Flex Serv Manuf J (2014) 26:540–564
DOI 10.1007/s10696-012-9166-z

A genetic local search algorithm for the multi-depot heterogeneous fleet capacitated arc routing problem

Tiantang Liu · Zhibin Jiang · Na Geng

Published online: 18 October 2012

Abstract This paper studies the multi-depot heterogeneous fleet capacitated arc routing problem (MDHCARP), a problem with rare research in the past, but with many applications in real life. The MDHCARP extends the capacitated arc routing problem (CARP) by considering both the multi-depot case and limited heterogeneous fleet constraints. We propose a genetic local search (GLS) algorithm for the MDHCARP. The GLS is appraised on simplified MDHCARP cases and on general MDHCARP instances from CARP files; and computational results show that the GLS outperforms an extended memetic algorithm and meanwhile they both improve best-known solutions of the simplified MDHCARP benchmark cases.

Keywords Multi-depot heterogeneous fleet capacitated arc routing problem · Genetic algorithm · Local search · Memetic algorithm

1 Introduction

The capacitated arc routing problem (CARP) introduced by Golden and Wong (1981) is an arc routing counterpart to the well-known vehicle routing problem. The CARP is defined on an undirected connected graph with edge costs and non-negative edge demands, where edges with positive-demands are called *required edges* or *tasks*. A fleet of homogeneous vehicles are based at a single depot. Any edge can be traversed any number of times by vehicles with each required edge being serviced only once. The objective is to find a set of vehicle routes of the least

T. Liu · Z. Jiang (✉) · N. Geng
Department of Industrial Engineering and Logistics Management, School of Mechanical Engineering, Shanghai Jiao Tong University, 800 Dong Chuan Road, Shanghai 200240, People's Republic of China
e-mail: zbjiang@sjtu.edu.cn

total cost of traversed edges, such that each route starts and ends at the depot while the total demand of each route does not exceed vehicle capacity.

The CARP is NP-hard in that it can be reduced to the NP-hard rural postman problem (RPP) (Lenstra and Rinnooy Kan 1976). During the 1980s and early 1990s, heuristics used to be the mainstream approach to solve the CARP and many heuristics were introduced, such as augment-merge (Golden and Wong 1981), path-scanning (Golden et al. 1983) and route-first, cluster-second (Ulusoy 1985), etc. In the recent decade, some meta-heuristics have been proposed, such as tabu search (Hertz et al. 2000; Brandão and Eglese 2008), variable neighborhood descent (Hertz and Mittaz 2001), guided local search (Beullens et al. 2003), memetic algorithms (Lacomme et al. 2001, 2004; Tang et al. 2009) and ant colony optimization (Santos et al. 2010), etc. A recent review can be found in Liu et al. (2008) and Corberán and Prins (2010).

Applications of the CARP include collection or delivery, street sweeping and snow removal, etc. However, the CARP has its limitation in modeling most real-life applications such as salt spreader routing problem and waste collection. Eglese (1994) and Li and Eglese (1996) study the salt spreader routing problem and propose that there can be multiple depot locations, different capacities of gritters (vehicles) and other constraints in this real-life problem. In municipal solid waste collection, large companies or public sectors usually have more than one depot and they have a fleet of vehicles characterized by different capacities and operating costs to collect or transport waste. Therefore, in this paper we consider two major extensions of the basic CARP, namely heterogeneous vehicles and multiple depots. The resulting new problem can be called the multi-depot heterogeneous fleet CARP (MDHCARP).

The heterogeneous fleet CARP (HCARP) has two versions, one is the heterogeneous fixed fleet CARP with a limited number of vehicles, and the other is the fleet size and mix CARP (FSMCARP) with unlimited ones. The FSMCARP is first introduced by Ulusoy (1985) who proposes the route-first, cluster-second method. Therein, a RPP is solved to form a giant tour, and then the giant tour is *partitioned* into feasible vehicle routes subject to the constraints by solving the shortest path problem on a transformed new graph. This idea is embedded in our genetic local search (GLS) algorithm. The multi-depot CARP (MDCARP) is studied firstly by Amberg et al. (2000). In their research, this problem is defined on an undirected graph with M depots. A fixed number of vehicles with heterogeneous capacity, but without fixed costs and variable costs, are stationed in each depot. Recently, Kansou and Yassine (2010) and Xing et al. (2010) have studied the MDCARP by using different meta-heuristics.

In this paper, we systematically study the NP-hard MDHCARP for the first time, and propose an effective genetic local search (GLS) algorithm for the problem. It has been shown that metaheuristics combining genetic algorithm (GA) with local search (LS) are powerful for the CARP and its variants (Lacomme et al. 2004; Mei et al. 2011; Tang et al. 2009), because it has the potential to exploit the global search advantage of GA and local search advantage of problem-specific LS. The remainder of this paper is organized as follows: Sect. 2 gives a formal and detailed description of the MDHCARP. The general framework and key components of the proposed

GLS are introduced in Sect. 3. Computational experiments on real-life applications and new generated data are presented in Sect. 4, and finally, Sect. 5 summarizes some concluding remarks.

2 Problem definition and notations

The MDHCARP can be described as follows. Given a *mixed* graph $G = (V, E \cup A)$ with a set V of n nodes, an undirected edge set E, a directed arc set A. $E \cup A$ is the set of links (edges and arcs), $c_{ij} = c_{ji}(\geq 0)$ is the cost (length) of a link $(i,j) \in E \cup A$, and $E_R \subseteq E$ and $A_R \subseteq A$ are the sets of *trequired* edges and arcs, respectively. Let $D \subseteq V$ be the set of *Mdepots*. A *heterogeneous* (different types) fleet of vehicles indexed in a set K is stationed at multiple depots. The number of vehicles of each type k in depot m is *fixed* (*limited*) and equals n_{mk} and the total number of vehicles of each type k is n_k. A fixed cost λ_{mk}, a variable cost per unit distance μ_{mk} and capacity Q_{mk} is associated to each vehicle of type k in depot m. The MDHCARP is to determine a set of routes in such a way that: (1) the number of vehicles used of each type k is not more than n_k, and even each type k from depot m is not more than n_{mk}; (2) each vehicle route starts and ends at the same depot; (3) each required link (task) is served exactly once; (4) the total demand of each route served by vehicle k from depot m does not exceed Q_{mk}, and its total duration does not exceed a preset value L_{mk}; (5) the total routing cost is minimized.

To describe the tasks clearly, each required arc is identified by a task number instead of one pair of nodes. Each arc $u \in A$ has a tail (start node) $a(u)$, a head (end node) $b(u)$ and a traversing (deadheading) cost $tc(u)$. Each required link (task) $u \in (E_R \cup A_R)$ has a demand $d(u)$, a serving cost $sc(u)$; and if u is an edge task, it has an inverse mark $inv(u)$. Task $inv(u)$ and u have the same traversing cost, demand and serving cost. Note that each edge task $u \in E_R$ can be served in either direction, i.e., only one of task u and $inv(u)$ is served. Table 1 summarizes the notations adopted by the paper.

3 The genetic local search algorithm

The genetic local search (GLS) algorithm is a combination of a population-based global search genetic algorithm (GA) and an individual-based local search. Our GLS has several main characters: (a) in chromosome encoding for the CARP with complex multi-depot and heterogeneous constraints, the GLS uses a simple permutation (sequence) of t tasks, without cluster and route delimiters (excluding depots); (b) a *multi-depot heterogeneous partition* (*MDH-Partition*) procedure is proposed to convert the indirect chromosomes encoding into solutions; (c) in population replacement, one chromosome P_r is selected in the parent population using one new replacement method described in Sect. 3.5.

The proposed GLS includes two phases, i.e., a main phase and a restart phase. Initially, each chromosome represents a potential MDHCARP solution through an encoding mechanism. Next, an initial population is generated, and the chromosomes

Table 1 List of notations about the MDHCARP

Notation	Meaning
$G = (V, E \cup A)$	Mixed graph
E_R	Set of required edges in G
A_R	Set of required arcs in G
V	Set of nodes in G
$a(u)$	Tail (start node) of arc u
$b(u)$	Head (end node) of arc u
$d(u)$	Demand of arc u
$tc(u)$	Traversing (deadheading) cost of arc u
$sc(u)$	Serving cost of arc u
$inv(u)$	Pointer to opposite arc of arc u
M	Number of depots
m	Depot node
K	Set of fleet of vehicles
n_k	Total number of vehicle of type k across depots
n_{mk}	Number of vehicle of type k in depot m
Q_{mk}	Capacity of vehicle of type k in depot m
L_{mk}	Maximum duration of vehicle of type k in depot m
λ_{mk}	Fixed cost of vehicle of type k in depot m
μ_{mk}	Variable cost of vehicle of type k in depot m per unit distance
t	Number of tasks

of the population are evaluated by the *MDH-Partition* procedure of Sect. 3.1.1 and sorted according to a fitness function.

Then, a typical iteration of the main phase proceeds as follows. Selection, crossover and local search operators are implemented to generate an offspring. The offspring is added into the parent population by applying a replacement operator. The new population is then kept sorted again and these operators are repeated until the stopping criterion of the main phase is reached. Finally, the restart phase keeps some best chromosomes of the main phase and generates other new chromosomes to form another initial population, and then the typical iteration is repeated until the termination condition of the restart phase is met. Table 2 shows the basic steps of the GLS. Several main components are described in detail in Sect. 3.1–3.5.

3.1 Chromosome structure and evaluation

As the MDHCARP belongs to the class of *multi-depot* routing problems, it is a natural idea to use depots as cluster delimiters, and even as route delimiters for each cluster in encoding corresponding to GAs proposed in the published multi-depot routing problem literature (Ho et al. 2008; Lau et al. 2010; Ombuki-Berman and Hanshar 2009; Kansou and Yassine 2010).

Table 2 General framework of the GLS

Step 1 The initial population is constructed using two heuristics and random generation (Sect. 3.2).
Step 2 Each iteration selects two parents P1 and P2 randomly, then two children C1 and C2 are obtained using one of order crossover (OX) and linear order crossover (LOX), and only one of C1 and C2 is randomly selected as child C (Sect. 3.3).
Step 3 The child C is converted into a MDHCARP solution by implementing the *MDH-Partition* procedure (Sect. 3.1.1), and then the solution undergoes the local search (LS) in a given probability p_m (Sect. 3.4).
Step 4 Two chromosomes are selected from the latter 2/3 subpopulation of the parent population, and the worse one P_r is replaced by the child C if C is not clone as the other chromosomes in the parent population (Sect. 3.5).
Step 5 The main phase stops after a maximum number of iterations (ni). After that, the restart procedure is implemented for nr times, where two best chromosomes are kept, and others are replaced by new randomly generated chromosomes, and then the main phase is repeated but with a higher local search probability p_r. That is to say, the total number of iteration is $(1 + nr) \times ni$ (Sect. 3.5).

Nevertheless, inspired by Lacomme et al. (2004) and Prins (2009), our chromosome T is a simple permutation (sequence) of t tasks, without cluster and route delimiters (excluding depots). Implicit shortest paths (deadheading paths) are between consecutive tasks. It can be viewed as a RPP or a giant tour traversed by one vehicle without capacity constraints from one depot. This kind of chromosome representation is adopted because: (1) the encoding is simple, and classical GA operators for the travelling salesman problem can be reused; (2) a chromosome can be converted into a feasible MDHCARP solution by the *MDH-Partition* procedure described in Sect. 3.1.1; (3) the indirect encoding is so flexible that it is suitable for extended problems including new constraints and objectives.

3.1.1 MDH-partition

Under the above chromosome structure, an MDHCARP solution can be obtained by decoding the chromosome using the *MDH-Partition* procedure. The fitness of each chromosome is evaluated based on the total cost of this corresponding solution.

Given a chromosome $T = (c_1, c_2, \ldots, c_t)$ where t corresponds to the number of tasks, the *MDH-Partition* procedure works on an auxiliary directed acyclic graph $G_a = (X, Y, Z)$, where X is a set of $t + 1$ vertex index from a dummy node 0 to t. Y is a set of arcs where one arc $(i,j) \in Y$ means that a trip serving tasks subsequence $(c_{i+1}, c_{i+2}, \ldots, c_j)$ is feasible in terms of capacity and duration, i.e., $load(i + 1, j) \leq Q_{mk}$ and $length(i + 1, j, m) \leq L_{mk}$ where $load(i + 1, j)$ is the load of the trip, $length(i + 1, j, m)$ is the length of this trip served by the vehicle from depot m. Z is the set of the weight of arcs where one weight z_{ij}^{mk} corresponds to the total cost of vehicle type k from depot m to serve tasks subsequence $(c_{i+1}, c_{i+2}, \ldots, c_j)$.

In each depot m, vehicles have k_m different types, with a number n_{mk} of vehicles for each type k. Vehicles (even the same size) from different depots will generate different costs, leading to z_{ij}^{mk} instead of z_{ij} in the CARP. The optimal partitioning of chromosome T corresponds to a shortest path from node 0 to node t in G_a, with no more than n_{mk} arcs for vehicle type k from depot m. Thus, this problem is a resource-constrained shortest path problem (RCSPP), which is an NP-hard problem but can be solved in pseudo-polynomial time based on dynamic programming.

In our dynamic programming approach, a set of labels is associated with each node i of the auxiliary graph G_a, and each label $L(v_{11},\ldots,v_{1k_1},\ldots,v_{M1},\ldots,v_{Mk_M},z)$ represents different paths from the origin 0 to node i with specific consumption of vehicle resources and cost. To be specific, $v:(v_{11},\ldots,v_{1k_1},\ldots,v_{M1},\ldots,v_{Mk_M})$ is the consumption combination of all types of vehicles, where v_{mk} indicates the number of vehicle type k used from depot m ($1 \le m \le M$, $1 \le k \le k_m$, $1 \le v_{mk} \le n_{mk}$); and z the corresponding cost of the path associated with the label. The number of possible vehicle consumption combinations is $\omega = \prod_{m=1,k=1}^{M,k_M}(n_{mk}+1)$, where null consumptions are considered.

To use a simpler notation, let $k_1 + k_2 + \ldots + k_M$ equal ξ which is the total number of vehicle types across all depots. Note that when we define label $L(v_{11},\ldots,v_{1k_1},\ldots,v_{M1},\ldots,v_{Mk_M},z)$ and compute ξ, even if vehicle types from different depots are the same, they are still regarded as different because vehicles of the same type and from different depots generate different costs when used; and the number of them may also be different. Then, each kind of possible vehicle consumption combination $v:(v_{11},\ldots,v_{1k_1},\ldots,v_{M1},\ldots,v_{Mk_M})$ with two-dimensional suffixes corresponds to a vector $\hbar:(\hbar_1,\hbar_2,\ldots,\hbar_\xi)$ with one-dimensional suffix and can be further converted into a scalar $q \in [0, \omega - 1]$. Thus, we can represent L_i^q as the qth label in node i for the path ending at node i, i.e., tasks subsequence $(c_1,c_2,\ldots,c_i)$, and then, $L_i^q.v$ as the partial fleet used, $L_i^q.v_{mk}$ as vehicle consumptions of type k from depot m, and $L_i^q.z$ as the corresponding cost. Given the label L_i^q and the arc $(i,j) \in Y$ (a trip c_{i+1} to c_j), then

$$load(i+1,j) = \sum_{l=i+1}^{j} d(c_l) \tag{1}$$

$$length(i+1,j,m) = tc(m,c_i) + \sum_{x=i+1}^{j}[sc(c_x) + tc(c_{x-1},c_x)] + tc(c_j,m) \tag{2}$$

In Eq. (1), $d(c_l)$ is the demand of task c_l, and in Eq. (2), the length includes the traversing cost tc from depot m to task c_i, from c_i to c_{i+1} until c_j and from c_j to the depot m, and the serving cost sc of any task from c_{i+1} to c_j. If the vehicle of type k from depot m is available and $load(i+1,j) \le Q_{mk}$ and $length(i+1,j,m) \le L_{mk}$, one label L_j^r for node j is generated by using recursive equations:

$$L_j^r.v_{mk} = L_i^q.v_{mk} + 1, 0 \le i < j \le t,\ 1 \le m \le M \tag{3}$$

$$L_j^r.z = L_i^q.z + z_{ij}^{mk}, 0 \le i < j \le t,\ 1 \le m \le M \tag{4}$$

In Eq. (4), $z_{ij}^{mk} = \lambda_{mk} + \mu_{mk} \cdot length(i+1,j,m)$. The time complexity of the *MDH-Partition* procedure is $O(t^2\omega\xi)$. Since a large number of labels increase the computing time, a dominance rule can be adopted.

In addition, largest capacity check is also useful, which means that if the vehicle with the largest capacity cannot satisfy the condition $load(i+1, j) \leq Q_{mk}$, all of other size of vehicles cannot have enough capacity to serve tasks subsequence $(c_{i+1}, c_{i+2}, \ldots, c_j)$. Note that since the number of vehicles of each type is fixed, when the total capacity of vehicles is slightly larger than the total demand, a chromosome may not be able to be converted into a solution by *MDH-Partition* procedure. In that case, we can add additional vehicles in the process of *MDH-Partition* without affecting the final solution.

3.1.2 An illustrative example

Consider one example of 4 edge tasks with their respective demands being 8, 14, 8, 9, and two types of vehicles and two depots. A single vehicle with a capacity of 30 (type 1) is stationed in depot 1, and a single vehicle with a capacity of 25 (type 2) is in depot 2. The detailed vehicle information is shown in Table 3. Figure 1a shows the chromosome tour $T = (c_1, c_2, c_3, c_4)$ with demands in brackets. Thin dotted lines represent shortest paths between any two nodes, and the numbers under $t = 4$ tasks are the serving costs.

The *MDH-Partition* procedure builds an auxiliary graph G_a with $t + 1$ nodes indexed from 0 to t, as shown in Fig. 1b. The initial label of node 0 is L_0^0 with $L_0^0.v = (0,0)$ and $L_0^0.z = 0$, i.e., no vehicles are used and no costs are generated. Arcs from node 0 to node 1 represent the trips that serve task c_1. To be specific, if task c_1 is served by the vehicle of type 1 from depot 1, it leads to the label L_1^1 with cost $L_1^1.z = 50 + 1.5 \times 36 = 104$, and if served by the vehicle of type 2 from depot 2, it leads to the label $L_1^2$with cost $L_1^2.z$ =40 + 1 × 26 = 66. But actually in our *MDH-Partition* procedure, neither of these two labels is generated. The reason is that if task c_1 is served by the vehicle of type 1 (type 2), the remaining capacity is only 25 (30) which is not enough to serve the remaining three tasks with a total demand of 31. Therefore, subsequent labels that arise from L_1^1 and L_1^2 are not all generated. In addition, the vehicle of type 1 cannot cover the total demands of four tasks. According to the largest capacity check technique of Sect. 3.1.1, neither can the small size vehicle of type 2, so neither of labels L_4^1 and L_4^2 is generated.

A shortest path from node 0 to node t in G_a (bold) indicates the optimal partitioning of T: two trips and a label L_4^3 obtained from label L_2^1, with vehicles consumption$L_4^3.v = (1,1)$ and total cost $L_4^3.z = 193$. The resulting MDHCARP solution is trip (0, c_1, c_2, 0) with a cost of 113 served by vehicle 1 from depot 1 and trip (0, c_3, c_4, 0) with a cost of 80 served by vehicle 2 from depot 2, as shown in Fig. 1c.

Table 3 Vehicle information

Vehicle type Vehicle position	1 Depot 1	2 Depot 2
n_{mk}	1	1
Q_{mk}	30	25
λ_{mk}	50	40
μ_{mk}	1.5	1

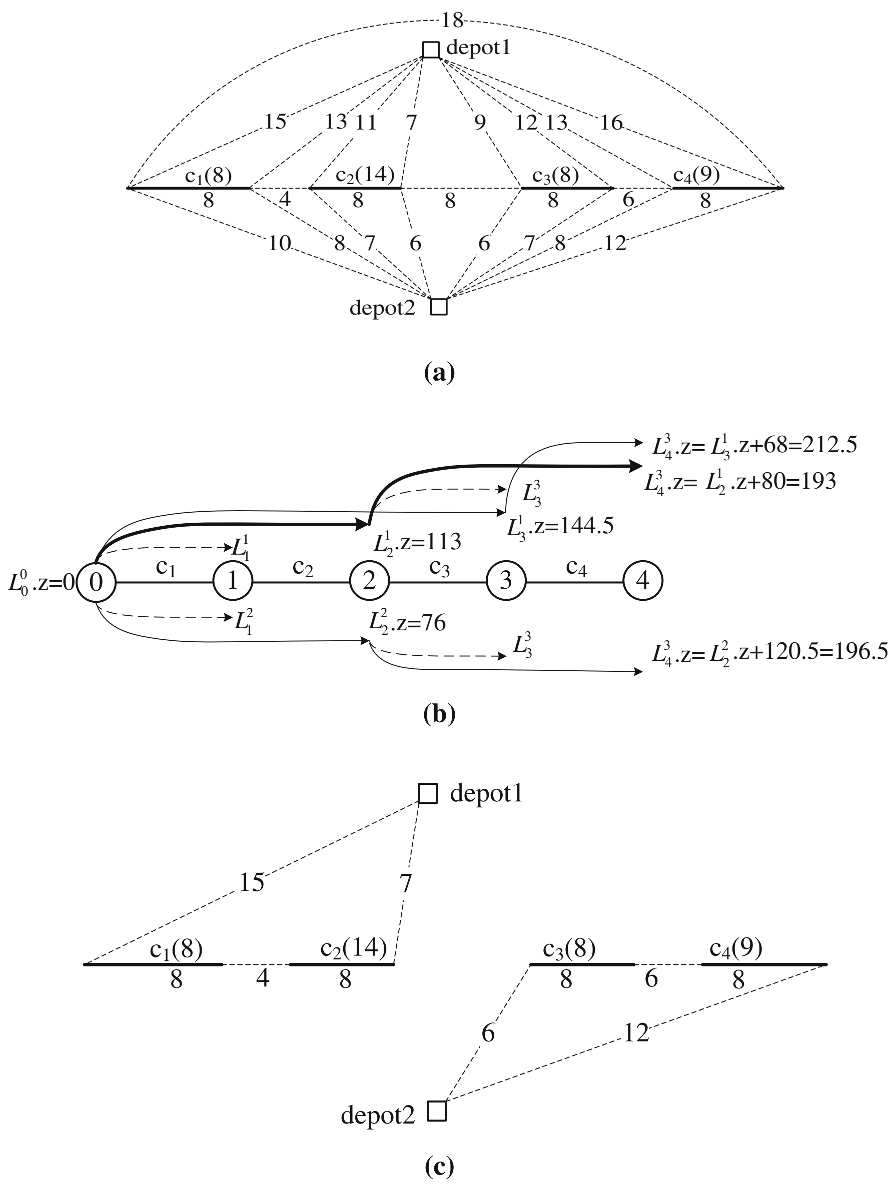

Fig. 1 Example of *MDH-Partition*.**a** Chromosome tour with 4 edge tasks. **b** Auxiliary graph G_a, labels and shortest path. **c** Resulting trips

3.2 Initial population

The population P is composed of ps chromosomes. The initial population consists of two good (low-cost) chromosomes and ps-2 random chromosomes. To be specific, two good chromosomes P_1 and P_2 are constructed by using matching-based algorithm (MBA) and modified path-scanning (MPS), both with local search. Note that when chromosomes are randomly generated, identical chromosomes (*clones*) may result in a premature convergence, and should therefore be avoided. Since exact detection for identical chromosomes is time consuming, we adopt a simpler and faster diversity condition, i.e., two chromosomes do not have the same cost. To meet this condition, we try *try_max* times to generate each random chromosome and decrease the number of chromosomes in the population if *try_max* times all fail.

All the above methods generate *ps* permutations of tasks (giant RPP tours) which are then converted into *ps* solutions by *MDH-Partition* procedure. Then each solution is concatenated into one chromosome, and all chromosomes are stored using an array in increasing cost order.

3.2.1 Matching-based algorithm

We recommend matching-based algorithm (MBA) to generate a very good chromosome, because this algorithm proposed by Frederickson (1979) for RPP has a worst-case bound of 3/2, which is similar to the travelling salesman problem heuristic of Christofides (1976).

The MBA works on the original undirected graph, and can be described as follows.

1. Construct a minimum cost spanning tree to connect several components of the original graph, and denote the new graph as G1.
2. Determine a minimum cost perfect matching (Edmonds and Johnson 1973) of the odd degree vertices of G1, denote the new graph as G2.
3. Find an Euler tour of G2 using the end-pairing algorithm (Edmonds and Johnson 1973), and the Euler tour is exactly the RPP tour.
4. Transform the representation of the RPP tour from nodes sequence into tasks sequence with an implicit shortest path between any two consecutive tasks, using pre-marked task numbers for each task.

3.2.2 Modified path-scanning

The original path-scanning algorithm is introduced by Golden et al. (1983) for the capacitated Chinese postman problem. This heuristic builds trip routes based on a greedy idea subject to vehicle capacity Q. In constructing each route, the sequence of tasks is extended by joining the task that looks most promising until Q or maximum trip length L or maximum time duration is exhausted. For a sequence ending at task f, the task closest to f is chosen as the next task. If multiple tasks satisfy this condition, five rules are employed to determine the next task g, not yet served: (1) maximize the distance from g to depot; (2) minimize the distance from g to depot; (3) maximize the ratio of demand/service cost of g, i.e., $d(g)/sc(g)$; (4) minimize the ratio of demand/service cost of g, i.e., $d(g)/sc(g)$; (5) use rule 1 if the vehicle is no more than half-full, or else use rule 2.

In our version, due to the coexistence of multi-depot and heterogeneous fleet with different capacities, a giant RPP tour is constructed by using modified path-scanning

(MPS), in which there is only one depot (the first depot) and only one vehicle where Q equals the total demand of overall tasks. Note that, MPS starts from the first depot, and chooses different next tasks to serve by implementing five different rules, until all tasks are chosen. Therefore, MPS will generate five different tours. Each of them is followed by the *MDH-Partition* procedure of Sect. 3.1.1, respectively. Then, five MDHCARP solutions are extracted, and only the best solution is kept.

3.3 Selection and crossover

The proposed GLS selects two parents P1 and P2 randomly from the sorted population. When the two parents are selected, order crossover (OX) (Davis 1985) and linear order crossover (LOX) (Falkenauer and Bouffouix 1991) are randomly selected to implement. Note that the OX and LOX generate two child chromosomes which are both kept in the original OX and LOX, but only one child randomly chosen is preserved in our OX and LOX. Fig. 2 gives one sample of OX with two parents P1 and P2 and two crossover points $i = 4$, $j = 7$, where ten tasks are undirected, i.e., $inv(1) = 11$, $inv(2) = 12$, and so on. Therefore, when child C1 is filled using P2($j + 1$) to P2($i - 1$), task 2 in P2 should be excluded because its opposite arc 12 is the same task that has been included in C1.

3.4 Local search

A local search (LS) is adopted with a fixed probability p_m in our GLS to produce a better offspring after each crossover. The LS works on a MDHCARP solution obtained by implementing the *MDH-Partition* procedure on the child C, because if it operates directly on the chromosome C without route delimiters, a large amount of time will be spent to evaluate each move of it. Let tasks i and j be served after tasks f and g in their respective routes and all move types are described below.

- M1: move task f after task g.
- M2: move two consecutive tasks (f, i) after task g.
- M3: swap task f and g.
- M4: swap task f and (g, j).
- M5: swap task (f, i) and (g, j).
- M6: 2-opt moves.

The LS scans each pair of tasks (f, g) in $o(t^2)$, and each iteration of the LS implements M1–M6 and stops when it finds the first improving move, and then the solution is updated and the next iteration is continued until all pairs of tasks are

				***i*=4**			***j*=7**			
P1:	1	4	5	7	9	10	12	3	6	8
P2:	13	2	10	8	6	15	7	9	1	4
C1:	8	6	15	7	9	10	12	1	4	13
C2:	9	10	12	8	6	15	7	3	1	4

Fig. 2 Example of OX crossover

scanned; however, the whole M1–M6 process for each pair of tasks is repeated as long as the solutions can be further improved.

There are several points to be noted in the LS. First, each type of move is implemented in the same route or in two different routes which may or may not be from the same depot. Second, in M1–M5, if a task *f* is moved to another position, it can appear in either as *f* or *inv* (*f*). Third, in M1 and M2, *g* can be the start depot of its route. Fourth, in 2-opt, if the move operates on two routes from different depots, we must reconnect the first or last task of the two routes to the two depots after 2-opt, to guarantee that a route starts and ends at the same depot (as shown in Fig. 3).

At last, some routes are removed if they become empty. The final solution of LS is converted into a chromosome by concatenating these routes and excluding route delimiters (depots). Then, the chromosome is converted into a solution by the optimal *MDH-Partition* which sometimes can bring a better solution for the same chromosome.

3.5 Replacement and termination

We propose a new replacement method, i.e., two chromosomes are selected from a subpopulation and the worse one P_r is replaced by the child *C* if *C* is not identical to any other chromosomes of the parent population. We test several kinds of subpopulation choices, such as the latter 1/2, the latter 2/3 and the whole of the parent population where chromosomes are stored in increasing cost order, and preliminary experiments show that the latter 2/3 is superior. After replacement, the *ps* chromosomes are stored in increasing cost order again.

As mentioned before, the proposed GLS includes two phases, i.e., a main phase and a restart phase. The main phase stops after a maximum number of iterations (*ni*). After that, the restart procedure is implemented for *nr* times, where two best chromosomes are kept, and others are replaced by new randomly generated chromosomes. Then, the main phase is repeated but with a higher local search probability p_r. That is to say, the total number of iterations is $(1 + nr) \times ni$.

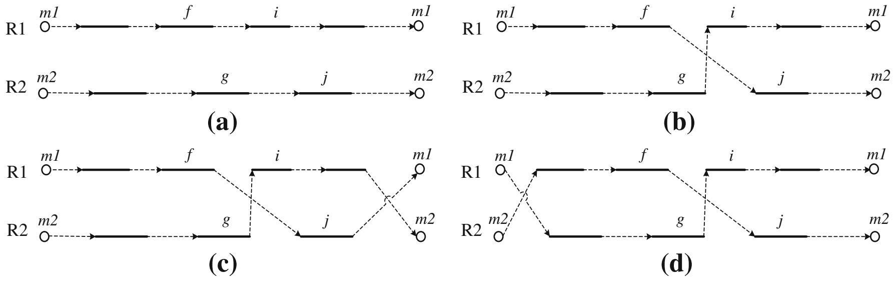

Fig. 3 Example of 2-opt move: **a** original two routes R1 and R2 from depot *m1* and *m2*, respectively; **b** the shortest paths linking *f* to *i* and *g* to *j* are replaced by the shortest paths from *f* to *j* and from *g* to *i*; **c** and **d** new routes are reconnected to depot *m1* and *m2* in two ways

4 Computational experiments

4.1 Test problems

The MDHCARP is a new problem and there are no benchmark instances. In this paper, we first test the real-life instances dealing with problems of route planning for winter gritting in the areas of Königstein and Wennigsen, Germany, which are from Amberg et al. (2000) and can be viewed as simplified MDHCARPs since no fixed costs and variable costs of vehicles are considered, and then generate new general MDHCARP data from CARP instances. To evaluate our GLS, especially for generated instances, we extend the memetic algorithm (Lacomme et al. 2004) to solve the MDHCARP for comparison. In the extended memetic algorithm (EMA), depots are used as cluster delimiters but not as route delimiters in chromosome encoding, as in Kansou and Yassine (2010). To be specific, a chromosome S consists of M sub-chromosome S_m where M corresponds to the number of depots. Each S_m is a sequence of tasks associated to depot m, without route delimiters. Then a chromosome is converted into a feasible MDHCARP solution by a *heterogeneous partition* (*H-Partition*) to each S_m of S. The *H-Partition* for the single depot HCARP *is a special case of* the MDH-Partition.

4.1.1 Simplified MDHCARP cases

The data of three real-life instances are shown in Table 4 and the corresponding sketch maps of the first two instances are shown in Fig. 4a and b. In Table 4, the first column stands for the instance name; |V|, $|E|$ and t indicate the number of nodes, the number of edges, and the number of tasks, respectively; T_D represents the total demand; M is the number of depots; m is the depot node; Q_{mk} is vehicle capacity of each type k in depot m; n_{mk} and CT_{mk} are the corresponding number of vehicle and time capacity, and fixed costλ_{mk} and variable cost μ_{mk} of each vehicle are not given. In Fig. 4, dashed lines are non-required edges which can be traversed but do not have to be served, solid lines represent required edges, and blue solid lines are parallel tasks.

The first instance is a relatively large size problem in the area of Königstein. Six vehicles are stationed in the same depot (node 8) but have different time and vehicle capacities. In this sense, it is a single-depot HCARP. The graph has 65 nodes and 93 required edges with total demand 202 (Amberg et al. (2000) state 94 required edges by mistake). The time capacity CT_{mk} is the maximum time duration of vehicle of type k from depot m and can be converted into the maximum trip length L_{mk}, since the average speed of vehicles is given as 30 km/h. Special conditions are that the vehicle with capacity $Q = 650$ must pass a load station node 65 and one edge (15, 16) is a narrow street and must be served by a small vehicle with $Q = 150$ or $Q = 250$.

The second instance concerns the area of Wennigsen. Six vehicles are stationed in two depot nodes 1 and 15. They have different vehicle capacities but no time capacities. The graph has 48 nodes and 55 required edges with a total demand of 229.4.

Table 4 Three simplified MDHCARP cases

Instance	$\|V\|$	$\|E\|$	t	T_D	M	m	Q_{mk}	n_{mk}	λ_{mk}	μ_{mk}	CT_{mk}
Königstein	65	101	93	202	1	8	65	1	–	–	3.5
							50	2	–	–	3.5
							25	2	–	–	3.0
							15	1	–	–	2.5
Wennigsen	48	56	55	229.4	2	1	45	5	–	–	–
						15	30	1	–	–	–
Wennigsen-modified	48	56	51	214.5	2	1	45	5	–	–	–
						15	30	1	–	–	–

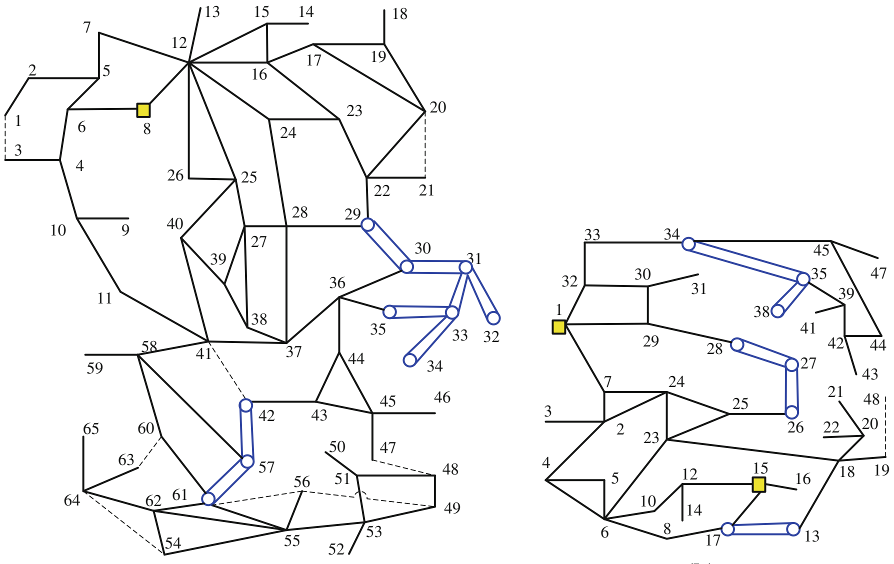
(a) Königstein **(b)** Wennigsen

Fig. 4 Graphs about Königstein problem (**a**) and Wennigsen problem (**b**)

The third instance (Wennigsen-modified) arises from the Wennigsen problem, and the only difference between them is that the third instance eliminates four of the required edges, i.e., (17, 13), (13, 18), (15, 17) and (18, 19), and thus has 51 required edges. Therefore, the total demand is decreased to 214.5.

4.1.2 New MDHCARP instances

Some general MDHCARP data are generated by adding depots and vehicle types in three sets of standard CARP instances (*gdb*, *val* and *egl* files) which can be downloaded from http://www.uv.es/~belengue/carp.html. The *gdb* set is 23 small size instances with 7–27 nodes and 11–55 edges; the *val* set includes 34 medium

Table 5 An example of general MDHCARP instances

Problem	|V|	|E|	Total capacity	M	m	Q_{mk}	n_{mk}	λ_{mk}	μ_{mk}
mdh-egl-s1-C	140	75	103 × 14	5	1	63	3	63	1.0
					29	83	3	83	1.2
					57	103	2	103	1.4
					85	123	3	123	1.6
					113	143	3	143	1.8

size instances with 24–50 nodes and 34–97 edges and the *egl* set contains 24 large size instances with 77–140 nodes and 98–190 edges. In the *gdb* and *val* sets, all the edges are required edges (tasks), and the *egl* set has 51–190 required edges and some non-required edges.

The general MDHCARP instances are generated from three benchmark sets of CARP problems: *gdb*, *val* and *egl*, so named *mdh-gdb*, *mdh-val* and *mdh-egl*, respectively, which are available upon request from the authors. For each instance, we change the original depot number from one depot into 2–5 depots, and replace the original homogeneous fleet with a heterogeneous fleet of 3–5 different vehicle types stationed at depots, according to the problem size. For the vehicles of each instance, their respective capacities Q_{mk} are in arithmetic progression, and their respective number n_{mk} is set to guarantee that the total capacity equals the original capacity; their fixed costs λ_{mk} are equivalent to their respective capacities; their unit variable cost μ_{mk} is 1.0, 1.2, …, 1.8. Table 5 gives the example of mdh-egl-s1-C generated from egl-s1-C.

4.2 Parameter settings

The GLS and EMA are implemented in C and executed on an Intel (R) Pentium (R) Dual 1.8 GHz PC under Windows XP.

Preliminary experiments were required to determine the best parameter settings. In these experiments, the following combinations of factors are tested: (1) population size *ps* tested at two levels, 30 and 50; (2) maximum value of *try_max* to generate each random non-clone chromosome tested at two levels, 10 and 20; (3) local search probability p_m in the main phase tested at four levels, 0, 0.15, 0.3 and 0.5; (4) local search probability p_r in the restart phase tested at three levels, 0.25, 0.5 and 0.75; (5) maximum iteration number of the main phase *ni* tested at two levels, 5,000 and 10,000; (6) maximum value of restarts *nr* tested at two levels, 10 and 20. Preliminary tests were done on the 34 instances of *mdh-val* generated problems.

The parameters based on the experimental results are set as follows: the population size *ps* is 30, the maximum value of *try_max* to generate each random non-clone chromosome is 10, the local search probability p_m and p_r in the main phase and restart phase are 0.5 and 0.75, respectively, the maximum iteration number of the main phase *ni* is 5,000, and the maximum number of restarts *nr* is 10.

4.3 Results on the simplified MDHCARP

For the Königstein problem, the comparison between the GLS and EMA and four strategies of Amberg et al. (2000) is shown in Table 6 (Boldface indicates the new best solution). Note that, in Amberg et al. (2000), each solution corresponds to the total length of additional edges (deadheading length); therefore, the total travelling length equals the solution plus the total length of required edges. Take the Königstein problem for instance, the best solution is 79.5, and the total travelling length should be 79.5 plus 202. Amberg et al. (2000) gives four strategies using Simulated Annealing and Tabu Search, i.e., SA t, STS s, CSM $c_1c_2c_3$ and REM r where SA, STS, CSM and REM are names of strategies, and t, s, $c_1c_2c_3$ and r are their respective parameters. In Table 6, we only show the best result of each strategy instead of giving all of them. Note that in Amberg et al. (2000), the CPU time is the average time of the last improvement instead of the total computing time.

In Table 6, the third and fourth columns are the best and average solutions of the EMA and GLS over 10 runs, respectively, the fifth column is the average computing time in seconds. From Table 6, we can find that even the average solutions of the GLS and EMA outperform all meta-strategies of Amberg et al. (2000), and the GLS is the best algorithm which has a better average solution and gets the new best solution in less time than EMA. The new best solution is 273.5 with an improvement of 2.84 % and the computing time is acceptable. The corresponding routes are given in detail in Königstein of Appendix.

For the Wennigsen problem, the comparison between the GLS and EMA and four strategies of Amberg et al. (2000) is shown in Table 7 (Boldface indicates the new best solution). From Table 7, we can find that the new best solution with cost 325.8 is obtained easily with an improvement of 1.81 % and likewise the GLS converges faster than the EMA. The corresponding routes are given in Wennigsen of Appendix.

For the Wennigsen-modified problem, the computing time and the number of iterations are not given in Amberg et al. (2000) whose best solution is 329.2 (104.7 plus 214.5). The GLS and EMA both can improve the result by 3.34 % after 0.70 and 0.92 s, respectively, and the corresponding routes of the new best solutions with cost 318.2, are shown in Wennigsen-modified of Appendix.

GLS and EMA outperform the metaheuristics of Amberg et al. (2000) in that GLS and EMA combine genetic algorithm (GA) with local search (LS), and they

Table 6 Comparison between the GLS and EMA and four published metastrateties for Königstein problem

Meta-strategy	Total length of additional edges	Best solution	Average solution	Sec.
SA 10	79.5	281.5	–	–
STS 10	79.5	281.5	–	–
CSM 1400 10 2	79.5	281.5	–	–
REM 1500	79.5	281.5	–	–
EMA	–	**273.5**	275.5	155.43
GLS	–	**273.5**	274	88.38

Boldface indicates the new best solution

Table 7 Comparison between the GLS and EMA and four published metastrateties for Wennigsen problem

	Total length of additional edges	Best solution	Average solution	Sec.
SA 120	102.4	331.8	–	–
STS 10	102.4	331.8	–	–
CSM 4700 10 2	102.4	331.8	–	–
REM 2700	102.4	331.8	–	–
EMA	–	**325.8**	**325.8**	6.44
GLS	–	**325.8**	**325.8**	3.22

Boldface indicates the new best solution

have the potential to exploit the global search advantage of GA and local search advantage of problem-specific LS.

4.4 Results on the general MDHCARP

The results of the general MDHCARP instances (*mdh-gdb*, *mdh-val* and *mdh-egl*) are presented in Tables 8, 9, 10. For each instance, two good initial solutions generated by heuristics MBA and MPS, and improved by LS, respectively, are given in columns 2–5, and the average solutions, the best solutions, and the average computing times in seconds over 10 runs for EMA and GLS are shown in columns 6–8 and columns 9–11, respectively. In Tables 8, 9, 10, if the best solution of the GLS ($\mathrm{GLS}_{\mathrm{bs}}$) is superior to that of the EMA ($\mathrm{EMA}_{\mathrm{bs}}$) then it is shown in bold.

The average solution quality of the algorithms is summarized in Table 11. For each file, the first row shows the average results and the second row reports the average deviations (in %) above the best-known solutions (BKS) which are our best solutions obtained from the $\mathrm{GLS}_{\mathrm{bs}}$. Main conclusions can be drawn from the results as follows.

1. The solutions of the GLS and the EMA are much better than two good initial solutions of the simple heuristics. For example, the average deviations of the initial solutions obtained from MBA + LS from the best solution $\mathrm{GLS}_{\mathrm{bs}}$ are 1.26, 5.30 and 9.12 % on *mdh-gab*, *mdh-val* and *mdh-egl* files, respectively.
2. The local search is effective. To be specific, MBA + LS improves MBA by 11.3, 4.5 and 7.9 % on three files, respectively. MPS + LS improves MPS by 3.6, 4.6 and 8.8 % on three files, respectively.
3. The GLS outperforms the EMA both in the results and in the computing time. To be specific, the average solutions of the GLS ($\mathrm{GLS}_{\mathrm{as}}$) over all instances are better than those of the EMA ($\mathrm{EMA}_{\mathrm{as}}$); for small size instances of the *mdh-gdb* set, the $\mathrm{EMA}_{\mathrm{bs}}$ obtains the same best solutions as the $\mathrm{GLS}_{\mathrm{bs}}$, but requires additional computing time; while for medium and large size instances, the average deviations of the $\mathrm{EMA}_{\mathrm{bs}}$ from the $\mathrm{GLS}_{\mathrm{bs}}$ are 0.17 and 0.42 %, respectively. It can be found that the GLS obtains new best solutions on 20 out

Table 8 Results of the mdh-gdb set

Problem	MBA	MPS	MBA + LS	MPS + LS	EMA_{as}	EMA_{bs}	Sec	GLS_{as}	GLS_{bs}	Sec.
mdh-gdb1	418.2	391.8	393.6	380.4	365.6	365.6	2.08	365.6	365.6	0.91
mdh-gdb2	449.8	426.8	409	407.6	397.2	397.2	1.75	397.2	397.2	0.89
mdh-gdb3	385.8	384.8	367.8	384.6	326.8	326.8	0.29	326.8	326.8	0.2
mdh-gdb4	386.8	393	361	357.6	353.8	353.8	1.6	353.8	353.8	0.52
mdh-gdb5	505.8	494.4	464.4	461.6	446.8	446.8	15.52	446.8	446.8	10.39
mdh-gdb6	394	384.4	376.0	384.4	353	353	1.75	353	353	1.08
mdh-gdb7	449.4	431.4	394.8	426.4	380.6	380.6	2.38	380.6	380.6	1.56
mdh-gdb8	733.8	721.8	674	695	639	636.6	30.51	638.2	636.6	19.4
mdh-gdb9	712.4	680.2	680.4	643.8	594.6	592.6	44.73	593.6	592.6	22.8
mdh-gdb10	424.6	422.4	372.2	386.8	358.6	357.6	2.78	358	357.6	1.79
mdh-gdb11	785.8	790.2	762.4	767.0	687.6	685.6	29.28	687	685.6	21.16
mdh-gdb12	892.6	879	836.8	826.4	754.4	754.4	1.12	754.4	754.4	0.58
mdh-gdb13	965	958.4	925	955.6	846.6	835.8	10.68	842	835.8	9.4
mdh-gdb14	244.6	243.8	233.4	222.4	217.4	217.4	3.44	217.4	217.4	2.96
mdh-gdb15	198.2	195.8	194.8	195.8	190.8	190.8	3.26	190.8	190.8	2.62
mdh-gdb16	306.4	289	282.8	273.8	263.2	262.8	13.9	262.8	262.8	10.86
mdh-gdb17	302.4	298.8	289.8	291.6	284.6	284.2	13.94	284.4	284.2	10.6
mdh-gdb18	378.2	370.4	371.8	360.4	346.2	344.8	28	346	344.8	15.54
mdh-gdb19	154.8	159.6	153.2	158.8	147.2	147.2	0.01	147.2	147.2	0
mdh-gdb20	318.6	290.8	252.8	283.4	249.8	249.8	7.48	249.8	249.8	5.64
mdh-gdb21	381.4	377	360	368.8	340.2	338.6	21.2	339.8	338.6	17.88
mdh-gdb22	472.4	471	460	455.2	444.6	442.6	51.24	443.6	442.6	31.07
mdh-gdb23	578.6	573.8	555.6	560.4	534.8	531.6	85.34	533.4	531.6	66.36
Average	471.3	462.1	418.1	445.6	414.1	412.9	16.19	413.6	412.9	11.05

of 34 *mdh-val* instances, and 17 out of 24 *mdh-egl* instances, which is better than the EMA. Compared with EMA_{as} and EMA_{bs}, GLS_{as} and GLS_{bs} save on average 0.2 and 0.4 % on two files. GLS outperforms EMA in that in GLS, the chromosome *T* is a permutation (sequence) of *t* tasks, without cluster and route delimiters (excluding depots) and an optimal possible MDHCARP solution can be extracted from a chromosome by the *MDH-Partition* procedure, while in

Table 9 Results of the mdh-val set

Problem	MBA	MPS	MBA + LS	MPS + LS	EMA$_{as}$	EMA$_{bs}$	Sec	GLS$_{as}$	GLS$_{bs}$	Sec.
mdh-val1A	626.6	646.8	621.2	615.8	606.8	606.0	22.64	606.0	606.0	9.88
mdh-val1B	637.8	648.4	629.8	631.2	572.4	567	39.12	570	567	25.86
mdh-val1C	710	702	688.6	688.8	621.6	614.4	20.34	617.2	614.4	13.12
mdh-val2A	652.8	635.2	598.6	621.2	596.2	595.6	13.09	595.6	595.6	8.22
mdh-val2B	693.6	686	636.4	634.2	630.6	630.6	2.69	630.6	630.6	1.42
mdh-val2C	832	828	803.8	758.6	710.6	710.6	2.95	710.6	710.6	2.44
mdh-val3A	262.8	259.6	256.2	247	242.6	242.6	12.71	242.6	242.6	7.75
mdh-val3B	261.2	264.6	256.2	252.8	243.4	243.4	8.44	243.4	243.4	6.05
mdh-val3C	314	305.4	281.6	293.2	261.6	261.6	4.92	261.6	261.6	3.92
mdh-val4A	1,242.8	1,216	1,175.4	1,178.8	1,141.6	1,134.6	76.98	1,138.4	1,134.6	37.99
mdh-val4B	1,240.4	1,232	1,192.6	1,175.8	1,147.8	1,144.6	73.85	1,144.6	**1,139.4**	46.55
mdh-val4C	1,264.6	1,238	1,157.8	1,175.8	1,137.6	1,130.6	75.22	1,129.4	**1,126.2**	53.56
mdh-val4D	1,338.8	1,332.4	1,276.6	1,248	1,176	1,173.4	71.94	1,174	**1,172.6**	65.07
mdh-val5A	1,243.8	1,250.2	1,182.6	1,189.8	1,153.8	1,150.4	79.16	1,151.2	**1,149.2**	56.4
mdh-val5B	1,267.6	1,253	1,211.4	1,182.8	1,156.2	1,153.2	73.44	1,152.8	**1,147.6**	53.8
mdh-val5C	1,287.6	1,284.4	1,233.6	1,196.2	1,154.2	1,148.8	68.96	1,152.6	**1,148**	41.61
mdh-val5D	1,382	1,388.4	1,286.2	1,265	1,203.2	1,199.8	69.41	1,200.2	**1,196.2**	43.3
mdh-val6A	828.6	820.6	790.2	792.6	770.6	769	31.41	769	769	17.42
mdh-val6B	808	796.8	768.2	780	745.4	743.8	44.32	743.8	743.8	21.86
mdh-val6C	907.8	883.6	869.8	853.8	808.2	808	37.6	808	808	18.72
mdh-val7A	1,006.6	1,024	986.6	957	934.4	933	66.4	933.4	**930.8**	42.92
mdh-val7B	1,008.2	1,049	967.4	999.2	936.4	934	68.34	933.8	**932.4**	48.46
mdh-val7C	1,123.6	1,130.6	1,096.8	1,044.6	981.4	974.4	78.7	977.6	974.4	55.7
mdh-val8A	1,161.6	1,143.6	1,106.2	1,113.6	1,061.2	1,057.4	72.82	1,058.4	**1,056.4**	53.11
mdh-val8B	1,187.8	1,144.2	1,099.4	1,112.8	1,063.4	1,062.4	67.49	1,061	**1,058.6**	48.86

Table 9 continued

Problem	MBA	MPS	MBA + LS	MPS + LS	EMA$_{as}$	EMA$_{bs}$	Sec	GLS$_{as}$	GLS$_{bs}$	Sec.
mdh-val8C	1,307.2	1,257.4	1,199.6	1,225.8	1,105.8	1,098.4	68.85	1,102.8	**1,094**	49.1
mdh-val9A	1,146.8	1,156.8	1,118.6	1,105.6	1,092.6	1,086.8	125.35	1,087.8	**1,085.8**	89.95
mdh-val9B	1,146.8	1,161	1,106.8	1,129.6	1,088.6	1,086	137.26	1,084.8	**1,082.4**	92.59
mdh-val9C	1,157	1,179.2	1,136.8	1,137	1,099.8	1,095.2	137.89	1,096.8	**1,094.6**	91.66
mdh-val9D	1,258.8	1,282	1,195.8	1,190.8	1,140.2	1,138.6	147.58	1,138.2	**1,133.6**	98.69
mdh-val10A	1,311.6	1,308.2	1,282.8	1,263.8	1,248.4	1,245.8	140.79	1,247	**1,244.6**	98.2
mdh-val10B	1,342.8	1,350.6	1,296.4	1,313	1,276	1,268.6	141.71	1,272.4	**1,268.4**	103.11
mdh-val10C	1,333	1,339.8	1,295.4	1,298.8	1,260	1,253.8	141.46	1,257.6	**1,252.2**	101.85
mdh-val10D	1,431.2	1,429.6	1,367	1,356.6	1,293	1,297.8	139.29	1,294	**1,289.8**	101.14
Average	1,021.3	1,018.5	975.7	971.5	931.2	928.2	69.5	929	926.6	47.36

Bold values indicate the best solution of the GLS (GLS$_{bs}$) is superior to that of the EMA (EMA$_{bs}$)

Table 10 Results of the mdh-egl set

Problem	MBA	MPS	MBA + LS	MPS + LS	EMA_{as}	EMA_{bs}	Sec	GLS_{as}	GLS_{bs}	Sec.
mdh-egl-e1-A	6,437	6,231	5,539	5,788.4	5,188	5,185.6	30.79	5,188	5,185.6	23.46
mdh-egl-e1-B	6,714.8	6,463	5,997	5,811.8	5,595.8	5,593	41.56	5,593	5,593	29.58
mdh-egl-e1-C	7,522.4	6,857	6,490.2	6,475.2	5,960.8	5,943.6	51.29	5,957.6	5,943.6	30.42
mdh-egl-e2-A	8,243	7,906.4	7,289.6	7,175	6,829.8	6,803.8	159.27	6,679.8	**6,679.8**	86.02
mdh-egl-e2-B	8,517.2	8,664.4	7,844.6	8,053.2	7,215	7,170.8	171.24	7,140.4	**7,102**	97.1
mdh-egl-e2-C	9,646.8	9,670.2	9,157.6	9,279.4	8,673	8,641.8	185.89	8,635.8	**8,629.8**	108.34
mdh-egl-e3-A	9,149.2	9,466.4	8,641	8,562	7,625.4	7,536.4	174.85	7,601.8	**7,532**	122.58
mdh-egl-e3-B	10,059.6	10,648.6	8,917	9,597	8,577.2	8,526.2	168.36	8,573.8	8,526.2	110.5
mdh-egl-e3-C	13,433	13,307.4	11,888.4	11,690.8	9,758.4	9,741	181.1	9,741	**9,715.2**	126.8
mdh-egl-e4-A	10,286.4	10,923.8	9,939.2	8,989.6	8,422.8	8,368	187.56	8,419.2	8,368	129.79
mdh-egl-e4-B	11,349.4	11,393.2	10,900	10,912.8	9,714.8	9,664	242.35	9,703.2	9,664	198.79
mdh-egl-e4-C	13,233.6	13,902	11,970.6	11,537.8	10,961.4	10,871.6	308.62	10,891.2	**10,862.8**	293.8
mdh-egl-s1-A	7,685.2	6,993.2	6,295	6,306.2	5,973	5,962.8	95.92	5,962.8	**5,952**	67.68
mdh-egl-s1-B	8,043.2	7,821.4	7,047.2	7,067.4	6,723.2	6,716.8	79.1	6,719.6	**6,707**	64.25
mdh-egl-s1-C	9,052.4	9,245	8,414	8,894.6	7,990.2	7,968.2	232.46	7,977.2	7,968.2	177.11
mdh-egl-s2-A	13,279.4	13,760.8	12,562.6	12,406.2	11,368.2	11,443.2	560.76	11,402.8	**11,361**	407.56
mdh-egl-s2-B	15,552	15,995	14,320.4	14,520.8	13,434.6	13,363.8	714.32	13,368.8	**13,285.8**	597.69
mdh-egl-s2-C	20,103.4	20,714.6	18,999.8	19,080.4	17,961.6	17,847.4	932.38	17,814	**17,773.8**	787.95
mdh-egl-s3-A	14,411.8	14,859.6	13,753.4	13,790.6	13,204.4	13,141.4	791.28	13,145.4	**12,968.6**	671.44
mdh-egl-s3-B	16,355	16,792.6	15,439.2	15,509.8	14,434	14,357.8	876.05	14,400.8	**14,318.6**	710.25
mdh-egl-s3-C	19,604.6	19,252.8	17,215.6	17,824.6	16,557	16,538	969.24	16,485.6	**16,476.4**	728.89
mdh-egl-s4-A	17,228	17,562.2	15,954.6	16,134.4	14,772.8	14,699.2	1,188.58	14,704.6	**14,632**	1,009.36
mdh-egl-s4-B	19,709.6	20,084	18,621	18,359.6	16,852.6	16,763.8	1,391.24	16,828.2	**16,740.2**	1,183.28
mdh-egl-s4-C	22,627.6	23,756.4	21,610.2	21,887.4	20,108	20,051.6	1,616.19	20,036.2	**19,846.8**	1,409.06
Average	12,426.9	12,594.6	11,450.3	11,485.6	10,579.3	10,537.5	472.93	10,540.5	10,493	382.15

Bold values indicate the best solution of the GLS (GLS_{bs}) is superior to that of the EMA (EMA_{bs})

Table 11 Average solution quality of the algorithms

Files		MBA	MPS	MBA + LS	MPS + LS	EMA_{as}	EMA_{bs}	GLS_{as}	GLS_{bs}	BKS
mdh-gdb	Average	471.3	462.1	418.1	445.6	414.1	412.9	413.6	412.9	412.9
	Avg. dev. (%) from BKS	14.14	11.92	1.26	7.92	0.29	0.00	0.17	0.00	
mdh-val	Average	1,021.3	1,018.5	975.7	971.5	931.2	928.2	929	926.6	926.6
	Avg. dev. (%) from BKS	10.22	9.92	5.30	4.86	0.50	0.17	0.26	0.00	
mdh-egl	Average	12,426.9	12,594.6	11,450.3	11,485.6	10,579.3	10,537.5	10,540.5	10,493	10,493
	Avg. dev. (%) from BKS	18.43	20.03	9.12	9.46	0.82	0.42	0.45	0.00	

EMA, a chromosome is a sequence S of M (the number of depots) sub-chromosome S_m where each S_m is a sequence of tasks associated to depot m. A feasible MDHCARP solution by a *heterogeneous partition* (*H-Partition*) to each S_m of S is a local optimal solution, and therefore worse than an optimal MDHCARP solution extracted from a chromosome T by the *MDH-Partition* procedure.

4. The computing times of the GLS and the EMA are both reasonable, even for large size problems.

5 Conclusions

This paper introduces the MDHCARP, a problem that despite its many real-life applications, has not received much attention in previous research on the CARP. The MDHCARP is to determine the least cost routes for a given heterogeneous fleet of vehicles to serve a set of *required* edges and arcs (tasks). The vehicles located at multi-depot are with different capacities, fixed costs and variable costs.

Due to its theoretical complexity, we propose the GLS and EMA for the MDHCARP, which extend the classic memetic algorithm (hybrid genetic algorithm) to both multi-depot and heterogeneous fleet constraints. The simplified MDH-CARPs from real-world cases are tested and new better solutions are found by the GLS and EMA in reasonable computation times. A large number of general MDHCARP instances generated from CARP instances are also tested and the results show that the GLS outperforms the EMA.

Note that our GLS is also effective for the multi-depot fleet size and mix CARP, where the number of available vehicles of each type is unlimited. The main modification is that the chromosome evaluation corresponds to the shortest path problem instead of the resource-constrained shortest path problem.

Our work is a foundation for future research on complicated constraints on the CARP. We plan to extend in several ways. First, an efficient lower bound can be provided to evaluate our GLS. Second, more complex and practical constraints can be involved, such as time windows and periodic demands for tasks. Due to the flexibility of the GLS, we intend to extend it to tackle these additional attributes. Finally, we also plan to propose some new operators to improve the performance of the GLS, as well as other more efficient hybrid metaheuristics combining population search and local search.

Acknowledgments This work was supported by the National Natural Science Foundation of China under research Grant No. 70872077 and the National Natural Science Foundation of China—Research Grants Council of Hong Kong (NSFC-RGC) joint research projects (No. 70831160527). The authors thank Professor Stefan Voß in Institute of Information Systems, Department for Business and Economics, University of Hamburg, Germany, who kindly provided the test data of Amberg et al. (2000). In addition, we thank the editor and the anonymous referees for their pertinent and constructive comments.

Appendix: New best solutions for three simplified MDHCARP instances

Königstein

See Tables 12, 13.

Table 12 Total cost = 273.5, 5 trip routes

Route	Vehicle capacity per route	Total demand per route	Total cost per route	Number of tasks per route
1	50	46	55	28
2	25	24	32.5	7
3	50	48.5	60.5	23
4	65	64.5	98	24
5	25	19	27.5	11

Table 13 Sequence of tasks per route

Route	Depot node	Sequence of tasks per routes (Each task is represented by one pair of nodes, and implicit shortest paths are between consecutive tasks)
1	8	12–24 24–23 23–22 22–20 21–22 22–29 29–30 31–30 30–31 31–33 35–33 33–34 34–33 33–31 31–32 32–31 33–35 35–36 36–30 30–29 29–28 28–27 27–39 39–38 38–27 27–25 25–26 26–12
2	8	12–7 7–5 5–2 2–1 3–4 5–6 6–8
3	8	6–4 4–10 9–10 10–11 11–41 41–58 58–59 58–60 60–61 61–57 57–42 42–43 43–45 46–45 45–47 45–44 43–44 44–36 36–37 37–38 39–40 40–25 25–12
4	8	16–23 24–28 28–37 37–41 42–57 57–61 61–62 62–55 54–55 55–56 55–53 53–49 49–48 48–51 51–50 51–53 53–52 53–54 54–62 62–64 64–65 64–63 58–57 41–40
5	8	12–13 12–16 17–20 20–19 19–18 19–17 17–16 16–15 14–15 15–12 12–8

Wennigsen

See Tables 14, 15.

Table 14 Total cost = 325.8, 6 trip routes

Route	Vehicle capacity per route	Total demand per route	Total cost per route	Number of tasks per route
1	45	35.4	48.2	7
2	45	44.4	74.9	11
3	45	37.6	39.5	10
4	30	26.1	34.3	8
5	45	41.7	70.7	11
6	45	44.2	58.2	8

Table 15 Sequence of tasks per route

Route	Depot node	Sequence of tasks per routes (Each task is represented by one pair of nodes, and implicit shortest paths are between consecutive tasks)
1	1	29–28 28–27 27–26 25–26 26–27 27–28 29–1
2	1	24–2 4–6 6–8 17–13 13–18 18–19 18–20 20–22 20–21 18–23 23–24
3	1	1–7 7–24 24–25 25–23 23–6 6–5 5–4 4–2 2–3 2–7
4	15	15–12 12–14 12–10 10–8 8–17 13–17 17–15 15–16
5	1	1–32 32–33 33–34 34–45 45–47 45–44 44–42 43–42 42–39 41–39 39–35
6	1	29–30 30–31 34–35 35–38 38–35 35–34 33–30 30–32

Wennigsen-modified

See Tables 16, 17.

Table 16 Total cost = 318.2, 6 trip routes

Route	Vehicle capacity per route	Total demand per route	Total cost per route	Number of tasks per route
1	45	33.9	35.8	9
2	45	27	33.1	6
3	30	23.4	31.1	6
4	45	44.3	89.3	11
5	45	44.1	50.6	8
6	45	41.8	78.3	11

Table 17 Sequence of tasks per route

Route	Depot node	Sequence of tasks per routes (Each task is represented by one pair of nodes, and implicit shortest paths are between consecutive tasks)
1	1	1–7 7–2 3–2 2–4 4–5 5–6 6–23 23–24 24–7
2	1	1–29 29–28 28–27 27–26 26–25 25–24
3	15	15–16 15–12 12–14 12–10 10–8 8–17
4	1	28–27 27–26 25–23 23–18 21–20 22–20 20–18 13–17 8–6 6–4 2–24
5	1	32–30 30–33 34–35 35–38 38–35 35–34 34–33 32–1
6	1	29–30 31–30 34–45 45–47 45–44 44–42 42–43 42–39 39–41 39–35 33–32

References

Amberg A, Domschke W, Voß S (2000) Multiple center capacitated arc routing problems: a tabu search algorithm using capacitated trees. Eur J Oper Res 124(2):360–376

Beullens P, Muyldermans L, Cattrysse D, Van Oudheusden D (2003) A guided local search heuristic for the capacitated arc routing problem. Eur J Oper Res 147(3):629–643

Brandão J, Eglese R (2008) A deterministic tabu search algorithm for the capacitated arc routing problem. Comput Oper Res 35(4):1112–1126

Christofides N (1976) Worst case analysis of a new heuristic for the traveling salesman problem. Report 388, Graduate School of Industrial Administration. Carnegie-Mellon University, Pittsburgh

Corberán A, Prins C (2010) Recent results on arc routing problems: an annotated bibliography. Networks 56(1):50–69

Davis L (1985) Applying adaptive algorithms to epistatic domains. In: Proceedings of the international joint conference on artificial intelligence, pp 162–164

Edmonds J, Johnson E (1973) Matching, Euler tours and the Chinese postman. Math Program 5(1):88–124

Eglese R (1994) Routeing winter gritting vehicles. Discrete Appl Math 48(3):231–244

Falkenauer E, Bouffouix SA (1991) genetic algorithm for job shop. In: Proceedings of the 1991 IEEE international conference on robotics and automation, pp 824–829

Frederickson G (1979) Approximation algorithms for some postman problems. J ACM 26(3):538–554

Golden B, Wong R (1981) Capacitated arc routing problems. Networks 11(3):305–315

Golden JS, DeArmon JS, Baker EK (1983) Computational experiments with algorithms for a class of routing problems. Comput Oper Res 10(1):47–59

Hertz A, Mittaz M (2001) A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. Transport Sci 35(4):425–434

Hertz A, Laporte G, Mittaz M (2000) A tabu search heuristic for the capacitated arc routing problem. Oper Res 48(1):129–135

Ho W, Ho G, Ji P, Lau H (2008) A hybrid genetic algorithm for the multi-depot vehicle routing problem. Eng Appl Artif Intell 21(4):548–557

Kansou A, Yassine A (2010) New upper bounds for the multi-depot capacitated arc routing problem. Int J Metaheuristics 1(1):81–95

Lacomme P, Prins C, Ramdane-Cherif W (2001) A genetic algorithm for the capacitated arc routing problem and its extensions. In: Boers EJW (ed) Applications of evolutionary computing, vol 2037/2001. Lecture Notes in Computer Science. Springer, pp 473–483

Lacomme P, Prins C, Ramdane-Cherif W (2004) Competitive memetic algorithms for arc routing problems. Ann Oper Res 131(1):159–185

Lau H, Chan T, Tsui W, Pang W (2010) Application of genetic algorithms to solve the multidepot vehicle routing problem. IEEE Trans Autom Sci Eng 7(2):383–392

Lenstra J, Rinnooy Kan AHG (1976) On general routing problems. Networks 6(3):273–280

Li L, Eglese R (1996) An interactive algorithm for vehicle routeing for winter-gritting. J Operat Res Soc 47(2):217–228

Liu T, Jiang Z, Chen F, Liu R, Liu S (2008) Combined location-arc routing problems: a survey and suggestions for future research. In: 2008 IEEE international conference on service operations and logistics, and informatics (SOLI'2008), Beijing, IEEE, pp 2336–2341

Mei Y, Tang K, Yao X (2011) Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem. IEEE Trans Evol Comput 15(2):151–165

Ombuki-Berman B, Hanshar F (2009) Using genetic algorithms for multi-depot vehicle routing. In: Bio-inspired algorithms for the vehicle routing problem. Springer, pp 77–99

Prins C (2009) Two memetic algorithms for heterogeneous fleet vehicle routing problems. Eng Appl Artif Intell 22(6):916–928

Santos L, Coutinho-Rodrigues J, Current J (2010) An improved ant colony optimization based algorithm for the capacitated arc routing problem. Transp Res Part B Methodol 44(2):246–266

Tang K, Mei Y, Yao X (2009) Memetic algorithm with extended neighborhood search for capacitated arc routing problems. IEEE Trans Evol Comput 13(5):1151–1166

Ulusoy G (1985) The fleet size and mix problem for capacitated arc routing. Eur J Oper Res 22(3): 329–337

Xing L, Rohlfshagen P, Chen Y, Yao X (2010) An evolutionary approach to the multidepot capacitated arc routing problem. IEEE Trans Evol Comput 14(3):356–374