2011;28:323–48.