OR Spectrum (2014) 36:401–421
DOI 10.1007/s00291-012-0289-0

REGULAR ARTICLE

A hybrid genetic algorithm for the multi-depot open vehicle routing problem

Ran Liu · Zhibin Jiang · Na Geng

Published online: 10 May 2012

Abstract This paper studies the multi-depot open vehicle routing problem (MDOVRP), a variant of the vehicle routing problem (VRP), in which vehicles start from several depots and are not required to return to the depot. Despite the vast amount of literature about VRPs, the MDOVRP has received very little attention from researchers. In this paper, a new hybrid genetic algorithm is presented for finding the routes that minimize the traveling cost of the vehicles. Computational results on a number of test instances indicate the proposed algorithm dominates the CPLEX solver and the existing approach in the literature. Meanwhile, experiments are conducted on multi-depot VRP benchmarks, and the results are compared with a sophisticated tabu search approach and an exact method.

Keywords Multiple depots · Open vehicle routing · Hybrid genetic algorithm

1 Introduction

The open vehicle routing problem (OVRP) is a variant of the classical vehicle routing problem (VRP) (Dantzig and Ramser 1959). The OVRP differs from the well-known VRP in that the vehicles do not return to the depot after serving the customers. The OVRP arises in several practical cases (Sariklis and Powell 2000; Russell et al. 2008).

R. Liu · Z. Jiang (✉) · N. Geng
Department of Industrial Engineering and Logistics Management,
School of Mechanical Engineering, Shanghai Jiao Tong University,
Shanghai 200240, China
e-mail: zbjiang@sjtu.edu.cn

R. Liu
e-mail: Liuran2009@gmail.com

N. Geng
e-mail: gengna@sjtu.edu.cn

For example, when a company does not own a vehicle fleet, or its private fleet is inadequate for fully satisfying customer demand, the company has to rent some vehicles to satisfy the demand of customers. Since the hired vehicles do not need to return to the depot after the completion of the task, the routing is open. In the OVRP, each customer is visited once by a single vehicle, the total demand of the customers assigned to a route does not exceed vehicle capacity, and the overall travel cost of the designed route set is minimized.

Since 2000, various heuristics and exact methods were proposed to solve the OVRP with some success. Sariklis and Powell (2000) first presented a heuristic method for solving the OVRP, based on a minimum spanning tree with penalties procedure. Some effective tabu search heuristics were implemented to solve the OVRP with different constraints, such as vehicle capacity and maximum route length (Brandão 2004; Fu et al. 2005). Tarantilis et al. (2005) solved the OVRP by adopting a simple threshold-accepting algorithm, called list based threshold accepting (LBTA) algorithm. The computational results on OVRP benchmark instances showed the LBTA algorithm was effective. Pisinger and Ropke (2007) presented an adaptive large neighborhood search (ALNS) approach to several variants of the VRP, including the OVRP. Li et al. (2007) proposed a record-to-record travel heuristic and a deterministic variant of simulated annealing to solve the OVRP. Recently, Repoussis et al. (2010) presented a hybrid evolution strategy for solving the OVRP. Based on integer linear programming (ILP) techniques, Salari et al. (2010) gave a heuristic improvement procedure for OVRP. Zachariadis and Kiranoudis (2010) proposed an OVRP meta-heuristic based on the static move descriptor technology, which reduces the computational complexity required for applying local search-based methods. For the exact method, Letchford et al. (2006) developed a branch-and-cut algorithm for the OVRP.

In terms of the OVRP, most researchers assume that all the vehicles start from the same central depot and end at a customer node, i.e., the single-depot OVRP. However, in many real-life transportation, logistics and supply chain practice, companies usually utilize a large vehicle fleet and operate from several depots or warehouses. The vehicles operate from one of several depots instead of only one. This kind of problem is much more complex than the single-depot OVRP, and can be called multi-depot OVRP (MDOVRP). The MDOVRP is also similar to another variant of the basic VRP, i.e., MDVRP (Chao et al. 1993; Renaud et al. 1996; Cordeau et al. 1997; Lim and Wang 2005; Lau et al. 2010). The MDVRP extends the VRP by allowing multiple depots, however, it requires that every vehicle route must start and end at the same depot. Clearly, the single-depot OVRP can be viewed as a special case of the MDOVRP. Since the single-depot OVRP is known to be NP hard in the strong sense (Brandão 2004), the MDOVRP is also NP hard. To the best of our knowledge, although many good research exist for the OVRP and MDVRP, only Tarantilis and Kiranoudis (2002b) presented a metaheuristic for solving the MDOVRP. In the work of Tarantilis and Kiranoudis (2002b), a real-life fresh meat distribution problem encountered by a major Greek industry distributing fresh meat from several depots to customers was studied. The problem was formulated as the MDOVRP and solved by a list-based threshold-accepting (LBTA) algorithm, which is a metaheuristic belonging to the class of threshold accepting based algorithms (Dueck and Scheuer 1990). The LBTA iteratively search the solution space guided by a control parameter, which is modified throughout the

iterations and is termed as threshold. Several kinds of local move methods are adopted in the LBTA, such as 2-opt move, 1-1 and 1-0 exchange moves, to search better neighbor solution and compute the new threshold value. For more details about the LBTA algorithm, readers refer to Tarantilis and Kiranoudis (2002b).

In this paper, we present a mixed integer programming (MIP) mathematical formulation for the MDOVRP. Meanwhile, we give an effective meta-heuristic solution approach, i.e., a hybrid genetic algorithm (HGA), for solving the MDOVRP. The HGA is constructed on the classical evolutionary algorithm framework (Reeves 2003) and hybridized with some kinds of local search methods (Sörensen and Sevaux 2006). It has been shown that the HGA is powerful for the classical VRP and its variants (Prins 2004, 2009; Ke et al. 2009; Mendoza et al. 2010). The remainder of this paper is organized as follows. In Sect. 2, the MDOVRP is formally introduced, and the mathematical formulation is defined. Then, our HGA for solving the MDOVRP is described in Sect. 3. In Sect. 4, we present a computational study that analyzes the proposed solution approach. Finally, we summarize our findings and suggest a future work direction in Sect. 5.

2 Problem definition and formulation

The MDOVRP can be defined as follows. Let $G = (V, A)$ be a directed graph, with $V = N \cup D$ is the vertex set, and A is the arc set. Vertex set $N = 1, 2, \ldots, n$ represents the customers to be served, and set $D = 1, 2, \ldots, m$ represents the uncapacitated depots. A travel cost c_{ij} is defined between each pair of vertices (i, j), $i, j \in V, i \neq j$. Each depot $d \in D$ stores and supplies products and has an unlimited fleet of identical vehicles with the same capacity of Q. Some researchers take into account the limited fleet constraint (also called capacitated depot). In this paper, we relax this constraint like the works of Salhi and Sari (1997), Branke et al. (2007) and Baldacci and Mingozzi (2009). Each customer $i \in N$ has a demand $q_i, 0 < q_i \leq Q$. For each depot, its demand equals zero. The MDOVRP consists of determining the routes of minimal traveling cost satisfying the following conditions: (1) every customer node is visited on exactly one route; (2) each vehicle starts from one of several depots, and end at the last customer it serves; (3) the total demand of the customers on any route does not exceed Q; (4) the traveling distance of any vehicle route cannot exceed a preset value H. In our proposed approach for solving the MDOVRP, we firstly set the distances from the customer nodes to each depot to be zero. That is to say, we set $c_{ij} = 0, \forall i \in N, j \in D$. Then, the following decision variables are used in the formulation:

$$x_{ijk} = \begin{cases} 1 & \text{if a vehicle starting from depot } k \text{ travels directly from customer } i \text{ to customer } j \\ 0 & \text{otherwise} \end{cases}$$

u_i An upper bound on the load of a vehicle upon leaving node i
h_i An upper bound on the traveling distance of a vehicle upon leaving node i

$$\text{Min} \sum_{i \in V} \sum_{j \in V} \sum_{k \in D} c_{ij} x_{ijk} \tag{1}$$

Subject to:

$$\sum_{i \in V} \sum_{k \in D} x_{ijk} = 1 \quad \forall j \in N \tag{2}$$

$$\sum_{i \in V} x_{ijk} = \sum_{i \in V} x_{jik} \quad \forall j \in V, k \in D \tag{3}$$

$$u_i - u_j + Q \cdot x_{ijk} \leq Q - q_j \quad \forall i \in N, k \in D, j \in V \backslash k \tag{4}$$

$$q_i \leq u_i \leq Q \quad \forall i \in N \tag{5}$$

$$u_k = 0 \quad \forall k \in D \tag{6}$$

$$h_i - h_j + H \cdot x_{ijk} \leq H - c_{ij} \cdot x_{ijk} \quad \forall i \in N, k \in D, \forall j \in N \backslash k \tag{7}$$

$$0 \leq h_i \leq H \quad \forall i \in N \tag{8}$$

$$h_k = 0 \quad \forall k \in D \tag{9}$$

$$x_{ijk} \in \{0, 1\} \quad \forall i \in V, j \in V, k \in D, i \neq j \tag{10}$$

The objective function (1) minimizes the total travelling distance. Constraints (2) ensure that each customer node is served by exactly once. Constraints (3) impose the degree balance for each node, including both the customer nodes and depots. Constraints (4)–(6) eliminate the sub-tours in the solution, and ensure that the given vehicle capacity is not exceeded, which is based on the Miller–Tucker–Zemlin sub-tour elimination for the classical VRP (Miller et al. 1960; Kara et al. 2004; Bolduc et al. 2010). Note that in constraints (4) j belongs to set $V \backslash k$, while in similar VRP sub-tour elimination constraints j belongs to the customer set N. Our constraints forbid not only the classical sub-tour but also a kind of special tour in our formulation, which connects two or more depots. For example, consider a tour $(d_1, n_1, d_2, n_3, d_1)$, where d_1, d_2 are two depots, n_1, n_2 are customer nodes. Supposed this tour starts from d_1, constraints (4) imply $u_{d2} \geq u_{n1} + q_{d2} = u_{n1} \geq q_{n1} > 0$. Since this inequality violates constraints (6), such kind of illegal tour is forbidden. Similarly, constraints (7)–(9) ensure that each vehicle route cannot exceed the distance span H.

This vehicle flow formulation is simple and flexible. Clearly, it can be used for the multi-depot closed VRP. Meanwhile, some OVRP researches consider the fixed cost of using a vehicle in the objective function (Repoussis et al. 2006, 2010; Li et al. 2007). If the objective is keeping the trade-off between route distance and vehicle fixed cost, we modify the distance matrix by adding the fixed cost of a vehicle to the distance from the depots to each customer. If the first objective is to find the minimum required vehicles and the second is to minimize the vehicle traveling cost, this can be satisfied by adding a sufficiently large number M to the distance from the depots to each customer. This forces the algorithm towards solutions with minimal number of vehicles to avoid the penalty.

3 Hybrid genetic algorithm

Genetic algorithms (GAs) are global search optimization techniques based on the mechanics of natural selection and reproduction, which have been found to be very

effective in solving complex optimization problems (Cvetkovic and Parmee 2002; Kim et al. 2003). In the area of VRPs, the basic and simple GA may not perform very well. Therefore, some good hybrid genetic algorithms (HGAs) are designed to compete with other powerful meta-heuristics, such as tabu search and simulated annealing. The process of HGA additionally generates the initial population using random and heuristic techniques for better values (Berger and Barkaoui 2004; Jeon et al. 2007), and uses the simple heuristics, e.g., local search procedures, to replace the simple mutation operators (Potvin and Bengio 1996; Prins 2004, 2009). In this work, we propose a HGA to solve the MDOVRP. The flowchart of the proposed HGA for the MDOVRP is shown in Fig. 1. The algorithm starts with an initial population, i.e., a set of solutions of the MDOVRP. Each solution in the population is called a chromosome, which is evaluated by the measure of fitness. Then, during each generation, the algorithm applies the crossover to two chromosomes (called parents) so as to get two new chromosomes (also called offspring). Then, the algorithm applies some local search methods to a new child chromosome. If the termination condition is satisfied, the algorithm reports the best chromosome found so far and stops. Otherwise, the algorithm goes on to repeat the evolution process. In the following sections, the basic components of the HGA, e.g., chromosome representation, population initialization, crossover method, and local search methods integrated in our HGA, are described.

3.1 Solution coding and decoding of chromosome

Chromosome used in the proposed genetic algorithm is encoded as permutation of all the customer nodes without trip delimiters, just like a giant traveling salesman problem (TSP) tour visiting all the customer nodes. Each chromosome has a cost, which corresponds to the cost of the best MDOVRP solution extractable from the chromosome. Such kind of chromosome cannot represent the MDOVRP solution directly. However, the advantage of such kind of chromosome coding is: (1) It is simpler to apply the crossover methods to the chromosome; (2) The algorithm can transmit good genetic information from parent chromosomes to children chromosomes when executing the crossover operator; (3) The algorithm can get the optimal MDOVRP solution for a sequence of all the customer nodes, i.e., for each chromosome the algorithm can get its optimal fitness. Similar chromosome encoding method is used in some excellent genetic algorithms for solving the VRP (Prins 2004) and its variant problems (Fallahi et al. 2008; Prins 2009). In this study, the following *split* procedure is adopted to evaluate the fitness of a chromosome, and to partition a chromosome into a set of feasible routes, i.e., the solution of the MDOVRP.

Given a chromosome $S = (s_1, \ldots, s_n)$, we build an acyclic auxiliary graph $T = (X, Y)$ with $n + 1$ nodes, where vertex set X contains nodes indexed from 0 to n, and Y is the directed arc set. One arc $(i, j), i < j$ is contained in set Y, iff a vehicle visiting customers subsequence $(s_{i+1}, \ldots, s_j)$ in S is feasible in terms of following load constraints (11) and traveling distance constraints (12). In constraints (12), z_{ij} is the cost of this arc (i, j).

$$\sum_{k=i+1}^{j} q_{Sk} \leq Q \tag{11}$$

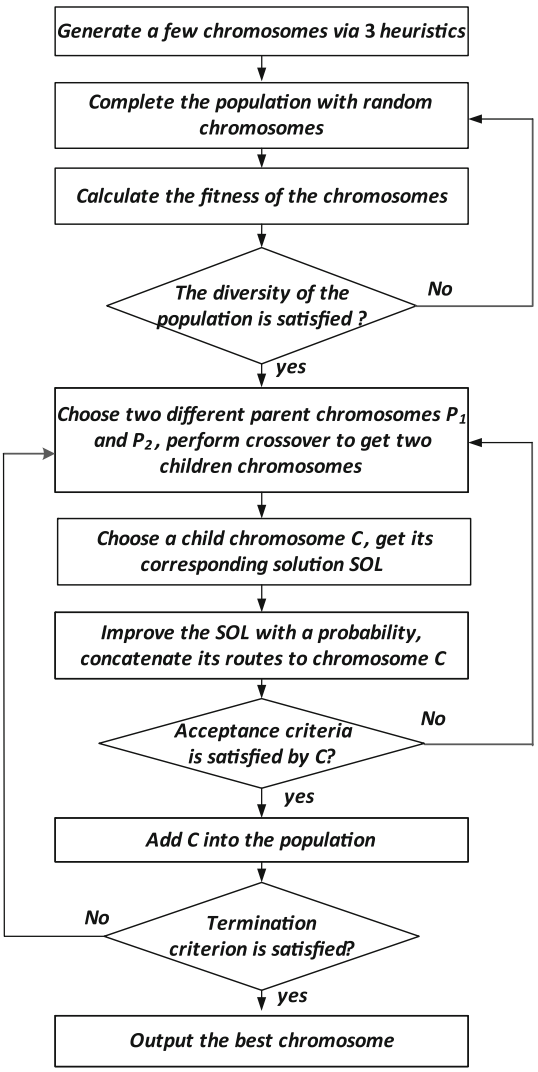

Fig. 1 General structure of the hybrid genetic algorithm

$$z_{ij} = \min\left(c_{d,Si+1} + \sum_{k=i+1}^{j} c_{Sk,Sk+1}\right) \leq H, \quad \forall d \in D \tag{12}$$

The optimal partition of the chromosome S corresponds to a shortest path problem, i.e., defining the shortest path from node 0 to node n in graph T. The cost of shortest path from node 0 to node n in graph T is adopted as the fitness of chromosome S. To describe chromosome partition method more clearly, an example is shown in Fig. 2. The first part of Fig. 2 shows a small instance with two depots (depots 1 and 2), and four customer nodes (a, b, c, d). The visiting sequence of nodes, i.e., the chromosome, is (a, b, c, d) with $Q = 10$, $H = 35$. The demand of each customer is shown in brackets,

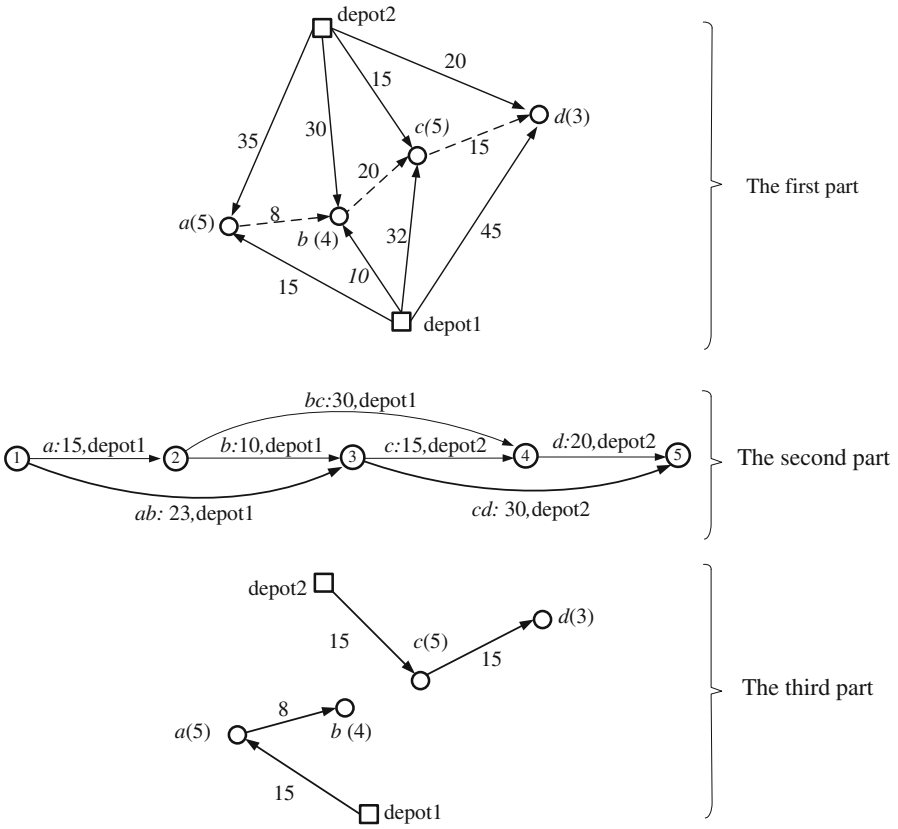

Fig. 2 The split procedure applied to a small instance

and the distances between each two nodes are given beside the arc. In the second part of Fig. 2, each arc in the auxiliary graph represents a possible vehicle route, with the label of traveling distance and the starting depot. The shortest path from nodes 1 to 5 is $(1 \rightarrow 3 \rightarrow 5)$ with the cost of 55. Then, the corresponding MDOVRP solution is given in the third part, consisting of two vehicle routes. Two routes start from deopt1 and depot2, separately.

3.2 Population structure and initialization

The HGA population consists of chromosomes of various costs (i.e., fitness), which are sorted in an ascending cost order. In the initial population, three chromosomes are generated via three heuristics, and the others are generated as random permutations of customer nodes.

The first heuristic is a simple two-stage procedure, based on the *saving* algorithm for the single-depot VRP (Clarke and Wright 1964). The saving algorithm is one of the most famous and effective heuristic for the VRP, which is based on the notion of *saving*. In our study, firstly each customer is assigned to the closest depot. Then, for

each depot a single-depot OVRP is solved by the saving algorithm. For the single-depot OVRP, the calculation of savings values should consider the fact that vehicles need not return to the depot. The second heuristic is based on the approach of *sweep* (Gillett and Miller 1974). Similarly, each customer is assigned to the closest depot. Then, for each depot feasible routes are created by rotating a ray centered at the depot and gradually including customers in a vehicle route until the capacity or route length constraint is attained. A new route is then initiated and the process is repeated until the entire plane has been swept. The third heuristic is proposed by Salhi and Sari (1997), which is more complex than the first two heuristics. Firstly, customers are divided into core customers and borderline customers by looking at the ratio between the distance to their nearest depot and the distance to the second-nearest depot. In this paper, we set the ratio to be 0.5. Then, the core customers are assigned to their nearest depots. We use the saving method to solve the VRPs for each depot with its core customers. Finally, borderline customers are inserted one by one in the order of decreasing opportunity cost, where opportunity cost is the absolute difference between inserting the customer into a tour starting at the closest depot, and inserting the customer into a tour starting at the second closest depot. The customer is inserted where the additional cost is minimal, and the vehicle capacity and route length constraint is satisfied. At this step, if a borderline customer cannot be inserted into all existing routes duo to vehicle capacity or route length constraint, a new route is generated, connecting this customer and the nearest depot.

Besides the chromosomes generated by three simple heuristic algorithms, the rest of the population is composed of chromosomes generated randomly, each of which is a random sequence of customer nodes. Each initial chromosome is partitioned by the split algorithm mentioned in previous section to define its exact cost (fitness).

3.3 Selection and crossover

In the proposed HGA, two parent chromosomes P_1 and P_2 are selected with the tournament selection method to generate their offspring. First, a few chromosomes are chosen randomly from the population. Then, the chromosome with the smallest cost is selected to be P_1. The procedure is repeated to get the second parent P_2. Selection pressure can be adjusted by changing the tournament size. With the increase in the tournament size, weak chromosomes will have smaller chance to be selected in the tournament selection. In this study, from the preliminary experimental results, we find that the tournament size of three can produce better result.

Since our chromosomes do not contain route and depot delimiters, they can undergo classical recycling crossover methods designed for the TSP. In the proposed algorithm, the crossover method of OX (Oliver et al. 1987) is used to perform the permutation representation, because OX rather concerns circular permutations. For a detailed description of OX used in this work, see Liu et al. (2009) and Prins (2009). The OX generates two children solutions. One child is selected randomly and undergoes local search methods (described in Sect. 3.4). Then, if the cost of this child chromosome is less than the worst chromosome in the existing population and this child does not violate the diversity of the population, it replaces one chromosome randomly selected in the worst half of the population.

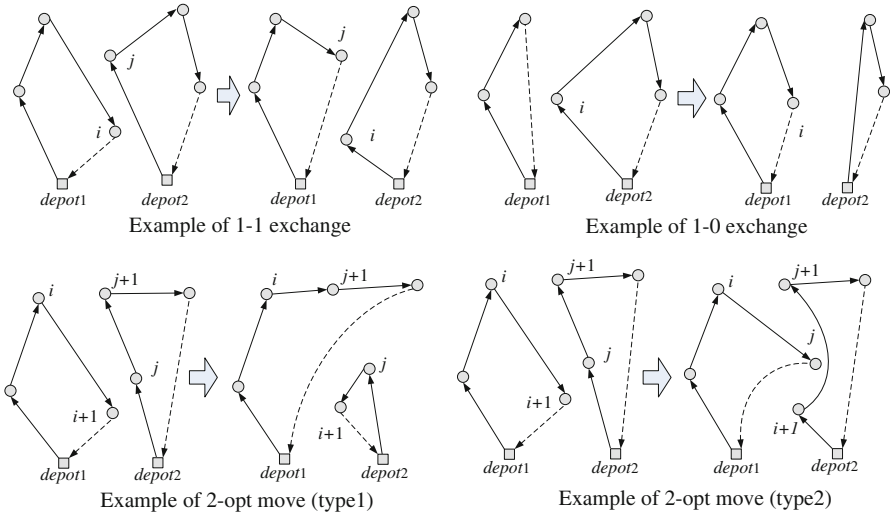

Fig. 3 Local search methods for the routes starting from different depots

3.4 Local search

The basic genetic algorithm with simple mutation methods (e.g., swapping or relocate some nodes) (Holland 1975) is hard to compete with other modern heuristics, such as simulated annealing and tabu search. So, in this work we hybridize the local search with the basic genetic algorithm to make the algorithm more effective. A large number of local search schemes were developed for the VRP, OVRP, MDVRP and some relative problems (Fleszar et al. 2009; Ho et al. 2008; Liu et al. 2010). In our work, three kinds of local search methods are used, i.e., 1-0 exchange move, 1-1 exchange move and 2-opt move. Tarantilis and Kiranoudis (2002b) and Prins (2009) also adopted these local search methods in their algorithms. The descriptions and pictorial illustrations of these local search methods for single and multiple routes originating from only one depot can be found in their works. Meanwhile, these methods are applied to various routes originating from different depots in our algorithm. Four examples in Fig. 3 show how 1-0 exchange, 1-1 exchange and 2-opt move are applied to customers served by the vehicles starting from two depots. In the top part of Fig. 3, the 1-0 exchange and 1-1 exchange are applied to two routes, which originate from depots 1 and 2, respectively. And, the bottom part of Fig. 3 demonstrates how the 2-opt move are adopted in such kind of case. Note that when 2-opt move is applied to two routes, there are two possible ways to re-connect route segments: the arcs $(i, i+1)$ and $(j, j+1)$ are replaced either by $(i, j+1)$ and $(j, i+1)$, or by arcs (i, j) and $(i+1, j+1)$.

These local search procedures are employed to the selected child chromosome with a probability. During the local search procedure, we adopt 'first-accept', not 'best-accept' strategy, i.e., during a local search procedure once a feasible and better solution is found, it is adopted as the new seed for repeating this local search. The local search on the selected child chromosome is repeated until no improving move can be found. Note that the local search procedures operate on a 'real' problem solution, not

on the chromosome itself. The local search result (a real solution) is then converted into its corresponding chromosome by concatenating its routes and deleting all the depots. Finally, the split algorithm is applied to this resulting chromosome since it may find a better partition and cost for the same chromosome.

3.5 Algorithmic variables and stopping criterion

The proposed genetic algorithm has some parameters, such as population size and local search probability. Based on the guidance given by Prins (2004) and our experimental results, we use a relative small population size of 30, and a low local search probability of 0.25.

It has been proven that controlling the diversity of population is important since it can diminish the risk of premature convergence (Hertz and Widmer 2003). In our work, a simple rule is imposed to keep the diversity of the HGA population, i.e., the costs of any two chromosomes must be different. This rule is checked when generating the initial population and trying to insert an offspring, which is generated from crossover operator and undergoes local search methods, into the population. When initializing the population, three chromosomes are generated by simple heuristic algorithms, the rest of the population are generated randomly. Then, the diversity of the population is checked. If the diversity requirement is not satisfied, all the violated chromosomes are removed from the population. That is to say, if several chromosomes have the same fitness, only one of them is reserved and the others are deleted. Then, the empty spaces in the population are filled with new randomly generated chromosomes. We checked the population again. This procedure is repeated until the population satisfies the diversity requirement. Similarly, after a child chromosome is generated and undergoes local search, we must confirm that it will not break the diversity of the population when inserting it into the population. Otherwise, this child chromosome is discarded.

The stopping criterion can be defined in terms of elapsed CPU time, the number of iterations, or number of iterations with no improvement in the best solution can be found. In the proposed HGA, we use a simple criterion: a variable is used to record the number of algorithm iterations. Once a pair of offspring is created, this variable increases by one. The algorithm stops when this variable reaches the defined limitation.

4 Computational experiments

4.1 Experiments on the MDOVRP instances

To the best of our knowledge, there are no benchmark test instances for the MDOVRP with which to test the HGA proposed in this paper. Thus, we modify the classical MDVRP benchmark instances and randomly generated our test instances with different scales. In this section, we explain how to generate the MDOVRP test instances, and report the results.

4.1.1 Experimental data

Firstly, we construct the MDOVRP test instances based on the MDVRP benchmark instances proposed by Cordeau et al. (1997), which are widely used in the literature. Clearly, most information in these MDVRP benchmark instances can be used for the MDOVRP directly. Note that in these instances the depot is capacitated, i.e., the number of the vehicles in each depot is limited, while in our MDOVRP such constrains are relaxed. Meanwhile, the heuristic approach proposed by Tarantilis and Kiranoudis (2002b) does not consider the vehicle route length constraint. To give a fair comparison, we also relax the constraint of the route length in this series of experiments (route length constraint will be taken into consider in the third series of experiments, in Sect. 4.2). Meanwhile, we find that there exist some *similar* MDVRP instances (e.g., $p12$, $p13$ and $p14$), between which the only difference is the maximum duration of the route. For such kind of similar MDVRP instances, only one of them is adopted in our experiment.

Meanwhile, the performance evaluation of the proposed approach on the MDOVRP was carried on three sets of randomly generated instances, which contain: (1) 5 depots and 50 customers; (2) 5 depots and 100 customers; and (3) 10 depots and 350 customers, respectively. In each test instance, the depots and customers spread randomly in a rectangle of width and height of 1,000. The demand quantity of each customer was randomly generated between 10 and 100. In first two sets of instances, the vehicle capacity is 500. In the third set large scale instances, we set the vehicle capacity to be 2,000. In all the test instances used in this paper, both travel cost and distance are given by the Euclidean distance between points, without rounding.

The performance of the proposed hybrid genetic algorithm was compared with (1) a powerful commercial MIP solver, CPLEX 12.2, and (2) the list-based threshold accepting (LBTA) heuristic proposed by Tarantilis and Kiranoudis (2002b) for solving the MDOVRP. As our demands to get the code of LBTA heuristic have been unfruitful, we developed our own implementation. All the algorithms were implemented in C++. All experiments are carried out on a 3.2 GHz Dual Core computer with a memory of 4 GB under Windows XP. We set a long time limit of 12 h on running CPLEX for each instance. For each instance, CPLEX runs with default settings when finding an optimal solution, or until exhausting the memory or predetermined maximum computation time. If CPLEX can solve the instance to optimality, or terminates because of exhausting the memory within 12 h, we report the running time of CPLEX. Otherwise, if CPLEX is still running when the maximum computation time is reached, the CPLEX computing time is considered to be 12 h. In our preliminary experiments, we also try different CPLEX parameter settings, e.g., strong branching, depth-first search, MIP emphasis feasibility or optimality, to tune Cplex's performance. But such settings cannot lead to obvious performance improvement.

4.1.2 Computational results for Cordeau's instances

We first test the proposed algorithm on 24 instances originated from MDVRP benchmarks (Cordeau et al. 1997), with 48 to 288 customers, and 2 to 6 depots. Two heuristics, HGA and LBTA, were run 20 times on each instance. The best results, the average results and average running time were got from these 20 runs. The setting

of HGA parameters is explained as follows. The population size is 30, the stopping criteria, i.e., the maximal number of algorithm iterations equals 30,000, and the local search probability is 0.25. The computational results are presented in Table 1, which contains the solution information obtained from the CPLEX, HGA and LBTA.

In Table 1, the first to forth columns '*Instance*', 'C_N', 'D_N' and 'Q' present the instance name, the number of the customers and the depots, and the vehicle capacity. The columns with the headers '$Cplex_{\mathrm{UB}}$', '$Cplex'_{\mathrm{LB}}$ and 't_{c}' refer to the solution value, the lower bound and running time (in seconds) of CPLEX solver, respectively. The columns headed 'HGA_{b}', 'HGA_{avg}' and 't_{h}' present the best and average HGA solution costs over 20 runs, and average computational time of one run in seconds. Similarly, the columns headed '$LBTA_{\mathrm{b}}$' and 't_{l}' present the corresponding best solution cost and running time of the LBTA. The column '*C–H%*' contains the percentage deviation between the CPLEX solution cost $Cplex_{\mathrm{UB}}$ and HGA best solution cost HGA_{b}, which is calculated as: $100 \times (Cplex_{\mathrm{UB}} - HGA_{\mathrm{b}})/Cplex_{\mathrm{UB}}$. And, the last column '*L-H%*' represents the deviation between best solution costs of two heuristics HGA and LBTA, calculated by $100 \times (LBTA_{\mathrm{b}} - HGA_{\mathrm{b}})/LBTA_{\mathrm{b}}$. If an instance is proven to be solved to optimality, it is marked with an asterisk.

Three remarkable conclusions can be drawn from these experimental results. First, the proposed HGA is able to provide high quality solutions for these instances. As shown in Tables 1 and 5, the instances are solved to optimality by CPLEX. The proposed HGA also obtains the optimal solutions for these instances within rather shorter running times. For other test instances, CPLEX fails to provide the optimal solution (its objective value cannot reach the lower bound), while HGA can get much better solutions. For total 24 instances, the average deviation between CPLEX objective values and HGA best solution costs is 19.93 %. Meanwhile, LBTA can find optimal solutions for four instances. For each instance the solution obtained by HGA is equal to, or better than the solution got by LBTA. For all other 20 test instances, the solutions found by the HGA clearly dominate the solutions of the LBTA. For all 24 test instances, the average gap between HGA and LBTA best solution costs (i.e., HGA_{b} and $LBTA_{\mathrm{b}}$) is 6.16 %. Second, we find that the proposed HGA is very robust with respect to different test instances. It always finds the best solution (among CPLEX, HGA and LBTA) for each instance. And, we see that HGA can obtain these good solutions in a reasonable computational time. The average running time of CPLEX for all these instances is about 270 min, while the average running time of HGA is less than 10 min. Compared with CPLEX solver, the running time of HGA is much shorter.

4.1.3 Computational results for randomly generated instances

In Tables 2, 3 and 4, we present the experimental results on three sets of randomly generated instances, where the first set is small scale, and the second and the third set are moderate and large size. Here, HGA uses the parameters as used for Cordeau's instances. We find that for the first set of small scale random instances, CPLEX can solve 9 out of 12 instances optimally. For the other 3 '*hard*' instances, CPLEX is able to provide good lower bounds. For the second set moderate size instances, CPLEX fails to get any optimal solution within the time limit of 12 h, but it still can provide feasible solutions. For large scale instances containing 350 customers and 10 depots,

Table 1 Computational result for MDOVRP on Cordeau's instances

Instance	C_N	D_N	Q	CPLEX			HGA			LBTA		GAP	
				$Cplex_{UB}$	$Cplex_{LB}$	t_c (s)	HGA_b	HGA_{avg}	t_h (s)	$LBTA_b$	t_l (s)	$C-H$ (%)	$L-H$ (%)
p01	50	4	80	386.18*	386.18	1,4891.2	386.18*	388.82	13.8	386.18*	7.3	0.00	0.00
p02	50	4	160	375.93*	375.93	11.2	375.93*	376.90	14.5	375.93*	8.1	0.00	0.00
p03	75	5	140	474.57	474.17	43,200.0	474.57	477.69	40.8	474.91	22.1	0.00	0.07
p04	100	2	100	721.31	598.58	7,942.0	667.02	669.49	86.0	719.28	30.7	7.53	7.27
p05	100	2	200	627.46	595.13	16,115.2	612.30	616.35	91.0	630.15	40.3	2.42	2.83
p06	100	3	100	614.93	579.76	43,200.0	614.93	621.13	91.4	614.93	41.7	0.00	0.00
p07	100	4	100	619.48	573.96	43,200.0	615.24	623.91	87.7	621.75	41.0	0.68	1.05
p08	249	2	500	4,869.25	2,157.35	12,129.0	2,913.63	2,911.51	1,089.4	3,172.91	622.1	40.16	8.17
p09	249	3	500	6,328.72	2,100.93	11,945.3	2,738.54	2,755.28	1,102.3	3,191.87	614.1	56.73	14.20
p10	249	4	500	7,740.98	2,038.94	902.9	2,577.48	2,616.02	1,118.8	2,871.69	637.3	66.70	10.25
p11	249	5	500	5,665.44	1,822.63	2,544.5	2,561.85	2,598.56	1,113.8	2,796.88	507.7	54.78	8.40
p12	80	2	60	953.26*	953.26	5.0	953.26*	955.80	49.3	956.26	31.7	0.00	0.31
p15	160	4	60	3,289.05	1,873.09	14,344.2	1,888.67	1,899.98	337.0	2,004.11	157.2	42.58	5.76
p18	240	6	60	4,795.26	2,714.80	10,245.2	2,830.78	2,854.81	1,038.2	3,194.26	712.1	40.97	11.38
pr01	48	4	60	647.03*	647.03	17.1	647.03*	647.03	13.1	647.03*	10.2	0.00	0.00
pr02	96	4	60	984.32	964.55	43,200.0	981.63	985.69	84.9	988.46	28.8	0.27	0.69
pr03	144	4	60	1,485.90	1,380.67	9,966.48	1,445.65	1,458.75	302.3	1,480.36	173.5	2.71	2.34
pr04	192	4	60	1,900.21	1,412.25	12,243.3	1,548.93	1,586.67	368.0	1,821.91	150.0	18.49	14.98
pr05	240	4	60	2,211.06	1,551.18	43,200.0	1,746.57	1,755.26	1,024.4	2,000.91	621.8	21.01	12.71
pr06	288	4	60	4,359.83	1,716.00	4,034.2	2,069.01	2,085.62	1,767.5	2,261.62	712.1	52.54	8.52
pr07	72	6	60	821.25*	821.25	154.2	821.25*	824.09	40.7	821.25*	19.9	0.00	0.00
pr08	144	6	60	1,312.46	1,211.21	11,545.7	1,266.87	1,277.79	251.7	1,278.40	127.5	3.47	0.90
pr09	216	6	60	1,683.81	1,407.75	43,200.0	1,643.56	1,666.02	778.5	1,658.78	711.6	2.39	0.92
pr10	288	6	200	5,813.10	1,867.47	1,829.8	2,037.22	2,076.08	1,740.9	3,196.68	726.3	64.95	36.27
Average				2,445.03		16,252.8	1,434.09	1,448.35	526.8	1,590.5	281.5	19.93	6.16

* Indicates the optimal solution

Table 2 Computational results on the first set random instances

Instance	C_N	D_N	CPLEX			HGA			LBTA		GAP	
			$Cplex_{\mathrm{UB}}$	$Cplex_{\mathrm{LB}}$	t_{c} (s)	HGA_{b}	HGA_{avg}	t_{h} (s)	$LBTA_{\mathrm{b}}$	t_{l} (s)	$C-H$ (%)	$L-H$ (%)
R1-1	50	5	4,499.05*	4,499.05	139.9	4,499.05*	4,500.09	6.0	4,499.05*	1.8	0.00	0.00
R1-2	50	5	4,677.94*	4,677.94	2,845.5	4,677.94*	4,678.75	10.2	4,677.94*	3.0	0.00	0.00
R1-3	50	5	4,431.80*	4,431.80	689.5	4,431.80*	4,432.11	7.1	4,438.10	2.1	0.00	0.14
R1-4	50	5	5,117.39	5,005.81	43,200.0	5,115.15	5,127.85	9.2	5,115.15	2.7	0.04	0.00
R1-5	50	5	4,709.15*	4,709.15	5,011.3	4,709.15*	4,728.01	9.7	4,709.15*	3.0	0.00	0.00
R1-6	50	5	4,412.95*	4,412.95	18.23	4,412.95*	4,413.00	5.8	4,412.95*	2.1	0.00	0.00
R1-7	50	5	4,399.10*	4,399.10	192.10	4,399.10*	4,352.19	6.6	4,401.87	1.7	0.00	0.06
R1-8	50	5	4,670.64	4,605.60	43,200.0	4,670.64	4,677.21	9.5	4,670.64	3.2	0.00	0.00
R1-9	50	5	5,028.58	4,904.39	43,200.0	5,027.91	5,029.01	11.3	5,028.91	3.5	0.01	0.02
R1-10	50	5	4,603.36*	4,603.36	4,234.3	4,603.36*	4,626.80	7.2	4,603.36*	1.9	0.00	0.00
R1-11	50	5	4,513.18*	4,513.18	555.9	4,513.18*	4,537.86	8.1	4,513.18*	2.0	0.00	0.00
R1-12	50	5	4,758.98*	4,758.98	38,218.9	4,758.98*	4,766.41	9.3	4,759.38	3.1	0.00	0.01
Average			4,651.84		15,125.5	4,651.60	4,655.77	8.3	4,652.47	2.5	0.00	0.02

* Indicates the optimal solution

Table 3 Computational results on the second set random instances

Instance	C_N	D_N	CPLEX			HGA			LBTA		GAP	
			$Cplex_{\mathrm{UB}}$	$Cplex_{\mathrm{LB}}$	t_c (s)	HGA_{b}	HGA_{avg}	t_{h} (s)	$LBTA_{\mathrm{b}}$	t_{l} (s)	$C-H$ (%)	$L-H$ (%)
R2-1	100	5	7,063.27	16,128	16,128.0	7,026.22	7,057.42	72.0	7,066.31	31.1	0.52	0.99
R2-2	100	5	7,325.15	43,200	43,200.0	7,263.30	7,303.69	72.5	7,327.75	30.9	0.84	0.88
R2-3	100	5	8,646.67	10,578	10,578.3	7,961.37	8,057.63	72.0	8,167.87	30.4	7.93	2.53
R2-4	100	5	7,640.12	43,200	43,200.0	7,552.12	7,599.49	72.1	7,617.51	35.9	1.15	0.86
R2-5	100	5	8,552.63	43,200	43,200.0	7,836.68	7,893.74	72.0	8,290.32	30.2	8.37	5.47
R2-6	100	5	8,290.25	40,920	40,920.5	8,078.52	8,092.45	78.3	8,092.91	36.2	2.55	1.40
R2-7	100	5	7,730.95	43,200	43,200.0	7,719.71	7,747.12	72.8	7,913.46	30.2	0.15	2.45
R2-8	100	5	8,390.99	43,200	43,200.0	8,052.16	8,163.92	72.0	8,391.53	30.0	4.04	4.04
R2-9	100	5	7,392.97	31,602	31,602.0	7,217.00	7,322.27	76.5	7,450.37	35.8	2.38	3.13
R2-10	100	5	8,208.49	7,168.17	43,200.0	43,200	8,123.81	72.0	8,209.79	36.1	1.72	1.74
R2-11	100	5	7,752.12	7,065.44	43,200.0	43,200	7,751.66	72.1	7,799.53	25.0	0.44	3.46
R2-12	100	5	8,100.30	7,551.56	29,754.2	29,754	7,896.17	84.5	8,157.41	30.2	3.24	3.92
Average			7,924.49		35,948.5	7,694.18	7,750.78	74.1	7,873.73	31.8	2.78	2.23

Table 4 Computational results on the third set random instances

Instance	C_N	D_N	**HGA**			**LBTA**		$L-H$ (%)
			HGA_{b}	HGA_{avg}	t_{HGA} (s)	$LBTA_{\mathrm{b}}$	t_{l} (s)	
R3-1	350	10	14,274.78	14,755.79	2,899.4	14,957.80	1,642.2	4.57
R3-2	350	10	14,302.28	14,687.76	2,917.5	15,086.12	1,695.0	5.20
R3-3	350	10	14,063.04	14,191.31	2,817.9	14,721.30	1,610.9	4.47
R3-4	350	10	14,621.31	14,729.87	3,000.9	15,177.62	1,770.4	3.67
R3-5	350	10	15,020.28	15,164.49	2,901.2	15,020.28	1,635.9	0.00
R3-6	350	10	13,582.80	13,615.44	2,987.1	15,830.20	1,508.6	14.20
R3-7	350	10	14,839.89	14,937.05	2,841.5	15,542.84	1,588.3	4.52
R3-8	350	10	13,702.25	13,959.19	2,871.0	14,184.94	1,662.9	3.40
R3-9	350	10	14,441.78	14,549.82	2,927.5	14,749.17	1,718.1	2.08
R3-10	350	10	15,209.81	15,289.54	2,996.3	15,881.18	1,634.0	4.23
R3-11	350	10	15,541.97	15,780.26	2,813.2	16,700.79	1,710.9	6.94
R3-12	350	10	14,837.43	14,967.59	2,926.7	15,787.96	1,627.5	6.02
Average			14,536.47	14,719.01	2,908.4	15,303.35	1,650.4	4.94

the CPLEX solver is unable to obtain feasible solution for all instances. Therefore, Table 4 does not present the CPLEX performance. It only compares the performances between HGA and LBTA for these large test instances.

The computational results shown in Tables 2, 3 and 4 are discussed as follows. Firstly, the performance of the proposed HGA is always better than CPLEX solver and LBTA in terms of solution quality. Concerning the small scale instances, the performances of CPLEX, HGA and LBTA are rather promising. As shown in Table 2, both HGA and CPLEX can get optimal solutions for 9 out of 12 small instances, and LBTA can find optimal solutions for 6 instances. For all 12 instances in Table 2, the average percentage deviation between HGA best solution costs and lower bounds provided by CPLEX is only 0.53 %, and the corresponding percentage ratio between LBTA and lower bound is 0.55 %. This proves that both two heuristics, HGA and LBTA, can find near-optimal solutions for small scale instances. For 12 moderate instances shown in Table 3, we find that HGA absolutely dominates the CPLEX solver, and the performance of the LBTA begins to deteriorate. For example, the average gap between HGA_{b} (the HGA best solution cost) and CPLEX solution cost is 2.78 %, the maximum and average gap between HGA_{b} and $LBTA_{\mathrm{b}}$ (best solution provided by the LBAT) is 5.47 and 2.23 %. Then, concerning the large scale instances shown in Table 4, we find that for each one instance the solution quality of HGA is much better than LBTA: the maximum and average percentage deviations between HGA_{b} and $LBTA_{\mathrm{b}}$ increase to 14.20 and 4.94 %. For total 36 random instances, the maximum and average percentage gaps between HGA_{b} and $LBTA_{\mathrm{b}}$ are 14.20 and 2.40 %, respectively. Summarizing the results shown in Tables 2, 3 and 4, we can say that the proposed HGA outperforms the CPLEX solver and the existing heuristic in terms of the quality of solutions. The superiority of the HGA is more apparent in medium and large size test instances.

Meanwhile, we find that the computing time of the HGA is still always reasonable. For medium size test instances, the proposed HGA can get the solution in 2 min. For

the large scale test instances, the average running time of HGA is less than 50 min. Of course, you may and the running time of the LBTA is less than that of the HGA. Compared with the LBTA, the improvement in the HGA solution quality is obtained by elapsing a larger CPU time. This may be due to two reasons. Firstly, compared with the local search operator used in the LBTA, our HGA adopts more complex local search methods as intensification strategy. Secondly, the frame of the genetic algorithm is more complex and time consuming than the basic structure of the LBTA. It has been shown that for some other computational optimization problems, e.g., job shop scheduling problems, the computational time of LBTA is less than most modern heuristics, such as tabu search and simulated annealing method (Tarantilis and Kiranoudis 2002a). However, on the other hand, though the running time of HGA is not as good as LBTA, in general, it is still acceptable. Furthermore, to give a fair comparison between HGA and LBTA, we adopt two strategies to improve LBTA. (1) We increase the running time of the LBTA to the same with the HGA for each randomly generated instance. We find that HGA still always outperforms the LBTA, leading to a maximum gap of 13.87 % (against 14.20 %) and average gap of 2.25 % (against 2.40 %). (2) We adopt a simple multi-start strategy to extend the running of LBTA to the same time of HGA. The fundamental principle of multi-start strategy used here is to create new seed solution, and let LBTA continue to search from the new seed. We let LBTA not only record the best solution obtained thus far, but also record the second best solution during its search. When the searching of LBTA falls into stagnation, it restarts from the second best solution obtained thus far. This procedure is repeated until the running time of the LBTA to the same with the HGA. Unfortunately, we find when LBTA integrated this multi-start strategy still provides slight improvement in its solution. The maximum percentage gaps between t_b and $LBTA_b$ is not changed, while the average gap is 2.23 %.

4.2 Experiments on the MDVRP

As stated above, in our formulation and algorithm the traveling distance (cost) from each customer node back to the depot is set to be 0. But if we do not adopt this setting, i.e., using the real value to represent the distance (cost) from each customer back to the depot, clearly our HGA can solve the closed MDVRP. It is very interesting to test our HGA on the MDVRP, and compare our results with other sophisticated heuristics and exact method. Here, 33 well-known MDVRP benchmark instances created by Cordeau et al. (1997) were chosen again. In Cordeau et al. (1997), the authors proposed a powerful tabu search heuristic, consisting of the GENI heuristic which was used to insert unrouted customers or remove customers from their current routes and then reinsert them into different routes. The tabu search proposed by Cordeau et al. (1997) has a good performance for the MDVRP. Thus, the results obtained by our HGA are compared with the results obtained by tabu search proposed by Cordeau et al. (1997). Note in Cordeau's tabu search the vehicles located at each depot is limited, and their s results reported in Cordeau et al. (1997) were got under this constraint. In our study, we relax this constraint and use Cordeau's heuristic to solve these benchmarks again. Meanwhile, our heuristic results are compare with the solutions achieved by the exact method proposed by Baldacci and Mingozzi (2009). In the work of Baldacci and

Table 5 Computational results on the MDVRP benchmarks

Instance	C_N	D_N	Q	H	HGA		Cordeau et al.	Baldacci and Maingozzi
					HGA_b	t_h (s)		
p01	50	4	80	∞	**576.87***	13.80	**576.87***	**576.87***
p02	50	4	160	∞	**473.53***	14.51	**473.53***	**473.53***
p03	75	5	140	∞	**640.65***	40.82	**640.65***	**640.65***
p04	100	2	100	∞	1,001.59	85.96	1,004.38	**999.21***
p05	100	2	200	∞	752.96	91.25	**751.16**	751.26
p06	100	3	100	∞	**876.50***	91.24	**876.50***	**876.50***
p07	100	4	100	∞	886.68	87.70	889.10	**881.97***
p08	249	2	500	310	4,439.29	1,101.54	**4,421.79**	–
p09	249	3	500	310	**3,888.49**	1,116.63	3,913.95	–
p10	249	4	500	310	3,666.98	1,133.34	**3,662.74**	–
p11	249	5	500	310	3,594.08	1,128.28	**3,573.06**	–
p12	80	2	60	∞	**1,318.95***	49.40	**1,318.95***	**1,318.95***
p13	80	2	60	200	**1,318.95**	49.28	**1,318.95**	–
p14	80	2	60	180	**1,360.12**	48.73	**1,360.12**	–
p15	160	4	60	∞	**2,505.42**	337.10	**2,505.42**	**2,505.42**
p16	160	4	60	200	**2,572.23**	369.20	**2,572.23**	–
p17	160	4	60	180	**2,709.09**	332.76	**2,709.09**	–
p18	240	6	60	∞	**3,702.85**	1,039.40	**3,702.85**	–
p19	240	6	60	200	**3,827.06**	1,040.72	**3,827.06**	–
p20	240	6	60	180	**4,058.07**	1,039.67	4,075.36	–
p21	360	9	60	∞	**5,474.84**	2,876.77	**5,474.84**	–
p22	360	9	60	200	5,712.10	2,853.40	**5,702.16**	–
p23	360	9	60	180	**6,078.75**	2,910.91	**6,078.75**	–
pr01	48	4	200	500	**861.32**	12.85	**861.32**	–
pr02	96	4	195	480	**1,275.61**	88.30	1,307.99	–
pr03	144	4	190	460	**1,799.93**	319.47	1,806.37	–
pr04	192	4	185	440	2,064.64	369.12	**2,055.90**	–
pr05	240	4	180	420	**2,338.57**	1,027.84	**2,338.75**	–
pr06	288	4	175	400	2,682.46	1,699.01	**2,674.02**	–
pr07	72	6	200	500	**1,073.63**	39.30	1,078.83	–
pr08	144	6	190	475	1,668.75	251.95	**1,666.63**	–
pr09	216	6	180	450	2,171.18	783.54	**2,162.40**	–
pr10	288	6	170	425	**2,824.90**	1,759.30	2,833.69	–
Average					2,430.21		2,430.77	

* Indicates the optimal solution

Mingozzi (2009), the fleet of vehicles located at each depot is unlimited, so their results can be adopted directly.

Table 5 reports the results obtained by our HGA, Cordeau's tabu search, and the exact method of Baldacci and Mingozzi. The first five columns describe the test

instance, where column '*Instance*' shows the problem name, columns 'C_N' and 'D_N' give the number of the customers and depots, columns 'Q' and 'H' show the vehicle capacity and maximum traveling distance. The following one major column displays the results of our HGA. The last two columns show best solution costs of tabu search and exact method. Note that Baldacci and Mingozzi (2009) do not consider all the instances, solving 9 out of 33 test instances. The best results among three methods are marked in bold.

Firstly, we compare two heuristic methods with the exact algorithm of Baldacci and Mingozzi (2009). The exact algorithm can solve to optimality seven out of nine instances. For these seven optimally solved instances, we find that both our HGA and Cordeau's tabu search can also find the optimal solutions. Then, we compare two heuristics. We find that Cordeau's tabu search is able to find 25 best solutions (among three methods) out of 33 instances considered, while our HGA can find 22 best solutions. The average value of HGA_b is equal to 2430.21, while the average value of Cordeau's best solution costs is 2430.77. Clearly, concerning the classical MDVRP, the performance of our HGA is satisfied and is competitive to the existing sophisticated methods.

5 Conclusions and future research

In this paper, we investigate MDOVRP, an extension of the classical MDVRP and the OVRP. An effective hybrid genetic algorithm is proposed to solve the problem. In the proposed approach, three classical heuristic algorithms are adopted to provide good initial solutions. A split method is designed for computing the exact fitness of each chromosome. Several simple and powerful local search methods are utilized to improve the offspring generated by the crossover procedure. The performance of the proposed HGA is tested on a range of test instances. Computational results indicate that the proposed HGA is a promising approach to solve the multi-depot open vehicle routing problem (MDOVRP). Meanwhile, since our algorithm is also capable of solving the classical MDVRP, computational experiments were carried out on the MDVRP benchmark instances taken from the literature. The computational results on the classical MDVRP are satisfied.

The MDOVRP studied in this paper can be extensively studied in the future. Its formulation could be extended to cover more operational constraints, such as time windows and heterogeneous fleet of vehicles. From the methodological point of view, one goal is to speed up the local search in the proposed genetic algorithm to reduce its running time.

Acknowledgments This work was supported by Research Grant from National Natural Science Foundation of China (no. 61104173). The authors wish to thank the editor and two anonymous referees who performed an extremely detailed analysis of the paper and suggested very valuable improvements. Thanks are also due to Professor Jean-François Cordeau for providing the solutions obtained using his tabu search approach, and giving valuable suggestions.

References

Baldacci R, Mingozzi A (2009) A unified exact method for solving different classes of vehicle routing problems. Math Program 120:347–380

Berger J, Barkaoui M (2004) A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. Comput Oper Res 31:2037–2053

Bolduc MC, Laporte G, Renaud J, Boctor FF (2010) A tabu search heuristic for the split delivery vehicle routing problem with production and demand calendars. Eur J Oper Res 202:122–130

Brandão J (2004) A tabu search algorithm for the open vehicle routing problem. Eur J Oper Res 157:552–564

Branke J, Schmidt C, Withopf M (2007) A fast look-ahead heuristic for the multi-depot vehicle routing problem. Wirtschaftinformatik(2) Proceedings 2007:80

Chao IM, Golden BL, Wasil E (1993) A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. Am J Math Manag Sci 13:371–406

Clarke G, Wright JW (1964) Scheduling of vehicles from a central depot to a number of delivery points. Oper Res 12:568–581

Cordeau J, Gendreau M, Laporte G (1997) A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30:105–119

Cvetkovic D, Parmee IC (2002) Preferences and their application in evolutionary multiobjective optimization. IEEE Trans Evol Comput 6:42–57

Dantzig GB, Ramser JH (1959) The truck dispatching problem. Manag Sci 6:80–91

Dueck G, Scheuer T (1990) Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. J Comput Phys 90:161–175

Fallahi AE, Prins C, Calvo R (2008) A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. Comput Oper Res 35:1725–1741

Fleszar K, Osman IH, Hindi KS (2009) A variable neighbourhood search algorithm for the open vehicle routing problem. Eur J Oper Res 195:803–809

Fu Z, Eglese R, Li L (2005) A new tabu search heuristic for the open vehicle routing problem. J Oper Res Soc 56:267–274

Gillett B, Miller L (1974) A heuristic algorithm for the vehicle-dispatch problem. Oper Res 22:340–349

Hertz A, Widmer M (2003) Guidelines for the use of meta-heuristics in combinatorial optimization. Eur J Oper Res 151:247–252

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

Holland J (1975) Adaptation in natural and artificial systems. University of Michigan Press

Jeon G, Leep HR, Shim JY (2007) A vehicle routing problem solved by using a hybrid genetic algorithm. Comput Ind Eng 53:680–692

Kara I, Laporte G, Bektas T (2004) A note on the lifted Miller–Tucker–Zemlin subtour elimination constraints for the capacitated vehicle routing problem. Eur J Oper Res 158:793–795

Ke T, Yi M, Xin Y (2009) Memetic algorithm with extended neighborhood search for capacitated arc routing problems. IEEE Trans Evol Comput 13:1151–1166

Kim KY, Park K, Ko J (2003) A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. Comput Oper Res 30:1151–1171

Lau HCW, Chan TM, Tsui WT, Pang WK (2010) Application of genetic algorithms to solve the multidepot vehicle routing problem. IEEE Trans Autom Sci Eng 7:383–392

Letchford A, Lysgaard J, Eglese R (2006) A branch-and-cut algorithm for the capacitated open vehicle routing problem. J Oper Res Soc 58:1642–1651

Li F, Golden B, Wasil E (2007) The open vehicle routing problem: algorithms, large-scale test problems, and computational results. Comput Oper Res 34:2918–2930

Lim A, Wang F (2005) Multi-depot vehicle routing problem: a one-stage approach. IEEE Trans Autom Sci Eng 2:397–402

Liu R, Jiang BZ, Liu X, Chen F (2010) Task selection and routing problems in collaborative truckload transportation. Transp Res Part E: Logist Transp Rev 46:1071–1085

Liu S, Huang W, Ma H (2009) An effective genetic algorithm for the fleet size and mix vehicle routing problems. Transp Res Part E: Logist Transp Rev 45:434–445

Mendoza JE, Castanier B, Guéret C, Medaglia AL, Velasco N (2010) A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. Comput Oper Res 37:1886–1898

Miller C, Tucker AW, Zemlin R (1960) Integer programming formulation of traveling salesman problems. J ACM 7:326–329
Oliver I, Smith D, Holland J (1987) A study of permutation crossover operators on the traveling salesman problem. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, pp 224–230
Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. Comput Oper Res 34:2403–2435
Potvin JY, Bengio S (1996) The vehicle routing problem with time windows part II: genetic search. Inf J Comput 8:165
Prins C (2004) A simple and effective evolutionary algorithm for the vehicle routing problem. Comput Oper Res 31:1985–2002
Prins C (2009) Two memetic algorithms for heterogeneous fleet vehicle routing problems. Eng Appl Artif Intell 22:916–928
Reeves C (2003) Genetic algorithms. Handbook of Metaheuristics, pp 55–82
Renaud J, Laporte G, Boctor FF (1996) A tabu search heuristic for the multi-depot vehicle routing problem. Comput Oper Res 23:229–235
Repoussis PP, Tarantilis CD, Bräysy O, Ioannou G (2010) A hybrid evolution strategy for the open vehicle routing problem. Comput Oper Res 37:443–455
Repoussis PP, Tarantilis CD, Ioannou G (2006) The open vehicle routing problem with time windows. J Oper Res Soc 58:355–367
Russell R, Chiang WC, Zepeda D (2008) Integrating multi-product production and distribution in newspaper logistics. Comput Oper Res 35:1576–1588
Sörensen K, Sevaux M (2006) MA|PM: memetic algorithms with population management. Comput Oper Res 33:1214–1225
Salari M, Toth P, Tramontani A (2010) An ILP improvement procedure for the open vehicle routing problem. Comput Oper Res 37:2106–2120
Salhi S, Sari M (1997) A multi-level composite heuristic for the multi-depot vehicle fleet mix problem. Eur J Oper Res 103:95–112
Sariklis D, Powell S (2000) A heuristic method for the open vehicle routing problem. J Oper Res Soc 51:564–573
Tarantilis CD, Kiranoudis CT (2002a). A list-based threshold accepting method for job shop scheduling problems. Int J Prod Econ 77:159–171
Tarantilis CD, Kiranoudis CT (2002b). Distribution of fresh meat. J Food Eng 51:85–91
Tarantilis CD, Ioannou G, Kiranoudis CT, Prastacos GP (2005) Solving the open vehicle routeing problem via a single parameter metaheuristic algorithm. J Oper Res Soc 56:588–596
Zachariadis EE, Kiranoudis CT (2010) An open vehicle routing problem metaheuristic for examining wide solution neighborhoods. Comput Oper Res 37:712–723