ELSEVIER

European Journal of Operational Research 103 (1997) 95-112

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

A multi-level composite heuristic for the multi-depot vehicle fleet mix problem [1]

S. Salhi [a,*], M. Sari [b]

[a] *Management Mathematics Group, School of Mathematics and Statistics, University of Birmingham, United Kingdom*
[b] *Department of Industrial Engineering, Algier's Polytechnic, Algeria*

Received 1 March 1996; accepted 1 August 1996

Abstract

The problem of simultaneously allocating customers to depots, finding the delivery routes and determining the vehicle fleet composition is addressed. A multi-level composite heuristic is proposed and two reduction tests are designed to enhance its efficiency. The proposed heuristic is tested on benchmark problems involving up to 360 customers, 2 to 9 depots and 5 different vehicle capacities. When tested on the special case, the multi-depot vehicle routing, our heuristic yields solutions almost as good as those found by the best known heuristics but using only 5 to 10% of their computing time. Encouraging results were also obtained for the case where the vehicles have different capacities. © 1997 Elsevier Science B.V.

Keywords: Heuristics; Heterogeneous fleet; Multiple depots; Reduction tests

1. Introduction

Distribution companies usually operate from several depots or warehouses, and utilise a large vehicle fleet which is not necessarily composed of vehicles of the same size. These vehicles obviously have different fixed costs to operate and to maintain, and require different variable costs to run. This class of multi-depot routing problems, which is more applicable but also more difficult to solve than its counterpart the multi-depot vehicle routing problem, has seldom received any attention from the OR/MS community. In this study, we attempt to address this practical distribution problem.

We now state our problem in more detail. As our input we have

- a set of customers with known location and demand,
- a fixed number of depots with known location and unlimited capacity,
- a known number of vehicle types and an unrestricted number of vehicles per type,
- vehicles with known capacities and known fixed and variable costs,
- an assumption that both the vehicle fixed cost and the mile operating cost increase with vehicle capacity.

Our aim is to simultaneously *construct a set of vehicle routes* and *determine the composition of the vehicle fleet* at a minimum total cost. This cost includes the sum of *the vehicle fixed cost* and *mileage cost*. The constraints we use here are the following:

* Corresponding author.

[1] Earlier versions of this paper were presented at *IFORS '93*, Lisbon and at *ADT'95*, Brunel.

0377-2217/97/$17.00 © 1997 Elsevier Science B.V. All rights reserved.
PII S0377-2217(96)00253-6

- both the vehicle capacity (the largest vehicle in case there are different types of vehicles) and the maximum distance travelled on any route must not be violated,
- each customer must receive his or her order in full and from one vehicle only,
- each vehicle makes one trip only and must start and end its journey at the same depot.

In the classical vehicle routing problem (VRP), all vehicles have the same capacity. Such an assumption does not always hold in practice as vehicles of different sizes are usually used. This problem, which we call the vehicle fleet mix (VFM) problem, adds another constraint to the VRP which is already a difficult *NP*-hard problem. Heuristic approaches as well as exact methods exist for the VRP, see Golden and Assad (1988) and Laporte (1992). Since the VFM is little known compared to the VRP, we provide a brief review of this topic but further information can be found in Salhi and Rand (1993).

Golden, Assad, Levy and Gheysens (1984) were the first to address the VFM problem. They developed several variants some of which are based on the savings method and others use tour partitioning. A lower bound procedure is also proposed. Gheysens, Golden and Assad (1984) used the number of vehicles found by this lower bound procedure to be the number of seed points, locate these seed points at some well chosen customers location, and then assign customers to these seed points by solving the generalised assignment problem. Desrochers and Verhoog (1991) put forward a saving based algorithm where the best combination of subtours is found using a matching based method. Salhi and Rand (1993) devised a route perturbation procedure (RPERT) which transforms a VRP solution into a VFM solution. Such a composite heuristic consists of several refinement phases, some of which allow an increase in routing cost while gaining a significantly more important reduction in vehicle fixed cost. Computational testing has shown that RPERT is superior to the previously known VFM heuristics. Some of these refinement modules will be briefly described later. Salhi, Sari, Saidi and Touati (1992) were the first to include variable mileage unit cost in some existing VFM heuristics. Osman and Salhi (1996) have recently adapted the tabu search heuristic, initially developed by Osman (1993) to solve the VRP, for the VFM problem. Some new best results were obtained at the expense of relatively larger computing time.

The multi-depot vehicle routing problem (MDVRP) seems to suffer from a shortage of published work although, in practice, it is unlikely that a distribution system operates from one single depot only. One commonly used technique for solving the MDVRP is a two phase approach; the customers are first allocated to their nearest depots and then for each depot the VRP is solved. Refinements are usually added to improve the obtained solutions. Past work on MDVRP is summarised in Table 1 and further details can be found in Chao, Golden and Wasil (1993) and also in Renaud, Laporte and Boctor (1994). Chao et al. developed a composite heuristic where a slight deterioration in the objective function is allowed in their one point move procedure. Refinements are also added. The latter paper uses diversification and intensification in the implementation of their tabu search based method. New best results are reported in the study of Renaud et al.

The MDVRP heuristic methods presented above all consider vehicles of the same capacity. We address here an important issue in distribution management where the vehicles have different capacities and different vehicle unit costs (fixed and running unit costs). To our knowledge, this is the first time this integrated distribution system, the multi depot vehicle fleet mix problem (MDVFM), is considered.

The purpose of this study is to

(i) address, for the first time, the MDVFM problem by simultaneously determining the allocation of customers to depots, the formation of the vehicle routes and the composition of the vehicle fleet,
(ii) present a multi level composite heuristic methodology, and implement it for the MDVFM problem by efficiently integrating and accordingly modifying some of the recently developed VFM heuristics,
(iii) introduce two reduction tests to enhance the efficiency of the proposed heuristic and which can also be applied to routing heuristics in general,
(iv) provide some results for the MDVFM problem which can then be used as initial benchmark solutions for future research.

The paper is organised as follows. Our methodology is outlined in Section 2 and the algorithm is described in Section 3. In Section 4, we provide the

Table 1
Brief summary of past work on MDVRP

Authors [year]	Solution methodology	Applications
Wren and Holliday (W-H) [1972]	method of saving in each depot and refinements	N/A
Cassidy and Bennett [1972]	similar to (W-H)	school meal delivery
Tillman and Cain (T-C) [1972]	method of saving with a modified distance formula	N/A
Gillett and Johnson [1976]	clustering procedure and sweep heuristic in each depot	N/A
Golden, Magnanti and Nguyen [1977]	use of borderline customers and a modified saving	newspaper delivery
Raft [1982]	multi-phase approach and additional refinements	N/A
Ball, Golden, Assad and Bodin [1983]	saving concepts and route first cluster second method	distribution of chemical product in the USA and Canada
Perl (P) [1987]	(T-C) modified distance formula and a variant of saving	N/A
Perl and Daskin [1984, 1985]	incorporate (P) in location routing formulation	distribution of manufacturing products in the USA
Benton [1986]	method of saving combined with a B-B based algorithm	delivery to retail outlets from a bakery in Indiana
Laporte, Nobert Taillefer [1988]	mixed integer linear formulation and LP relaxation	N/A
Klot, Gal and Harpaz [1992]	combination of LP and heuristics	distribution of dairy products in Haifa
Min, Current and Schilling [1992]	combination of exact methods and heuristics for backhauls	distribution of hardware products in the USA
Chao, Golden and Wasil [1993]	composite heuristics that uses infeasibility and refinements	N/A
Renaud, Laporte and Boctor [1994]	tabu search with intensification and diversification	N/A

necessary mathematical expressions. In Section 5, we discuss how we implement our heuristic emphasising the benefits of the two reduction tests which we developed. The computational results are presented in Section 6. We highlight some comments on the use of our approach in Section 7. Our conclusion and some research avenues are given in the last section.

2. The methodology used

The MDVFM problem can be formulated as a mixed integer linear program. This is similar to the one given by Perl and Daskin (1985) except that here vehicle unit costs are not necessarily constant. It can be shown that exact methods are suitable for problems of limited size only. Heuristics seem to offer the best way to solve this complex combinatorial problem. In this study, we present a general framework for constructive based heuristics, which we shall refer to as multi-level composite heuristics.

2.1. A general framework for multi-level composite heuristic search

In heuristic search, the main aim is to construct a model which can be *easily understood* and which provides *good* solutions in a *reasonable* amount of computing time. This class of methods is, in our view, an Art and a Science at the same time, since human concepts, mathematical logic and computational skills are all used in an integrated manner. In heuristic search, there are two main types of approaches, namely (i) composite heuristics and (ii) local search techniques. Both (i) and (ii) start with an initial solution, preferably feasible although not necessarily so.

In local search, the solution is allowed to deteriorate or become infeasible. This additional flexibility obviously helps to avoid, in many circumstances, the problem of being trapped in some local optima. Here, the main drawbacks are mainly the setting of the control parameters and the computational burden which may be required to obtain the solution. However, more recently several attempts were made to make such methods more efficient. Tabu Search, Simulated Annealing and Genetic Algorithms among others fall into this category, see Reeves (1993) for basic ideas on this topic.

In simple composite heuristic, the solution is improved using a given refinement procedure, and the method stops when there is no further improvement. Such a solution is then considered to be the best solution which is a local optima with respect to the selected procedure. One way to direct the search out of a local optima is by using other refinement modules for which current local optimality does no longer hold. Our aim is not to restrict ourselves to using 1, 2 or at most 3 modules as usually applied in simple composite heuristics but to introduce as much distortion to the solution as possible as long as the total computing time remains acceptable. The choice of the refinement procedures and the sequential order in which they are used can be critical. One possible skeleton of our approach, which we refer to as a multi-level composite heuristic, is given below. For clarity of presentation, we use p levels where the value of p will depend on the problem, and the procedures used within each composite heuristic.

2.2. A p-level composite heuristic

Level 1 : Generate an initial solution.

Level 2 : Apply a composite heuristic of type I.

Level 3 : Apply a composite heuristic of type II and go to level 2.

⋮

Level p : Apply a composite heuristic of type $p-1$ and go to level 2.

Level $p+1$: Record the final solution.

In level 1, the solution can be generated randomly, by good guess, using experience or simply found by a suitable greedy heuristic. In level 2, the composite heuristic of type I consists of several refinement modules which are implemented in sequence. These modules need to be fast in their implementation. This can be accomplished either because the procedures themselves are fast, or because they are applied to a small and well restricted neighbourhood. In level 3, the composite heuristic of type II consists of fewer refinements which are usually more powerful but requiring relatively more computing time. Some of these may be part of the refinements of the previous level but applied to a larger neighbourhood. The next levels can be distinguished by using refinement modules which are more and more powerful and which require a larger and larger amount of computing time.

3. The proposed heuristic

3.1. The basic idea

In this section, we adapt the proposed methodology to the MDVFM problem. The heuristic we put forward consists of 3 levels. In level 1, a starting solution for the MDVFM problem is found. This is done as follows: Initially, the VFM (VRP) is solved within each depot with certain customers left unassigned. The method of obtaining these unassigned customers, which we call *borderline customers*, will be discussed later in the paper. Then, each unassigned customer is inserted into an existing route, or an empty route, using an appropriate selection/insertion rule. In level 2, a composite heuristic, which attempts to improve on the solution found in each depot when taken separately, is introduced. In level 3, a composite heuristic which considers all depots simultaneously is used. To cut down on the computational burden, some reduction tests and computing considerations are embedded into most of our refinement procedures. Since some of the refinement modules used here are taken from RPERT and modified accordingly for multiple depots, we refer to such a method as MULTI-RPERT. The main steps of this algorithm are given below.

3.2. The algorithm MULTI-RPERT

Step 0: Set the borderline parameter $\varepsilon = \varepsilon_0$, say 0.7 and a step size Δ (say 0.1).

Step 1: Using the value of ε, determine the borderline customers and assign the other customers to their nearest depots.

Step 2: For each depot and its associated customers, solve the VFM problem using a suitable VFM heuristic.

Step 3: Insert the borderline customers into the most economical partial routes, including empty routes, using a suitable selection/insertion rule.

Step 4: Improve the VFM solution within each depot using a suitable VFM composite heuristic of type I.

Step 5: Apply a composite heuristic of type II which involves routes from different depots.

Step 6: Repeat steps 3 and 4 until a suitable stopping criterion is met. If $\varepsilon = \varepsilon_{max}$ stop else set $\varepsilon = \varepsilon + \Delta$ and go to step 1.

3.3. Clarification of the above steps

We now give some clarification of the above steps. The necessary mathematical formulations will be presented in the next section. Level 1 consists of the first four steps, level 2 and level 3 correspond to step 4 and step 5 respectively.

In step 1, the customers are split into two subsets; one for the borderline customers and the other for the remaining customers who will be, at this stage, served from their nearest depot. A customer who is situated nearly half way between his two nearest depots is considered to be a borderline customer. The explicit measure of defining a borderline customer, which is also used in Golden, Magnanti and Nguyen (1977), will be discussed later. At this stage, a borderline customer remains unassigned. This idea of borderline customers is also useful in the design of one of our reduction tests which we shall present in Section 5.

In step 2, we used the modified multiple giant tour based procedure as outlined by Salhi et al. (1992). This is briefly reviewed here.

First, a giant tour (travelling salesman tour) passing through all customers including the depot is generated. Starting from the depot and following the direction of the giant tour, all the possible feasible tours, each of which starts and ends at the depot, are defined. The feasibility checks used are both capacity and time restrictions.

Second, a directed network is constructed as follows;

- each node corresponds to the last city of a feasible route,
- each arc denotes a feasible route that serves all customers situated between the depot, the first and the last city of this feasible route, and back to the depot,
- the distance on the arc is the least total cost (fixed and running cost) of serving that feasible route. The running cost includes the cumulative mileage cost of going from the depot to the first city of this route, through all customers of this route and then back to the depot. The vehicle fixed cost associated with this route is the vehicle cost of the smallest (cheapest) vehicle that can accommodate all customers of this route.

Third, the shortest path is found using the algorithm of Dijsktra. In our example, the shortest path consists of a set of feasible routes, and the total length of the path is the total cost of the solution.

This procedure is repeated starting from different giant tours. The overall least cost solution is chosen and used in subsequent steps of the algorithm. In this study, we constructed five giant tours; one using the nearest neighbour, the other the least insertion cost procedure, and the remaining three tours are generated randomly. The detailed description on how to generate the giant tour and how to construct the directed network can be found in Salhi et al.

In step 3, a selection/insertion rule which is based on the maximum opportunity cost is used. The opportunity cost criterion, also known as regret cost, has been successfully applied in finding good initial solutions for the transportation problem, see Balakrishnan (1990). Once a customer is selected, the process is repeated until all borderline customers are assigned. Since in this selection/insertion rule, one customer is picked one at a time, the updating process is carried out using some computing considerations to avoid recalculating already known information. More details on this computing aspect will be given in Section 5.

In step 4, for each depot the current solution is used as a starting solution in our composite heuristic of type I. This is composed of some refinement modules taken from RPERT (Salhi and Rand, 1993) and two new ones which we have developed to further enhance our results. As these refinement modules are also used in steps 4 and 5, we now briefly give their main outlines.

- *matching* (for each route, assign the smallest vehicle that can accommodate the total demand of those customers served in this route. This is useful when the starting solution is a VRP solution),
- *allocate and swap* (reallocate customer from one route to another or shift them between routes. Note that here we may use larger or smaller vehicles),
- *combine and share* (combine routes to use less larger vehicles or split a route to be served by more smaller vehicles),
- *reduce* (delete the entire route by allocating its customers into other routes),
- *drop/relax* (try to make a route served by the next smaller vehicle in size by allocating some of its customers in other routes),
- *relax/comb* (it may not be cheaper to combine two routes but if it could have been if one or two customers of the combined route are allocated somewhere else and a positive overall improvement may now be possible),
- *relax/share* (similar to relax/comb except it is a splitting of a route rather than a combining. It may not be worthwhile splitting a route into 3 subroutes, say, but it could have been profitable if such a route is split into 2 subtours instead and some of its customers reallocated into other routes).

In addition to the above modules we introduced two new ones which include:

allocate in chain: This sits between the modules *allocate* and *reduce*. This is based on removing a chain of customers, say of size two, and inserting those customers of the chain in other routes. Note that those customers may not be necessarily put into the same route especially in VFM, where larger vehicles are allowed to be used.

perturb-vfm: This is an extension of the refinement module *perturb*, previously developed by Salhi and Rand (1987) for the VRP, to the VFM. This procedure considers three routes simultaneously. For instance, a customer is taken from route one and inserted in the best place of one of the other routes. Time and capacity restrictions are not imposed at this stage. The route that attracted such a customer, though it becomes now infeasible, is considered for further testing by inserting some of its customers into a third route. It is only at this stage where feasibility checks are carried out for the second and the third routes. The new solution is then accepted if the solution is feasible and the overall improvement is positive. This procedure is repeated for all customers and all routes.

In step 5, our composite heuristic of type II is similar to the one used in step 4 except that some of the refinement procedures are modified accordingly for multi-depot problems. For instance, care is taken when performing the combining process so that both end points of the combined route should originate from the same depot. Since the neighbourhood at this level is rather large, we introduced a reduction test which defines only those pairs of customers that could affect the solution and hence eliminate a large number of combinations from investigation that would have been tested otherwise. We shall describe this reduction test and other computing aspects in Section 5.

Step 6 is the termination phase. Our search terminates either when step 4 or step 5 do not yield an improvement or when the number of times steps 4 and 5 are performed is greater than a prescribed number, say 5. In our experiments, the first stopping criterion

was usually found to be the most binding.

Although this multi-level composite heuristic consists of several refinement procedures, the computing time used is found to be much shorter when compared with the time needed for the other recently developed heuristics. This important feature of this approach will be shown in our computational results section.

4. Mathematical expressions

In this section, we provide the mathematical notations, pseudo codes of our selection/insertion rule and the necessary mathematical expressions.

4.1. Notation

$N = \{1, \ldots, n\}$ the set of customers.

$M = \{1, \ldots, m\}$ the set of depots.

$D_{i,j}$ = the Euclidean distance between customer i and j $(i, j \in N)$.

d_i^k = the Euclidean distance between customer i $(i \in N)$ and depot k $(k \in M)$.

q_i = the demand of the ith customer.

K = the number of vehicle types.

CAP_s = the capacity of a vehicle of type s $(s = 1, \ldots, K)$.

VC_s = the variable unit cost per mile of a vehicle of type s.

VF_s = the fixed unit cost of a vehicle of type s.

$p_i(p_i')$ = the nearest (second nearest) depot to customer i.

E_i = the set of depots which can attract customer i.

$A(i)$ = the most economical depot to serve customer i $(A(i) \in E_i)$.

ε = a prescribed positive value for the borderline parameter $(0 < \varepsilon \leq 1)$.

B = set of borderline customers.

$\nu(\varepsilon) = |B|$ the number of borderline customers based on the value of ε.

C_i^k = the least extra cost of serving customer i from depot k.

R_k = set of routes served from depot k.

r_j^k = the jth route served from depot k $(j \in \{1, \ldots, |R_k|\})$.

$Q(r_j^k)$ = the sum of demand of customers in route r_j^k.

$L(r_j^k)$ = the length of route r_j^k.

$TC(r_j^k)$ = the total cost of serving route r_j^k.

4.2. Determination of the borderline customers

For each customer $i \in N$ do

begin

compute $\rho = d_i^{p_i} / d_i^{p_i'}$,

if $\rho \leq \varepsilon$ allocate customer i to its nearest depot (say depot p_i) and set $E_i = \{p_i\}$, otherwise customer i is considered a borderline customer and it may, at a later stage, be allocated to either the nearest or the second nearest depot, set $B = B \cup \{i\}$ and $E_i = \{p_i, p_i'\}$.

end

set $\nu(\varepsilon) = |B|$.

4.3. The opportunity cost selection/insertion rule

This rule is based on the concept of penalty opportunity cost. This represents the penalty cost of not being served from the most economical depot. The customer not yet assigned and having the highest opportunity cost (or regret cost) value is first allocated to its most economical depot. The computation of C_i^k will be discussed in the next subsection. This rule is given in the following pseudo-code format.

- For each customer not yet assigned (i.e.; $i \in B$) do

  begin

  for each depot $k \in E_i$ compute C_i^k,

  compute $C_i^{A(i)} = \text{Min}\{C_i^k; k \in E_i\}$,

  compute $OPC_i = C_i^k - C_i^{A(i)}$ where $k \in E_i$ and $k \neq A(i)$.

  end

- Determine the customer producing the maximum opportunity cost value, say customer j, where $OPC_j = \text{Max}\{OPC_i; i \in B\}$.
- Assign customer j to depot $A(j)$ and set $B = B - \{j\}$.
- Update the necessary information.
- Repeat the entire process until $|B| = 0$.

4.4. Computation of the least insertion cost C_i^k

The least cost of serving customer i ($i \in B$) from depot k ($k \in E_i$), C_i^k, is computed as follows. Note that such a customer can be either inserted into an existing route served from this kth depot, or can form a new route from afresh, originating at this kth depot. Let us examine both options:

Option 1 (insert customer i in an existing route):
For each route $r_j^k \in R_k$ ($j = 1, \ldots, |R_k|$) do

begin

compute the total cost of serving r_j^k with and without customer i.

$$TC(r_j^k) = VC_{s'}.L(r_j^k) + VF_{s'},$$
$$TC(r_j^k \cup \{i\}) = VC_s.[L(r_j^k) + [D_{i1,i} + D_{i,i2} - D_{i1,i2}]] + VF_s, \quad (1)$$

compute the extra cost of inserting customer i in r_j^k.

$$EXC_i^{r_j^k} = TC(r_j^k \cup \{i\}) - TC(r_j^k)$$
$$= [L(r_j^k).(VC_s - VC_{s'}) + (VF_s - VF_{s'})] + VC_s.[D_{i1,i} + D_{i,i2} - D_{i1,i2}] \quad (2)$$

end

where the least cost for inserting customer i in route r_j^k is between the two successive customers i_1 and i_2.

s and s' denote the type of the smallest vehicles which can accommodate $(Q(r_j^k) + q_i)$ and $Q(r_j^k)$ respectively. In other words s and s' are the smallest integers such that

$$CAP_s \geq Q(r_j^k) + q_i > CAP_{s-1} \quad \text{and}$$
$$CAP_{s'} \geq Q(r_j^k) > CAP_{s'-1}.$$

Note that if route r_j^k was already served by a vehicle of type s (i.e.; $s = s'$), the insertion cost defined by (2) will reduce to the following equation:

$$EXC_i^{r_j^k} = VC_s.(D_{i1,i} + D_{i,i2} - D_{i1,i2}). \quad (3)$$

The best route which attracts customer i is found among all routes served from depot k and yielding the least $EXC_i^{r_j^k}$ value.

$$\Delta_{k,i}^1 = \text{Min}\{EXC_i^{r_j^k} \ \forall j = 1, \ldots, |R_k|\} \quad (4)$$

Option 2 (insert customer i in an empty route):
Customer i forms its own route originating from depot k. The cost required to serve this new route is defined by

$$\Delta_{k,i}^2 = VF_s + VC_s.(2d_i^k). \quad (5)$$

Using (4) and (5) the least extra cost of assigning customer i to depot k is found by

$$C_i^k = \text{Min}\{\Delta_{k,i}^1, \Delta_{k,i}^2\}. \quad (6)$$

5. Gaining efficiency through reduction tests

In this section, we describe how we used the proposed heuristic and how we speeded up some of its steps by avoiding performing unnecessary calculations. This is achieved by using (i) algorithmic considerations that do not affect the quality of the solutions and (ii) reduction tests that cut down computing time at the expense of a negligible loss in solution quality. In (i), as many of our procedures use the insertion based rules in a repetitive manner, it is worth noting that some information is either unaffected or slightly changed from one iteration to the next. These may include for each customer not yet assigned, the insertion cost, the best insertion place in the selected route and in other routes, the cumulative demand and the new total time with such a customer if inserted in any route, the remaining demand of the route loosing such a customer as well as the time left to serve such a route, etc. For more details on this computational aspect, see Osman and Salhi (1996). Recording this necessary information help in reducing the computing time. In this section we concentrate on (ii).

We have devised two reduction tests which have enhanced the performance of MULTI-RPERT greatly,

especially in larger problems. The first one is built for single depot routing whereas the second one is designed for multi-depot routing problems. These reduction tests generate a large cut in computing time at the expense of negligible loss in solution quality. The main idea is to construct a logical $(n+m).(n+m)$ matrix, say $POS(j_1, j_2)$ where j_1 and $j_2 \in N \cup M$. We say that customer j_1 could be inserted next to customer j_2 only if $POS(j_1, j_2)$ is `true`. Initially we set the entire matrix to `false`. In this section we first describe the two reduction tests and then produce an approximation of the rate of decrease in the time complexity of our refinement modules for illustration.

5.1. The Within Depot Reduction Test (WDRT)

For each depot $i \in M$ do

begin

determine the set of customers whose nearest depot is i, say S_i where $S_i = \{j \in N$ such that $d_j^i < d_j^k \forall k \in M\}$,

for all $j_1, j_2 \in S_i$ set $POS(j_1, j_2)$=`false` and $POS(j_1, i) = POS(i, j_1)$=`true`,

compute the average distance between customers and the depot, say $DIST(i)$ where $DIST(i) = \sum_{j \in S_i} d_j^i / |S_i|$,

for each pair of customers, say j_1 and $j_2 \in S_i$ do

begin

compute the angle $\alpha(j_1, i, j_2)$,

if $\alpha(j_1, i, j_2) \leq \theta_{\min}$ set

$POS(j_1, j_2) = POS(j_2, j_1)$ = `true`, (C_1)

else

if $\alpha(j_1, i, j_2) \leq \theta_{\max}$ and if either (C_2)

$d_{j1}^i \leq DIST(i)$ and

$d_{j2}^i \leq DIST(i)$, (C_3)

or $d_{j1}^i \leq d_{j2}^i/2$, (C_4)

or $d_{j2}^i \leq d_{j1}^i/2$, (C_5)

then set $POS(j_1, j_2) = POS(j_1, j_2)$ = `true`.

end

end

In (C_1) the possibility of having petal shaped routes is taken into account within a certain angle limit, say $\theta_{\min}$ which we set to $\pi/3$. In other words, each customer is allowed to be inserted in another route within this angle limit. In (C_2) we relax (C_1) slightly using $\theta_{\max}$, which we set to $2\pi/3$, while we impose another kind of restrictions which are based on proximity from the depot. These 3 last conditions (C_3, C_4, C_5) have the tendency to allow circular routes near the depot to exist.

Preliminary experiments showed that the above values for $\theta_{\min}$ and $\theta_{\max}$ are large enough neither to affect the quality of the solution, nor to require a large cpu time. Other values were also tested but the selected ones appear to be the most promising. It may be useful to derive a functional expression for these two bounds but in our testing this proved unnecessary as the loss in quality was rather negligible, see the computational results section.

5.2. The Between Depot Reduction Test (BDRT)

This is based on defining a relationship between borderline customers and those customers close to them. These borderline customers are found using a fixed ratio ε which we set to 0.7. This value is small enough to allow as many combinations that could affect the solution as possible. For each borderline customer, we determine those customers who are situated within a certain radius of this borderline customer. These customers do not need to be borderline customers and can belong to any depot. The main steps of this reduction test are given below.

- Define the set of borderline customers, say B, using the value of $\varepsilon = 0.7$.
- For all $j_1 \in B$ do

  begin

  Compute the radius $RAD(j_1) = 1/2(d_{j1}^{p'_{j1}})$,

  set $POS(j_1, p'_{j_1}) = POS(j_1, p_{j_1})$ = `true`,

  for all $j_2 \in N - S_i$ with $i = p_{j1}$ do

  begin

  If $D_{j_1,j_2} \leq RAD(j_1)$ set $POS(j_1, j_2) = POS(j_1, j_2)$=`true`.

  end

  end

Note that if the reduction test (BDRT) is used without WDRT the set $N - S_i$ becomes N instead, as the customers in S_i would need to be checked as well. We used the second nearest depot instead of the nearest

in calculating the radius to have more flexibility, but other choices would also be useful.

5.3. Approximating the rate of decrease in the time complexity

In this section we give an approximation of the rate of decrease of one of our refinement modules. This can be performed similarly for the other improvement procedures. The module *allocate* was chosen as an example. For simplicity but without loss of generality, we assume that the customers are uniformly distributed.

(a) case of WDRT: Consider depot i ($i \in M$) and let n_i be the number of customers in S_i. Note that the time complexity of the module *allocate* is $O(kn_i^2)$. Here each customer is tested for its best insertion place and hence this will require at least n_i operations. The overall best insertion over all customers will be accomplished in at least n_i^2 operations. This process can be repeated, say k times, until there is no improvement.

Using WDRT, the time complexity of the module *allocate* reduces to $O(kn_i.Z_i)$ where Z_i denotes the approximate number of customers as defined by the WDRT conditions. A rough approximation of Z_i can be $(n_i/3 + n_i/6)$ where $n_i/3$ is defined by condition (C_1) and $n_i/6$ by the set of conditions $(C_2) \cap (C_3 \cup C_4 \cup C_5)$. This leads to a time complexity of $O(kn_i^2/2)$ and a rate of reduction of $1/2$.

(b) case of BDRT: Consider now the entire system (i.e., all depots are included). Our procedure *allocate* has a time complexity of $O(kn^2)$ where n is now the total number of customers. Let $\nu(\varepsilon)$ be the number of borderline customers based on the value of ε and Z the approximate number of customers within a given radius of a borderline customer as defined by BDRT conditions. Using BDRT, the time complexity of this module is reduced drastically to $O(k\nu(\varepsilon).Z)$, and the rate of reduction is of the order of $\nu(\varepsilon)/n.(Z/n)$. For instance if Z is approximated by $n/2m$ and $\nu(\varepsilon)$ by $n/3$, the time complexity becomes $O(n/3.(n/2m))$. This yields a rate of reduction which can be approximated by $1/6m$.

Note that the overall reduction time is not exactly equal to the cumulative reduced amounts generated by all the refinement modules. This is because many other tasks are performed between the modules and also because of the assumptions used in our approximation. To be more explicit, in the computational results section, we provide an estimate in the rate of reduction using basic linear regression.

6. Computational experiments

The proposed heuristic is coded in Fortran 77 and executed on a VAX 4000-500 computer at the University of Birmingham. Our heuristic is executed for 4 values of ε starting from $\varepsilon = 0.7$ with an increment of 0.10, and the least cost solution is chosen. This kind of restart is useful as it provides a new solution from which the search will take place. Two scenarios are used to test the performance of the present heuristic.

(i) We compare our results with the ones obtained by the two best MDVRP heuristics, namely the heuristic by Chao et al. (1993) and the one by Renaud et al. (1994). The 23 problem tests given in Chao et al. and the 3 problems provided by Perl and Daskin (1985) and Perl (1987) are used for this purpose. These vary in size from 50 to 360 customers and from 2 to 9 depots. Note that the vehicle fixed cost and mileage cost are set to 0 and 1 respectively for all the problems tested.

(ii) Since no benchmark solutions exist for the MDVFM, we used the input data related to customers and depots as the ones given in (i), and we considered a fleet of 5 different vehicle types. Their respective capacities (CAP), their vehicle unit running cost (VC) and their vehicle unit fixed cost (VF) are generated centred around the vehicle data used in (i). The values of the vehicle capacity ($\hat{Q}$), see Table 2, are used as center points for CAP. A value of 1 for the vehicle unit running cost and an arbitrary value of 100 for the vehicle fixed cost are used as center points for both the VC and the VF values respectively. In other words, we have

$$CAP_k = (0.4 + 0.2k)\hat{Q};$$
$$VC_k = 0.7 + 0.1k;$$
$$VF_k = 70 + 10k; \quad k = 1, 2, 3, 4, 5.$$

For convenience, these 5 types of vehicles are denoted by A, B, C, D, and E with A being the vehicle with the smallest capacity.

Table 2
Performance of MULTI-RPERT vs. the best MDVRP heuristics

Problem number	n	m	$\hat{Q}$	TM	Renaud, Laporte & Boctor	Chao, Golden & Wasil	Proposed method[b]	Proposed method[b,d]	Proposed method[c]	Number of vehicles
1	55	4	100	∞	421[p]	-	**416.4**	426.5	**416.4**	12
2	85	3	100	∞	666[p]	-	**647.2**	654.7	**646.6**	18
3	85	3	160	∞	540[p]	-	**513.7**	518.9	**507.6**	11
4	50	4	80	∞	**576.9**	**576.9**	587.8	597.3	587.8	11
5	50	4	160	∞	**473.5**	**473.5**	484.6	496.7	476.7	5
6	75	5	140	∞	**641.2**	**641.2**	645.9	662.0	644.5	11
7	100	2	100	∞	**1003.9**	1012.0	1047.9	1054.8	1042.3	15
8	100	2	200	∞	**750.3**	756.5	777.2	793.7	777.2	8
9	100	3	100	∞	**876.5**	879.1	888.6	913.6	888.6	16
10	100	4	100	∞	**892.6**	893.8	918.9	931.6	911.8	15
11	249	2	500	310	**4485.1**	4511.6	4513.3	4674.4	4513.3	25
12	249	3	500	310	**3937.8**	3950.9	4005.3	4086.8	4005.3	26
13	249	4	500	310	**3669.4**	3727.1	3824.7[+]	3884.0[+]	3836.4[+]	26
14	249	5	500	310	**3648.9**	3670.2	3714.3	3794.4	3714.3	26
15	80	2	60	∞	**1318.9**	1327.3	1326.8	1340.7	1326.8	8
16	80	2	60	200	**1318.9**	1345.9	**1318.9**	1324.3	**1318.9**	8
17	80	2	60	180	1365.7	1372.5	**1360.1**	1364.3	**1360.1**	8
18	160	4	60	∞	**2551.4**	2610.3	2586.7	2600.7	2579.3	16
19	160	4	60	200	**2572.2**	2605.3	2584.5	2606.1	2584.5	16
20	160	4	60	180	2731.4	2816.6	**2720.2**	2723.0	**2720.2**	16
21	240	6	60	∞	**3781.0**	3877.4	3853.3	3887.8	3822.1	24
22	240	6	60	200	**3827.1**	3863.9	3867.9	3876.4	3863.9	24
23	240	6	60	180	4097.1	4272.0	**4074.8**	4088.9	**4074.8**	24
24	360	9	60	∞	**5656.5**	5791.5	5788.5	5835.5	5788.0	36
25	360	9	60	200	**5718.0**	5857.4	5742.6	5810.5	5765.9	36
26	360	9	60	180	6145.6[+]	6494.6[+]	**6106.6**	6138.9	**6106.6**	36
Average solutions [a]					2697.4	2753.2	2727.9	2761.3	2726.8	
% above the best average [a]					0.0	2.07	1.13	2.36	1.09	
Worst deviation (%) [a]					0.64	6.78	4.23	5.41	4.50	

n and m denote the number of customers and depots respectively.
$\hat{Q}$ and TM represent the maximum capacity of a vehicle and the maximum travel time respectively.
[p] denotes that those solutions are obtained by Perl (1987).
bold denotes the best or better than the existing best solutions.
[a] means that the calculations are based on the bottom 23 problems.
[b,c] denotes that MULTI-RPERT is used with and without reduction tests respectively.
[d] denotes the average solution values over the 4 runs ($\varepsilon = 0.7, 0.8, 0.9$ and 1.0).
[+] shows the problem where the worst solution occurs.

The evaluation of our heuristic is assessed through the two main criteria, namely the quality of the solutions and the computation time needed to obtain those solutions. In this analysis, empirical testing is used.

6.1. Solution quality

In scenario (i), MULTI-RPERT behaves rather well as it produces solutions which lie within an average of 1.13% above the best published results, but requiring

a fraction of the computational time of the other two best MDVRP heuristics, see Table 2. In addition, the proposed heuristic yields 7 new best results out of the 26 problems tested. This came to us as a surprise, given that our heuristic is primarily designed to solve the MDVFM and not the MDVRP. The detailed results of these 7 new best solutions can be collected from the first author. Note that our heuristic would exhibit even better behaviour if compared with the standard solutions found by the other two methods. In this case, the average deviation above the best standard results would be only 0.88%.

To provide an average behaviour of the present heuristic, we recorded the average solution values over the 4 runs. According to our experiments, MULTI-RPERT still behaves reasonably well as shown in Table 2. It has also been observed that the most promising values of ε were found to be 0.8 and 0.9. In other words, if the computing cut off time is very limited, say of the order of 2 or 3 minutes, using those two ε values only will still produce solutions which are much above the average solutions as given in Table 2, but requiring half the total time as shown in Table 4. The computing time for these methods will be discussed in the next subsection.

In scenario (ii), to assess the performance of MULTI-RPERT we found the best MDVRP solution by solving the MDVRP problem 5 times, one for each vehicle capacity. Here, the total cost in the solution is computed using both the vehicle fixed cost and the vehicle running cost of the corresponding vehicle. A computational comparison between these two solutions is displayed in Table 3. Based on our input data, it can be shown that an average reduction of 1.6% in total cost could be gained if a mixed fleet is used instead of an homogeneous one. The largest reduction was around 6%, see problems no. 4 and 9. Note that, in some circumstances a solution with an homogeneous fleet may be the best option, see problems no. 5 and 20. Although in practice it may be difficult for a company who operates an homogeneous fleet to opt for a mixed vehicle fleet, the benefits of examining different capacities can be two-fold; (a) the company will be aware of a benchmark solution and (b) the company will have the option to match their existing fleet with the new solution as much as possible in future vehicle purchase. Note that in both (a) and (b) the cost and the inconvenience caused by changing a part or the entire existing fleet needs to be carefully measured for any decision to be finally made.

6.2. Computational effort

The total computing time for MULTI-RPERT is the cumulative time of the 4 runs. The average total time when computed over the bottom 23 problems is only 2.3 minutes for the proposed heuristic, whereas the heuristic of Chao et al. required over 17 minutes and the one by Renauld et al. needed 19 minutes. In addition, the largest time used by the proposed heuristic is just 14.1 minutes, whereas in Chao et al. it was over 2 hours and in Renaud et al. this necessitated around 1 hour and 10 minutes, see Table 4. As a summary, MULTI-RPERT is found to be about 7 to 8 times faster than the two top performers, besides being very robust as its standard deviation was approximately 3% when compared to Chao et al. and Renaud et al. whose figures were 29% and 17% respectively. Note that the cpu times recorded for the other two methods are for the standard solutions which are slightly inferior in quality to those reported in Table 2. However, we would also like to point out that MULTI-RPERT is executed on the VAX 4000-500 machine whereas the other two methods are run on Sun Sparc stations, and hence any final comparison should consider the relative speed of those machines.

For the MDVFM, the computational effort is generally found to be larger than in the case of the MDVRP. This is mainly because the feasibility checks based on vehicle capacity are less strict here as larger vehicles are allowed to be used and hence a larger number of combinations need to be evaluated. This increase is more noticeable when the binding constraint in the MDVRP was the vehicle capacity and not the time restriction, as here this constraint is relaxed as larger vehicles are tested. For instance for the 360 customers problem and 9 depots, it took nearly 3 times longer to solve the problem without time restriction (problem no. 24) than the one with the tightest restriction (problem no. 26), see Table 4 for details. Such a characteristic was not apparent in the MDVRP scenario. In summary, our heuristic took approximately 4.1 minutes on average and 27.1 minutes in the worst case.

The computing time seems to depend not only on the size of the problem but also on the number of depots. For instance, for the MDVRP the computing

Table 3
Solution quality of MULTI-RPERT on MDVFM problems

Problem number	MDVFM solution		Best MDVRP solution	
	cost	Fleet configuration	cost	Fleet configuration
1	1428.4	B^2CE^6	1423.2	E^8
2	2131.9	CD^4E^8	2228.5	E^{13}
3	1499.5	CE^7	1528.3	E^8
4	1526.7	CD^4E^3	1617.8[+]	D^9
5	992.8[+]	BC^3D	973.5	C^5
6	1611.1	$BCDE^5$	1685.3	E^8
7	2361.9	D^3E^8	2363.7	E^{11}
8	1498.4	CD^2E^3	1550.3	C^{15}
9	2277.5	CE^{10}	2404.6	E^{12}
10	2297.1	$BC^2D^2E^7$	2351.0	D^{13}
11	6718.6	$ABC^5D^6E^8$	6881.7	E^{19}
12	6211.4	$AC^3D^7E^9$	6368.9	D^{23}
13	6018.7	BCD^4E^{13}	6249.6	E^{19}
14	6030.8	$C^8D^7E^6$	6094.3	D^{21}
15	2108.2	B^3C^4D	2118.9	C^8
16	2126.8	BC^7	2118.9	C^8
17	2160.1	C^8	2160.1	C^8
18	4116.2	$A^2B^4C^8D^2$	4151.4	C^{16}
19	4178.9	$A^2B^2C^{10}D^2$	4172.2	C^{16}
20	4344.1	A^2BC^{14}	4320.2	C^{16}
21	6217.0	$A^2B^4C^{12}D^4E$	6181.0	C^{24}
22	6233.6	$A^5B^2C^{12}D^5$	6227.1	C^{24}
23	6493.1	A^2BC^{22}	6474.8	C^{24}
24	9184.6	$AB^5C^{13}D^6E^6$	9256.6	C^{36}
25	9332.0	$A^8B^6C^{15}D^8$	9318.0	C^{36}
26	9706.6	C^{36}	9706.6	C^{36}
Average solution	4178.6		4227.5	
Average deviation (in %)	0.14		1.58	
Worst deviation (in %)	1.98		5.97	

'cost' includes both the vehicle fixed cost and mileage cost.
A^pB^q denotes the fleet consists of p vehicles of type A and q vehicles of type B.
[+] shows the problem where the worst solution occurs.

time for the 249 customers problem with 2 depots took more than three times as long to solve than the one with 5 depots. A similar trend is also observed for the MDVFM case, see Table 4.

6.3. *Impact of the two reduction tests*

The effect of introducing reduction tests has shown to be very useful in cutting down the total computing time. A drastic cut of over 90% especially for larger benchmark test problems is obtained. On average, when solving the MDVRP, our heuristic took about 2.3 minutes and 18.6 minutes with and without reduction tests respectively, see Table 4.

It is also worth noting that the cpu time required for the proposed heuristic is nearly constant for problems having a similar number of customers and number of

Table 4
Total computing times for each of the three methods (in mins)

Problem number	n	m	$\hat{Q}$	TM	MDVRP				MDVFM
					Chao, Golden & Wasil[1]	Renaud, Laporte & Boctor[2]	Proposed method[3]	Proposed method[4]	Proposed method[3]
1	55	4	100	∞	N/A	N/A	< 0.1	0.3	0.1
2	85	3	100	∞	-	-	0.3	1.2	0.6
3	85	3	160	∞	-	-	1.1	2.7	0.6
4	50	4	80	∞	1.1	3.2	< 0.1	0.2	0.1
5	50	4	160	∞	1.2	4.8	< 0.1	0.4	< 0.1
6	75	5	140	∞	1.8	5.8	0.1	1.1	0.2
7	100	2	100	∞	2.2	11.4	1.3	1.5	0.6
8	100	2	200	∞	2.4	12.8	0.3	4.4	2.0
9	100	3	100	∞	2.1	8.4	0.3	0.9	0.5
10	100	4	100	∞	4.8	6.8	1.9	1.9	0.6
11	249	2	500	310	24.1	69.4+	14.1+	33.8	27.1+
12	249	3	500	310	20.1	41.2	7.1	24.9	13.6
13	249	4	500	310	7.2	43.0	4.5	26.1	7.2
14	249	5	500	310	16.7	36.4	4.1	26.2	7.4
15	80	2	60	∞	2.8	5.4	0.3	2.1	0.5
16	80	2	60	200	0.7	4.8	0.3	1.6	0.5
17	80	2	60	180	13.0	2.6	0.3	1.8	0.4
18	160	4	60	∞	2.3	15.5	1.1	10.3	2.3
19	160	4	60	200	6.1	11.1	1.1	10.8	2.1
20	160	4	60	180	6.5	5.8	0.7	6.9	1.2
21	240	6	60	∞	8.6	23.2	1.9	28.9	4.2
22	240	6	60	200	22.3	22.0	1.9	25.2	3.8
23	240	6	60	180	14.6	10.0	1.4	30.5	2.0
24	360	9	60	∞	78.5	48.7	3.6	63.9	13.4
25	360	9	60	200	132.4+	33.5	3.7	60.5	8.7
26	360	9	60	180	24.4	17.3	3.3	64.5+	4.9
	Average solutions[a]				17.2	19.3	**2.3**	18.6	4.1
	Worst solution[a]				132.4	69.4	**14.1**	64.5	27.1

[1] run on a Sun 4/370.
[2] run on a Sun Sparc station 10.
[3,4] run on a VAX 4000-500 with and without reduction tests respectively.
bold denotes the method requiring the least computing time.
+ shows the problem that uses the largest amount of computing time.
Other notations are as defined in Table 2.

depots, see Table 4. This important property does not seem to exist with the other two heuristics or with ours if used without reduction tests. With regard to solution quality, introducing these reduction tests did not seem to have a serious effect, as the average loss in quality is only around 0.2%, see Table 2.

We have also carried out experiments to assess the impact each of the two reduction tests has on reducing the total computing time. It has been found that the solutions without reduction test WDRT only, did not affect the solution at all. Our heuristic without WDRT is around 2 to 3 times slower than with it depending on the number of depots and the tightness of the constraints. The reduction test BDRT, on the contrary, is the one which contributed to the small deterioration in solution quality but at the expense of higher compu-

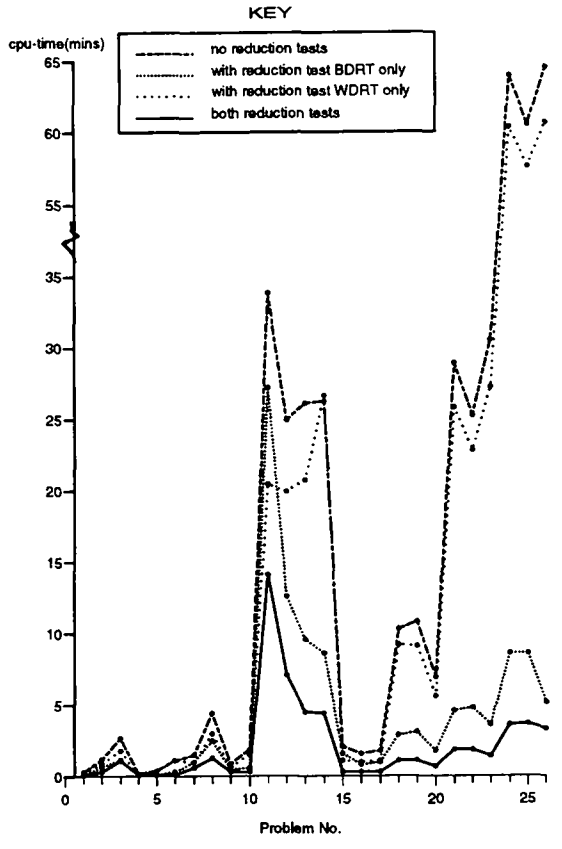

Fig. 1. A graph showing the effect of reduction tests on computing times.

tational effort. This required an amount of computing time which is nearly as large as the time without both reduction tests, except in problems with 2 depots. Fig. 1 illustrates the effect on computational time of these reduction tests. For clarity of presentation, we did not show in Fig. 1 the cpu time for the other two heuristics as their pattern is similar to ours when used without reduction tests.

Using the detailed cpu times found with and without the reductions tests, we carried out a simple linear regression to find these rates of decrease. Consider T_0, T_1, T_2, T_3 to be the time needed to find a solution without both reduction tests, with reduction test WDRT only, with reduction test BDRT only, and with both reduction tests when taken together respectively.

Let $T_k = \alpha_k T_0$ for $k = 1, 2, 3$ be the model which we would like to fit with α_k being the correction factor. The estimated values of α_k with their respective R^2 values are found as follows:

$\alpha_1 = 0.20\ (R^2 = 0.30)$,

$\alpha_2 = 0.91\ (R^2 = 0.98)$ and

$\alpha_3 = 0.10\ (R^2 = 0.27)$.

Since the values of the coefficient of correlation R^2 showed the lack of fit, we split the whole set of problems into two classes, one for those problems with 3 or less depots and the other for the remaining problems. Using this partitioning, more consistent results are obtained, though these are still not perfect.

Case of $m \leq 3$:

$\alpha_1 = 0.70\ (R^2 = 0.94)$,

$\alpha_2 = 0.67\ (R^2 = 0.97)$ and

$\alpha_3 = 0.37\ (R^2 = 0.96)$.

Case of $m \geq 4$:

$\alpha_1 = 0.15\ (R^2 = 0.58)$,

$\alpha_2 = 0.93\ (R^2 = 0.99)$ and

$\alpha_3 = 0.07\ (R^2 = 0.60)$.

In the light of these results, the reduction test WDRT is found to be useful as its impact in cutting down the total computing is extremely important, especially for problems with more than 3 depots. When the number of depots is 2 or 3, both the WDRT and BDRT are nearly equally important in reducing the computing time. As a summary it has been found that when using these two reduction tests, only 37% of the total time is needed for problems with 3 or less depots and only 7% of the total time for the other problems.

7. Some comments

In this section we highlight two comments; one is related to the heuristic itself and the other is more linked to the implementation of the model in practice.

7.1. General observations on MULTI-RPERT

In this subsection we point out two aspects related to our heuristic. In (i), the quality of the solutions as well as the computational time required to obtain

those solutions before and after refinements are briefly examined. In (ii), we explain some features regarding the implementation of this heuristic when more than one value of ε is used.

(i) It can be shown that the solution obtained without any refinements (e.g.; up to step 3) does take negligible computing time but at the expense of poorer solution quality. On average, an improvement of approximately 6.5% and 4.7% over the initial solution for MDVRP and MDVFM is found respectively when applying the refinements. Most of the improvement is obtained within the first two iterations. Both steps 3 and 4 produce nearly equal, large improvements at the first iterations.

(ii) When executing MULTI-RPERT, we observed the following properties of $\nu(\varepsilon)$.
   (a) If $\varepsilon_1 < \varepsilon_2$ and $\nu(\varepsilon_1) = \nu(\varepsilon_2)$ then there is no need to run the heuristic for ε_2.
   (b) If $\nu(\varepsilon_1) = 0$ and $\varepsilon_1 < 1$ then there is no need to run the heuristic for other values of $\varepsilon \geq \varepsilon_1$.
   (c) If $\varepsilon_1 < \varepsilon_2$ and customer i was not a borderline customer when using ε_1, then customer i can not become a borderline customer when using ε_2.

In (a) we avoid executing the entire heuristic again for ε_2 as the same solution will be found. In (b) we run the heuristic for the value of ε_1 and then stop. Note that (b) is a special case of (a). In (c) the determination of the borderline customers is made much quicker although this gain is not significant with relation to the total computing time.

7.2. *A possible implementation of MULTI-RPERT*

For our model to be easily applicable we would like to discuss two important points which may be of help at the implementation stage.

(i) In this study there are two sets of decision variables; one considers vehicle routes and the other the composition of the vehicle fleet. The former is done at an operational level (which may change from one day to the next) and the latter is more of a medium term decision (this may change only after a certain amount of time, say 1 or 2 years). The question is how we can rely on a variable input such as routing when choosing the right vehicle fleet. One commonly used way is to find the composition of the fleet first and then use that input, which in our view can be very restrictive, to solve the routing problem. In this study, we attempt to integrate the routing factor into the decision of determining the vehicle fleet. This new input is less constraining and more informative than that obtained previously. One way to proceed is to apply this approach on a set of simulated problems to obtain a set of solutions, then based on this new information, construct a more appropriate composition of the fleet. Such a vehicle fleet when used in the simulated problems should generate a routing cost which is as minimal as possible. This concept of consistency and robustness is discussed by Rosenhead (1989) and successfully applied by Salhi and Nagy (1997) when analysing the location routing problem.

(ii) In our testing, the values used for the vehicle fixed cost are based on vehicle capacity only. This is done for illustration but note that in practice those values will be generated based on several factors namely drivers salaries, insurance, regular maintenance, road tax, etc.

Note that the above comments (i) and (ii) can arise in both types of problems namely MDVRP and MDVFM.

8. Conclusion and possible research directions

In this study, we have addressed an important distribution problem, the multi depot vehicle fleet mix, which seems to have not been investigated in the past. We have put forward a structure for multi-level composite heuristics. Two reduction tests are devised; one for the single depot routing and the other for multi-depot routing problems. Their effect is found to be remarkably important as only a tiny fraction of the original computing time is needed and this is achieved with a negligible loss in quality.

The proposed approach is tested on benchmark problems varying in size from 50 to 360 customers, 2 to 9 depots, and 5 different vehicle capacities. When tested on the special case, the MDVRP, the proposed heuristic yields solutions which are on average just over 1% above the best solution but requiring only 5

to 10% of the computing time of those best heuristic methods. In addition, we found 7 new best solutions out of the 26 test problems given in the literature. For the case of the MDVFM, we adapted some MDVRP solutions for comparison as no benchmark solutions exist. Encouraging results are also obtained in this scenario.

Our multi-level composite heuristic constitutes a good compromise between simple composite heuristics and better but more time consuming tabu search based approaches. In addition, since our heuristic is fast, it can be incorporated within a local search heuristic such as tabu search, simulated annealing and/or genetic algorithms.

The reduction tests, which we developed in this study, can also be used without any difficulties in many of the existing routing heuristics. The focus in the design of good reduction tests is, in our view, an important step forward in improving the efficiency of many routing heuristics in particular and heuristic search based methods in general.

As for future research, it may be useful to

(i) investigate the issue where there is restriction on the number of vehicles per type and also on the capacity of the depots,

(ii) conduct a simulation study to analyse the consistency of the integrated models which combine short with medium term decisions. This is an implementation issue which can, in our view, be of great help to both the researchers and the end users.

Acknowledgements

We are thankful to Steve Welch for preparing Fig. 1 for us. We are also grateful to the referees for their constructive comments.

References

[1] Balakrishnan, N. (1990), "Modified Vogel's approximation method for the unbalanced transportation problem", *Applied Mathematics Letters* 3, 9–11.

[2] Ball, M., Golden, B.L., Assad, A., and Bodin, L. (1983), "Planning for truck fleet size in the presence of a common-carrier option", *Decision Science* 14, 103–120.

[3] Benton, W.C. (1986), "Evaluating a modified heuristic for the multiple vehicle scheduling problem", Working paper RS86-14, College of Administration Science, The Ohio State University, Columbus, OH.

[4] Cassidy, P., and Bennett, B. (1972), "TRAMP-A multi depot vehicle dispatching problem", *Operational Research Quarterly* 23, 151–163.

[5] Chao, I.M., Golden, B.L., and Wasil, E. (1993), "A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions", *American Journal of Mathematical & Management Sciences* 13, 371–401.

[6] Desrochers, M., and Verhoog, T.W. (1991), "A new heuristic for the fleet size and mix vehicle routing problem", *Computers & Operations Research* 18, 263–274.

[7] Gillett, B.E., and Johnson, J.W. (1976), "Multi-terminal vehicle-dispatching algorithm", *Omega* 4, 711–718.

[8] Gheysens, F.J., Golden, B.L., and Assad, A. (1984), "A comparison of techniques for solving the fleet size and mix vehicle routing", *Operations Research Spektrum* 6, 207–216.

[9] Golden, B.L., and Wasil, E. (1987), "Computerized vehicle routing in the soft drink industry", *Operations Research* 35, 6–17.

[10] Golden, B.L. and Assad, A., eds. (1988), *Vehicle Routing: Methods and Studies*, Elsevier Sciences Publishers.

[11] Golden, B.L., Magnanti, T.L., and Nguyen, H.Q. (1977), "Implementing vehicle routing algorithms", *Networks* 7, 113–148.

[12] Golden, B.L., Assad, A., Levy, L., and Gheysens, F.G. (1984), "The fleet size and mix vehicle routing", *Computers & Operations Research* 11, 49–66.

[13] Klots, B., Gal, S., and Harpaz, A. (1992), "Multi-depot and multi-product delivery optimization problem with time and service constraints", IBM Israel Report 88-315, Science and Technology, Haifa, Israel.

[14] Laporte, G., Nobert, Y., and Taillefer, S. (1988), "Solving a family of multi-depot vehicle routing and location-routing problems", *Transportation Science* 22, 161–172.

[15] Laporte, G. (1992), "The vehicle routing problem: an overview of exact and approximate algorithms", *European Journal of Operational Research* 59, 345–358.

[16] Min, H., Current, J., and Schilling, D. (1992), "The delivery depot vehicle routing problem with backhauling", *Journal of Business Logistics* 13, 259–288.

[17] Osman, I.H. (1993), "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem", *Annals of Operations Research* 41, 421–451.

[18] Osman, I.H., and Salhi, S. (1996), "Local search strategies for the vehicle fleet mix problem", in: Rayward-Smith, V.J., Osman, I.H., Reeves, C.R., and Smith, G.D., eds., *Modern Heuristic Search Techniques* Wiley, 131–154.

[19] Perl, J. (1987), "The multi-depot routing allocation problem", *American Journal of Mathematical & Management Sciences* 7, 8–34.

[20] Perl, J., and Daskin, M.S. (1985), "A warehouse location-routing problem", *Transportation Research* 19B, 381–396.

[21] Raft, O.M. (1982), "A modular algorithm for an extended scheduling problem", *European Journal of Operational Research* 11, 67–76.

[22] Reeves, C.R. (1993), *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publications.

[23] Renaud, J., Laporte, G., and Boctor, F.F. (1994), "A Tabu Search Heuristic for the Multi-Depot Vehicle Routing Problem", Working Paper CRT 94-44, Centre de Recherche sur les Transports, University of Montreal, Canada.

[24] Rosenhead, J. (1989), *Rational Analysis for a Problematic World: Problem structuring methods for complexity, uncertainty and conflict*, Wiley.

[25] Salhi, S., and Nagy, G. (1997), "Consistency and robustness in location-routing", *Studies in Locational Analysis* 11 (in press).

[26] Salhi, S., and Rand, G.K. (1993), "Incorporating vehicle routing into the vehicle fleet composition problem", *European Journal of Operational Research* 66, 313–330.

[27] Salhi, S., and Rand, G.K. (1987), "Improvements to vehicle routing heuristics", *Journal of the Operational Research Society* 38, 293–295.

[28] Salhi, S., Sari, M., Saidi, D., and Touati, N. (1992), "Adaptations of some vehicle fleet mix heuristics", *Omega* 20, 653–660.

[29] Tillman, F.A., and Cain, T.M. (1972), "An upper bound algorithm for the single and multiple terminal delivery problem", *Management Science* 18, 664–682.

[30] Wren, A., and Holliday, A. (1972), "Computer scheduling of vehicles from one or more depots to a number of delivery points", *Operational Research Quarterly* 23, 333–344.