Transportation Research Part B 82 (2015) 20–35

Contents lists available at ScienceDirect

Transportation Research Part B

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

An evolutionary local search for the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints

Zhenzhen Zhang[a,1], Lijun Wei[b,*], Andrew Lim[c,d]

[a] Department of Management Sciences, City University of Hong Kong, Tat Chee Ave, Kowloon Tong, Hong Kong
[b] School of Information Technology, Jiangxi University of Finance and Economics, Nanchang, Jiangxi 330013, People's Republic of China
[c] International Center of Management Science and Engineering, School of Management and Engineering, Nanjing University, Nanjing 210093, People's Republic of China
[d] Department of Industrial & Systems Engineering, National University of Singapore, Singapore

ARTICLE INFO

Article history:
Received 25 May 2015
Revised 15 August 2015
Accepted 1 October 2015
Available online 11 November 2015

Keywords:
Routing
Loading
Evolutionary local search
3L-CVRP
3L-FCVRP

ABSTRACT

This study introduces a new practical variant of the combined routing and loading problem called the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints (3L-FCVRP). It presents a meta-heuristic algorithm for solving the problem. The aim is to design routes for a fleet of homogeneous vehicles that will serve all customers, whose demands are formed by a set of three-dimensional, rectangular, weighted items. Unlike the well-studied capacitated vehicle routing problem with 3D loading constraints (3L-CVRP), the objective of the 3L-FCVRP is to minimize total fuel consumption rather than travel distance. The fuel consumption rate is assumed to be proportionate to the total weight of the vehicle. A route is feasible only if a feasible loading plan to load the demanded items into the vehicle exists and the loading plan must satisfy a set of practical constraints.

To solve this problem, the evolutionary local search (ELS) framework incorporating the recombination method is used to explore the solution space, and a new heuristic based on open space is used to examine the feasibility of the solutions. In addition, two special data structures, Trie and Fibonacci heap, are adopted to speed up the procedure. To verify the effectiveness of our approach, we first test the ELS on the 3L-CVRP, which can be seen as a special case of the 3L-FCVRP. The results demonstrate that on average ELS outperforms all of the existing approaches and improves the best-known solutions for most instances. Then, we generate data for 3L-FCVRP and report the detailed results of the ELS for future comparisons.

1. Introduction

Recently, many Chinese cities were shrouded for several days in thick fog and haze, and the Chinese government is beginning to recognize the need to reduce emissions and improve air quality. Freight transportation is essential for economic development, but it is also harmful to the environment, as a significant portion of freight transportation is carried out by trucks run on diesel engines, which are the major source of carbon dioxide (CO_2) emissions. Thus, reducing fuel consumption can directly reduce

* Corresponding author. Tel.: +8615970638521.
E-mail addresses: zhenzhenzhang222@gmail.com (Z. Zhang), villagerwei@gmail.com (L. Wei), lim.andrew@cityu.edu.hk (A. Lim).
[1] Tel.: +852 64067667; fax: +852 34420188.

http://dx.doi.org/10.1016/j.trb.2015.10.001
0191-2615/

carbon emissions. In addition, fuel consumption accounts for as much as 60% of the operating cost of a vehicle, according to a study by Sahin et al. (2009). Therefore, reducing fuel consumption can also reduce operating costs.

Motivated by these facts, we study a new variant of the combined routing and loading problem, which we call the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints (3L-FCVRP). This problem is more practical than the classic vehicle routing problems (VRP). For companies, the operating cost is usually measured by the cost of consumed fuel rather the total travel distance, as fuel consumption of a vehicle is the product of the fuel consumption rate and the travel distance. The fuel consumption rate varies with the current weight of the vehicle, as shown by Xiao et al. (2012). Second, the routing plan is meaningful only if all of the goods can be loaded into the vehicle given a series of practical constraints. Thus, it is necessary to consider routing and loading simultaneously. Moreover, methodologically, the combined routing and loading problem requires to balance the efforts of exploring the solution space and identifying the loading plans. A method that simply combines the sophisticated VRP and packing algorithms may not obtain a good result within a reasonable computational time.

To solve this problem, an evolutionary local search (ELS) is used to intensively search the solution space. In addition, a multi-level diversification strategy is incorporated to diversify the search trajectory. The block exchange method exhaustively searches the neighbor regions of the current solution, and the recombination helps the search jump to other promising regions. These strategies are vital for solving this highly constrained problem successfully. The route pool helps to fully exploit the historical solution information by storing promising routes and constructing new promising solutions that are used to adjust the search trajectory. To ensure that the solution is loading-feasible, the loading check procedure is invoked for each temporal solution. Thus, our search only moves within the feasible solution space. Although the basic moves and accelerating strategies are borrowed from the work of Wei et al. (2014), we make the following contributions. First, a new practical variant of VRP is introduced. Second, a totally different search framework, ELS, is used to guide the search. The ELS is easier to implement than the adaptive variable neighborhood search (Wei et al., 2014) and obtains better results, as illustrated by our experiments. Third, a new packing heuristic based on the open space (Section 5.1) is developed to efficiently find the loading plans. Finally, most of the best-known results of 3L-CVRP instances are updated. More precisely, two special data structures, Trie and Fibonacci heap, are adopted to speed up the procedure. The Trie structure not only helps avoid duplicate loading checks of the same routes, but also makes it available to spend increasing effort on the frequently encountered routes. These routes are usually promising to be in the final best solution. The Fibonacci heap helps to avoid duplicating objective calculations of moves, and only invoke the loading check procedure for parts of promising moves.

The remainder of this paper is organized as follows. Section 2 gives an overview of the relevant literature. Section 3 provides the detailed description of 3L-FCVRP. The entire ELS search framework and ingredient functions are presented in Section 4. Section 5 describes the local search procedure for the loading component. The extensive results for the 3L-CVRP and 3L-FCVRP are reported in Section 6. Finally, Section 7 concludes the paper.

2. Literature review

The original combined routing and loading problem, the capacitated vehicle routing problem with two-dimensional loading constraints (2L-CVRP), was proposed by Iori et al. (2007). It aims at minimizing the total distance traveled by all of the vehicles and assumes that products cannot be stacked on top of each other. Based on the 2L-CVRP, Leung et al. (2013) proposed the heterogeneous fleet vehicle routing problems with two-dimensional loading constraints (2L-HFVRP). Recently, Khebbache-Hadji et al. (2013) introduced the time window into the 2L-CVRP and proposed the two-dimensional loading capacitated vehicle routing problem with time windows (2L-CVRPTW).

Gendreau et al. (2006) first proposed the capacitated vehicle routing problem with three-dimensional loading constraints (3L-CVRP), which considers many of the practical constraints on loading, such as the fragility, support, and last in first out (LIFO) constraints. The 3L-CVRP can be seen as a special case of the 3L-FCVRP by assuming that the fuel consumption rate of a vehicle is constant regardless of the weight of the vehicle. Zachariadis et al. (2012) studied the pallet-loading vehicle routing problem (PPVRP) by integrating routing with pallet-loading constraints. In this problem, the products are first stacked into pallets, which are then loaded onto the vehicles. Recently, Zachariadis et al. (2013) proposed a very complex problem called the pick-up and delivery routing problem with time windows and pallet loading (PDRP-TWP). Surveys of routing problems with 2D and 3D loading constraints are provided by Wang et al. (2009), Iori and Martello (2010) and Iori and Martello (2013). We refer the reader to the papers by Bektaş and Laporte (2011); Franceschetti et al. (2013); Santos et al. (2010); Zachariadis et al. (2015) for more variants of the routing problems.

These vehicle routing problems with loading constraints combine two NP-hard problems, thus an exact approach can only be used to solve small instances. The only exact approach was proposed by Iori et al. (2007) for 2L-CVRP, which can solve instances with up to 35 customers in 24 h. Most previous studies used a meta-heuristic to solve the routing sub-problem. These approaches include tabu search (Bortfeldt, 2012; Doerner et al., 2007; Gendreau et al., 2006; 2008; Leung et al., 2011b; Tao and Wang, 2015; Tarantilis et al., 2009; Zachariadis et al., 2009; Zhu et al., 2012), ant colony optimization (Fuellerer et al., 2009; 2010), simulated annealing (Leung et al., 2013; 2010), honey bee mating optimization (Ruan et al., 2013), multi-start evolutionary local search (Duhamel et al., 2011), and variable neighborhood search (Wei et al., 2014; 2015). To the best of our knowledge, the variable neighborhood search frameworks proposed by Wei et al. (2015) and Wei et al. (2014) are the state-of-the-art methods for the 2L-CVRP and 3L-CVRP, respectively.

Previous studies adapted the classical approaches to the loading component; for example, the bottom-left fill method (Chazelle, 1983; Gendreau et al., 2006; Zhu et al., 2012), max touching perimeter heuristic (Gendreau et al., 2006; Lodi et al., 1999; Zachariadis et al., 2009), min area heuristic (Iori et al., 2007; Zachariadis et al., 2009), lowest reference line best-fit heuristic (Leung et al., 2011a; 2013; 2010), and the least waste first (Tao and Wang, 2015; Wei et al., 2009; 2015).

Only a few studies have discussed fuel consumption in routing problems. Piecyk and McKinnon (2010) pointed out that decisions in road freight transport have a significant effect on carbon emissions. Kara et al. (2007) defined cumulative VRPs (CumVRPs), where the objective is to minimize vehicle fuel consumption and the fuel consumption rate is assumed to be proportional to the total weight of the vehicle. However, the details of the formulation of the fuel consumption values are not provided. By analyzing collected data, Xiao et al. (2012) proposed a linear model of fuel consumption rate associated with the vehicle load. A similar measurement method was also used by Harris et al. (2011) to assess the effect of traditional cost optimization on the final carbon emissions. More details of the fuel consumption are discussed in the survey by Lin et al. (2014). Another problem that considers fuel consumption is the pollution-routing problem (Bektaş and Laporte, 2011; Franceschetti et al., 2013; Koç et al., 2014). Another similar problem is the load-dependent VRP (Zachariadis et al., 2015).

Recombination, one of the adaptive memory-based methods, is attracting increasing attention from researchers. It has been proved useful to improve local search methods. This rationale was first introduced by Rochat and Taillard (1995). A special structure is used to store good routes during the procedure. A new provisional solution is constructed by probabilistically selecting stored routes, and this is improved by a tabu search. This method was also used by Li et al. (2012) to solve the heterogeneous fixed fleet open vehicle routing problem (HFFOVRP). Subramanian et al. (2012) constructed the new solution via solving a set partitioning formulation by means of a mixed integer programming solver CPLEX. Recently, Chang et al. (2013); 2012) proposed a novel block mining and recombination method for the traveling salesman problem (TSP) and the permutation flow-shop scheduling problem (PFSP), respectively. Meaningful segments are extracted from the local optimal solution and archived in a pool. Subsequently, these segments are used to generate new promising solutions, which are added to the population to improve the genetic algorithm. A similar idea was presented by Tarantilis and Kiranoudis (2002); 2007), but they used a different method to extract the good segments and generate solutions.

3. Problem description

Following previous studies, the 3L-FCVRP is defined on a complete graph $G = (V, E)$, where $V = \{0, 1, \ldots, N\}$ is the vertex set containing a central depot (node 0) and N customers, and $E = \{[i, j] | i, j \in V, i \neq j\}$ is the undirected edge set. Each edge $[i, j] \in E$ associates with a travel distance d_{ij} between vertices i and j. A fleet of K homogeneous vehicles is located at the depot, and the vehicle has self-weight Q_0, weight capacity Q, and a three-dimensional rectangular loading space of length L, width W, and height H. Each vehicle has an opening at the rear door, which is as large as the $(W \times H)$ area. Each customer i $(i = 1, \ldots, N)$ requires a set of items denoted as I_i, whose total weight is q_i; the size of set I_i is denoted as m_i. Each item $I_{ik} \in I_i$ $(k = 1, \ldots, m_i)$ has length l_{ik}, width w_{ik}, and height h_{ik}. So I_i is the set of items required by customer i and any item in I_i can be denoted as I_{ik}. In addition, a fragility flag f_{ik} is associated with each item I_{ik}, and f_{ik} is set as 1 if the item is fragile, and otherwise 0 (Gendreau et al., 2006). By analyzing the collected data, Xiao et al. (2012) found that, for any vehicle, the fuel consumption per unit distance $fc(q)$ is a linear function depending on the total weight of the vehicle, that is $fc(q) = a(Q_0 + q) + b$, where a and b are constant factors and q is the carried load. This formula can be rewritten as $fc(q) = fc_0 + \frac{fc^* - fc_0}{Q} q$, where $fc_0 = aQ_0 + b$ is the no-load fuel consumption rate and $fc^* = a(Q_0 + Q) + b$ is the full-load fuel consumption rate. In this study, fc_0 is set to 1 and fc^* is set to 2, as in the previous study (Xiao et al., 2012). Thus, the fuel consumption for any edge $[i, j]$ by a vehicle with load q is $fc(q) \cdot d_{ij}$. It is worth noting that for the same edge, the fuel consumption differs with different vehicle loads. Unlike the classic VRP, the objective of the 3L-FCVRP is to design routes for given vehicles that will serve every customer while minimizing the total fuel consumption. Thus, in this problem, the cost is defined as the total fuel consumption. A feasible solution must satisfy the following vehicle constraints.

1. Each vehicle must start and terminate at the central depot. Note that not all K vehicles have to be used.
2. **No-split demand**. All of the demands of each customer must be served, and each customer must be visited exactly once by only one vehicle.
3. **Weight constraint**. The total weight of the items in one route must not exceed the weight capacity Q of the vehicle.

In this study, we use the Cartesian coordinate system shown in Fig. 1, where the width W is parallel to the x-axis, the height H is parallel to the y-axis, and the length L is parallel to the z-axis. A feasible route must satisfy the following constraints in terms of loading aspect.

4. **Loading constraint**. The items are loaded into the vehicle surface with their edges parallel to the sides of the vehicle and no overlap exists between any pair of items. The item can only be rotated 90° horizontally and the vertical orientation is fixed.
5. **Support constraint**. The base of any item must be partially supported by the vehicle or other items. The percentage of supported area should not be less than α, which is usually set as 0.75 in the literature (Gendreau et al., 2006).
6. **Fragility constraint**. A fragile item can be placed on top of any other item, but a non-fragile item can only be stacked on other non-fragile items.

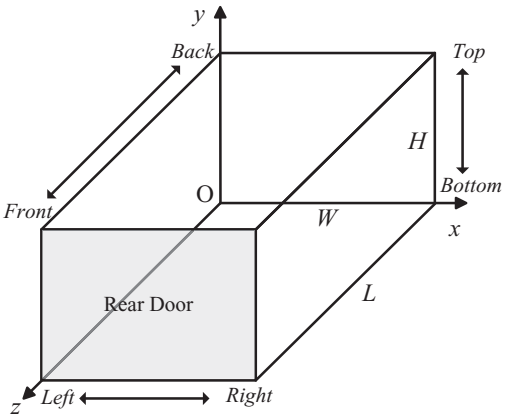
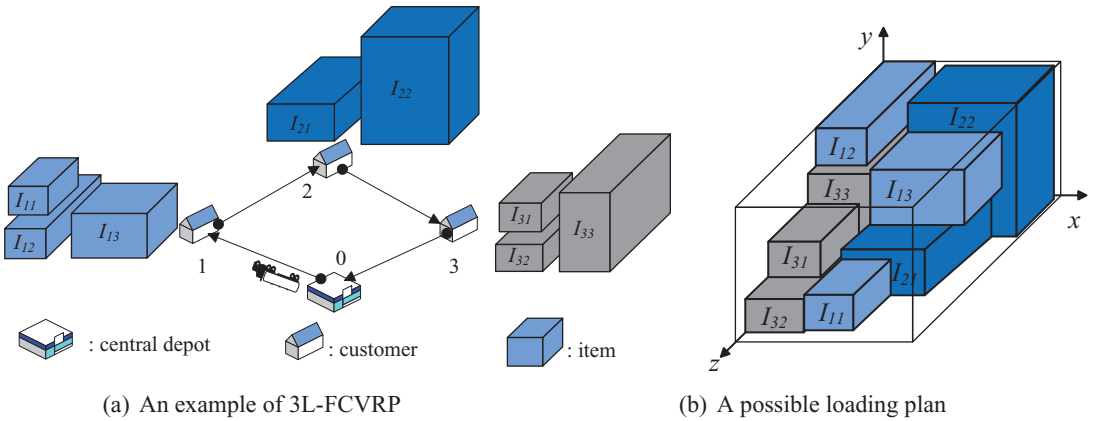


Fig. 1. 3D coordinate system.



(a) An example of 3L-FCVRP (b) A possible loading plan

Fig. 2. An example of 3L-FCVRP and loading plan.

7. **LIFO constraint**. For any customer pair i and j, assuming j is visited later than i, no items demanded by j can be placed above any item $I_{ik}(k = 1, \ldots, m_i)$ or between I_{ik} and the rear of the vehicle.

Fig. 2 gives an example of a 3L-FCVRP and the possible loading plan for the sample route.

4. Evolutionary local search algorithm for exploring the solution space

In this study, the evolutionary local search (ELS) algorithm is used to explore the solution space, and a multi-level perturbation strategy is incorporated into the algorithm to diversify the search trajectory. The ELS is an improved version of iterated local search (ILS) in which several perturbed points of the current solution are generated and improved instead of only one (Prins, 2009). Our method takes advantage of the ability of the ELS to better investigate the neighborhoods of the current local optimum, but also adopts the recombination strategy to diversify the search to other promising regions. The recombination strategy is vital for highly constrained problems, because it helps the search jump to other promising regions by fully exploiting the historical solutions.

The framework of our method is outlined in Algorithm 1. The initial solution is constructed with the modified saving method described in Section 4.1. For a given solution *S*, *nd* child points are obtained using the BlockExchange (Section 4.3) procedure, then improved with the LocalSearch (Section 4.2) with basic moves (Lines 6–11), and the best child replaces *S* (Line 12). In addition, each local optimal child is stored to maintain the route pool *RP* (Line 9). This iteration is repeated to exhaustively investigate the neighboring spaces. After that, the Recombination (Section 4.4) procedure is invoked to drive the search into a new promising region by generating a new solution based on the fruitful routes stored in the pool (Line 15). The search is repeated until the runtime exceeds the specified time limit. The *nonImpMax* is the parameter controlling when to invoke the Recombination procedure. In our implementation, its value is set to 10, and *nd* is set to min (10, *K*) by some simple tests.

All of the ingredient procedures used in the ELS are described extensively in the following sections. Finally, the ways in which the special data structures, Fibonacci heap and Trie, are used to accelerate the algorithm are presented in Section 4.5. During all of these procedures, we use the LocalSearchPack procedure (see Section 5) to check the loading feasibility of a route. For convenience, cost is used instead of fuel consumption in the description of the algorithm.

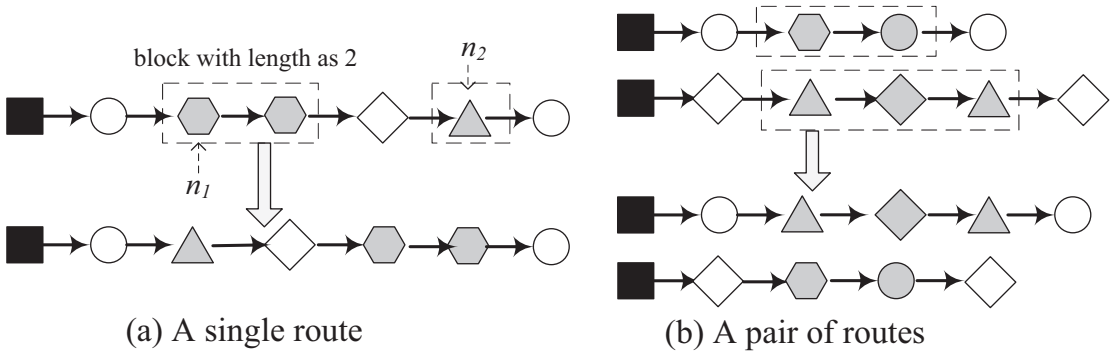


Fig. 3. An example of the block exchange method.

4.1. The modified saving method

The saving algorithm (Clarke and Wright, 1964) is an excellent method for constructing the initial solution. At the beginning, each customer is assumed to be served by one vehicle. Then, the pair of routes whose merger would result in the most saving cost is merged into a single route. This is repeated until no more improving merges exist. The saving cost is defined as the total cost of two old routes minus the cost of the new route. Of course, the new route must be feasible given all of the constraints. The classic saving method is modified to allow for the limited number of vehicles in our problem. If the number of routes exceeds the number of vehicles, the merge procedure continues until the feasible solution is obtained or no feasible merges exist. If the solution is still infeasible in terms of number of routes, the excess routes with the least volume utilization are removed and the customers on these routes are marked as unserved; then, a repair procedure is invoked to obtain a complete solution.

The repair procedure begins by sorting the customers by decreasing volume of required items and then inserting them into the solution one by one. More specifically, the customers are sequentially inserted into the position and route with the least incremental cost, and the resulting route must be packing-feasible. If a customer cannot be inserted into any route, it will be inserted into a randomly chosen route by randomly removing some customers gradually until the route is feasible. This procedure is repeated until a feasible solution is obtained. Fortunately, it is easy and fast to construct a feasible solution with this method. In our experiments, we test the five versions of the problem with all 39 test instances. First, we find that the vehicles in every instance are usually sufficient. Second, if the customers with high demand are inserted first, it is easy to serve the rest of the customers.

4.2. Local search procedure

The LocalSearch procedure aims at improving the solution quickly. Therefore, three types of basic moves are used: *shift*, in which one customer is shifted to another position on the same route or to a different route; *exchange*, in which the positions of two customers on the same route or on two different routes are exchanged; and *2-opt* (or *2-opt*$*$), in which the sequence between the two given customers is reversed if they are in the same route or the end parts after the two specified customers of different routes are interchanged. In our algorithm, the best feasible improvement strategy is adopted. More precisely, one of the three types is randomly selected with the same probability. Then, all of the possible moves of the chosen type are checked, and the feasible move with the best cost improvement is identified and performed. Here, a move is defined as feasible if the resulting solution is feasible in terms of all constraints. The procedure terminates when no move can produce an improved solution. For a given instance with N customers and K vehicles, there are $(N+K)$ positions for the *shift* type and $(N+K)$ points to partition the routes for *2-opt(*$*$)*; therefore, the cardinality of possible moves are $N(N+1)/2$ for *exchange*, and $N(N+K)$ for *shift* and *2-opt(*$*$)*. It is time-consuming to identify the best move, especially for this problem, as the loading check procedure must be invoked frequently. To accelerate the procedure, two special data structures, Fibonacci heap and Trie, are used. They are discussed in Section 4.5.

4.3. Block exchange structure

To explore a wide range of neighboring regions of the current solution, the block exchange is used to generate several starting points for the local search. The current solution is usually a local optima obtained by the local search, so the block exchange must have the ability to drive the search to the space that basic moves cannot reach; however, the strength should not be too strong to move the search to a totally different region before the near spaces of the current solution have been carefully investigated.

A block is a sequence of consecutive customers with an arbitrary length. The BlockExchange procedure exchanges the positions of two blocks, as depicted in Fig. 3. To execute this method, it is necessary to specify two customers, n_1 and n_2, which denote the starts of blocks, and the lengths of blocks, as shown in Fig. 3. Note that the different shapes do not have special means;

they are just used to clearly depict the operator. Because the probability of obtaining a feasible resulting solution decreases with the difference between the lengths of two blocks, we limit the length of the block to 4. Note that one of the two blocks might be empty.

4.4. Recombination for a new promising solution

The principle of recombination is to construct a promising solution using meaningful historical solutions as a new starting point for the search. It guides the search to a new promising space. Initially, the route pool is set as empty. Then, only the local optimal solutions are stored in the pool to make the pool promising (see Line 9 of Algorithm 1). The characteristic of the ELS makes it quite suitable for maintaining the pool, because it provides diversified local optimas. When obtaining a local optimum, each route is tagged with the value of the solution cost, and added to the pool as follows. The procedure first checks whether the route is in the pool. If the route does not exist, the route is added to the pool. Otherwise, the value of the route tag is updated with the least solution cost. The routes in the pool are maintained according to the non-decreasing values of the tags. Finally, if the pool size exceeds the specified limit, the excess part of routes are discarded. In this study, the size limit is set as ten times of the number of vehicles, which is a reasonable limit for providing diversified routes.

To balance the quality and diversification of the obtained solution, the routes are probabilistically selected to construct the new solution as follows. As a common rule, the route with a higher rank should be selected with higher probability. Thus, the probability of ith route is defined as $2(M-i+1)/(M(M+1))$, where M is the number of useful routes in the pool. This definition guarantees that the sum of the probabilities of all of the routes equals 1. The roulette wheel method is used to select routes. After choosing one route, other routes that share customers with the chosen route are marked as useless, and the ranking of the routes and the number M are modified correspondingly. This procedure is repeated until no other route can be selected. Note that some customers might remain unserved; in this case, the repair procedure given in the Section 4.1 is invoked to obtain a complete solution.

4.5. Acceleration strategies

Two data structures, Trie and Fibonacci heap, are used to accelerate the search. The Trie can help to avoid duplicate loading check for the same route, as shown in Section 5. The Trie has been used by Wei et al. (2014); 2015) to accelerate the search. The Fibonacci heap is used to quickly find the best move and to reduce the calls of the loading check procedure.

The Fibonacci heap (Fredman and Tarjan, 1987) is a special priority queue with the capability of fast insertion, update, deletion and retrieving minimum, which was first used by Zachariadis and Kiranoudis (2010); 2011) to accelerate the local search procedure in the VRP. The basic moves have a property that ensures at most two routes are involved in a move. Thus, it is only necessary to re-evaluate the costs of parts of moves. Storing the cost information of all moves in the heap, prevents duplicated cost calculations in the identification of the best move in each iteration of the local search, and ensures that only a part of moves need to be updated after performing a move. Thus, one Fibonacci heap is used for each of the move types described in Section 4.2. Here, our update procedure is slightly different from previous studies, because our cost function is quite different than the one used in the classic VRP problem. In particular, more moves need to be re-evaluated. After performing a move on single route R_1 (or two routes R_1 and R_2), the costs of the stored moves of all three types that are related to the customers on R_1 (or R_2) need to be updated. The pseudo-code is shown in Algorithm 2.

In addition, with the help of the Fibonacci heap, the time-consuming loading check can be restricted to promising moves. More precisely, the most improving move is retrieved from the heap, and then the loading check procedure is called. If the move is feasible, this move is performed immediately and no other loading check is called. Otherwise, the procedure is repeated until a feasible move is encountered. Thus, the use of a Fibonacci heap dramatically speeds up the algorithm.

5. Local search for the loading subproblem

When a temporal solution is obtained by a given move, its loading feasibility must be examined. Thus, the loading check is a frequently invoked sub-problem. The LocalSearchPack procedure, which is given as Algorithm 3, is used to examine the loading feasibility of a route. It takes as input a route tour R containing the ordered customers visited by a vehicle. If the procedure finds a feasible loading for this vehicle, it returns a const TRUE; otherwise, it reports a const FALSE. Two cases are classified based on n, which is the number of items required by the customers in R.

In the first case, where n is not larger than a user defined parameter max_enumeration, all of the permutations of I are enumerated and the process OpenSpaceHeuristic is called on each permutation. Given an ordered list of items I, the OpenSpaceHeuristic (see Section 5.1 for details) tries to pack the items into the vehicle surface one by one according to the given order of I. It terminates once an item fails to be placed or all items are placed. The number of packed items is returned at last. If it finds a permutation on which the result returned by OpenSpaceHeuristic is n, TRUE is returned; otherwise, FALSE is returned.

In the second case, where n is larger than the user defined parameter max_enumeration, a local search procedure is used to test a part of the permutations of I. Initially, it uses the sorting rules to generate a sequence I of the items demanded by the customers in R (Line 9). Here, two sorting rules are used. For each sorted sequence of items I, it uses the same heuristic loading procedure OpenSpaceHeuristic to pack the items. If the number returned by OpenSpaceHeuristic is n, the LocalSearchPack halts with TRUE. Otherwise, it tries to optimize the solution in terms of the number of packed items through a random local search.

Algorithm 1 LocalSearchPack procedure

```
LocalSearchPack( R )
    // Input: R, the ordered customers visited by a vehicle
 1  let n be the number of items demanded by the customers in R
 2  if n ≤ max_enumeration
 3      I = all items demanded by the customers in R
 4      for each permutation I* of I
 5          if OpenSpaceHeuristic( I* ) = n
 6              return TRUE
 7      return FALSE
 8  for each sorting rule
 9      I = sorted sequence of all items demanded by the customers in R
10      p = OpenSpaceHeuristic( I )
11      if p = n return TRUE
12  for k = 1 to n
13      Generate new sequence I* by swapping two randomly selected items in I
14      p* = OpenSpaceHeuristic( I* )
15      if p* ≥ p
16          I = I*, p = p*
17      if p = n return TRUE
18  return FALSE
```

The number of iterations is limited to the number of items n (Lines 12–17). In each iteration, a new sequence is generated by randomly swapping the positions of two items; the new sequence is accepted if it is not worse than the old one. At any point, if the result returned by the OpenSpaceHeuristic is n, the procedure halts with TRUE. If this does not occur during any of the iterations, FALSE will be returned finally.

The sorting rules are the same as those in Wei et al. (2014), which are defined as follows. First, each item i is associated with a vector $p(i)$ consisting of four components. Item i will be placed before j if $p(i)$ is lexicographically greater than $p(j)$ (i.e., two vectors are compared component by component, and the order of the first different component decides the order of the two items). In the first sorting rule, the components of $p(i)$ are

$$p(i) = (ord(i) \cdot LIFO, -f(i) \cdot FRAGILITY, l_i \cdot w_i \cdot SUPPORT, l_i \cdot w_i \cdot h_i), \tag{1}$$

where

1. LIFO, FRAGILITY, and SUPPORT: the flags indicating whether the LIFO, fragility, and support constraints are set, respectively. If such a constraint is enforced, the corresponding flag is 1, and otherwise 0.
2. ord(i): the visiting order of the customer to whom item i belongs.
3. f(i): if the item i is fragile, this value is 1, and otherwise 0.
4. l_i, w_i, and h_i: the length, width, and height of item i, respectively.

The components of $p(i)$ in the second sorting rule are

$$p(i) = (ord(i) \cdot LIFO, -f(i) \cdot FRAGILITY, l_i \cdot SUPPORT, l_i \cdot w_i \cdot h_i), \tag{2}$$

The only difference occurs in the third component. The third component in the first sorting rule is the base area of the item, whereas it is the length of the item in the second sorting rule.

To speed up our loading procedure, a special data structure, Trie, is used to keep track of the loading feasibility of the routes that have been already examined. Similar to the Trie structure used by Wei et al. (2014) for the 3L-CVRP and Wei et al. (2015) for the 2L-CVRP, the Trie not only stores the feasibility information of the routes, but also the number of times the LocalSearchPack is called on this route (callNum). Given a route R, we first retrieve the stored information from the Trie. If the stored feasibility information is TRUE or callNum reaches the maximum allowed number maxCallTime, the stored feasibility information will be returned; otherwise, we call the LocalSearchPack on this route again, and update the feasibility information with the new results. The number callNum is updated in the following way. If the number of items demanded by the customers in R is not larger than max_enumeration (user defined parameter used in Algorithm 3), we set the callNum to maxCallTime directly. This is because we have enumerated all of the permutations of items, there is no need to call the LocalSearchPack on this route again. Otherwise, we increase callNum by one. In our implementation, the maximum calling time maxCallTime is set to $\max(n, 100 \times (1 - V_I/(L \cdot W \cdot H)))$, where n is the number of items, V_I is the volume of all items, and $L \cdot W \cdot H$ is the volume of the vehicle surface. As a

larger ratio between the volumes of items and vehicle, lower the probability that a feasible loading pattern exists for this route, it is reasonable to spend less effort on this route, so we set maxCallTime to a smaller value.

The Trie could save time in three ways. First, for a route whose feasible loading pattern has already been found and stored, Trie can avoid unnecessary calls of LocalSearchPack on the same route. Second, limiting the number of calls of LocalSearchPack on the same route ensures that limited effort is spent on routes without feasible loading plans. Third, the dynamic setting of maximum allowed calling number allows our approach to focus on the promising solution.

5.1. Open space based heuristic

The new packing heuristic OpenSpaceHeuristic is given as Algorithm 4, which takes a sequence of items I as input. It places

Algorithm 2 OpenSpaceHeuristic procedure

```
OpenSpaceHeuristic( I )
    // Input: I, a set of ordered items
 1  S.x = 0, S.y = 0, S.z = 0, S.w = W, S.l = L, S.l = H
 2  spaceList = { S }
 3  while spaceList ≠ ∅ and I ≠ ∅
 4      i = first non-packed item in I
 5      Find the first space S that can accommodate i
 6      if such S is found
 7          Place i at S and remove S from spaceList
 8          Update the open space list spaceList and remove i from I
 9      else
10          Break the loop
11  return the number of packed items
```

the items in I one by one according to the given order by the input. During the whole loading process, it maintains a list of *open spaces* as candidate positions for an item. The *open space* is a cuboid space with one plane coinciding with the rear door. The coordinate of an open space is defined by the corner closest to the origin, and the size is defined by the size of the cuboid. An open space S is described by the following six attributes:

- x, y, z: the coordinates of the corner closest to the origin ; and
- w, l, h: the width, length, and height of S.

An open space a is said to be dominated by another open space b if they share the same coordinate but a is totally contained by b, i.e., the length (resp. width, height) of a is not larger than that of b.

Initially, the vehicle surface is represented by an open space with size $L \times W \times H$ located at the original point. Each iteration of the loading process first sorts the open spaces in ascending order, first by z-coordinate, then by x-coordinate, then by y-coordinate. Then, for the first non-packed item i, the spaces are successively examined to see whether i can be accommodated. If there is an adequate space, the item i is placed at the deepest-bottom-left corner of this space, the space list is updated, and the process continues to the next item. If no such space exists, the process is terminated and the number of packed items is returned. The above process is repeated until all items are placed or an item fails to find a space to accommodate it. The number of packed items is returned at last.

Each time an item i is placed at the open space S, the open space list is updated in two steps. In the first step, the space S is removed from the open space list and up to three sub-spaces of S are introduced, as shown in Fig. 4. These three sub-spaces are generated by selecting five planes (must include the rear door) from S and then selecting one plane from i to form a cuboid.

Note that the placed item i may intersect with other open spaces in the list. Thus, in the second step, we check each space S' in the open space list. If S' intersects with i, we remove S' from the list and introduce up to four sub-spaces of S', as shown in Fig. 5. These four sub-spaces (S'_1, S'_2, S'_3, and S'_4 in Fig. 5) are generated by selecting five planes (must include the rear door) from S' and then selecting one plane from i to form a cuboid. In Fig. 5, S'_1 is formed by all the planes except the bottom of S' and the top plane of i. S'_2 is formed by all the planes except the left of S' and the right plane of i. S'_3 is formed by all the planes except the right of S' and the left plane of i. S'_4 is formed by all the planes except the back of S' and the front plane of i.

Intuitively, the dominated open space is redundant. Thus, before we add an open space S to the open space list, we check whether S is dominated by any open space already in the list. S is added to the list only if it is not dominated by another space in the list.

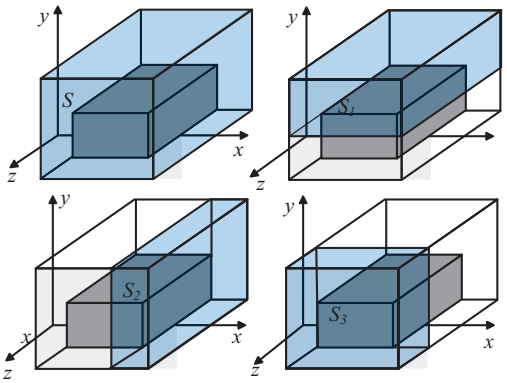


Fig. 4. Step 1 of updating the open space after placing an item.

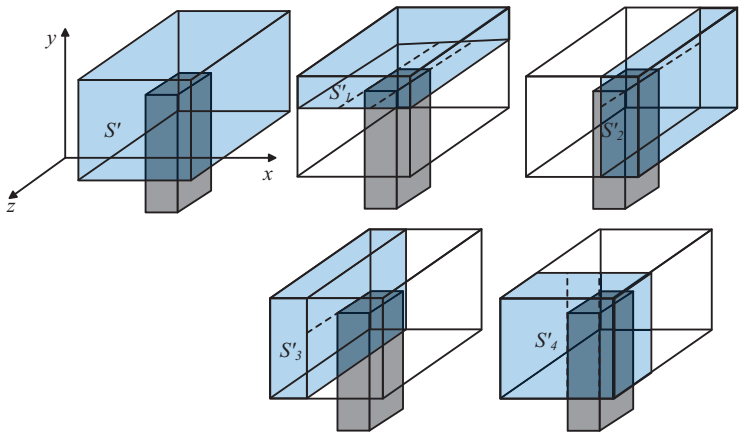


Fig. 5. Step 2 of updating the open space after placing an item.

The *open space* is similar to the *maximal space* proposed by Parreno et al. (2008) for the single container loading problem. The *maximal space* is the largest rectangular box that is interior-disjoint with the placed items. However, there are two major differences. First, one of the planes of an open space must lie in the rear door of the container, but no such constraint is required for maximal space. Second, the domination rule is different. A maximal space a is dominated by b if a is contained by b. So if a is contained by b but their coordinates are different, a is dominated by b if they are maximal spaces but a is not dominated by b if they are open spaces.

6. Computational results

The instances of 3L-CVRP consider many diverse attributes of customer distribution and item dimensions, and has been widely used as a benchmark for other algorithms. Thus, 3L-CVRP instances are also used to test our new 3L-FCVRP, but with a different objective function. A brief description of the benchmark is given in Section 6.1. We assess the performance of the proposed algorithm on 3L-CVRP in Section 6.2 and report the results of 3L-FCVRP in Section 6.3. We compare the solutions of 3L-FCVRP and 3L-CVRP in Section 6.4.

Our algorithm is coded in C++ and executed on an Intel Xeon E5430 with a 2.66 GHz (Quad Core) CPU and 8 GB RAM running the CentOS 5 Linux operating system. In the experiments, each instance is assigned a time limit according to its customer number N: 900 CPU seconds for $N \leq 25$, 1800 CPU seconds for $25 < N < 50$, and 3600 s for $N \geq 50$. The ELS algorithm is run 10 times by setting the random seed as 1 to 10 for each instance. The support ratio α is set to 0.75 as in previous studies. The user defined parameter max_enumeration used in the LocalSearchPack procedure is set to 8. The setting of these parameters are shown in Table 1. Other parameters are set in reasonable ranges, so we do not spend much time on tuning their values. All of the test data and the detailed computational results for both routing and loading plans can be downloaded from our website (Zhang et al., 2015).

6.1. Benchmark instances

There are two sets of benchmark data. The first set includes 27 instances and was introduced by Gendreau et al. (2006). The second set was proposed by Tarantilis et al. (2009); it contains 12 instances and has more variation in customer size and item dimensions.

The first set was derived from classic CVRP data by introducing 3-D items for each customer. More precisely, the length L, width W, and height H of vehicles are set to 60, 25, and 30, respectively. The customer distribution and demand are the same as in the CVRP instance. However, for each customer, the item number is a random number within the [1, 3] interval, and the item dimensions are randomly generated in the interval between 20% and 60% of the corresponding vehicle edges. Each item is given a 20% probability of being fragile.

The second set contains 50–125 customers located at $(r\cos(\theta), r\sin(\theta))$, and the depot location is (0,0); r and θ are randomly selected from [10, 100] and $[0, 2\pi]$ respectively. The vehicle dimensions are set as $L = 60$, $W = 30$, and $H = 30$, respectively. There are three classes of items: Class 1 contains small items with dimensions equal to 20%–40% of the corresponding vehicle dimensions; Class 2 contains large items whose dimensions are within the interval [40%, 60%] of vehicle dimensions; and Class 3 contains diverse items by setting the factor interval as [10%, 70%]. Each item has a 20% probability of being fragile. The number of items required by each customer is randomly taken from the intervals [2, 4], [1, 2], and [1, 3] for classes 1, 2, and 3, respectively. The demand of each customer is randomly distributed in the interval [5, 35].

6.2. Results for the 3L-CVRP

The 3L-CVRP can be seen a special case of our 3L-FCVRP if the fuel consumption is only related to the travel distance. Thus, the proposed algorithm is compared with algorithms for the 3L-CVRP available in the literature, including TS (Gendreau et al., 2006), GTS (Tarantilis et al., 2009), ACO (Fuellerer et al., 2010), DMTS (Zhu et al., 2012), VRLH1 (Bortfeldt, 2012), HA (Ruan et al., 2013), TS-ILA (Tao and Wang, 2015), and AVNS (Wei et al., 2014). The computational environments for these approaches are summarized in Table 7 in Appendix A. Similar to ACO, DMTS, VRLH1, TS-ILA, and AVNS, we run the ELS 10 times, and we compare the average distance obtained for each instance during the ten runs.

Table 2 compares the average distances of the two sets of instances; the best solutions are marked in bold. We also test other versions of the problem by removing one or two of the constraints. For each version, the average distance of all instances in each set is listed. From this table, we can see that our approach outperforms all existing approaches in all test versions. The constraint that affects the ELS the most is the LIFO and the one that affects the ELS the least is fragility, which are similar to the results for ACO and DMTS. Note that for the AVNS, we do not list the results of the set 2, as some of the final solutions (the instances 9 and 12 in set 2) use more vehicles than the limit.

Tables 3 and 4 compare the detailed results of each instance in sets 1 and 2 of the 3L-CVRP respectively; the best solution is marked in bold. To save space, we only report the detailed results of the selected approaches that gave excellent performances. The column d_{avg} is the average distance of each instance found by each approach, and ttb(s) is the average time to find the final solution. The column BKS gives the best-known d_{avg} found by previous algorithms. For our approach, we also report a column

Table 1
Parameter setting.

Parameters	Explanation	Value
Time limit	$N \leq 25$	900 s
	$25 < N < 50$	1800 s
	$N \geq 50$	3600 s
Running times		10
Random seed		1–10
α	Support ratio	0.75
Max_enumeration	Parameter used in LocalSearchPack	8

Table 2
Comparison of the average distance between the ELS and existing approaches in the 3L-CVRP instances set.

Set	Constraints	GTS	ACO	DMTS	VRLH1	HA	TS-ILA	AVNS	ELS
1	All	997.18	966.67	962.08	953.79	960.10	941.59	942.12	**928.96**
1	No fragility	965.14	945.04	941.08	934.38	935.92	924.12	920.00	**909.77**
1	No support	934.96	919.69	913.43	902.55	903.18	881.62	892.04	**877.43**
1	No LIFO	950.59	916.25	887.71	912.05	912.47	892.67	874.32	**868.69**
1	Loading only	876.39	856.66	846.44	864.54	858.07	840.41	842.13	**839.88**
2	All	3240.71	–	3181.66	3363.81	–	3175.01	–	**3052.43**
2	No fragility	3164.46	–	–	3199.85	–	3059.33	–	**2937.24**
2	No support	3240.71	–	–	2963.56	–	2906.26	–	**2840.01**
2	No LIFO	3240.71	–	–	2947.87	–	2893.07	–	**2816.11**
2	Loading only	3240.71	–	–	2846.16	–	2806.09	–	**2751.81**

Table 3
Comparison between ELS and existing approaches in 3L-CVRP set 1 instances. All constraints are imposed.

Id	Name	VRLH1		HA		TS-ILA		AVNS		BKS	ELS		
		d_{avg}	ttb(s)	d_{avg}	ttb(s)	d_{avg}	ttb(s)	d_{avg}	ttb(s)	d_{avg}	d_{avg}	ttb(s)	gap(%)
1	E016−03m	302.02	72.3	303.21	98.9	302.02	53.5	302.02	66.3	302.02	302.02	3.2	0.00
2	E016−05m	334.96	0.9	334.96	4.6	334.96	6.3	334.96	0.2	334.96	334.96	0.2	0.00
3	E021−04m	401.44	182.0	398.05	93.9	**381.37**	116.2	387.84	156.8	**381.37**	385.53	365.6	1.09
4	E021−06m	437.54	16.1	440.68	46.8	437.19	14.0	437.19	27.3	437.19	437.19	20.3	0.00
5	E022−04g	451.03	182.6	452.56	64.0	**436.79**	149.3	447.58	425.8	**436.79**	443.17	151.9	1.46
6	E022−06m	498.38	23.6	498.56	196.9	**498.32**	31.6	501.20	7.4	**498.32**	501.06	3.4	0.55
7	E023−03g	772.49	133.1	790.23	317.0	**768.94**	84.9	771.02	66.8	**768.94**	771.07	19.5	0.28
8	E023−05s	821.35	139.1	820.67	98.9	**805.77**	120.4	808.55	199.3	**805.77**	813.13	325.2	0.91
9	E026−08m	645.81	24.3	635.5	353.1	631.68	13.7	630.73	1.9	630.13	630.13	6.3	0.00
10	E030−03g	827.29	175.1	836.21	410.9	828.99	258.6	828.60	979.1	827.29	**824.69**	473.5	−0.31
11	E030−04s	815.62	136.4	825.75	197.8	780.61	278.9	776.25	614.0	776.25	**776.19**	220.5	−0.01
12	E031−09h	630.46	14.0	626.59	89.5	614.60	145.8	610.23	28.7	610.23	**610.23**	21.6	−0.00
13	E033−03n	2694.81	268.4	2739.8	319.8	**2636.85**	369.4	2703.62	843.3	**2636.85**	2656.72	495.0	0.75
14	E033−04g	1413.59	311.6	1469.38	268.4	1398.77	588.1	1436.25	962.7	1398.77	**1369.22**	1079.4	−2.11
15	E033−05s	1355.5	311.5	1369.69	356.6	1352.76	615.9	1351.38	1062.5	1351.38	**1338.35**	1295.0	−0.96
16	E036−11h	705.05	3.4	703.15	431.7	698.92	4.0	698.61	4.0	698.61	698.61	6.0	0.00
17	E041−14h	917.96	2.5	872.05	374.8	866.40	9.5	866.40	167.0	866.40	866.40	17.3	0.00
18	E045−04f	1228.98	309.5	1250.86	325.7	1228.47	1634.1	1240.87	1290.2	1228.47	**1223.64**	1104.4	−0.39
19	E051−05e	753.87	416.5	780.37	1374.8	763.09	718.4	750.17	2138.0	750.17	**744.33**	1494.9	−0.78
20	E072−04f	596.42	427.0	605.59	1337.0	590.99	2941.8	579.50	2324.3	579.50	**570.82**	2441.6	−1.50
21	E076−07s	1107	443.4	1119.45	1247.9	1096.53	2301.4	1086.26	2581.0	1086.26	**1063.20**	2510.8	−2.12
22	E076−08s	1171.49	423.5	1167.28	1294.6	1155.81	1241.8	1164.65	2196.9	1155.81	**1140.54**	2689.2	−1.32
23	E076−10e	1135.46	425.8	1171.77	1105.7	1130.08	1924.9	1117.77	2163.8	1117.77	**1094.33**	2887.9	−2.10
24	E076−14s	1128.82	411.1	1136.27	2001.1	1122.80	2526.8	1116.34	2053.1	1116.34	**1101.67**	2282.7	−1.31
25	E101−08e	1428.8	453.0	1426.34	1458.8	1417.09	4536.2	1391.74	2333.5	1391.74	**1359.25**	2846.1	−2.33
26	E101−10c	1625.31	430.6	1585.46	3354.7	1605.11	3017.7	1584.44	2619.0	1584.44	**1545.67**	2958.3	−2.45
27	E101−14s	1550.85	435.0	1562.18	3140.2	1538.10	6025.7	1512.92	2231.9	1512.92	**1479.73**	3065.5	−2.19
Avg		953.79	228.6	960.10	754.2	941.59	1101.1	942.12	1020.2	936.47	**928.96**	1066.1	−0.55

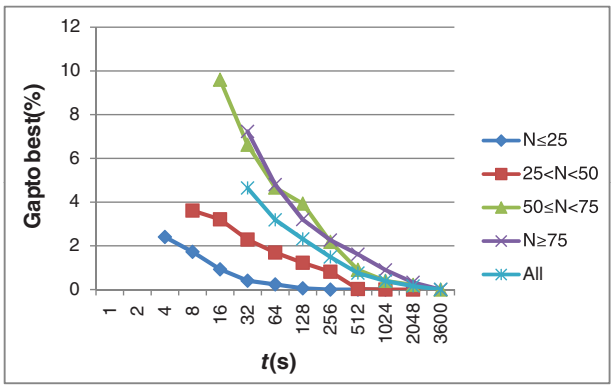
Table 4
Comparison between ELS and existing approaches in 3L-CVRP set 2 instances. All constraints are imposed.

Id	Name	DMTS		VRLH1		TS-ILA		AVNS		BKS	ELS		
		d_{avg}	ttb(s)	d_{avg}	ttb(s)	d_{avg}	ttb(s)	d_{avg}	ttb(s)	d_{avg}	d_{avg}	ttb(s)	gap(%)
1	50–1	1451.14	1404.4	1456.30	407.7	1434.56	1849.00	1417.88	384.8	1417.88	1417.88	83.8	0.00
2	50–2	2222.50	390.0	2225.20	113.6	2246.71	889.60	2196.95	1464.6	2196.95	**2190.52**	51.7	−0.29
3	50–3	1817.52	3356.8	1821.90	416.6	1761.66	1969.30	1730.25	2038.6	1730.25	**1689.78**	2555.8	−2.34
4	75–1	2078.00	2565.0	2067.47	415.8	2068.50	1179.60	2013.00	2196.4	2013.00	**2009.39**	2399.6	−0.18
5	75–2	3166.69	858.2	3078.92	334.5	3047.06	1469.50	**3006.27**	1876.3	**3006.27**	3033.81	2839.1	0.92
6	75–3	2540.58	5019.7	2534.36	415.5	2512.62	3217.30	2376.77	2180.6	2376.77	**2355.00**	2935.3	−0.92
7	100–1	2613.04	7448.5	2645.02	469.3	2598.52	6029.70	2509.44	1919.5	2509.44	**2486.82**	3000.1	−0.90
8	100–2	4237.80	835.3	4269.22	207.3	4254.91	1436.30	4200.80	1636.0	4200.80	**4188.86**	2374.2	−0.28
9	100–3	4263.59	9639.5	5007.87	428.5	4248.79	7128.40	–	–	4190.92	**3785.60**	2861.8	−9.67
10	125–1	3260.23	8416.3	3294.00	441.7	3273.37	3130.00	3176.37	1873.7	3176.37	**3159.33**	1990.1	−0.54
11	125–2	5450.58	1854.6	5563.91	320.3	5536.29	4180.00	**5373.55**	1158.6	**5373.55**	5396.24	2141.6	0.42
12	125–3	5078.30	14219.3	6401.59	439.3	5093.25	7936.20	–	–	5078.30	**4915.93**	1993.5	−3.20
Avg		3181.66	4667.3	3363.81	367.5	3173.02	3514.90	–	–	3105.88	**3052.43**	2102.2	−1.42

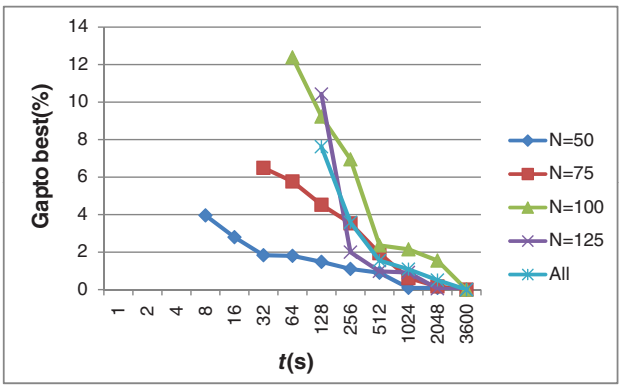
gap(%), which is the relative gap between the solution found by our approach and the BKS. For example, assuming that the BKS for an instance is d_{best} and the average distance found by the ELS is d_{ELS}, then gap(%) is calculated as $100 \times (d_{ELS}/d_{best} - 1)$. We can see that ELS improves the solutions for 15 of the 27 instances in set 1 and 9 of the 12 instances in set 2. According to these two tables, our algorithm can find the best solution in a reasonable length of time.

The detailed results obtained by the ELS for all instances in sets 1 and 2 of the 3L-CVRP are given in Appendix B.

To investigate the convergence behavior of the ELS, we perform an additional experiment on these instances (the random seed is set to 1). For each instance, the solution found by the ELS at the time of $1, 2, 4, \ldots, 2048, 3600$ s is recorded. The results are shown as Fig. 6, where the instances are grouped by the number of customers N. For each group, we plot the gap between the average distance and final average distance as the computational time increases. More specifically, assuming that the final average distance found by the ELS is d_{final} and the average distance found at time t is d_t, then the gap(%) is defined as $100 \times (d_t/d_{final} - 1)$. These figures show that the ELS exhibits rapid convergence for most of these instances. The ELS converges after 64 s on all small instances with $N \leq 25$, and after 512 s in the medium instances with $25 < N < 50$. For the instances with $N \geq 50$, the ELS converges after 2048 s in most of the instances.

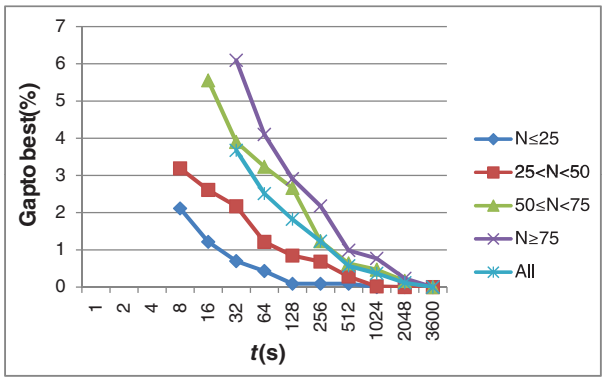


(a) Convergence of the ELS in data set 1

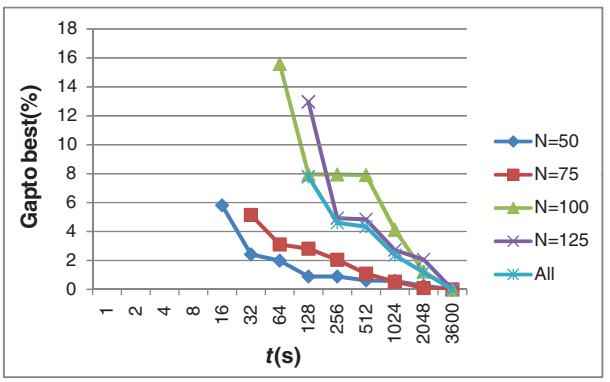


(b) Convergence of the ELS in data set 2

Fig. 6. Convergence of the ELS in the 3L-CVRP data sets.



(a) Convergence of the ELS in data set 1



(b) Convergence of the ELS in data set 2

Fig. 7. Convergence of the ELS in the 3L-FCVRP data sets.

6.3. Results for the 3L-FCVRP

As the 3L-FCVRP is developed in this study, there are no previous results for comparison. Here, we report the detailed results of our experimental tests. Table 5 provides the minimum fuel consumption fc_{bst}, average fuel consumption fc_{avg}, and the average time to find the best solution $ttb(s)$ over 10 runs for all versions of each instance. The row gap_1 is calculated as $(fc_{avg} - fc_{bst})/fc_{bst} * 100$ for each problem version. The results demonstrate that our algorithm is rather stable for this problem, especially for the loading-only version with the smallest gap 0.17%. The largest gap (0.85%) occurs in the version with all constraints. To investigate the effect of loading constraints on the final solution, the improvement gap_2 of each version over the version with all constraints is calculated, such as $(fc_{avg}^{No fragility}/fc_{avg}^{All constraints} - 1) * 100$ for the No fragility version. The results show that fragility is the least influential constraint with an average improvement of 2.99%, followed by the support constraint (6.02%). The LIFO is the hardest constraint, and solutions without LIFO can be improved by 7.33%. Finally, when only the 3D-loading constraint is imposed, the improvement reaches 9.18% on average.

A similar method is used to investigate the convergence behavior of the ELS in the 3L-FCVRP. The results are shown as Fig. 7. The algorithm converges rapidly in small-scale instances, whose best solutions are found in about 128 s. It converges after 1024 s in medium instances. For larger instances, the solutions continue to be improved after 2048 s.

6.4. Comparison of solutions to the 3L-FCVRP and 3L-CVRP

To evaluate the improvements of the 3L-FCVRP, it is necessary to calculate the fuel consumption in the 3L-CVRP solutions, and to compare them to the results of the 3L-FCVRP. The comparison is shown in Table 6, where the average fuel consumption is given in columns 3L-FCVRP and 3L-CVRP. The improvements are at least 3.16%, and can reach 5.07%. The largest improvements are in the versions without LIFO constraints, such as the No LIFO and Loading Only versions, followed by the No Support and No Fragility versions. The least improvement is obtained in the version that considers all constraints. In general, more improvements

Table 5
Detailed results of the ELS in the 3L-FCVRP instances.

Inst.	All constraints			No fragility			No LIFO			No support			Loading only		
	fc_{bst}	fc_{avg}	ttb(s)	fc_{bst}	fc_{avg}	ttb(s)	fc_{bst}	fc_{avg}	ttb(s)	fc_{bst}	fc_{avg}	ttb(s)	fc_{bst}	fc_{avg}	ttb(s)
1	396.93	396.93	3.8	391.59	391.59	1.5	385.40	385.40	0.3	385.40	385.40	0.3	385.40	385.40	0.2
2	466.34	466.34	0.3	466.34	466.34	0.2	466.34	466.34	0.1	466.34	466.34	0.1	466.34	466.34	0.1
3	545.02	545.02	63.3	530.14	530.14	14.6	500.03	500.03	5.7	499.86	500.47	286.8	499.86	499.86	1.5
4	612.28	612.28	0.9	612.28	612.28	0.4	607.95	607.95	0.4	607.95	607.95	0.2	607.95	607.95	0.2
5	583.37	585.00	275.0	579.64	581.29	170.5	553.47	553.47	22.1	561.72	561.72	94.1	536.54	537.04	0.4
6	704.50	704.50	4.7	693.42	693.42	0.5	693.42	693.42	0.5	693.42	693.42	0.7	693.42	693.42	0.5
7	896.20	896.20	6.8	874.25	874.25	5.0	842.63	842.63	2.2	864.38	864.38	3.0	842.63	842.63	1.5
8	928.97	928.97	4.1	921.27	927.27	355.1	861.82	861.82	64.8	908.65	908.65	45.9	857.12	857.12	9.2
9	892.17	892.17	7.0	892.17	892.17	4.0	888.43	888.43	3.1	892.17	892.17	2.7	888.43	888.43	2.1
10	982.48	982.48	618.5	933.91	943.73	1026.7	892.65	895.11	397.5	896.29	898.12	727.4	854.31	861.14	143.6
11	927.36	927.82	409.9	922.32	923.18	684.6	868.65	868.65	34.2	881.06	881.06	582.3	868.65	868.65	5.5
12	855.97	855.97	19.4	855.97	855.97	3.8	849.93	849.93	21.1	852.58	852.58	641.5	849.93	849.93	1.6
13	3141.94	3141.94	55.5	3073.46	3090.67	860.1	2905.45	2911.53	556.7	2936.10	2951.61	794.7	2774.74	2779.54	18.2
14	1710.65	1724.29	1469.0	1675.78	1677.12	443.6	1618.97	1628.55	883.4	1642.13	1644.29	1149.3	1508.40	1508.70	866.1
15	1682.38	1682.38	613.7	1663.39	1664.83	1067.4	1498.92	1509.41	1385.9	1535.03	1594.94	1373.1	1471.92	1475.98	578.0
16	976.49	976.49	2.2	976.49	976.49	2.4	976.49	976.49	2.2	976.49	976.49	2.9	976.49	976.49	2.5
17	1230.19	1230.19	167.9	1225.62	1225.62	53.0	1222.95	1222.95	14.4	1222.95	1222.95	14.2	1222.95	1222.95	19.8
18	1372.17	1378.27	1242.7	1315.92	1326.39	1291.3	1298.97	1306.82	1193.6	1299.01	1307.49	1250.8	1245.64	1259.03	1091.4
19	906.33	908.27	1968.2	887.90	888.38	1995.9	837.45	839.74	1062.8	864.19	864.80	1195.1	821.18	827.88	1529.8
20	632.03	633.18	2407.8	614.66	614.87	1824.1	595.15	597.14	3035.8	596.52	599.15	2915.3	560.58	561.90	1899.4
21	1270.44	1277.54	2566.7	1238.33	1245.48	2890.0	1149.67	1158.44	3015.4	1185.63	1191.95	2840.3	1132.51	1136.24	2290.9
22	1388.73	1391.75	2343.6	1363.83	1364.99	2493.3	1279.60	1283.52	2506.0	1316.21	1319.96	2458.8	1248.26	1254.53	2629.2
23	1409.81	1418.27	2765.4	1393.07	1396.31	2513.2	1326.39	1328.32	2847.1	1339.71	1348.83	2493.5	1288.10	1291.03	2548.1
24	1544.25	1548.83	2565.5	1502.50	1506.98	1860.9	1462.62	1470.02	2560.2	1465.41	1473.41	1749.3	1456.95	1458.12	1689.1
25	1618.11	1621.18	2712.7	1579.29	1588.23	3035.0	1482.82	1496.58	2908.4	1509.02	1517.61	3064.4	1425.41	1432.59	2783.6
26	1879.26	1891.78	2731.6	1827.94	1832.12	2677.4	1768.82	1770.94	2271.0	1791.74	1795.91	2806.8	1687.77	1691.03	2380.7
27	1892.11	1908.75	2987.4	1858.46	1864.30	2466.6	1745.67	1747.79	2635.5	1779.63	1787.94	2683.5	1700.72	1706.26	2685.0
28	1991.37	1991.37	21.2	1991.37	1991.37	5.2	1991.37	1991.37	4.4	1991.37	1991.37	4.9	1991.37	1991.37	3.4
29	2710.33	2710.33	86.4	2655.72	2655.72	37.6	2574.26	2574.26	56.4	2625.34	2625.34	21.0	2541.08	2541.08	8.8
30	2257.97	2263.43	1827.5	2166.33	2192.92	2562.3	2107.80	2109.61	1125.4	2110.31	2110.31	1282.3	2062.09	2062.09	525.0
31	2857.64	2857.64	824.1	2857.64	2857.64	191.0	2848.50	2848.50	1199.3	2848.50	2850.78	1557.6	2848.50	2848.50	760.1
32	3912.02	3928.70	2628.5	3848.09	3861.59	2192.8	3639.39	3639.39	526.3	3801.90	3814.07	2539.6	3537.23	3537.29	839.7
33	3246.27	3264.52	2887.3	3185.23	3207.23	2624.4	3089.57	3093.14	2124.2	3120.49	3121.83	2070.1	3041.31	3041.34	251.4
34	3480.18	3498.78	2196.3	3472.35	3472.69	1702.1	3462.84	3466.29	2512.6	3470.72	3472.19	1307.3	3451.05	3451.06	1649.1
35	5227.16	5252.78	2491.6	5202.76	5212.71	2162.9	5047.71	5061.91	1745.7	5142.63	5153.44	2414.8	5003.36	5016.16	2510.5
36	4848.47	5031.59	2759.6	4410.08	4451.62	2654.2	4131.84	4175.47	2736.1	4129.96	4145.19	2953.2	3956.26	3975.43	2456.9
37	4491.69	4497.78	2451.6	4481.53	4490.19	2530.1	4477.86	4479.08	2698.9	4491.26	4492.50	2231.6	4477.86	4478.80	1789.8
38	6924.39	6985.64	2168.9	6804.32	6832.76	2092.3	6490.33	6540.41	1824.5	6674.36	6729.41	2663.4	6406.29	6417.08	2287.0
39	6670.87	6937.20	1854.7	5921.81	6148.86	2540.6	5125.67	5183.38	3311.2	5211.69	5272.81	2983.7	5023.34	5033.03	2598.0
Avg.	2052.95	2070.42	1287.5	1996.59	2008.45	1257.4	1912.00	1918.57	1212.7	1938.16	1945.87	1313.0	1877.23	1880.34	996.4
gap_1(%)		0.85			0.59			0.34			0.40			0.17	
gap_2(%)				−2.74	−2.99		−6.87	−7.33		−5.59	−6.02		−8.56	−9.18	

Table 6
Comparison of average fuel consumption in solutions for the 3L-FCVRP and 3L-CVRP.

Constraints	Set 1			Set 2		
	3L-FCVRP	3L-CVRP	gap(%)	3L-FCVRP	3L-CVRP	gap(%)
All constraints	1167.66	1204.61	3.16	4101.65	4244.52	3.48
No fragility	1146.46	1191.17	3.90	3947.94	4084.34	3.46
No LIFO	1098.57	1148.50	4.54	3763.57	3954.33	5.07
No support	1115.17	1155.39	3.61	3814.94	3967.74	4.01
Loading only	1071.86	1124.12	4.88	3699.44	3881.57	4.92

are obtained in set 2, except in the No Fragility version. Thus, the results demonstrate that it is deserved to study the 3L-FCVRP specially.

Table 7
Computational environments.

Algorithm	CPU	RAM	Running times
TS (Gendreau et al., 2006)	Pentium IV 3 GHz	512 MB	1
GTS (Tarantilis et al., 2009)	Pentium IV 2.8 GHz	1 GB	1
ACO (Fuellerer et al., 2010)	Pentium IV 3.2 GHz	2 GB	10
DMTS (Zhu et al., 2012)	Xeon E5520 2.26 GHz	8 GB	10
VRLH1 (Bortfeldt, 2012)	Intel 3.17 GHz PC(Core2 Duo E8500)	2 GB	10
HA (Ruan et al., 2013)	Pentium IV 2.3 GHz	1 GB	1
TS-ILA (Tao and Wang, 2015)	Intel (R) Pentium IV 2.67 GHz	4 GB	10
AVNS (Wei et al., 2014)	Intel Xeon E5430 with a 2.66 GHz (Quad Core) CPU	8 GB	10
ELS	Intel Xeon E5430 with a 2.66 GHz (Quad Core) CPU	8 GB	10

7. Conclusions

This study introduces and tests the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints (3L-FCVRP). It aims to reduce carbon emissions and practical operating costs. Fuel consumption is closely related to travel distance and total weight of the vehicle. To make the resulting plan practical, we simultaneously consider fuel consumption and loading freight into vehicle parameters. As this problem combines the well-known routing and loading problems, it is a challenging NP-hard problem of high complexity.

To solve this highly constrained problem, the evolutionary local search (ELS) framework is used to intensively investigate the solution space, and the recombination method is incorporated to help the search jump to new promising regions. The route pool stores the routes of the local optima, which are then probabilistically selected to construct new promising solutions. These solutions become the new starting points of the search, so the search trajectory is adjusted timely in a manner. The ELS works well with the recombination procedure, as it produces various local optima that diversify the pool. To guarantee that the solutions are loading-feasible, an open space based heuristic is invoked to identify the loading plan. In addition, two special data structures, Trie and Fibonacci heap, are adopted to speed up the algorithm. The proposed algorithm is assessed first on the 3L-CVRP, which is a special case of 3L-FCVRP. The results demonstrate that our algorithm outperforms all of the other methods and improves most of the best-known solutions. Furthermore, the required runtime is short. Thus, our algorithm is quite effective and efficient for this problem.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (grant no. 71401065) and the Natural Science Foundation of Jiangxi Province (grant no. 20151BAB217006).

Appendix A. Computational environments of the approaches to the 3L-CVRP

See Table 7.

Appendix B. Details results of the ELS of each instance in sets 1 and 2 of the 3L-CVRP

Tables 8 and 9 list the detailed results of the ELS of each instance in sets 1 and 2 of the 3L-CVRP, respectively. The columns d_{bst}, d_{avg}, and *ttb*(s) are the minimum distance, average distance, and average time to find the best solution for each instance, respectively. The row *avg.gap*(%) is the average *gap*(%) in all instances, where the *gap*(%) of an instance is calculated as $100 \times (d_{avg}/d_{bst} - 1)$. The results show that our algorithm is rather stable for the 3L-CVRP. The largest *avg.gap*(%) is 1.50 for the All Constraints version in set 2.

References

Bektaş, T., Laporte, G., 2011. The pollution-routing problem. Transportation Research Part B 45, 1232–1250.
Bortfeldt, A., 2012. A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. Computers & Operations Research 39, 2248–2257.
Chang, P.-C., Huang, W.-H., Wu, J.-L., Cheng, T., 2013. A block mining and re-combination enhanced genetic algorithm for the permutation flowshop scheduling problem. International Journal of Production Economics 141, 45–55.
Chang, P.-C., Huang, W.-H., Zhang, Z.-Z., 2012. A puzzle-based genetic algorithm with block mining and recombination heuristic for the traveling salesman problem. Journal of Computer Science and Technology 27, 937–949.
Chazelle, B., 1983. The bottomn-left bin-packing heuristic: an efficient implementation. IEEE Transactions on Computers C-32, 697–707.
Clarke, G., Wright, J.W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568–581.
Doerner, K.F., Fuellerer, G., Hartl, R.F., Gronalt, M., Iori, M., 2007. Metaheuristics for the vehicle routing problem with loading constraints. Networks 49, 294–307.
Duhamel, C., Lacomme, P., Quilliot, A., Toussaint, H., 2011. A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle routing problem. Computers & Operations Research 38, 617–640.
Franceschetti, A., Honhon, D., Woensel, T.V., Bektaş, T., Laporte, G., 2013. The time-dependent pollution-routing problem. Transportation Research Part B 56, 265–293.
Fredman, M.L., Tarjan, R.E., 1987. Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM 34, 596–615.

Fuellerer, G., Doerner, K.F., Hartl, R.F., Iori, M., 2009. Ant colony optimization for the two-dimensional loading vehicle routing problem. Computers & Operations Research 36, 655–673.
Fuellerer, G., Doerner, K.F., Hartl, R.F., Iori, M., 2010. Metaheuristics for vehicle routing problems with three-dimensional loading constraints. European Journal of Operational Research 201, 751–759.
Gendreau, M., Iori, M., Laporte, G., Martello, S., 2006. A tabu search algorithm for a routing and container loading problem. Transportation Science 40, 342–350.
Gendreau, M., Iori, M., Laporte, G., Martello, S., 2008. A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. Networks 51, 4–18.
Harris, I., Naim, M., Palmer, A., Potter, A., Mumford, C., 2011. Assessing the impact of cost optimization based on infrastructure modelling on CO_2 emissions. International Journal of Production Economics 131, 313–321.
Iori, M., Martello, S., 2010. Routing problems with loading constraints. TOP 18, 4–27.
Iori, M., Martello, S., 2013. An annotated bibliography of combined routing and loading problems. Yugoslav Journal of Operations Research 23, 311–326.
Iori, M., Salazar-González, J.-J., Vigo, D., 2007. An exact approach for the vehicle routing problem with two-dimensional loading constraints. Transportation Science 41, 253–264.

Table 8
Detailed results of the ELS for 3L-CVRP set 1 instances.

Id	Name	All constraints			No fragility			No support			No LIFO			Loading only		
		d_{bst}	d_{avg}	ttb(s)	d_{bst}	d_{avg}	ttb(s)	d_{bst}	d_{avg}	ttb(s)	d_{bst}	d_{avg}	ttb(s)	d_{bst}	d_{avg}	ttb(s)
1	E016−03m	302.02	302.02	3.202	302.02	302.02	1.3	298.70	298.70	0.8	297.65	298.60	0.3	297.65	297.65	0.3
2	E016−05m	334.96	334.96	0.179	334.96	334.96	0.2	334.96	334.96	0.1	334.96	334.96	0.1	334.96	334.96	0.1
3	E021−04m	385.53	385.53	365.607	373.01	378.26	19.3	362.27	362.27	61.4	362.27	362.27	5.6	362.27	362.27	1.5
4	E021−06m	437.19	437.19	20.264	430.89	430.89	35.5	430.89	430.89	0.3	430.89	430.89	0.2	430.89	430.89	0.1
5	E022−04g	442.14	443.17	151.91	431.54	431.97	385.8	395.64	408.27	437.5	406.50	406.50	7.7	395.64	395.96	2.7
6	E022−06m	501.06	501.06	3.438	495.85	495.85	4.1	495.85	495.85	1.3	495.85	495.85	0.8	495.85	495.85	0.2
7	E023−03g	771.07	771.07	19.537	759.96	759.96	6.1	750.38	750.69	283.7	732.52	732.52	1.8	725.44	731.81	1.7
8	E023−05s	811.17	813.13	325.198	798.61	801.82	284.5	775.77	777.56	375.1	735.14	735.14	63.5	730.66	730.66	10.1
9	E026−08m	630.13	630.13	6.26	630.13	630.13	4.7	630.13	630.13	1.8	630.13	630.13	3.2	630.13	630.13	1.6
10	E030−03g	823.46	824.69	473.484	765.89	771.74	1301.2	745.53	747.08	279.0	739.89	741.61	315.9	706.30	707.10	218.3
11	E030−04s	776.19	776.19	220.536	767.89	767.89	790.6	720.97	720.97	125.9	718.25	718.25	17.8	718.25	718.25	3.3
12	E031−09h	610.23	610.23	21.595	610.23	610.23	9.3	610.00	610.00	20.5	610.00	610.00	12.0	610.00	610.00	4.2
13	E033−03n	2648.93	2656.72	494.993	2605.25	2609.44	1257.0	2458.83	2458.83	342.4	2437.62	2437.62	429.6	2306.04	2309.75	121.9
14	E033−04g	1363.04	1369.22	1079.371	1342.34	1344.24	1057.3	1294.41	1299.79	1201.4	1286.41	1290.85	1215.0	1184.27	1186.05	427.5
15	E033−05s	1338.22	1338.35	1295.009	1323.41	1328.35	1158.5	1202.91	1209.56	1281.7	1182.11	1184.24	581.2	1149.92	1155.60	656.9
16	E036−11h	698.61	698.61	6.011	698.61	698.61	3.2	698.61	698.61	4.0	698.61	698.61	2.9	698.61	698.61	3.8
17	E041−14h	866.40	866.40	17.304	866.40	866.40	18.0	863.27	863.27	18.9	866.40	866.40	5.9	861.79	861.79	11.2
18	E045−04f	1201.85	1223.64	1104.383	1163.62	1179.60	866.4	1130.38	1136.05	1176.8	1129.33	1141.98	1366.6	1092.01	1096.54	1180.9
19	E051−05e	741.64	744.33	1494.948	713.01	714.90	2262.0	688.56	690.68	1985.8	672.58	678.01	1573.7	656.96	660.28	1216.0
20	E072−04f	570.17	570.82	2441.629	555.64	556.75	1903.7	538.90	540.60	2745.4	538.46	539.36	2889.6	503.90	505.33	2574.5
21	E076−07s	1044.26	1063.20	2510.777	1023.10	1030.40	2738.7	984.07	985.63	2008.8	958.39	969.85	3086.3	923.74	928.93	2402.9
22	E076−08s	1133.06	1140.54	2689.198	1117.30	1118.36	2268.9	1068.49	1072.92	3116.5	1039.11	1045.79	2722.5	1001.63	1004.60	2184.5
23	E076−10e	1084.17	1094.33	2887.943	1071.15	1076.97	2384.0	1018.30	1024.90	2710.5	1002.78	1007.98	2506.9	959.49	960.44	1353.7
24	E076−14s	1092.46	1101.67	2282.749	1068.96	1070.09	1913.9	1053.60	1054.94	2385.0	1048.21	1049.20	1611.1	1035.80	1035.80	1228.9
25	E101−08e	1338.35	1359.25	2846.143	1314.32	1320.87	2646.2	1244.48	1255.54	2986.1	1240.39	1249.27	3152.4	1166.99	1175.95	3256.1
26	E101−10c	1530.05	1545.67	2958.288	1489.61	1494.94	2924.9	1461.86	1467.30	3074.6	1452.29	1457.23	2753.2	1353.48	1363.80	2573.5
27	E101−14s	1472.11	1479.73	3065.461	1433.80	1438.26	2625.1	1358.59	1364.59	3242.7	1337.74	1341.49	3013.0	1285.70	1287.72	2610.1
Avg.		924.02	928.96	1066.13	906.94	909.77	1069.3	874.68	877.43	1106.2	866.09	868.69	1012.5	837.72	839.88	816.5
Avg.gap(%)			0.41			0.29			0.33			0.26			0.21	

Table 9
Detailed results of the ELS for 3L-CVRP set 2 instances.

Id	Name	All constraints			No fragility			No support			No LIFO			Loading only		
		d_{bst}	d_{avg}	ttb(s)	d_{bst}	c_{avg}	ttb(s)	d_{bst}	d_{avg}	ttb(s)	d_{bst}	d_{avg}	ttb(s)	d_{bst}	d_{avg}	ttb(s)
1	50–1	1417.88	1417.88	83.8	1417.88	1417.88	31.5	1417.88	1417.88	26.3	1417.88	1417.88	9.6	1417.88	1417.88	4.1
2	50–2	2190.52	2190.52	51.7	2154.40	2154.40	77.7	2131.22	2131.22	67.8	2105.17	2105.17	19.3	2072.68	2072.68	48.8
3	50–3	1682.66	1689.78	2555.8	1627.17	1637.15	2583.1	1554.88	1556.10	1337.5	1581.92	1594.63	2084.7	1512.97	1521.63	75.6
4	75–1	2009.09	2009.39	2399.6	2003.57	2008.69	1296.5	2003.57	2003.57	240.4	2003.57	2003.57	69.4	2003.57	2003.57	27.3
5	75–2	3028.38	3033.81	2839.1	2966.88	2976.13	1703.3	2941.05	2942.38	1267.1	2827.22	2827.22	857.3	2725.97	2725.97	447.2
6	75–3	2325.44	2355.00	2935.3	2302.18	2330.02	2398.7	2244.63	2247.36	1215.0	2230.94	2236.90	1595.9	2195.84	2198.38	809.2
7	100–1	2472.15	2486.82	3000.1	2430.79	2443.21	1961.9	2434.11	2438.17	2594.7	2430.79	2431.53	1440.6	2423.29	2423.29	739.4
8	100–2	4176.55	4188.86	2374.2	4134.14	4150.42	2703.2	4085.77	4092.68	2613.5	4029.98	4034.25	2266.0	3966.25	3969.29	2549.4
9	100–3	3596.23	3785.60	2861.8	3270.43	3325.33	2758.5	3086.82	3104.79	2826.4	3118.75	3137.14	2945.2	2957.70	2962.42	2611.3
10	125–1	3156.69	3159.33	1990.1	3152.86	3155.45	2076.6	3150.15	3153.60	2800.6	3147.50	3149.26	2411.9	3147.50	3147.90	794.8
11	125–2	5326.31	5396.24	2141.6	5269.38	5310.75	1792.2	5211.78	5238.22	2121.0	5115.65	5140.23	2804.8	4984.33	5000.92	2630.5
12	125–3	4545.43	4915.93	1993.5	4202.78	4337.49	2600.1	3726.76	3754.17	3050.5	3669.17	3715.53	3195.2	3568.04	3577.82	2724.0
Avg.		2993.94	3052.43	2102.2	2911.04	2937.24	1831.9	2832.39	2840.01	1680.1	2806.55	2816.11	1641.7	2748.00	2751.81	1121.8
Avg.gap(%)			1.50			0.76			0.21			0.30			0.13	

Kara, Ä., Kara, B.Y., Yetis, M.K., 2007. Energy minimizing vehicle routing problem. In: Dress, A., Xu, Y., Zhu, B. (Eds.), Combinatorial Optimization and Applications. In: Lecture Notes in Computer Science, vol. 4616. Springer, Berlin Heidelberg, pp. 62–71.
Khebbache-Hadji, S., Prins, C., Yalaoui, A., Reghioui, M., 2013. Heuristics and memetic algorithm for the two-dimensional loading capacitated vehicle routing problem with time windows. Central European Journal of Operations Research 21, 307–336.
Koç, Ç., Bektaş, T., Jabali, O., Laporte, G., 2014. The fleet size and mix pollution-routing problem. Transportation Research Part B 70, 239–254.
Leung, S.C., Zhang, D., Sim, K.M., 2011a. A two-stage intelligent search algorithm for the two-dimensional strip packing problem. European Journal of Operational Research 215, 57–69.
Leung, S.C.H., Zhang, Z., Zhang, D., Hua, X., Lim, M.K., 2013. A meta-heuristic algorithm for heterogeneous fleet vehicle routing problems with two-dimensional loading constraints. European Journal of Operational Research 225, 199–210.
Leung, S.C.H., Zheng, J., Zhang, D., Zhou, X., 2010. Simulated annealing for the vehicle routing problem with two-dimensional loading constraints. Flexible Services and Manufacturing Journal 22, 61–82.
Leung, S.C.H., Zhou, X., Zhang, D., Zheng, J., 2011b. Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem. Computers & Operations Research 38, 205–215.
Li, X., Leung, S.C.H., Tian, P., 2012. A multistart adaptive memory-based tabu search algorithm for the heterogeneous fixed fleet open vehicle routing problem. Expert Systems with Applications 39, 365–374.
Lin, C., Choy, K., Ho, G., Chung, S., Lam, H., 2014. Survey of green vehicle routing problem past and future trends. Expert Systems with Applications 41, 1118–1138.
Lodi, A., Martello, S., Vigo, D., 1999. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. INFORMS Journal on Computing 11, 345–357.
Parreno, F., Alvarez-Valdes, R., Tamarit, J.M., Oliveira, J.F., 2008. A maximal-space algorithm for the container loading problem. INFORMS Journal on Computing 20, 412–422.
Piecyk, M.I., McKinnon, A.C., 2010. Forecasting the carbon footprint of road freight transport in 2020. International Journal of Production Economics 128, 31–42.
Prins, C., 2009. A grasp × evolutionary local search hybrid for the vehicle routing problem. In: Pereira, F., Tavares, J. (Eds.), Bio-inspired Algorithms for the Vehicle Routing Problem. In: Studies in Computational Intelligence, vol. 161. Springer Berlin Heidelberg, pp. 35–53.
Rochat, Y., Taillard, E.D., 1995. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1, 147–167.
Ruan, Q., Zhang, Z., Miao, L., Shen, H., 2013. A hybrid approach for the vehicle routing problem with three-dimensional loading constraints. Computers & Operations Research 40, 1579–1589.
Sahin, B., Yilmaz, H., Ust, Y., Guneri, A.F., Gulsun, B., 2009. An approach for analysing transportation costs and a case study. European Journal of Operational Research 193, 1–11.
Santos, L., Coutinho-Rodrigues, J., Current, J.R., 2010. An improved ant colony optimization based algorithm for the capacitated arc routing problem. Transportation Research Part B 44, 246–266.
Subramanian, A., Penna, P.H.V., Uchoa, E., Ochi, L.S., 2012. A hybrid algorithm for the heterogeneous fleet vehicle routing problem. European Journal of Operational Research 221, 285–295.
Tao, Y., Wang, F., 2015. An effective tabu search approach with improved loading algorithms for the 3L-CVRP. Computers & Operations Research 55, 127–140.
Tarantilis, C.D., Kiranoudis, C.T., 2002. Boneroute : an adaptive memory-based method for effective fleet management. Annals of Operations Research 115, 227–241.
Tarantilis, C.D., Kiranoudis, C.T., 2007. A flexible adaptive memory-based algorithm for real-life transportation operations: two case studies from dairy and construction sector. European Journal of Operational Research 179, 806–822.
Tarantilis, C.D., Zachariadis, E.E., Kiranoudis, C.T., 2009. A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem. IEEE Transactions on Intelligent Transportation Systems 10, 255–271.
Wang, F., Tao, Y., Shi, N., 2009. A survey on vehicle routing problem with loading constraints. In: Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization, pp. 602–606.
Wei, L., Zhang, D., Chen, Q., 2009. A least wasted first heuristic algorithm for the rectangular packing problem. Computers & Operations Research 36, 1608–1614.
Wei, L., Zhang, Z., Lim, A., 2014. An adaptive variable neighborhood search for a heterogeneous fleet vehicle routing problem with three-dimensional loading constraints. IEEE Computational Intelligence Magazine 9, 18–30.
Wei, L., Zhang, Z., Zhang, D., Lim, A., 2015. A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. European Journal of Operational Research 243, 798–814.
Xiao, Y., Zhao, Q., Kaku, I., Xu, Y., 2012. Development of a fuel consumption optimization model for the capacitated vehicle routing problem. Computers & Operations Research 39, 1419–1431.
Zachariadis, E.E., Kiranoudis, C.T., 2010. An open vehicle routing problem metaheuristic for examining wide solution neighborhoods. Computers & Operations Research 37, 712–723.
Zachariadis, E.E., Kiranoudis, C.T., 2011. A local search metaheuristic algorithm for the vehicle routing problem with simultaneous pick-ups and deliveries. Expert Systems with Applications 38, 2717–2726.
Zachariadis, E.E., Tarantilis, C.D., Kiranoudis, C.T., 2009. A guided tabu search for the vehicle routing problem with two-dimensional loading constraints. European Journal of Operational Research 195, 729–743.
Zachariadis, E.E., Tarantilis, C.D., Kiranoudis, C.T., 2012. The pallet-packing vehicle routing problem. Transportation Science 46, 341–358.
Zachariadis, E.E., Tarantilis, C.D., Kiranoudis, C.T., 2013. Designing vehicle routes for a mix of different request types, under time windows and loading constraints. European Journal of Operational Research 229, 303–317.
Zachariadis, E.E., Tarantilis, C.D., Kiranoudis, C.T., 2015. The load-dependent vehicle routing problem and its pick-up and delivery extension. Transportation Research Part B 71, 158–181.
Zhang, Z., Wei, L., Lim, A., 2015. Supplement for 'An evolutionary local search for the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints'. July 2015. [Online]. Available: www.computational-logistics.org/orlib/3l-fcvrp. (accessed 16.10.15.).
Zhu, W., Qin, H., Lim, A., Wang, L., 2012. A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3L-CVRP. Computers & Operations Research 39, 2178–2195.