Computers & Operations Research 37 (2010) 950–959

Contents lists available at ScienceDirect

Computers & Operations Research

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

Two-phase heuristic algorithms for full truckloads multi-depot capacitated vehicle routing problem in carrier collaboration

Ran Liu[a], Zhibin Jiang[a,*], Richard Y.K. Fung[b], Feng Chen[a], Xiao Liu[a]

[a]Department of Industrial Engineering and Logistics Management, School of Mechanical Engineering, Shanghai Jiao Tong University, 800 Dong Chuan Rd., Shanghai 200240, PR China
[b]Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong

ARTICLE INFO

Available online 18 August 2009

Keywords:
Collaborative transportation
Multi-depot
Full truckloads
Lower bound
Heuristic

ABSTRACT

Collaborative transportation, as an emerging new mode, represents one of the major developing trends of transportation systems. Focusing on the full truckloads multi-depot capacitated vehicle routing problem in carrier collaboration, this paper proposes a mathematical programming model and its corresponding graph theory model, with the objective of minimizing empty vehicle movements. A two-phase greedy algorithm is given to solve practical large-scale problems. In the first phase, a set of directed cycles is created to fulfil the transportation orders. In the second phase, chains that are composed of cycles are generated. Furthermore, a set of local search strategies is put forward to improve the initial results. To evaluate the performance of the proposed algorithms, two lower bounds are developed. Finally, computational experiments on various randomly generated problems are conducted. The results show that the proposed methods are effective and the algorithms can provide reasonable solutions within an acceptable computational time.

1. Introduction

Nowadays, shippers and carriers are looking for ways to operate more efficiently in response to ever stricter customer demands, surging costs of fuel and carrier insurance, and more intensive market competition in the transportation industry. Shippers and carriers have developed various strategies to improve the efficiency of their internal operations with focus on reducing individual operating costs. However, more opportunities exist for increasing overall profit through collaboration among shippers and carriers, since asset repositioning is very common in the transportation industry. Asset repositioning is empty movement from a delivery location to a pickup location. It is reported that in the USA, about 18% of daily truck movements are empty [1]. If asset repositioning could be reduced, the total transportation cost will decrease. Hence, some companies have adopted a new transportation model called collaborative transportation (CT), which brings together all logistics participants to improve the overall performance of transportation planning and scheduling.

CT can be classified in two ways, i.e., as the collaboration among shippers and that among carriers. When shippers consider collaborating, their goal is to identify sets of lanes that can be submitted to a carrier as a bundle, in the hope that this results in more favorable rates. A lane corresponds to a shipment delivery from an origin to a destination with one full truckload. The truckload shipper collaboration problem is formulated as the lane cover problem (LCP), i.e., covering a set of lanes with a set of cycles of minimum cost. It is proven that the distance constrained lane covering problem (DCLCP) and the cardinality constrained lane covering problem (CCLCP) are NP-hard and thus a greedy algorithm is proposed for solving the CCLCP [1].

Compared to shipper collaboration, carrier collaboration has received less research attention. At present, most carriers collect freight requests from shippers and then optimize the vehicle routing individually. However, carriers may also benefit from the collaboration if they form an organizational system to reduce overall system-wide costs and thus increase each partner's profit.

In this paper, we tackle a special optimization problem in the carrier collaboration system called the multi-depot capacitated arc routing problem with full truckloads (MDCARPFL). We may assume that the collaborative carriers receive a set of transportation orders of specific load (maybe either smaller or larger than the vehicle capacity), the pickup location (i.e., the origin) and the delivery location (i.e., the destination). If each carrier provides only full truckload transport between different locations, the carriers are required to fulfil the orders at minimal cost, using a fleet of vehicles located at several depots. The problem can thus be formulated as the MDCARPFL and

* Corresponding author. Tel./fax: +86 21 34206065.
E-mail address: zbjiang@sjtu.edu.cn (Z. Jiang).

0305-0548/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2009.08.002

has significant applications in carrier collaboration. However, solving the MDCARPFL for minimizing asset repositioning in a logistics network is not easy. Existing exact methods, as proposed by Arunapuram et al. [2], can solve relatively simple problems optimally. For large scale problem instances, as typically found in carrier collaboration, it is not practicable to find the optimal solution. Therefore, powerful heuristic algorithms should be established to tackle real-world larger-scale instances.

The rest of this paper is organized as follows. Section 2 introduces the relevant literature. A mathematical model for the MDCARPFL is developed in Section 3. Section 4 puts forward two lower bounds for the problem. Section 5 proposes the heuristic algorithms for solving the problem. Computational experiments are described in Section 6. Finally, conclusions and future work are given in Section 7.

2. Literature review

The MDCARPFL is a variant of the capacitated arc routing problem (CARP), introduced by Golden and Wong [3]. The CARP is probably the most significant problem in the area of arc routing, since it arises naturally in a number of practical contexts [4–6]. The CARP consists of finding a set of vehicle routes of minimum cost, such that every required edge is serviced by one vehicle, each route starts and ends at the depot and the total demand serviced by a route does not exceed the vehicle capacity. The CARP is NP-hard. Golden and Wong [3] showed that even 1.5-approximation for the CARP is also NP-hard. Exact methods for the CARP have only been able to solve small problems to optimality [7–10]. To solve problems of a realistic size, researchers have resorted to heuristic algorithms.

Dror [11] discusses different heuristics for the CARP, including simple constructive heuristics and some metaheuristic algorithms. Hertz and Mittaz [12] give an adaptation of variable neighborhood search (VNS) to the CARP. Muyldermans et al. [13] develop 2-opt and 3-opt local search algorithms for the arc and general routing problems. Two forms of the 2-opt and 3-opt approaches are applied to the problems, which are simpler than the methods developed by Hertz et al. [14]. Based on these procedures, Beullens et al. [15] introduce a guided local search heuristic for the CARP. Experiments on standard benchmark problems and the newly developed instances indicate that the algorithm is capable of finding optimal or near-optimal solutions within a limited computation time. Lacomme et al. [16] present some powerful memetic algorithms for the CARP. Based on the tabu search algorithm, Eglese and Li [17], Brandão and Eglese [18] and Greistorfer [19] present the algorithms for the CARP. Brandão and Eglese [18] propose a deterministic algorithm that does not require the use of random parameter values, so that the results are fully reproducible, while Greistorfer [19] makes use of the scatter search. Ergun et al. [1] propose a greedy algorithm for the shipper collaboration problem, which is similar to the CARP, but ignores the constraint that each vehicle tour must start and finish at a designated depot.

Contrary to the CARP, the multiple depots CARP (MDCARP) has received relatively less attention. Eglese [20] presents a two-stage solution procedure for the MDCARP. In the first stage, an Eulerian graph is partitioned into small cycles, and cycles are aggregated into routes using a greedy saving heuristic. In the second stage, a simulated annealing algorithm is adopted to improve the solution. Amberg et al. [21] investigate a two-phase algorithm for the MDCARP. First, the problem of finding the specific routes is modeled as the capacitated minimum spanning tree problem and a heuristic is applied to yield initial solutions. Second, two metaheuristics are applied to get better routes.

To the best of our knowledge, only Arunapuram et al. [2] have presented an exact algorithm for solving the MDCARPFL so far. They introduce a column generation method that takes advantage of the special structure of the linear programming sub-problems at the nodes of the branch-and-bound tree. However, the exact algorithm is unable to solve the instances when the number of lanes exceeds 200.

Although many heuristics have been proposed for the CARP, they cannot be applied directly because of three major differences between the CARP and MDCARPFL, which have not been considered together in existing literature. First, in the CARP each vehicle starts and returns to one designated depot, while in the MDCARPFL vehicles are located at several depots. Second, in the CARP each order only has one full-truck demand, while the MDCARPFL allows each order demand to be any integer multiple of full truckloads. Finally, in the MDCARPFL all the orders are directed, while in the CARP transportation orders are undirected. So far little research has been done on this important and complex problem. Arunapuram et al. [2] introduce a complicated exact method for the MDCARPFL, which can only solve small-scale problems. To tackle large-scale instances, efficient heuristics are required.

3. Problem formulation

The MDCARPFL studied in this paper can be described formally as follows. A fleet of vehicles is located at several depots, and a number of transportation orders need to be served by the vehicles. Each vehicle must start and end at the same depot. The objective of the problem is to determine the tours for the vehicles that serve all the orders and minimize the total shipping costs. The underlying assumptions of the model are given as follows:

(1) All orders are known in advance and kept unchanged during the transportation process.
(2) There is no transportation order between the depot and the customer node.
(3) All the vehicles are identical. The vehicle capacity is Q.
(4) The vehicle shipping costs are equivalent to the travel distance.
(5) A vehicle's tour must not exceed a given distance span H. This restriction ensures that the maintenance intervals for vehicles are respected. For some problems the restriction is that a tour must not exceed a given time span, which can be dealt with analogously.
(6) The vehicles only provide full truckload transportation. That means the goods are transported directly from the pickup location to the delivery location. If an order has the demand of q, the serving number to this order is $\lceil q/Q \rceil$ (the smallest integer no less than q/Q).

To clearly describe the MDCARPFL, an example with 2 depots and 10 orders is shown in Fig. 1, where the dashed lines represent the empty vehicle movements, and the solid lines correspond to the transportation lanes. Among all orders, order (A, B) and order(C, D) are represented as two separate lanes, since each of them has two full truckloads. Therefore, there are 12 lanes in Fig. 1. All the lanes are covered by three tours, i.e., Tour 1, Tour 2 and Tour 3. Each tour is assigned to one vehicle, starting from a depot, serving a number of lanes and finally returning to the departing depot. One order may be served by one or several vehicles in this problem. For example, order (A, B) is transformed into two lanes and served by one vehicle, i.e., vehicle 3. Order (C, D) is also expressed as two lanes, but they are covered by Tour 1 and Tour 2 separately, which means vehicle 1 and vehicle 2 fulfil order (C, D) jointly.

The exact formulation for the MDCARPFL is given as follows.

Parameters:

D	*the set of depots*
V	*the set of vehicles*

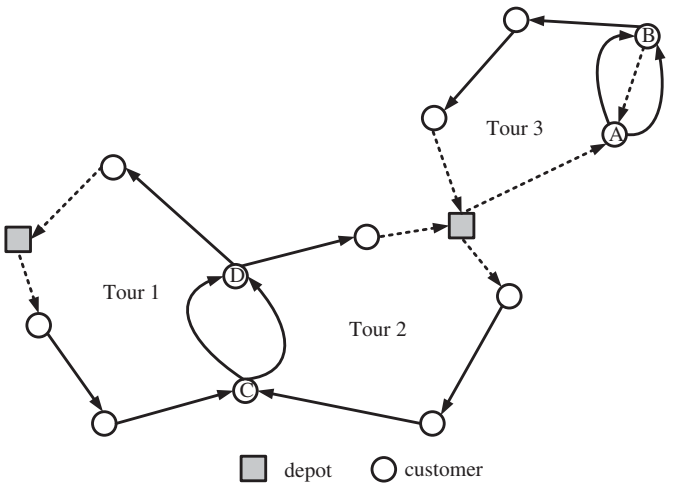

Fig. 1. MDCARPFL example.

P^+ the set of pickup locations
P^- the set of delivery locations
P the set of $P^+ \cup P^-$
q_{ij} the non-negative service requirement that transported from origin i to destination j, $\forall i \in P^+, j \in P^-$
c_{ij} the distance from location i to location j, $\forall i,j \in P \cup D$

Decision variables:

x_{ij}^v the number of trips from location i to location j performed by vehicle v

Objective:

$$\min \sum_{v \in V} \sum_{i \in P \cup D} \sum_{j \in P \cup D} c_{ij} x_{ij}^v \tag{1}$$

Subject to:

$$\sum_{v \in V} x_{ij}^v \geq \left\lceil \frac{q_{ij}}{Q} \right\rceil \quad \forall i \in P^+,\ j \in P^- \tag{2}$$

$$\sum_{j \in P \cup D} x_{ij}^v - \sum_{j \in P \cup D} x_{ji}^v = 0 \quad \forall v \in V,\ i \in P \cup D \tag{3}$$

$$\sum_{i \in D} \sum_{j \in P \cup D} x_{ij}^v \leq 1 \quad v \in V \tag{4}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij}^v - M \sum_{i \in S} \sum_{j \in (P \cup D) \setminus S} x_{ij}^v \leq 0 \quad v \in V,\ S \subseteq P \tag{5}$$

$$\sum_{i \in P \cup D} \sum_{j \in P \cup D} c_{ij} x_{ij}^v \leq H \quad v \in V \tag{6}$$

$$x_{ij}^v \geq 0 \text{ and integer} \quad \forall i,j \in P \cup D, v \in V \tag{7}$$

The objective (1) is to minimize the sum of vehicle travel distances. Constraints (2) ensure that all the orders must be satisfied by the vehicles. Note that an order between one pair of locations may be served either by various vehicles or by one vehicle for many times. Constraints (3) specify the flow balance. If a vehicle enters any location (i.e., a depot, a pickup location or a delivery location), it has to leave this location. Constraints (4) state that a vehicle starts from only one depot. Since constraints (3) ensure the balance equation, constraints (4) also imply that each vehicle tour must start and end at the same depot. Constraints (5), the so-called *isolated subtour elimination* constraints, impose that the solution dose not contain any illegal isolated subtour, where M is a 'sufficient large' positive number. Constraints (5) stipulate that each cut $(P \cup D \setminus S, S)$ defined by a vertex set S, is crossed by at least one transportation lane. In this case, isolated subtour elimination constraints are different from famous subtour elimination constraints in the TSP [22] and the basic VRP [23]. In the TSP and the VRP, the subtour elimination constraints ensure that there is no subtour in the solution. However, a subtour is legal if it is part of a vehicle tour in our problem. Constraints (6) imply that the distance span of the tour is respected.

Since the proposed heuristic algorithms for the MDCARPFL are based on the graph theory, a detailed description of the MDCARPFL, as the graph theoretic problem, is provided as follows. Given a complete directed Euclidean graph $G = (N,A)$ with vertex set N, arc set A, and order set $A' \subseteq A$, each $a \in A'$ represents one transportation order and is associated with a non-negative load demand q_a, which can be transferred to the arc covered time $\lceil q_a/Q \rceil$ according to the full truckload transport. Let $\hbar$ represent the set of directed closed chains in G. Each $r \in \hbar$ satisfies $r \cap A' \neq \varnothing$, contains a depot, and covers arc a r_a times. Let l_r denote the distance of chain r. x_r is a 0–1 variable indicating whether chain r belongs to the optimal solution or not. The problem can be formulated as follows.

Objective:

$$\min \sum_{r \in \hbar} l_r x_r \tag{8}$$

Subject to:

$$\sum_{r \in \hbar} r_a x_r \geq \left\lceil \frac{q_a}{Q} \right\rceil \quad \forall a \in A' \tag{9}$$

$$l_r \leq H \quad \forall r \in \hbar \tag{10}$$

$$x_r = \begin{cases} 1, & \textit{chain } r \textit{ is selected} \\ 0, & \textit{else} \end{cases} \quad \forall r \in \hbar \tag{11}$$

In the above formulation, objective (8) minimizes the total distance of all the vehicle tours. Constraints (9) clarify that each order must be served by one or more vehicles. Constraints (10) require that the distance of a tour is no longer than the distance span.

4. Lower bounds for the problem

As stated above, the MDCARPFL is a variant of the CARP and even harder than the basic CARP. There is no efficient exact algorithm for solving the MDCARPFL with large problem sizes. If there are hundreds of the customer nodes, orders and lanes in the MDCARPFL, it cannot be solved by commercial IP solver software (e.g., Cplex 10.0) due to excessive memory requirements or excessive running time requirements. Therefore, heuristic algorithms are used for tackling the MDCARPFL. A tight lower bound is essential for evaluating the quality of the heuristic solutions. In this section, two lower bounds, LB1 and LB2, are proposed for the MDCARPFL. LB1 is proposed by Gronalt et al. [24] for the pickup and delivery problem with full truckloads. It is formulated as simple network flow LP, and can be adopted for the MDCARPFL directly. However, LB1 ignores the travel distances between the customer locations and the depots. Lifting LB1 into LB2 is done by considering the travel distances between the depots and customer locations. Therefore, LB2 always gives results that are no worse than the results from LB1. The computational

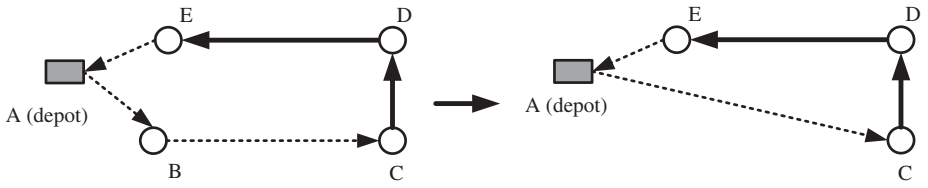

Fig. 2. The first visited node must be the pickup node.

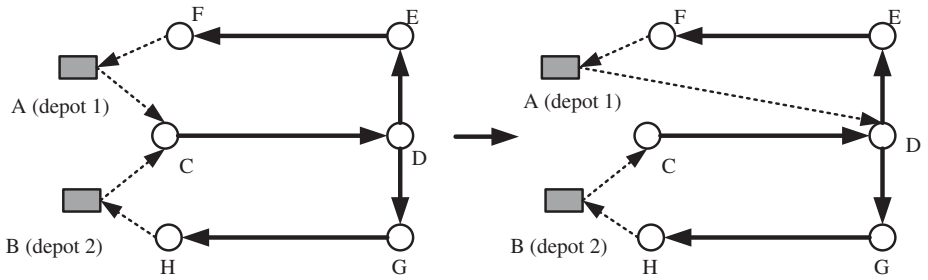

Fig. 3. The number of visits to each node is constrained.

experiments that we present in Section 6 show that in many cases LB2 are clearly superior to LB1.

4.1. Lower bound LB1

Lower bound LB1 is formulated as follows. First constraints (4)–(6) in the IP model formulated above are relaxed. Let x_{ij} indicates total number of movements (loaded and empty) from location i to location j. The revised objective function and the constraints can be reformulated as the following LP.

Objective

$$\min \sum_{i\in P}\sum_{j\in P} c_{ij}x_{ij} \tag{12}$$

Subject to:

$$x_{ij} \geq \left\lceil \frac{q_{ij}}{Q} \right\rceil \quad \forall i \in P^+,\ j \in P^- \tag{13}$$

$$\sum_{j\in P} x_{ij} = \sum_{j\in P} x_{ji} \quad \forall i \in P \tag{14}$$

$$x_{ij} \geq 0 \quad \forall i,j \in P \tag{15}$$

The objective (12) minimizes the total vehicle travel distance, both load and empty. Constraints (13) imply that all the transportation requirements must be met. Constraints (14) ensure that at each location, the number of incoming vehicle movements equals the number of outgoing vehicle movements. Constraints (15) ensure that variable x_{ij} is non-negative.

4.2. Lower bound LB2

Before formulating LB2, some characteristics of the MDCARPFL solution were analyzed. As shown in the left-hand part of Fig. 2, the closed chain (A, B, C, D, E, A) represents a vehicle tour. Node B is visited immediately after vehicle has started from depot A, but node B is not a pickup node. The sum of the distances of arc (A, B) and arc (B, C) is greater than the distance of arc (A, C). We can lift the original chain into chain (A, C, D, E, A), as illustrated in the right-hand part of Fig. 2. Therefore, it can be concluded that a vehicle must first visit a pickup node after starting from the depot. Similarly, the last node visited by a vehicle before returning to the depot must be a delivery node.

If a pickup node is chosen as the first node after the vehicle has started from the depot, the number of the tours passing this pickup node must satisfy specific constraint. As shown in the left-hand part of Fig. 3, there are two closed chains, each of them representing a tour. The first one is chain1 (A, C, D, E, F, A) and the second is chain2 (B, C, D, G, H, B). In each chain, the vehicle first visits node C after starting from the depot. But there is only one lane originating from node C. After lane (C, D) has been covered by chain2, node C is 'not' a pickup node in chain1. Therefore, chain1 should be lifted into chain3 (A, D, E, F, A), as shown in the right-hand part of Fig. 3.

LB2 is formulated based on the following parameters and variables:

V_n the lower bound to the minimum number of vehicles needed to serve all the orders.

x_{ij}^d the number of tours in the solution, each of which satisfies that location i is first visited after vehicle starting from depot d, and location j is last visited before vehicle returning to depot d.

y_{ij} the number of trips from location i to location j.

The final IP formulation of LB2 can be written as:

Objective:

$$\min \sum_{d\in D}\sum_{i\in P^+}\sum_{j\in P^-}(c_{di}+c_{jd})x_{ij}^d + \sum_{i\in P}\sum_{j\in P} c_{ij}y_{ij} \tag{16}$$

Subject to:

$$\sum_{d\in D}\sum_{i\in P^+}\sum_{j\in P^-} x_{ij}^d \geq V_n \tag{17}$$

$$\sum_{d\in D}\sum_{j\in P^-} x_{ij}^d \leq \sum_{j\in P^-}\left\lceil \frac{q_{ij}}{Q} \right\rceil \quad \forall i \in P^+ \tag{18}$$

$$\sum_{d\in D}\sum_{i\in P^+} x_{ij}^d \le \sum_{i\in P^+}\left\lceil \frac{q_{ij}}{Q}\right\rceil \quad \forall j \in P^- \tag{19}$$

$$y_{ij} \ge \left\lceil \frac{q_{ij}}{Q}\right\rceil \quad \forall i \in P^+,\ j \in P^- \tag{20}$$

$$\sum_{d\in D}\sum_{j\in P} x_{ij}^d + \sum_{j\in P} y_{ji} = \sum_{d\in D}\sum_{j\in P} x_{ji}^d + \sum_{j\in P} y_{ij} \quad \forall i \in P \tag{21}$$

$$y_{ij} \ge 0 \text{ and integer} \quad \forall i \in P,\ j \in P \tag{22}$$

$$x_{ij}^d \ge 0 \text{ and integer} \quad \forall i \in P^+,\ j \in P^-,\ d \in D \tag{23}$$

In objective (16), the first term represents the travel distances between the depots and the customer locations; while the second term implies the travel distances between the customer locations. The ultimate objective is to minimize the sum of vehicle travel distances. Constraints (17) account for the availability of tours whereas the number of connections between the depots and the customer locations is bounded by a minimum. Constraints (18) and (19) are proposed according to the characteristics of the solution that are mentioned above. Constraints (20) ensure that all the orders are served. Constraints (21) specify the flow balance in the network.

While tackling LB2 for the proposed problem, V_n can be computed as $V_n = \lceil LB1/H \rceil$. Note that LB2 can be improved if a better value of V_n can be found. Detailed discussion about the value of V_n will be given in Section 6.2.

5. Solution approach

In this section, a two-phase heuristic method is introduced to approach the MDCARPFL effectively and efficiently. In the first phase, a set of cycles is created to cover all the lanes. In the second phase, close chains are constructed, each of which corresponds to a vehicle tour. Finally, a set of local search approaches is adopted to improve the initial solution.

5.1. The first phase: cycle construction

Ergun et al. [1] propose a simple greedy algorithm, i.e., 'generating cycles first, choosing cycles second' approach, for the CCLCP. First, the directed cycles of cardinality less than or equal to a prespecified number k are generated. Second, in each iteration a cycle is chosen that maximizes the 'cover factor' of a cycle, i.e., the ratio of the distance of the lanes covered by the cycle and the total distance of the cycle. The main differences between this approach and our problem lie in two main facts. As in the CCLCP, there is no restriction on the maximum distance of a cycle, but the cardinality of a cycle must not exceed a prespecified number. Furthermore, the CCLCP assumes one truckload demand for each order, whereas our problem allows each order demand to be any integer multiple of full truckloads. Therefore, we extend this approach with respect to the two points. First, the distance span constraint is checked when a cycle is chosen. The infeasible cycle is rejected. In addition, the algorithm records the 'covering number' CN for each order, when a cycle is chosen to cover the orders. For each order a with load demand q_a, its initial covering number CN_a equals $\lceil q_a/Q \rceil$. When order a is covered by a cycle in the iteration, CN_a decreases by 1. If CN_a is greater than 0, order a is valid and needs to be covered by cycles. Otherwise, it is invalid. The improved greedy algorithm can be presented as follows.

First phase Algorithm: cycle construction

```
Input parameter k
generate cycle set C, which represents the set of all directed cycles
  in graph G of cardinality less than or equal to k
U := A'
for each ∈ A'
    CN_a = ⌈q_a/Q⌉
end for
C_choice = ∅, where C_choice is the set of cycles chosen to cover the
  orders
repeat choose one cycle c ∈ C
    calculate the total distance of c: l_c
    join cycle c to the depot which is closest to cycle c, to generate
      a tour.
    improve the tour by applying the refining procedure described
      in Section 5.3.1. let t_c denote the distance of the tour
    if t_c ≤ H then
        calculate the 'cover factor' of the cycles. The cover factor of
      cycle c is: ξc = Σ_{e∈c∩U} l_e/l_c, where l_e is the distance of arc e
    else
        remove cycle c from C
    end if
until all ∈ C have been chosen
repeat choose c_max ∈ C, which has the maximal cover factor
    C_choice := C_choice ∪ c_max
    repeat choose an arc a ∈ c_max ∩ U
        CN_a = CN_a − 1
        if CN_a= = 0 then
            U := U\a
        end if
    until all arcs in max ∩ U have been chosen
    repeat choose cycle c ∈ C
        calculate and update the cover factor ξc
    until all ∈ C have been chosen
until =∅
output C_choice
```

Note that the more cycles generated, the better the quality of the solution that may be obtained. However, with the increase of the cycle cardinality, the number of cycles becomes prohibitively large very quickly, especially for large-scale instances. In such situations, it is dramatically time consuming to generate all possible cycles. Therefore, we have to make a trade-off between generating cycles and improving solution quality. Ergun et al. [1] find that the greedy algorithm solution is close to the optimal solution of the CCLCP when cycle cardinality equals 5, but the gap is unknown for the DCLCP. In our algorithm, we also try to restrict the cardinality of a cycle to be at most 5.

Another important parameter, the maximum number of repositioning arcs in a cycle, should be considered in the algorithm. We ignore this constraint in order to get more accurate solutions for the problem, i.e., the maximum number of repositioning arcs in a cycle is at most 2.

5.2. The second phase: closed chain construction

In the second phase, a set of closed chains is constructed to cover the cycles. Each closed chain corresponds to a vehicle tour. Note that the closed chains are generated based on the cycles rather than individual arcs. This may lead to suboptimal solutions. However, it has the advantage of retaining the structure of the cycles and

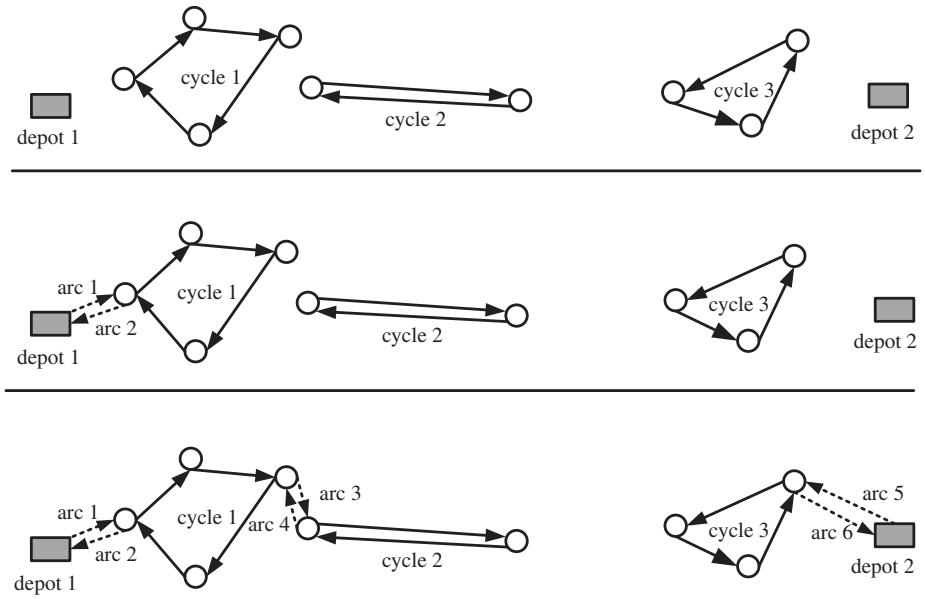

Fig. 4. The closed chain constructing procedure.

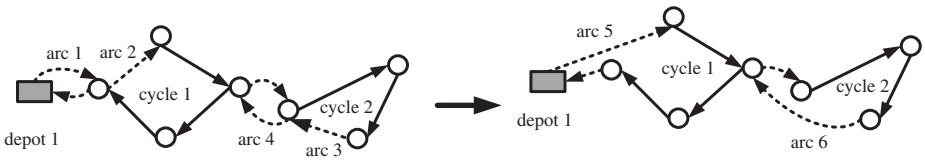

Fig. 5. The example of repositioning arcs merging.

reducing the size of the problem. The closed chain construction algorithm used in this paper is described as follows.

First, when all the cycles are unassigned (if a cycle is connected to a depot, it is called assigned), we calculate the distances between the cycles and the depots. The distance between a cycle and a depot is the shortest distance between the cycle's nodes and the depot. The first closed chain is constructed by adding two arcs between the closest depot and cycle. Then, the distances between the depots and unassigned cycles, and the distances between the assigned cycles and unassigned ones are calculated. The distance between two cycles equals the shortest distance between two nodes, each of which belongs to one cycle. An unassigned cycle is assigned to an existing chain, if the minimal distance is between them and the distance of the new chain is no more than vehicle travel distance span. Otherwise, if the minimal distance is between a depot and an unassigned cycle, a new chain is generated, i.e., two arcs are added to connected them together. The procedure is repeated until all cycles are assigned.

To describe the chain construction procedure more clearly, an example with two depots and three cycles is considered, as illustrated in Fig. 4. When all the cycles are unassigned (as shown in the top part of Fig. 4), the shortest distance between the depots and the cycles is (depot1, cycle1). A new chain is created, which consists of cycle1, depot1, arc1 and arc2 (as shown in the middle part of Fig. 4). Then, the shortest distance between the depots and unassigned cycles is (depot2, dycle3). The shortest distance between the assigned and unassigned cycles is (cycle1, cycle2). Since the former is less than the latter, arc3 and arc4 are added between cycle1 and cycle2. And last, the second chain is generated to cover cycle3, which is composed of cycle3, arc5, arc6 and depot2 (as shown in the bottom part of Fig. 4).

5.3. Improvement strategies

In general, the quality of the solutions obtained with this approach is good, but in most cases can be further improved.

5.3.1. Repositioning arcs merge

We find that two repositioning arcs may be connected together in the solution obtained by the initial solution step. Because of the triangle inequality, such two repositioning arcs should be replaced by one shorter repositioning arc. An example of repositioning arcs merge is shown in Fig. 5. Cycle1 and cycle2 are generated in the first phase of the heuristic, where arc2 and arc3 are the repositioning arcs in the cycles. Arc1 and arc4 are added in the second phase. In the merge procedure, arc1 and arc2 are deleted and replaced by arc5. Similarly, arc3 and arc4 are replaced by arc6.

5.3.2. Improved methods based on local search

Some classical local search methods are developed for VRP and ARP, which modify the heuristic initial solution and get a better result. Most iterative improvement methods applied to VRP and ARP are edge exchanging [13–15,25]. For example, the famous λ-opt heuristic for VRP removes λ edges from the tour, and reconnects the λ remaining segments in all possible ways. The procedure stops at a local minimum when no further improvements can be obtained. Since in the proposed heuristic the basic components of the chains are cycles, it is a natural idea to obtain neighboring solutions by swapping or shifting cycles. Since the number of the cycles chosen in the first phase is much less than the number of arcs in the solution, dealing with the cycles is more quickly than dealing with the arcs. Two types of neighborhoods (i.e., intra-route and inter-route cycle neighborhoods) are adopted to improve the initial solutions.

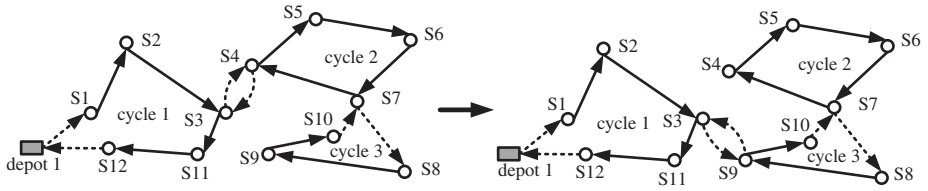

Fig. 6. The cycle rearrangement operator within one chain.

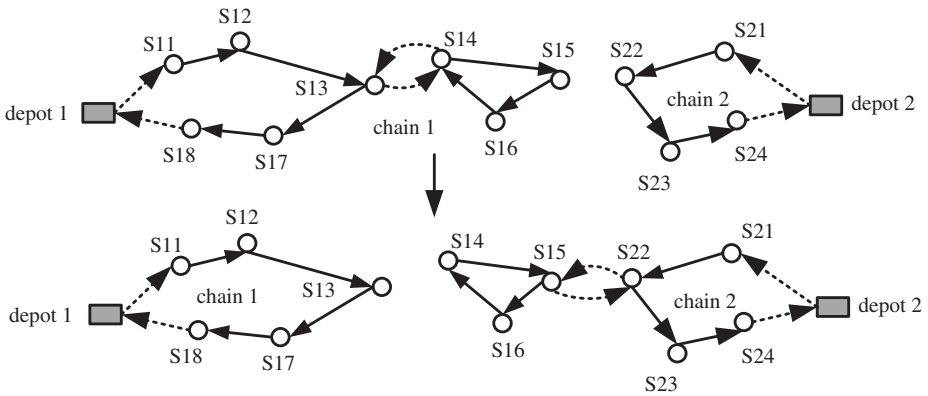

Fig. 7. The cycle shift operator between two chains.

The first local search procedures operate on each vehicle tour separately. We try to exchange the positions of two cycles within one chain, and take a cycle from its current position and insert it into another position of the chain. The whole neighborhood of an initial solution is explored. Infeasible or inferior solutions are discarded in the evaluation. While one better solution is found, it is adopted as the new seed solution for repeating the search procedure. The procedure stops when no additional improvements can be obtained. Fig. 6 illustrates an example of moving one cycle within a chain. Before the improvement operation, the cycles' sequence is cycle1, cycle2 and cycle3. After the rearrangement, the cycles' sequence is cycle1, cycle3 and cycle2.

The inter-route improvement methods simultaneously operate on two vehicle tours. We move one cycle from its current chain and insert it to any possible positions in another chain, and swap two cycles between two chains. The search process terminates when no improved solution can be obtained. Fig. 7 shows an example of shifting a cycle from chain1 to chain2. Before the shifting operation, chain1 consists of 9 nodes, i.e., depots 1 and nodes s11–18. Chain2 consists of 5 nodes, i.e., depot2 and nodes s21–24. After the shifting operation, nodes s14–16 are moved from chain1 to chain2.

6. Computational experiments

6.1. Experimental data

The proposed heuristic is assessed on a number of test problems in complete Euclidean graphs to analyze its performance. Nine parameters are considered while the problems are created:

(1) The region where customer nodes are located
(2) The number of customer nodes
(3) The number of depots
(4) The number of orders
(5) The number of lanes
(6) The spread types of customer nodes
(7) The distance span of vehicle tour
(8) The percentage p of 'long orders'
(9) The distance between depot and its closest customer node

Each test problem is generated over a rectangle region, where the customer nodes are located. The straight-line distance is adopted as the cost and distance between two locations. The test problems are divided into two sets with respect to the geographical data of the customer nodes. For the first set problems, i.e., problems C1 to C6, clusters are introduced to represent metropolitan areas or other geographical clusters of points. The clusters are randomly located in the rectangle region, each of which is a circle with radius equals 1. All the customer nodes are located in the clusters equally. For the second set problems, i.e., problems U1 to U3, the customer nodes are randomly located in the rectangle region.

For each problem, order set and lane set are created in three steps. First, the order set is created. Each order is specified by a pickup node and a delivery node, both chosen from the customer nodes. For first set problems, an order is called 'long order', if its pickup node and delivery node belong to various clusters. Otherwise, it is called 'short order'. We use a long percentage p, i.e., randomly select p% orders from the set of long orders and the rest from the set of short orders. For second set problems, each order is specified by two randomly chosen customer nodes. Second, we let each order's truckload shipment equals 1. Now the number of orders equals the number of lanes. And last, we randomly choose an order and increase its truckload shipment by 1, until the number of lanes reaches the stopping condition.

Concerning the depot locations, the test problems are grouped into two categories. For problems C5, C6 and U3, all the depots are outside the rectangle and are far from the customer nodes. The distance between the depot and its closest customer node is 20. For the rest problems, the depots are located inside the rectangle and are close to the customer nodes. The distance between each depot and its closest customer node is 5. Problems C2 and C5 comprise the same structure of orders and lanes, but different depot locations.

Table 1
Basic input parameters of nine sets of test problems.

Problem	Node	Depot	Order	Lane	Cluster	L–P	Min-D-N	Region	H_{min}
C1	100	3	50	100	4	80%	5	30×30	67.1
C2	200	4	150	200	8	50%	5	40×40	127.6
C3	200	4	300	500	8	50%	5	40×40	127.9
C4	500	6	800	1000	25	80%	5	90×90	185.7
C5	200	4	150	200	8	50%	20	40×40	166.2
C6	500	6	800	1000	25	80%	20	90×90	231.2
U1	200	3	100	200	–	–	5	10×20	55.3
U2	200	3	300	500	–	–	5	10×20	56.6
U3	200	3	300	500	–	–	10	10×20	78.9

Table 2
Computational results for the instances in problem C1.

H	Cycles	Chains	L_1	T_1 (s)	L_2	T_2 (s)	LB	Gap (%)
1000	42	3	2556.1	0.00	2498.7	0.00	2480.2	0.75
800	42	4	2568.5	0.08	2505.9	0.00	2488.3	0.71
600	42	7	2578.6	0.11	2515.8	0.06	2496.5	0.77
400	42	7	2598.9	0.12	2541.2	0.07	2513.4	1.11
300	42	10	2628.8	0.17	2558.5	0.08	2530.7	1.10
200	42	15	2675.4	0.17	2611.6	0.13	2565.8	1.79
150	42	21	2726.4	0.22	2660.8	0.09	2610.9	1.91
100	42	38	2890.9	0.31	2827.0	0.09	2697.2	4.81
80	45	44	2940.7	0.37	2883.0	0.09	2767.4	4.18
70	56	56	3301.8	0.20	3237.1	0.10	2828.8	14.43
Average				0.18		0.07		3.15

The same principles apply to C4 and C6, U2 and U3.

The test problem generation is detailed in Table 1. Columns 1–9 indicate the test problem, the number of customer nodes, the number of depots, the number of orders, the number of lanes, the number of clusters, the percentage of 'long orders' (L–P), the distance between the depot to its closest customer nodes (Min-D-N) and the rectangle region, respectively.

Ten instances are generated within each problem. All the instances within a problem have the same underlying graph, i.e., the structure of orders, lanes and depots. The only difference comes from the distance span H. For notational convenience, an instance within a problem is named as the problem's label with its distance span. For example, an instance in problem C1 with distance span 100 is denoted by C1–100. Note that for each problem, distance span must be no less than a minimal value H_{min}, or there is no feasible solution to the problem. The minimal distance span H_{min} can be found as follows. The minimal chain that covers a lane and passes one depot is a triangle, whose vertices are the depot node, the pickup node and delivery node of the lane. For each problem, we first compare the triangles determined by one lane with different depots, and get the 'smallest triangle' determined by this lane. Then, among the set of 'smallest triangles' determined by all the lanes, the distance of the largest one is H_{min}. In Table 1, the last column presents the minimal distance span H_{min} for each problem.

6.2. Results analysis

The algorithms were implemented in ANSI C and tested on an Intel P4 3 GHz PC with 2 GB memory, under Windows XP. The computational results are presented in Tables 2–10.

Tables 2–7 show results obtained by the proposed algorithms and LB2s for 60 test instances. Column 1 identifies the travel distance span H. Columns 2 and 3 display the number of cycles and the number of chains generated in the first and the second phase of the heuristic, respectively. Columns 4 and 5 reflect the solution cost provided by the two-phase heuristic and the corresponding running

Table 3
Computational results for the instances in problem C2.

H	Cycles	Chains	L_1	T_1 (s)	L_2	T_2 (s)	LB	Gap (%)
3000	84	3	7182.9	0.56	7031.2	0.75	6909.7	1.76
2000	84	4	7198.1	0.69	7044.0	1.90	6914.4	1.87
1000	84	6	7310.6	0.81	7111.3	5.57	6930.2	2.61
800	84	10	7376.7	0.82	7150.6	2.05	6942.8	2.99
600	84	13	7385.4	1.02	7152.9	7.94	6967.5	3.67
400	84	20	7585.4	1.41	7326.3	0.72	7028.7	4.23
300	84	27	7688.5	1.78	7422.1	4.52	7092.3	4.65
200	84	47	8128.2	2.63	7725.5	2.99	7242.1	6.67
150	90	63	8319.2	2.69	8051.3	4.63	7423.0	8.46
130	94	74	8928.2	3.52	8609.1	4.64	7557.3	13.92
Average				1.59		3.57		5.08

Table 4
Computational results for the instances in problem C3.

H	Cycles	Chains	L_1	T_1 (s)	L_2	T_2 (s)	LB	Gap (%)
3000	175	6	17032.9	9.36	16650.2	9.21	16497.1	0.93
2000	175	9	17106.3	9.69	16742.3	19.20	16501.8	1.46
1000	175	17	17239.1	10.86	16848.5	17.53	16546.3	1.83
800	175	22	17336.5	11.53	16856.5	46.17	16575.1	1.70
600	175	30	17500.8	13.38	17008.5	64.22	16633.1	2.26
400	175	46	17795.3	15.07	17251.2	32.72	16758.5	2.94
300	175	64	18135.5	18.11	17436.0	61.82	16897.8	3.19
200	175	106	18943.4	25.01	18097.0	25.13	17238.7	4.98
150	185	150	20001.4	26.20	19124.7	40.89	17648.0	8.37
130	185	173	20329.0	20.93	19675.8	26.47	17950.6	9.61
Average				16.01		34.34		3.73

Table 5
Computational results for the instances in problem C4.

H	Cycles	Chains	L_1	T_1 (s)	L_2	T_2 (s)	LB	Gap (%)
5000	369	10	46123.9	47.47	–	–	44135.5	4.51
4000	369	13	46227.6	48.21	–	–	44142.3	4.72
3000	369	16	46271.2	55.42	–	–	44151.1	4.80
2000	369	24	46386.5	49.71	–	–	44184.9	4.98
1000	369	47	47020.2	53.53	–	–	44430.4	5.83
800	369	60	47256.9	57.09	–	–	44570.2	6.03
600	369	81	47711.3	48.95	–	–	44822.1	6.45
400	369	130	48728.9	50.37	–	–	45346.3	7.46
300	369	216	50626.2	59.29	–	–	46758.6	8.27
200	369	298	53621.6	71.59	–	–	47095.7	13.86
Average				54.16				6.69

Table 6
Computational results for the instances in problem U1.

H	Cycles	Chains	L_1	T_1 (s)	L_2	T_2 (s)	LB	Gap (%)
2000	81	1	1968.6	0.32	1902.7	0.02	1764.3	7.84
1000	81	2	2004.2	0.33	1909.0	1.04	1771.8	7.74
800	81	3	2039.6	0.41	1917.1	1.14	1779.4	7.74
600	81	4	2046.5	0.41	1925.1	2.51	1779.4	8.19
400	81	6	2075.9	0.50	1942.9	4.21	1796.8	8.13
300	81	8	2118.3	0.66	1972.7	3.03	1818.2	8.50
200	81	11	2153.6	0.81	2011.2	3.35	1833.5	9.69
100	81	25	2329.8	1.39	2184.8	4.91	1933.5	13.00
80	81	33	2407.9	1.81	2229.3	8.56	1999.5	11.49
60	81	52	2573.1	2.62	2428.2	2.38	2104.7	15.37
Average				0.93		3.12		9.77

Table 7
Computational results for the instances in problem U2.

H	Cycles	Chains	L_1	T_1 (s)	L_2	T_2 (s)	LB	Gap (%)
3000	178	2	4753.4	8.83	4646.2	5.94	4433.5	4.80
2000	178	3	4797.4	7.32	4653.4	21.56	4439.5	4.82
1000	178	5	4832.0	9.12	4665.7	67.42	4451.5	4.81
800	178	6	4852.5	10.48	4671.7	61.20	4457.5	4.81
600	178	10	5009.5	9.11	4694.6	112.00	4469.5	5.04
400	178	21	5113.6	13.62	4722.3	239.90	4497.9	4.99
200	178	48	5431.9	36.55	4847.6	302.20	4583.9	5.75
100	178	61	5661.4	55.87	5180.4	433.30	4789.5	8.16
80	178	78	5758.9	64.60	5411.5	290.75	4933.8	9.68
60	178	125	6233.3	106.87	5816.2	239.55	5196.6	11.92
Average				32.24		177.38		6.48

Table 8
Computational results for the instances in problem C5.

H	L_2	LB1	LB2	GAP1 (%)	GAP2 (%)
3000	7150.3	6903.8	6955.5	3.57	2.80
2000	7189.6	6903.8	6975.7	4.14	3.07
1000	7364.1	6903.8	7059.2	6.67	4.32
800	7458.4	6903.8	7117.2	8.03	4.79
600	7607.8	6903.8	7215.8	10.20	5.43
400	7990.7	6903.8	7496.4	15.74	6.59
300	8578.8	6903.8	7924.26	24.27	8.26
200	9639.5	6903.8	8409.7	39.63	14.62
180	10250.8	6903.8	8911.3	48.48	15.03
170	10691.6	6903.8	9248.2	54.87	15.61
Average				21.56	8.05

Table 9
Computational results for the instances in problem C6.

H	L_2	LB1	LB2	GAP1 (%)	GAP2 (%)
5000	45765.7	44127.7	44446.5	3.71	2.97
3000	46443.2	44127.7	44689.5	5.25	3.92
2000	47251.3	44127.7	45047.6	7.08	4.89
1000	48637.4	44127.7	46160.2	10.22	5.37
800	49878.4	44127.7	46752.0	13.03	6.69
600	51008.5	44127.7	47741.4	15.59	6.84
500	52238.9	44127.7	48852.2	18.38	6.93
400	53427.6	44127.7	49830.0	21.07	7.22
300	56923.1	44127.7	52181.9	29.00	9.09
260	59294.8	44127.7	53890.7	34.37	10.03
Average				15.77	6.40

Table 10
Computational results for the instances in problem U3.

H	L_2	LB1	LB2	GAP1 (%)	GAP2 (%)
3000	4674.2	4421.9	4459.5	5.70	4.81
2000	4694.3	4421.9	4479.4	6.16	4.80
1000	4734.2	4421.9	4519.4	7.06	4.75
800	4774.3	4421.9	4539.4	7.97	5.17
600	4816.8	4421.9	4579.4	8.93	5.18
400	5082.3	4421.9	4659.6	14.93	9.07
300	5244.6	4421.9	4769.3	18.60	9.96
200	5530.7	4421.9	4908.5	25.07	12.68
100	6461.0	4421.9	5476.0	46.11	17.99
80	7079.0	4421.9	5910.1	60.09	19.78
Average				20.06	9.42

time in seconds. Columns 6 and 7 show the same information of the improvement strategies. Column 8 indicates the lower bound LB2 for each instance. Column 9 shows the percentage gap between LB2 and final solution cost. The last line gives the average running time of two-phase heuristic algorithm and improvement strategies, and average percentage deviation to LB2.

Note that when computing the lower bound LB2 for a test instance from formulations (16)–(23), we may find a better value of V_n than $\lceil LB1/H \rceil$ during the computational experiments. For example, given a set of instances $(n_1,\ldots,n_{10})$, which is generated based on one problem and the only difference comes from the vehicle travel distance spans: $(h_1,\ldots,h_{10})$. Without loss of generality, let $h_1 > h_2 > \cdots > h_{10}$. Lower bound LB1 for these test instances is the result of formulations (12)–(15), i.e., LB1. First, we calculate LB2 for test instance n_1 from formulations (16)–(23) by setting $V_{n1}=\lceil LB1/h_1 \rceil$. Then, when computing LB2 for n_2, we compare the values of $\lceil LB1/h_2 \rceil$ and $\lceil LB2_{n1}/h_2 \rceil$. If $\lceil LB2_{n1}/h_2 \rceil$ is larger than $\lceil LB1/h_2 \rceil$, it is adopted as the value of V_{n2}, and is input into formulations (12)–(15) to get $LB2_{n2}$. The procedure is repeated until LB2s for 10 test instance $n_1,\ldots,n_{10}$ are gotten.

As shown in Tables 2–7, we can find that the solution costs obtained by the proposed heuristic are close to the lower bounds. For some test instances, such as C1-1000 and C3-3000, the gaps between the solution costs and LB2 are less than 1%. For all 60 test instances, the largest percentage gap between the solution cost and LB2 is 15.37%, and the average percentage gap is 5.82%.

Tables 2–7 also show that for various instances generated within one problem, the number of cycles, the number of chains and the proposed heuristic results increase simultaneously, with the decrease of the distance span. The processing time of local search procedure increases with the increase of lanes. When the number of lanes reaches 1000, it takes too long to execute the local search. Therefore, Table 5 gives the solutions without local search improvement, which are adopted as the final results. Similarly, for the test instances within problem C6, the solutions without local search improvement are adopted as the final results.

Solution costs and two lower bounds for problems C5, C6 and U3 are given in Tables 8–10. In these tables, columns 1–4 record the distance span, solution cost (L_2), LB1 and LB2. The last two columns identify the percentage gaps between the solution cost and two lower bounds.

First, we try to compare the existing lower bound LB1 with new lower bound LB2 in Tables 8–10. It is depicted that for each test instance GAP2 is always smaller than GAP1. For each problem, GAP1 and GAP2 both increase with the decrease in distance span. However, GAP1 increases much more rapidly than GAP2. We choose problem C5 as an example. When we decrease the distance span from 3000 to 170, GAP1 increases from 3.57% to 54.87%, whereas GAP2 only increases from 2.80% to 15.61%. It indicates that LB2 is more promising than LB1, especially for the test instance with tight vehicle travel distance span. This conclusion can be seen clearly in Fig. 8, where two curves represent GAP1 and GAP2 for 10 test instances within problem C5.

Furthermore, we see that GAP2 keeps acceptable when the depot locations are changed. As stated above, problems C2 and C5 comprise the same basic structure. The only difference comes from the depot locations. The same principles apply to problems C4 and C6, problems U2 and U3. As shown in Tables 3, 5, and 7–10, the values of GAP2 increase when the depots are far away from the customer nodes. However, the increase in GAP2 is not prominent. For problems C2, C4 and U2 the average gaps between the heuristic solution costs and LB2 are 5.08%, 6.69% and 6.48%, respectively. When the depot locations are changed (i.e., the depots are departing from the customer locations), the average gaps for problems C5, C6 and U3 are 8.05%, 6.40% and 9.42%, respectively. The results show that the

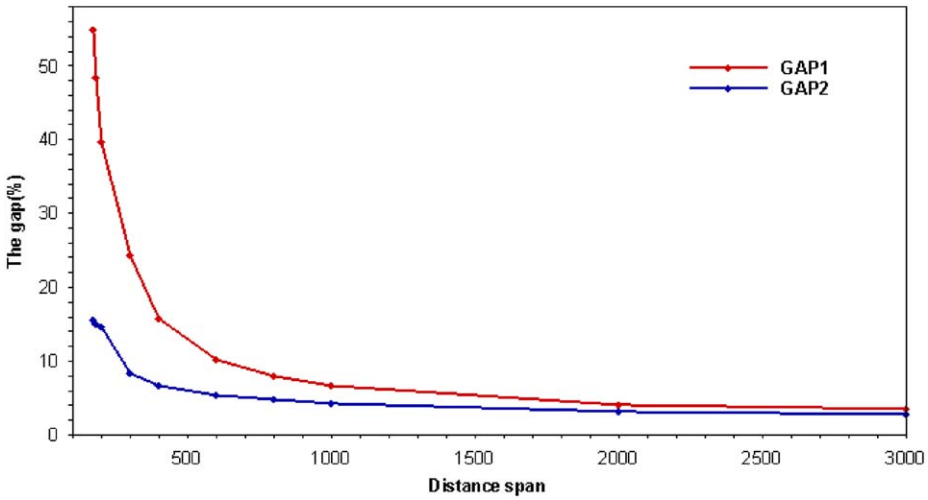

Fig. 8. The percentage deviations of the heuristic solution values over LB1 and LB2 for problem C5.

proposed method yields robust results which do not heavily depend on the depot locations.

7. Conclusions and future research

This paper discusses an important problem, i.e., the multi-depot capacitated arc routing problem with full truckloads, which is an extension of the CARP with wide applications in carrier collaboration. An exact formulation is presented. A set of heuristics is put forward to solve this problem of a realistic size. To validate the proposed heuristic, two lower bounds are designed. The algorithm is tested on different types of problems. The results demonstrate that the proposed heuristic provides high-quality solutions in a reasonable computing time. The impact of vehicle distance span and depot locations on the solution quality is also explored. In conclusion, the proposed algorithms can provide robust solutions.

For future research, attentions can be focused on the extension of MDCARPFL, MDCARPFL with time windows. Properly incorporating timing considerations in MDCARPFL is of critical importance to practical viability. In the MDCARPFL with time windows, the service at each order must start and end within an associated time window. It is interesting and challenging to design the algorithm for obtaining a high quality solution to the MDCARPFL with time windows in an acceptable time.

Acknowledgments

This work was supported by Research Grant from National Natural Science Foundation of China (no. 70872077, 70771063), National Natural Science Foundation of China/Research Grants Council of Hong Kong joint research projects (no. 70831160527), and a General Research Fund (GRF) of the Research Grants Council (RGC) of HKSAR (no. RGC 113609).

We are indebted to the anonymous reviewers, whose comments and suggestions have greatly improved content and presentation of the material in this paper.

References

[1] Ergun Ö, Kuyzu G, Savelsbergh M. Shipper collaboration. Computers & Operations Research 2007;34(6):1551–60.
[2] Arunapuram S, Mathur K, Solow D. Vehicle routing and scheduling with full truckloads. Transportation Science 2003;37(2):170–82.
[3] Golden BL, Wong RT. Capacitated arc routing problems. Networks 1981;11(3):305–15.
[4] Eglese RW, Murdock H. Routeing road sweepers in a rural area. Journal of the Operational Research Society 1991;42(4):281–8.
[5] Eiselt H, Gendreau M, Laporte G. Arc routing problems, part II: the rural postman problem. Operations Research 1995;43(3):399–414.
[6] Assad AA, Golden BL. Arc routing methods and applications. Ball MO. et al., editor. Handbooks in operations research and management science, vol. 8, Amsterdam: Elsevier; 1995. p. 375–483.
[7] Hirabayashi R, Saruwatari Y, Nishida N. Tour construction algorithm for the capacitated arc routing problem. Asia-Pacific Journal of Operational Research 1992;9(2):155–75.
[8] Belenguer JM, Benavent E. A cutting plane algorithm for the capacitated arc routing problem. Computers & Operations Research 2003;30(5):705–28.
[9] Longo H, De Aragão MP, Uchoa E. Solving capacitated arc routing problem using a transformation to the CVRP. Computers & Operations Research 2006;33(6):1823–37.
[10] Baldacci R, Maniezzo V. Exact methods based on node-routing formulations for undirected arc-routing problems. Networks 2006;47(1):52–60.
[11] In: Dror M, editor. Arc routing: theory, solutions and applications. Boston: Kluwer; 2000.
[12] Hertz A, Mittaz M. A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. Transportation Science 2001;35(4):425–34.
[13] Muyldermans L, Beullens P, Cattrysse D, Oudheusden VD. Exploring variants of 2-opt and 3-opt for the general routing problem. Operations Research 2005;53(6):982–95.
[14] Hertz A, Laporte G, Hugo PN. Improvement procedures for the undirected rural postman problem. INFORMS Journal on Computing 1999;11(1):53–62.
[15] Beullens P, Muyldermans L, Cattrysse D, Van Oudheusden D. A guided local search heuristic for the capacitated arc routing problem. European Journal of Operational Research 2003;147(3):629–43.
[16] Lacomme P, Prins C, Ramdane-Cherif W. Competitive memetic algorithms for arc routing problems. Annals of Operations Research 2004;131(1):159–85.
[17] Eglese RW, Li LYO. A tabu search based heuristic for arc routing with a capacity constraint and time deadline. In: Osman IH, Kelly JP, editors. Meta-heuristics: theory and applications. Dordrecht: Kluwer; 1996. p. 633–50.
[18] Brandão J, Eglese RW. A deterministic tabu search algorithm for the capacitated arc routing problem. Computers & Operations Research 2008;35(4):1112–26.
[19] Greistorfer P. A tabu scatter search metaheuristic for the arc routing problem. Computers & Industrial Engineering 2003;44(2):249–66.
[20] Eglese RW. Routing winter gritting vehicles. Discrete Applied Mathematics 1994;48(3):231–44.
[21] Amberg A, Domschke W, Voß S. Multiple center capacitated arc routing problems: a tabu search algorithm using capacitated trees. European Journal of Operational Research 2000;124(2):360–76.
[22] Dantzig G, Fulkerson R, Johnson S. Solution of a large-scale traveling salesman problem. Operations Research 1954;2(1):393–410.
[23] Toth P, Vigo D. Models, relaxations and exact approaches for the capacitated vehicle routing problem. Discrete Applied Mathematics 2002;123(1):487–512.
[24] Gronalt M, Hartl RF, Reimann M. New savings based algorithms for time constrained pickup and delivery of full truckloads. European Journal of Operational Research 2003;151(3):520–35.
[25] Bräysy O, Gendreau M. Vehicle routing problem with time windows, part I: route construction and local search algorithms. Transportation Science 2005;39(1):104–18.