European Journal of Operational Research 228 (2013) 56–71

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Integrated distribution and loading planning via a compact metaheuristic algorithm

Emmanouil E. Zachariadis [a,*], Christos D. Tarantilis [b], Chris T. Kiranoudis [a]

[a] Department of Process Analysis and Plant Design, National Technical University of Athens, Athens, Greece
[b] Department of Management Science and Technology, Athens University of Economics and Business, 28 Hydras Str., Athens 11362, Greece

ARTICLE INFO

Article history:
Received 11 May 2012
Accepted 20 January 2013
Available online 31 January 2013

Keywords:
Logistics
Loading constraints
Combinatorial optimization
Metaheuristics

ABSTRACT

The present article examines a vehicle routing problem integrated with two-dimensional loading constraints, called 2L-CVRP. The problem is aimed at generating the optimal route set for satisfying customer demand. In addition, feasible loading arrangements have to be determined for the transported products. To solve 2L-CVRP, we propose a metaheuristic solution approach. The basic advantage of our approach lies at its compact structure, as in total, only two parameters affect the algorithmic performance. To optimize the routing aspects, we propose a local-search method equipped with an effective diversification component based on the regional aspiration criteria. The problem's loading requirements are tackled by employing a two-dimensional packing heuristic which repetitively attempts to develop feasible loading patterns. These attempts are effectively coordinated via an innovative, simple-structured memory mechanism. The overall solution framework makes use of several memory components for drastically reducing the computational effort required. The performance of our metaheuristic development has been successfully evaluated on benchmark instances considering two distinct versions of the loading constraints. More specifically, the algorithm managed to improve or match the majority of best known solution scores for both problem versions.

1. Introduction

The distribution of goods constitutes one of the most important operations in the modern business environment, with numerous everyday applications. For this reason, researchers and practitioners have been devoted to both modeling and optimizing such transportation logistics activities. The most widely known and studied distribution model is the vehicle routing problem (VRP) which examines the distribution of products from a central location to a set of geographically dispersed customers with the use of a homogeneous vehicle fleet. This classical VRP version can be regarded as an idealized model of the routing problems met in practice, obtained after making simplifying assumptions on various practical elements of real-life distribution systems. The increase of computational power provided by modern computer systems together with methodological advances allow researchers to investigate more realistic distribution models which incorporate complex operational characteristics which in the past were thought to be too hard to tackle. Following this research stream, composite routing and packing problems have been recently modeled and introduced in the literature. The central characteristic of these problems is that apart from optimizing the vehicle routing operations, the physical dimensions of transported goods are taken into account, so that feasible loading arrangements of goods must be identified. In other words, these integrated models promote a more accurate and effective decision making compared to traditional vehicle routing approaches which employ oversimplifying one-dimensional volume and weight capacity constraints, even when solid products of different shapes and sizes are transported.

The present article studies and solves the first composite routing-packing problem that appeared in the vehicle routing literature, called the Capacitated Vehicle Routing Problem with Two-Dimensional Loading constraints (2L-CVRP). The 2L-CVRP model was originally introduced by Iori et al. (2007). It considers that a homogeneous vehicle fleet based in a central depot must be routed to deliver requested products to a set of customer locations. The products to be transported are thought to be rectangular and non-stackable items. The 2L-CVRP model is aimed at determining the optimal routes for executing the necessary delivery operations, as in the case of the well-known classical Capacitated Vehicle Routing Problem (Laporte, 2009). Moreover, for each of the generated routes, the problem calls for the determination of a feasible two-dimensional orthogonal loading arrangement of the transported items onto the vehicle loading surface. This loading requirement is closely related to the two-dimensional bin packing problem (2BPP) which involves the minimization of the number of identical rectangular bins required for packing a predetermined set of rectangular items.

* Corresponding author. Tel.: +30 6945028191; fax: +30 2107723228.
E-mail addresses: ezach@mail.ntua.gr (E.E. Zachariadis), tarantil@aueb.gr (C.D. Tarantilis), kyr@chemeng.ntua.gr (C.T. Kiranoudis).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.01.040

In terms of 2L-CVRP solution methodologies, Iori et al. (2007) propose an exact branch-and-cut algorithm for the routing aspects, whereas the loading requirements of the problem are tackled via a branch-and-bound approach. The algorithm is capable of optimally solving 2L-CVRP instances of up to 25 customers and 91 items within 24 hours of computational time. To deal with larger-scale instances, Gendreau et al. (2008) propose a metaheuristic solution approach that employs tabu search for optimizing the routing characteristics, whereas feasible loading patterns are identified via a heuristic procedure based on the Touching Perimeter rule (Lodi et al., 1999). Zachariadis et al. (2009) describe a hybridization of tabu-search and guided local search which operates in parallel with a collection of heuristics for developing feasible item loadings. The work of Fuellerer et al. (2009) describes an Ant Colony Optimization approach combined with several two-dimensional packing heuristics in pursuit of feasible loading patterns. Another algorithmic design has been proposed by Leung et al. (2011). Their algorithm deals with the routing aspects via a hybridization of tabu search and extended guided local search, whereas loading feasibility of routes is investigated by a set of two dimensional packing heuristics. The most recent paper on the 2L-CVRP model is due to Duhamel et al. (2011). The authors propose a hybrid algorithm which combines the powers of the greedy randomized adaptive search (GRASP) and evolutionary local search (ELS) strategies. The loading aspects are dealt with by firstly relaxing the corresponding two-dimensional bin packing problems into simpler resource constrained project scheduling (RCPSP) ones. The generated routes are then modified in order to satisfy the original 2L-CVRP loading requirements. A vehicle routing problem with similar two dimensional loading constraints has been introduced by Malapert et al. (2008), examining the pick-up and delivery extension of the basic 2L-CVRP model.

Regarding vehicle routing problems integrated with loading constraints, Gendreau et al. (2006) have generalized 2L-CVRP by imposing three-dimensional loading constraints for the transportation of rectangular and stackable boxes. The problem is referred to as the capacitated vehicle routing problem with three-dimensional loading constraints (3L-CVRP). The authors solve the examined problem via a tabu search metaheuristic development. Further solution approaches for the 3L-CVRP model have been published by Tarantilis et al. (2009), Fuellerer et al. (2010), Ruan et al. (2011), Zhu et al. (2012) and Bortfeldt (2012). Another routing problem with three-dimensional packing requirements which imposes time-window constraints has been studied by Moura and Oliveira (2009). Zachariadis et al. (2012) examine an integrated routing-packing model of great practical importance, called the pallet packing vehicle routing problem. As its name suggests, the examined model considers that transported boxes must be feasibly packed into pallets which are then loaded onto the vehicles. A routing application with loading constraints faced by an Austrian wood product retailer is discussed in the studies of Doerner et al. (2007b) and Tricoire et al. (2011). The interested reader is referred to the work of Iori and Martello (2010) which reviews published vehicle routing problems integrated with loading constraints.

The purpose of the present article is to propose an innovative and effective solution approach for the 2L-CVRP. In terms of the routing aspects, we propose a local-search optimization approach, coordinated via a single-parameter policy which adapts to the progress of the conducted search. Regarding the loading constraints, they are tackled via a packing heuristic which attempts to identify feasible packing arrangements for the transported goods. The packing heuristic behavior is controlled by an innovative memory mechanism which is aimed at generating diverse packing structures in order to maximize the probability of obtaining feasible loading arrangements. Finally, various memory components are used to reduce the computational effort dedicated for examining loading feasibility.

The basic advantage of the overall solution framework lies at its compact and simple structure. Contrary to previously published complex 2L-CVRP approaches, our algorithm makes use of two search parameters in total, one for the routing and one for the loading aspects. This feature is very important, as it promotes a robust performance and eliminates the time required for tuning experiments, when problems of diverse characteristics are solved. The proposed solution approach was tested on 2L-CVRP benchmark problems. It produced fine quality results, improving several best known solutions.

The remainder of the present article is organized as follows: Section 2 describes the 2L-CVRP model in detail. In Section 3, we present the routing optimization component of the proposed methodology, whereas the heuristic procedure for investigating the loading constraints is discussed in Section 4. Finally, Section 5 provides and evaluates the obtained computational results, followed by Section 6 which concludes the paper.

2. Description of the 2L-CVRP model

The 2L-CVRP model is defined on a complete graph $G = (V,A)$, where $V = \{0, 1, \ldots, n\}$ is the vertex set and A is the arc set. Vertex 0 represents the depot where a fleet of k vehicles is available. Each vehicle has a maximum carrying load of weight Q and carries a loading surface with length and width equal to L and W, respectively. With each arc $(i,j) \in A$ is associated a known and non-negative travel cost c_{ij} reflecting the distance, the travel-time or the actual monetary cost required for traveling between vertices i and j. Let N denote the set of customer vertices ($N = V\backslash\{0\}$). Each customer $i \in N$ is considered to demand a set of items $T_i = \{0, \ldots, m_i\}$ of total weight equal to q_i. The length and width dimensions of an item $j \in T_i$ are denoted by l_{ij} and w_{ij}, respectively.

The goal of the 2L-CVRP is to identify the set of routes which minimize the total required travel cost. The produced routes are subject to the following constraints:

- Each route starts and terminates from and to the central depot, respectively.
- Each customer vertex is visited once by exactly one route.
- The total weight of the items required by the customer subset assigned to each route does not exceed the vehicle capacity Q.
- All items demanded by the customers visited by a route are transported by the vehicle implementing this route.
- For each vehicle, there exists a feasible, two-dimensional, orthogonal packing of the transported items onto the loading surface (loading constraints).

An example solution of a 2L-CVRP instance that involves 5 customers, 12 items and 2 vehicles is depicted in Fig. 1.

The loading constraints of the 2L-CVRP guarantee that all items demanded by the customers visited by a single route can be orthogonally packed into the loading surface of the corresponding vehicle. Obviously, loaded items must not overlap, nor exceed the vehicle surface boundaries. In practical cases, additional loading constraints can be imposed to ensure that unloading operations are efficiently performed. This is discussed by Iori et al. (2007) who introduce the *sequence* (often referred to as LIFO) constraint which ensures that whenever a customer i is visited, all T_i items can be unloaded by employing a sequence of straight movements (one per item) parallel to the length dimension of the vehicle surface. In other words, no item of customer j to be served later than customer i can be placed between items of i and the loading/unloading door of the vehicle. Moreover, depending on the

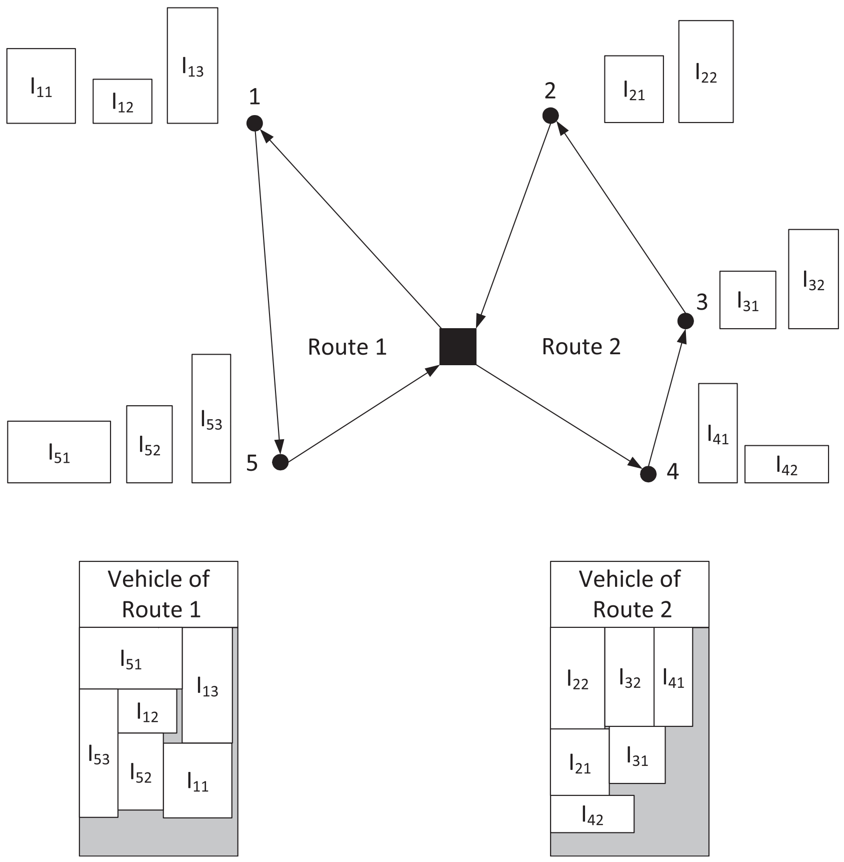

Fig. 1. An example 2L-CVRP solution.

operational requirements, items might have to be placed on the loading surfaces with fixed orientation (their *l*- and *w*-edges to be parallel to the *L*- and *W*-edges of the loading surface respectively), or item rotations (of 90°) may be allowed.

In the following and according to the classification scheme of Fuellerer et al. (2009), we distinguish between four 2L-CVRP loading constraint configurations, depending on whether the sequence constraint is considered and whether item rotations are allowed:

2|UO|L: no sequence constraint, fixed orientation.
2|UR|L: no sequence constraint, rotations allowed.
2|SO|L: sequence constraint, fixed orientation.
2|SR|L: sequence constraint, rotations allowed.

Note that in the present paper, we solve the 2|UO|L and 2|SO|L versions which were originally tackled by Iori et al. (2007).

3. The proposed solution approach

As previously mentioned, the 2L-CVRP can be regarded as the composition of two NP-hard optimization problems: the CVRP, as far as the routing characteristics are concerned and the 2BPP, regarding the imposed loading requirements. Subsequently, the overall problem is very hard to be solved to optimality, especially when instances of practical sizes are examined. To tackle such problems, the decision-maker has to focus on approximate solution methodologies. In this context, to deal with the routing aspects of 2L-CVRP, we have developed a local-search metaheuristic approach. The proposed method starts off with a simple construction heuristic for generating an initial 2L-CVRP solution. Then, the core of our methodology is employed for producing the final solution. The basic elements of the proposed algorithm are the following:

- The overall search is controlled with the use of a new variation of the promises concept (Zachariadis and Kiranoudis, 2012) which depends on only one parameter.
- Local-search is efficiently performed using the Static Move Descriptor (SMD) concept (Zachariadis and Kiranoudis, 2010) for statically encoding local-search moves.
- The use of special priority queue structures (Fibonacci Heaps) eliminates unproductive loading feasibility investigations.
- Several memory components are employed to eliminate redundant loading feasibility investigations.

In the following, we provide a detailed analysis of the proposed routing optimization approach. The loading feasibility is examined with the use of the procedure presented in Section 4.

3.1. Building an initial 2L-CVRP solution

To obtain an initial 2L-CVRP solution, we apply a simple constructive method which is primarily aimed at maximizing the utilization of the vehicle loading area, and secondarily minimize the total travel cost.

In the beginning, k empty routes are opened (one route per vehicle) and unserved customers of N are inserted into set U. Let C_i ($i = 1, \dots, k$) denote the customer set assigned to route i. At each iteration, we identify the route i^*-customer j^* pair which maximizes the loading area utilization $u(i,j) = \sum_{p \in C_i} \sum_{r \in T_p} w_{pr} l_{pr} + \sum_{r \in T_j} w_{jr} l_{jr}$, while the tentative item set $\bigcup_{r \in C_{i*} \cup \{j*\}} T_r$ can be feasibly loaded onto the loading surface and the capacity constraints are satisfied $\left(\sum_{r \in C_{i*} \cup \{j*\}} q_r \leqslant Q\right)$. Customer j^* is removed from the set of unserved customers U and inserted into the position of route i^* which minimizes the travel cost increase. Note that if the sequence constraint is considered, for every route-customer pair, all possible insertion positions are examined and the method selects the minimum cost insertion which leads to a route that satisfies all loading constraints. The iterative procedure terminates when no feasible customer-route pair can be identified ($U \neq \emptyset$), or when all customers have been assigned to some route ($U = \emptyset$). This implies that the constructive method generates a complete or partial solution which consists of feasible 2L-CVRP routes.

3.2. Local-Search for improving the 2L-CVRP solution

To generate the final 2L-CVRP solution, we propose a simple-structured local search algorithm. In the following, we go through the basic ingredients of the algorithm which are later discussed in greater detail. Our approach iteratively performs structural modifications to a candidate 2L-CVRP solution. These modifications are defined by a blend of three quadratic operators, widely used in vehicle routing approaches: 1–0 Exchange (customer relocation), 1–1 Exchange (customer swap), 2-opt (route crossover). As mentioned earlier, local-search is efficiently performed with the use of the Static Move Descriptor (SMD) strategy. The algorithm acts according to the best admissible move policy, i.e. tentative moves of the employed operators are exhaustively examined and the best-quality move (in terms of the problem's objective function) is identified and selected to be applied to the candidate solution. This move-selection policy promotes an intensive search towards locally optimal solutions. To balance the algorithm behavior so that a broad solution space exploration is achieved, we incorporate a new variation of a diversification component, called promises, based on the regional aspiration criteria of tabu search approaches (Zachariadis and Kiranoudis, 2012). Finally, we note that our approach explores the feasible solution space, or in other words, visits solutions which consist of routes that respect all of the 2L-CVRP constraints. Loading feasibility investigations are discussed in Section 4.

3.3. The adopted diversification component

Our algorithm is equipped with a diversification mechanism related to the regional aspiration criteria of tabu search (Glover and Laguna, 1997) and the attribute based-hill climber (ABHC) heuristic (Whittley and Smith, 2004; Derigs and Kaiser, 2007). This mechanism has a twofold contribution: (a) eliminates the risk of cycling around poor-quality local optima, and (b) promotes drastic structural modifications on the candidate solution, so that diverse regions of the search space are investigated.

Let c_a denote a tag (hereafter referred to as promise tag) associated with each arc $a \in A$. In addition, let E_m and C_m ($E_m, C_m \subseteq A$) denote the set of arcs eliminated and introduced to the candidate solution, respectively, whenever a local search move m is applied. In the beginning of the improvement method, every arc promise tag is initialized to $+\infty$ ($\forall a \in A$: $c_a = +\infty$). The diversification mechanism works as follows: when a move m is applied to a solution S of cost $z(S)$, the promise tags of the eliminated arcs are set equal to $z(S)$ ($\forall a \in E_m$: $c_a = z(S)$). During later evaluations of tentative moves, a move m' transforming a solution S' to S'' is considered admissible (*promise-keeping*), if and only if $\forall a \in C_{m'}: z(S'') < c_a$. Note that the adopted criterion of move admissibility can be seen as a more aggressive (diversification causing) variation of the ABHC strategy which would allow move m' from S' to S'', if $\exists a \in C_{m'}: z(S'') < c_a$.

The above-described mechanism of filtering out tentative moves has an excessive diversification character, driving the algorithm in very poor-quality regions of the search-space. To overcome this issue and achieve a fair algorithmic behavior, a promise tag re-initialization scheme must be adopted. In our previously published solution approaches, this was made by periodically resetting the promise cost tags to $+\infty$, every time a fixed number of iterations were completed. Further experimentation led us to a more effective and simple rule which adapts to the conducted search: the proposed approach reinitializes the promise tags to $+\infty$, when the cost of the candidate 2L-CVRP solution $z(S)$ is greater than $\sigma \cdot z(S^*)$, where σ is a user-defined search parameter and $z(S^*)$ denotes the objective function value of the best-quality solution encountered through the search process. Loosely speaking, the proposed scheme ensures that when the search is driven to low-quality regions, it is released to perform cost reducing moves for returning into promising areas of the solution space. Preliminary runs indicated that this adaptive scheme manages to obtain slightly better results compared to the original periodic re-initialization strategy. Values of σ taken from the [1.02,1.05] interval resulted in a very effective performance, thus we fixed σ at 1.035.

3.4. The role of the Static Move Descriptor representation

The proposed algorithm employs three local-search quadratic operators. To efficiently use these operators, we use the Static Move Descriptor (SMD) strategy. In the following, we provide a brief description of the basic SMD principles for completeness of presentation. More details are available in Zachariadis and Kiranoudis (2010), where the SMD strategy was originally introduced.

Each tentative move defined by the local-search operators is statically encoded into an SMD instance which also contains a cost tag corresponding to the cost of the encoded move, i.e. the objective function change that this move would cause, if it was applied to the candidate solution. Each time a move m is performed, only a subset of the solution arcs is removed (E_m) and replaced with a newly-introduced arc set (C_m). To keep the cost tags of the complete SMD population (of every tentative move) updated according to the modified solution state, only the tags of the SMD instances depending on the E_m and C_m sets have to be re-evaluated. The cost tags of the rest SMD instances remain unaffected, and thus their cost re-evaluation is unnecessary.

To better present the SMD strategy used for encoding local search moves, we provide Fig. 2. It illustrates how a local search move is encoded into an SMD instance. In addition, it depicts the application of the encoded move to an example solution of a 2L-CVRP with eight customers. Each SMD instance belongs to a local search operator type which defines the mechanism of the move. It also includes the solution point where the encoded move is applied. We note that the SMD instance type and move point information is static, viz. independent of the shape of the candidate solution. The type of the SMD instance presented in Fig. 2is 1–1 Exchange, meaning that it encodes a 1–1 Exchange local search move. The move point is a pair of vertices $n_1 = 2$ and $n_2 = 6$. Thus, the presented SMD instance corresponds to the exchange of service positions for vertices 2 and 6. Regarding the SMD instance cost tag (objective function modification), we observe that it is equal to the sum of the costs of the arcs to be generated minus the sum of the costs of the arcs to be removed from the solution ($c_{52} + c_{27} + c_{16} + c_{63} - c_{56} - c_{67} - c_{12} - c_{23}$). Applying the encoded local search move requires the cost tag recalculation for the SMD instances whose tags were modified by the applied move. Fig. 2 presents the updated cost tag for the presented SMD instance according to the modified solution state ($c_{56} + c_{67} + c_{12} + c_{23} - c_{52} - c_{27} - c_{16} - c_{63}$). The cost tags of the SMD instances which are not affected by the move application are valid according to the modified solution state, and thus their cost tag recalculation is

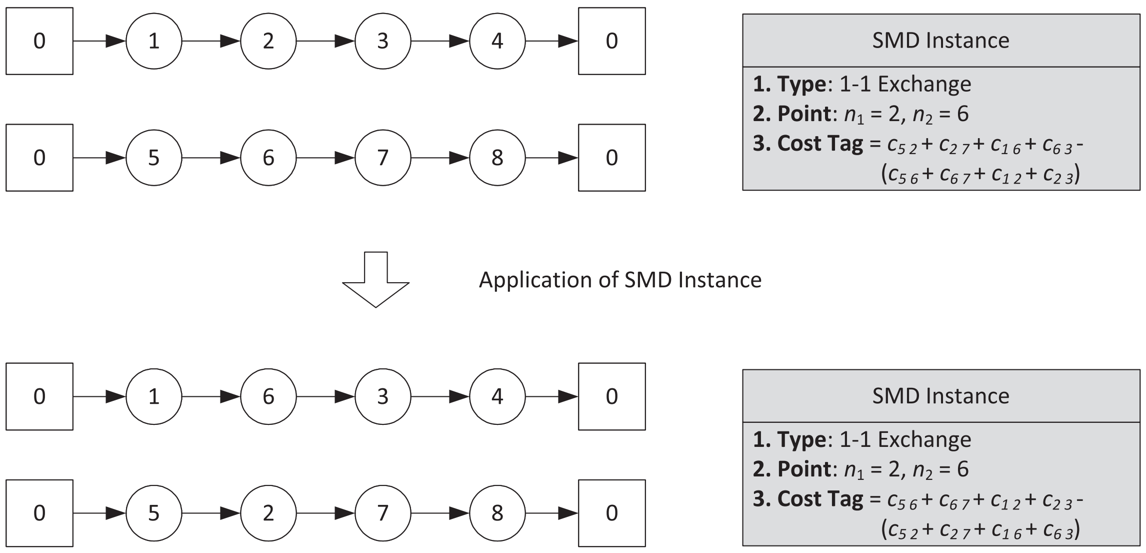
0
1
2
3
4
0
0
5
6
7
8
0
SMD Instance
1. Type: 1-1 Exchange
2. Point: $n_1 = 2$, $n_2 = 6$
3. Cost Tag = $c_{52} + c_{27} + c_{16} + c_{63}$ - $(c_{56} + c_{67} + c_{12} + c_{23})$
Application of SMD Instance
0
1
6
3
4
0
0
5
2
7
8
0
SMD Instance
1. Type: 1-1 Exchange
2. Point: $n_1 = 2$, $n_2 = 6$
3. Cost Tag = $c_{56} + c_{67} + c_{12} + c_{23}$ - $(c_{52} + c_{27} + c_{16} + c_{63})$

Fig. 2. Encoding local-search moves into SMD instances.

redundant. For instance, the cost tag of the SMD instance encoding the position exchange of vertices 4 and 8 of Fig. 2 is equal to $c_{38} + c_{80} + c_{74} + c_{40} - c_{34} - c_{40} - c_{78} - c_{80}$, both before and after the presented move application.

As mentioned earlier, our algorithm employs the best admissible move strategy. This means that, at each iteration, the lowest-cost SMD instance, or in other words the best quality tentative move, must be located. To do so, SMD instances are stored into Fibonacci Heaps which are special priority queue structures offering very fast insertion, minimum retrieval and deletion capabilities (Fredman and Tarjan, 1987). Another advantage which naturally arises from keeping the SMD instances sorted according to their cost, is that the two-dimensional packing methods for examining the loading feasibility of tentative moves (Section 4) are executed only for the best quality moves, or in other words only for the actual candidates for being selected and performed (Zachariadis and Kiranoudis, 2010). Thus, unproductive loading feasibility investigations are avoided. This algorithmic feature is crucial when dealing with routing problems of complex feasibility constraints, because the total CPU time required strongly depends on the calculations dedicated for the necessary feasibility checks.

Finally, the SMD representation provides a straightforward way for memorizing obtained packing feasibility information, helping the algorithm to eliminate duplicate and unnecessary executions of the loading feasibility methods, and thus accelerating the search process. This will be further explained in Section 4.4 where the employed memory mechanisms are discussed.

3.5. The overall local-search framework

After the initial solution of the constructive method (Section 3.1) has been generated, the proposed local-search metaheuristic is initiated. Firstly, the SMD instances of the three employed local search operators are generated and their cost tags are evaluated according to the structure of the initial solution. The SMD instances are then inserted into three Fibonacci Heaps (one heap per operator type).

When this initiation stage has been completed, the algorithm applies the iterative core of the improvement methodology. Each iteration involves the execution of the following steps:

1. One of the three employed local search operator is randomly selected, with all three operators sharing equal selection probability.
2. The lowest-cost SMD instance of the selected type is retrieved from the corresponding Fibonacci Heap. As previously stated, this SMD instance must encode a local-search move that is both promise-keeping and satisfies all 2L-CVRP constraints. To locate this SMD instance, the minimum-cost tag SMD instances are successively popped out of the heap and the method examines their promise-keeping status, their capacity constraint feasibility and their loading constraint feasibility. Note that loading feasibility is evaluated last, as it is the most time-consuming of the necessary checks: the promise-keeping status and the capacity constraint examination are performed in constant time, whereas the loading feasibility is investigated by the application of the packing heuristic presented in Section 4. We note that if no SMD instance which is both feasible and promise-keeping can be identified, the method reinitializes all arc promise tags to $+\infty$ ($\forall\ a \in A$: $c_a = +\infty$). In other words, the proposed method drops the promise-keeping requirement, to avoid entrapment of the search process.
3. The local-search move encoded by the selected SMD instance is applied to the candidate solution. The cost tags of the affected SMD instances are updated according to the modified solution state (Section 3.4). The necessary cost update operations for the employed local search operators are thoroughly presented in Zachariadis and Kiranoudis (2010). In addition, the appropriate arc promise tags are set (Section 3.3).
4. If the candidate solution is partial (i.e. unserved customers exist), the method attempts to insert unserved customers into the candidate solution. Customers are inserted into the position minimizing the routing cost increase, given that the resulting solution satisfies all 2L-CVRP constraints.

The above-described set of steps is repeatedly applied until a certain termination condition is reached. Under our implementation, the algorithm terminates after 50,000 iterations have been executed. This limit proved to be enough to obtain high-quality solutions, as shown in Section 5.

At this point, we would like to make a brief comment concerning the examination of loading constraints. As already explained, the moves encoded by the SMD instances are investigated in terms of the 2L-CVRP loading constraints via the application of the heuristic procedure described in Section 4. This heuristic has been designed to apply several attempts in pursuit of feasible packing arrangements for the examined tentative routes. However, for cost-increasing moves (moves that deteriorate the quality of the candidate solution), the loading feasibility procedure applies only one packing attempt. This design was originally adopted mainly for accelerating the algorithm. However, preliminary runs indicated that after weakening the loading feasibility method for cost-increasing moves (i.e. narrowing the accessible solution space), higher quality final solutions were obtained.

4. Examining the 2L-CVRP loading constraints

The routing component of our solution approach investigates the loading feasibility of tentative routes via a two-dimensional packing heuristic. The basic feature of the proposed heuristic is that it employs a collection of attempts to feasibly pack the transported items. Each of these attempts successively inserts items into the loading surface, based on the extreme-point rationale of Crainic et al. (2008). The acceleration effect of the SMD concept allows us to design a more powerful packing methodology, compared to our previous approach (Zachariadis et al., 2009). More specifically, the proposed packing method incorporates two new ideas:

- The heuristic employs a memory mechanism to store packing characteristics. Stored information is used to systematically diversify the loading structures generated by the packing attempts, and thus increase the probability for identifying a feasible loading arrangement.
- Each attempt does not insert items in a predetermined order. Instead, each item placement step is capable of inserting any item into any of the available loading positions.

4.1. Loading position management

Each attempt of generating a feasible loading arrangement involves the successive placement of items on the loading surface. To define the item insertion positions, we maintain a list that contains the coordinates of the available loading positions.

Let the loading surface be described by a two-dimensional Cartesian coordinate system (w, l). The origin of the axes $(0, 0)$ represents the backmost and leftmost point of the loading surface, whereas the loading door lies at the linear segment connecting $(0, L)$ and (W, L). In the beginning, when the loading surface is empty, the list of available loading positions contains only point

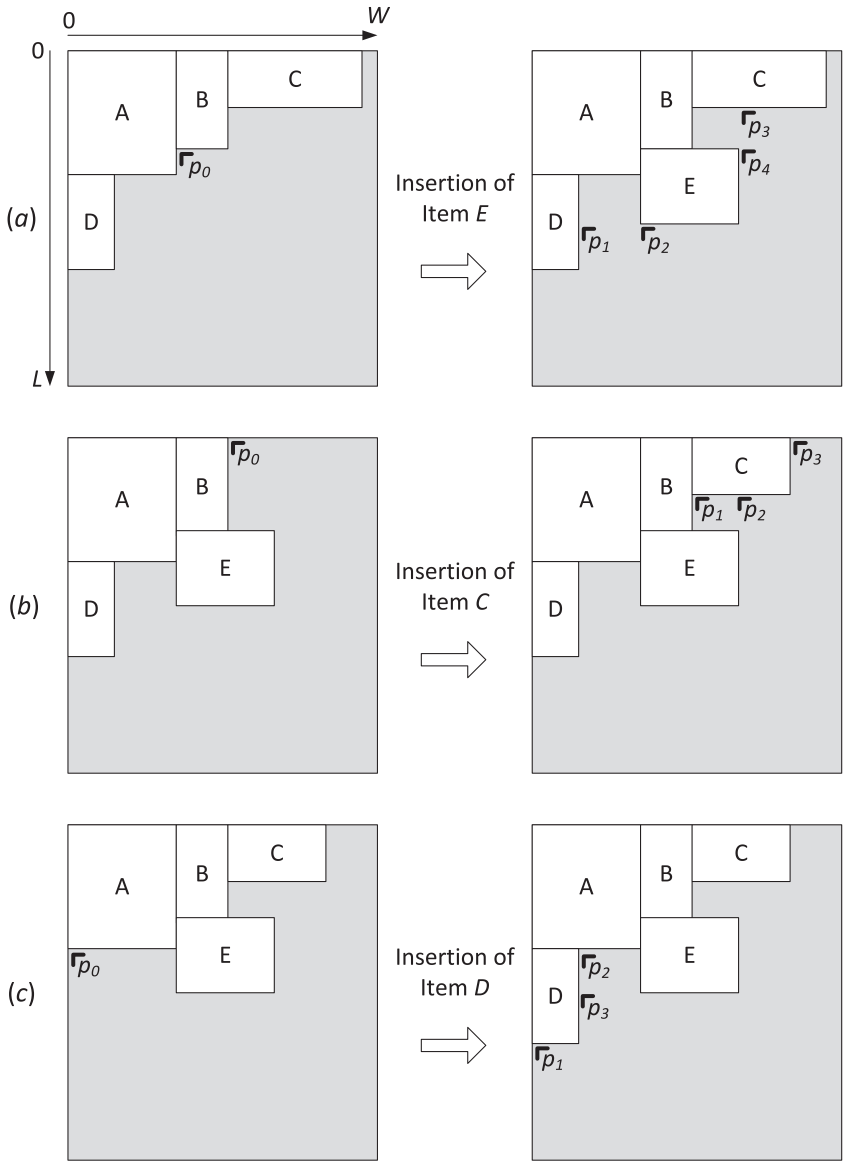

Fig. 3. Loading position management.

(0,0). Each time an item is inserted into a loading position, this occupied position is deleted from the list. It is replaced by a set of new loading positions, responsible for accommodating subsequent items. The mechanism for updating the contents of the list of available loading positions is graphically depicted in Fig. 3, where three example item insertions are presented.

In Fig. 3a item E is inserted with its back left corner in position p_0 which is removed from the list. Four new positions are created: the first one (p_1) lies on the minimum w- non-occupied point of the line defined by item's E front edge, the second position (p_2) is located on the front left corner of E, whereas the third position (p_3) is created on the minimum l-, non-occupied point of the line defined by item's E right edge. Finally, the fourth position (p_4) is at the back right corner of E. Fig. 3b presents another type of position creation in the front edge of the inserted item. More specifically, item C is placed with its back left corner in p_0. Position p_0 is eliminated from the list. It is substituted by p_1 lying on the front left corner of C, p_3 on the back right corner of C and p_2 that is generated on the front edge of item C. Regarding its w- coordinate, it lies on the right edge of an already inserted item (E) which in front of C. A similar case of position creation is depicted in Fig. 3c. Item D is placed in p_0. As a result, positions p_1 and p_2 are generated on the front left and the right back corners of item D, respectively. Position p_3 is created on the intersection of the right edge of item D and the line defined by the front edge of item E which has been already loaded on the right of D.

4.2. The memory mechanism for controlling the packing heuristic

The proposed packing heuristic performs several attempts for feasibly arranging the items into the loading surface. These various attempts are interconnected and coordinated through the use of a memory mechanism, responsible for storing the number of times that a given partial loading has been visited through the overall heuristic procedure. The above-described memory mechanism is implemented as a hashtable structure (*System.Collections.HashTable* class of *.NET Framework 2.0*). Hashtable keys contain the information necessary for encoding encountered partial loadings (item ids, placement coordinates), whereas the value assigned to each key corresponds to the number of times that the encoded partial loading has been encountered. To better describe the hashtable structure, we provide an example together with Fig. 4. Consider that the heuristic procedure applies the first attempt to feasibly pack the items of route 1 of Fig. 1. In the first step of this attempt, item I_{11} is placed in position $(w,l) = (0,0)$. To store this partial

Table 1
Characteristics of the 2L-CVRP benchmark instances.

		Class 2			Class 3			Class 4			Class 5		
Inst	n	t	k	u%	t	k	u%	t	k	u%	t	k	u%
1	15	24	3	78	31	3	82	37	4	70	45	4	61
2	15	25	5	52	31	5	59	40	5	53	48	5	39
3	20	29	5	68	46	5	77	44	5	62	49	5	54
4	20	32	6	58	43	6	58	50	6	63	62	6	47
5	21	31	4	72	37	4	75	41	4	76	57	5	53
6	21	33	6	54	40	6	63	57	6	72	56	6	46
7	22	32	5	71	41	5	66	51	5	67	55	6	49
8	22	29	5	63	42	5	71	48	5	68	52	6	38
9	25	40	8	57	61	8	61	63	8	60	91	8	53
10	29	43	6	74	49	6	66	72	7	73	86	7	63
11	29	43	6	77	62	7	74	74	7	83	91	7	63
12	30	50	9	62	56	9	52	82	9	66	101	9	58
13	32	44	7	69	56	7	68	78	7	77	102	8	59
14	32	47	7	65	57	7	65	65	7	61	87	8	49
15	32	48	6	76	59	6	84	84	8	72	114	8	72
16	35	56	11	55	74	11	57	93	11	64	114	11	49
17	40	60	14	46	73	14	42	96	14	51	127	14	40
18	44	66	9	72	87	10	75	112	10	77	122	10	58
19	50	82	11	77	103	11	83	134	12	79	157	12	61
20	71	104	14	84	151	15	83	178	16	81	226	16	69
21	75	114	14	84	164	17	82	168	17	70	202	17	61
22	75	112	15	82	154	16	81	198	17	82	236	17	66
23	75	112	14	86	155	16	83	179	16	83	225	16	72
24	75	124	17	81	152	17	77	195	17	82	215	17	66
25	100	157	21	83	212	21	85	254	22	83	311	22	65
26	100	147	19	84	198	20	82	247	20	87	310	20	75
27	100	152	19	84	211	22	82	245	22	78	320	22	71
28	120	183	23	83	242	25	83	299	25	84	384	25	72
29	134	197	24	85	262	26	83	342	28	85	422	28	74
30	150	225	29	83	298	30	87	366	30	86	433	30	70
31	199	307	38	84	402	40	85	513	42	86	602	42	70
32	199	299	38	84	404	39	85	497	39	86	589	39	73
33	199	301	37	85	407	41	84	499	41	87	577	41	71
34	240	370	46	85	490	49	86	604	50	86	720	50	72
35	252	367	45	85	507	50	85	634	50	90	762	50	74
36	255	387	47	86	511	51	86	606	51	83	786	51	74

t: Total items to be transported.
k: Number of vehicles.
u%: Utilization of the loading surfaces.

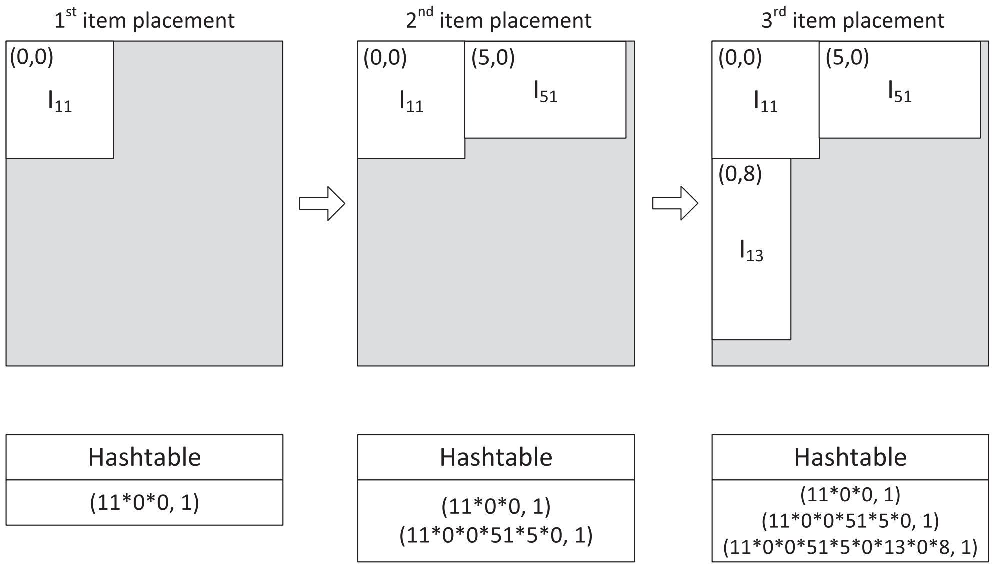

Fig. 4. Memorizing encountered partial loadings.

loading, we prepare the corresponding key which contains the relevant information: item id, followed by the w- and l-coordinates of the item's back left corner position. These entries are divided with the use of a standard operator ($*$). As this is the first packing attempt, obviously this loading has been observed only once. Thus, the first entry added in the hashtable is (#*key*, #*value*) = $(11 * 0 * 0, 1)$. In the second step, item I_{51} is placed with its back left corner at position $(w,l) = (5,0)$. As a result, the second entry is (#*key*,# *value*) = $(11 * 0 * 0 * 51 * 5 * 0, 1)$. In the third step, item I_{13} is inserted into position $(w,l) = (0,8)$. Consequently, the third hashtable entry is (#*key*,#*value*) = $(11 * 0 * 0 * 51 * 5 * 0 * 13 * 0 * 8, 1)$. If during subsequent packing attempts (within the same heuristic), any of the aforementioned partial loadings are visited, the value attributes will be augmented, accordingly.

4.3. The overall structure of the packing heuristic

To test whether a tentative route r that serves customer set C_r is feasible regarding the 2L-CVRP loading constraints, the proposed heuristic initializes the hashtable memory structure of Section 4.2 and then performs up to μ packing attempts, in search for a feasible loading arrangement for item set $R_r = \bigcup_{i \in C_r} T_i$.

Each packing attempt begins from an empty loading surface. The list of available loading positions P is initialized to contain only the point $(0,0)$, as described in Section 4.1. Then, item insertions are iteratively performed. At each iteration, a binary value λ is randomly selected from $\{0,1\}$, with equal probability. Then, for every feasible (no item overlaps, no boundary violations) item-position pair (i,p) $(i \in R_r, p \in P)$, we calculate the following utility function:

$$u(i,p) = \text{TP}(i,p) - \lambda \text{ M times}(i,p),$$

where TP (i,p) is the perimeter of item i touching the edges of already loaded items, or loading surface boundaries, if inserted into position p, as per the Touching Perimeter heuristic (Lodi et al., 1999); times (i,p) denotes the number of times that the loading to be produced, if item i is placed into p, has been encountered through the heuristic process (retrieved from the hashtable structure described in Section 4.2); and M is a large positive number. Let (i^*,p^*) denote the item-position pair maximizing the proposed utility function. Item i^* is removed from the set R_r and inserted into position p^*. The list of available loading positions P is updated, as

Table 2
Results on pure CVRP instances of Class 1.

	EGTS + LBFH		ACO		GRASPxELS		BAS	PRMP		
	z	*t*	*z*	*t*	*z*	*t*		*z*	*gap%*	*t*
1	**278.73**	1.6	**278.73**	0.1	**278.73**	0.0	278.73	**278.73**	0.00	0.0
2	**334.96**	1.3	**334.96**	0.1	**334.96**	0.0	334.96	**334.96**	0.00	0.0
3	**358.40**	2.9	**358.40**	0.2	**358.40**	0.0	358.40	**358.40**	0.00	0.0
4	**430.88**	2.1	**430.88**	0.3	**430.88**	0.0	430.88	**430.88**	0.00	0.0
5	**375.28**	2.3	**375.28**	0.3	**375.28**	0.0	375.28	**375.28**	0.00	0.0
6	**495.85**	2.6	**495.85**	0.3	**495.85**	0.0	495.85	**495.85**	0.00	0.0
7	**568.56**	2.6	**568.56**	0.2	**568.56**	0.0	568.56	**568.56**	0.00	0.0
8	**568.56**	2.8	**568.56**	0.2	**568.56**	0.0	568.56	**568.56**	0.00	0.0
9	**607.65**	4.3	**607.65**	0.6	**607.65**	0.0	607.65	**607.65**	0.00	0.0
10	**535.74**	5.0	535.80	2.3	535.80	0.0	535.74	535.80	−0.01	0.1
11	**505.01**	4.5	**505.01**	0.8	**505.01**	0.0	505.01	**505.01**	0.00	0.0
12	**610.00**	13.5	**610.00**	1.5	**610.00**	0.2	610.00	**610.00**	0.00	0.2
13	**2006.34**	6.8	**2006.34**	1.3	**2006.34**	0.0	2006.34	**2006.34**	0.00	0.3
14	**837.67**	10.3	**837.67**	4.1	**837.67**	0.2	837.67	**837.67**	0.00	0.1
15	**837.67**	6.9	**837.67**	2.8	**837.67**	0.0	837.67	**837.67**	0.00	0.4
16	**698.61**	17.3	**698.61**	2.0	**698.61**	0.0	698.61	**698.61**	0.00	0.3
17	**861.79**	14.2	**861.79**	3.3	**861.79**	0.0	861.79	**861.79**	0.00	1.6
18	**723.54**	12.5	**723.54**	9.5	**723.54**	8.3	723.54	**723.54**	0.00	3.6
19	**524.61**	21.5	**524.61**	7.9	**524.61**	0.3	524.61	**524.61**	0.00	2.1
20	**241.97**	55.4	**241.97**	56.1	**241.97**	4.5	241.97	**241.97**	0.00	7.2
21	**687.80**	62.2	690.20	26.5	**687.60**	1.4	687.60	**687.60**	0.00	3.8
22	**740.66**	62.5	742.91	57.4	**740.66**	2.1	740.66	**740.66**	0.00	2.8
23	835.32	113.2	845.34	55.9	**835.26**	3391.3	835.26	**835.26**	0.00	48.7
24	1034.56	125.7	1030.25	49.8	1026.60	53.3	1026.60	**1024.69**	0.19	38.1
25	**826.14**	128.1	830.82	167.1	827.39	2.4	826.14	**826.14**	0.00	8.6
26	**819.56**	85.8	**819.56**	175.7	**819.56**	0.4	819.56	**819.56**	0.00	11.2
27	1091.40	166.7	1100.22	190.5	**1082.65**	486.5	1082.65	**1082.65**	0.00	172.3
28	**1040.70**	199.3	1062.23	252.5	1042.12	129.8	1040.70	1042.12	−0.14	71.2
29	1163.38	469.8	1168.13	769.1	**1162.96**	549.6	1162.96	**1162.96**	0.00	121.9
30	1040.73	972.0	1041.05	310.3	1033.42	2165.9	1033.42	**1028.42**	0.48	267.5
31	1358.01	2667.9	1341.89	521.8	1306.07	5096.1	1306.07	**1299.56**	0.50	353.8
32	1324.82	1974.8	1334.26	517.7	1303.52	4492.4	1303.52	**1296.91**	0.51	312.0
33	1312.97	1386.6	1331.69	476.6	1301.06	4842.1	1301.06	**1299.55**	0.12	434.1
34	711.98	2710.5	712.32	614.5	713.51	3007.4	711.98	**709.82**	0.30	328.2
35	875.07	2076.5	868.12	1452.6	870.63	2616.5	868.12	**866.06**	0.24	396.3
36	586.58	2405.8	616.69	1588.3	592.87	5264.7	586.58	**585.46**	0.19	228.9
Avg									*0.07*	

EGTS + LBFH: The algorithm of Leung et al. (2011) – Intel Core 2 Duo 2.0 GHz, Visual C++.
ACO: The Ant Colony Optimization method of Fuellerer et al. (2009) – Pentium IV 3.2 GHz, C++.
GRASPxELS: The method of Duhamel et al. (2011) – Opteron 2.1 GHz, C++.
PRMP: The proposed methodology – Intel Core 2 Duo E6600 2.4 GHz, Visual C#.
BAS: Best algorithmic solution (among EGTS + LBFH, ACO and GRASPxELS).
Bold entries correspond to higher quality solutions.
z: Cost of the highest quality solution achieved.
t: CPU time (in seconds) elapsed when the best quail solution was identified.
gap%: Percent gap between our solution scores and BAS (*gap%* = 100(BAS − z_{PRMP})/BAS).

described in Section 4.1. Furthermore, the hashtable structure is updated as per the modified loading arrangement. The aforementioned steps are repeated until all items have been loaded ($R_r = \emptyset$), or no more items can be feasibly packed into the available loading positions ($R_r \neq \emptyset$). If no complete arrangement is achieved, the loading surface is emptied, the list of available loading positions is reset to $P = \{(0,0)\}$, set R_r is re-initialized to $\bigcup_{i \in C_r} T_i$, and the heuristic employs the next loading attempt.

At this point, it is useful to provide some comments on the criterion used for deciding on item placements. The proposed utility function consists of two terms: the first one intensifies the search towards a feasible loading, as it applies a deterministic greedy criterion, whereas the second term has a diversification role forcing each attempt to construct packing patterns not previously examined. Preliminary runs, without the use of the λ parameter, revealed an over-diversified and ineffective behavior, as the greedy touching perimeter criterion was completely overridden by the role of the memory component. As a result, the method performed inappropriate item insertions which drastically reduced the probability of identifying a feasible loading structure. The use of λ which is the only parameter used throughout the proposed packing heuristic, resulted in a good interplay between intensifying and diversifying the search for feasible loading structures.

When the *sequence* constraint is considered, the heuristic procedure is applied exactly as described above with a slight modification in the utility function used for selecting item-position pairs: let v_j denote the visit order of customer $j \in C_r$ in the examined route r. For the second route of Fig. 1, $v_4 = 1$ as customer 4 is visited first on the route, followed by customers 3 and 2 ($v_3 = 2$, $v_2 = 3$). For every item $i \in R_r$ required by a customer $j \in C_r$, we set $v_i = v_j$. The utility function used is:

$$u(i,p) = \mathrm{TP}\ (i,p) + M_2\ v_i - \lambda\ M_1\ \mathrm{times}(i,p),$$

where M_1 and M_2 are both large positive numbers, with $M_1 \gg M_2$. In this way, items with high visit orders (served late on the route) are given higher priority of being loaded first into the rear part of the loading surface, letting free space for items to be delivered earlier to be placed on the front, so that they can be directly unloaded from the vehicle. Obviously, item insertions are performed only if they respect the *sequence* constraint.

As mentioned earlier, the two-dimensional packing heuristic can perform up to μ attempts for identifying a feasible item arrangement. Large μ values promote a thorough investigation of the loading feasibility of examined routes, at the expense of additional computational effort. In our implementation, we use $\mu = 1000$, for which the packing procedure performs very well within reasonable computational times. The aforementioned limit is used for investigating the loading feasibility of cost-reducing local-search moves. For tentative moves that increase the total

Table 3
Results for the 2|UO|L version of 2L-CVRP.

Inst.	Class 2				Class 3				Class 4				Class 5			
	z_{bst}	z_{avg}	gap%	t	z_{bst}	z_{avg}	gap%	t	z_{bst}	z_{avg}	gap%	t	z_{bst}	z_{avg}	gap%	t
1	278.73	278.73	0.00	0.3	284.52	284.52	0.00	0.3	282.95	282.95	0.00	0.2	278.73	278.73	0.00	0.7
2	334.96	334.96	0.00	0.2	352.16	352.51	0.10	0.2	334.96	334.96	0.00	0.3	334.96	334.96	0.00	0.3
3	387.70	387.70	0.00	0.4	394.72	394.72	0.00	0.4	364.45	364.45	0.00	0.4	358.40	358.40	0.00	0.4
4	430.88	430.88	0.00	0.4	430.88	430.88	0.00	0.3	447.37	447.83	0.10	0.2	430.88	430.88	0.00	0.2
5	375.28	375.28	0.00	1.4	381.69	381.69	0.00	1.2	383.87	383.87	0.00	0.8	375.28	375.28	0.00	0.9
6	495.85	495.85	0.00	0.2	498.16	498.48	0.06	0.2	498.32	498.32	0.00	0.3	495.85	495.85	0.00	0.3
7	725.46	725.46	0.00	1.8	678.75	678.75	0.00	2.0	700.72	700.72	0.00	0.7	657.77	657.77	0.00	2.0
8	674.55	674.55	0.00	2.8	738.43	738.67	0.03	3.0	692.47	692.47	0.00	2.9	609.90	613.25	0.55	1.7
9	607.65	607.65	0.00	0.1	607.65	607.65	0.00	0.1	625.10	625.10	0.00	3.1	607.65	610.97	0.54	3.2
10	689.68	689.68	0.00	26.9	620.33	620.33	0.00	21.0	710.87	710.87	0.00	25.8	686.12	686.12	0.00	33.9
11	694.47	694.47	0.00	29.9	706.73	706.73	0.00	21.6	786.85	786.85	0.00	23.8	624.82	624.82	0.00	33.5
12	610.57	610.57	0.00	1.7	610.00	610.00	0.00	1.2	614.23	614.23	0.00	2.0	610.23	610.23	0.00	0.9
13	2585.72	2585.72	0.00	46.7	2436.56	2436.56	0.00	35.8	2607.66	2607.66	0.00	73.4	2334.78	2334.78	0.00	54.9
14	1038.20	1038.20	0.00	203.3	1003.52	1003.52	0.00	186.3	981.00	981.00	0.00	158.5	880.80	894.43	1.52	109.1
15	1017.95	1017.95	0.00	24.8	1166.17	1169.19	0.26	27.2	1187.30	1192.44	0.43	15.1	1160.20	1160.20	0.00	13.5
16	698.61	698.61	0.00	0.7	698.61	698.61	0.00	0.5	703.35	703.35	0.00	6.4	698.61	698.61	0.00	8.8
17	870.86	872.59	0.20	3.0	861.79	862.21	0.05	3.0	861.79	861.79	0.00	1.9	861.79	861.88	0.01	1.7
18	1004.99	1004.99	0.00	33.9	1081.98	1081.98	0.00	26.0	1119.55	1119.55	0.00	21.6	921.29	921.29	0.00	50.6
19	762.58	763.37	0.10	27.2	772.20	776.88	0.60	26.7	780.39	782.41	0.26	25.4	651.97	651.97	0.00	17.9
20	525.75	530.28	0.85	562.2	522.01	522.01	0.00	433.7	540.58	542.90	0.43	608.1	473.41	474.94	0.32	605.0
21	997.94	1008.60	1.06	228.3	1119.88	1124.79	0.44	268.5	974.08	976.41	0.24	149.4	879.21	885.18	0.67	319.8
22	1035.81	1043.53	0.74	210.3	1058.87	1062.29	0.32	154.6	1062.78	1064.99	0.21	177.9	934.60	940.85	0.66	123.7
23	1035.18	1043.86	0.83	365.4	1079.03	1081.78	0.25	426.8	1075.36	1078.38	0.28	218.4	939.88	942.47	0.28	336.5
24	1178.07	1181.80	0.32	111.9	1080.88	1086.46	0.51	129.0	1111.27	1117.77	0.58	187.0	1048.33	1049.96	0.16	850.3
25	1412.72	1417.84	0.36	831.6	1370.27	1378.37	0.59	935.8	1409.98	1414.01	0.29	867.2	1172.08	1180.27	0.69	1052.2
26	1281.51	1284.15	0.21	247.1	1344.10	1356.21	0.89	285.1	1400.38	1409.00	0.61	673.6	1221.16	1231.09	0.81	408.2
27	1322.34	1329.22	0.52	560.1	1377.99	1383.49	0.40	407.7	1319.87	1322.73	0.22	413.4	1251.95	1253.90	0.16	371.7
28	2565.73	2596.23	1.17	3847.1	2608.27	2622.57	0.55	3222.2	2627.03	2637.53	0.40	2760.9	2320.81	2327.37	0.28	4977.5
29	2211.01	2230.27	0.86	2667.3	2094.60	2123.00	1.34	1994.0	2253.91	2264.97	0.49	1478.5	2132.55	2160.52	1.29	1202.8
30	1817.77	1827.48	0.53	2020.7	1835.79	1843.95	0.44	2335.8	1837.31	1839.99	0.15	2204.6	1551.50	1552.67	0.08	2046.2
31	2279.39	2294.77	0.67	3081.9	2287.11	2302.56	0.67	3202.5	2394.00	2401.35	0.31	3079.4	2016.03	2020.19	0.21	2345.8
32	2272.93	2292.39	0.85	3674.7	2246.74	2258.85	0.54	2739.8	2282.76	2287.57	0.21	3568.2	1984.95	1993.83	0.45	4872.4
33	2259.29	2283.56	1.06	1325.1	2371.86	2384.38	0.53	1050.9	2408.76	2410.78	0.08	2769.2	2006.82	2026.03	0.95	2714.0
34	1179.76	1185.98	0.52	4303.3	1204.87	1210.70	0.48	3274.5	1209.13	1213.04	0.32	3502.3	1035.13	1036.76	0.16	3126.5
35	1383.93	1398.41	1.04	3268.2	1446.20	1455.67	0.65	3673.8	1511.99	1517.89	0.39	2240.2	1262.56	1264.07	0.12	1843.9
36	1711.12	1734.77	1.36	3786.4	1781.75	1798.67	0.94	4008.1	1672.44	1682.09	0.57	4866.7	1512.46	1516.04	0.24	4321.1
Avg			*0.37*				*0.30*				*0.18*				*0.28*	

z_{bst}: Cost of highest quality solution identified over the ten runs.
z_{avg}: Average solution score achieved over the ten runs.
gap%: The percent gap between the best and average solution scores over the ten runs ($gap\% = 100(z_{avg} - z_{bst})/z_{avg}$).
t: Average CPU time elapsed when the best solutions of the ten runs were identified.

travel cost, we use $\mu = 1$. This matter is discussed in detail in Section 3.5.

4.4. Memory components used for storing loading feasibility information

Loading constraint investigations are repeatedly invoked by the routing optimization component of Section 3. Consequently, a great part of the total CPU time required by the overall 2L-CVRP solution approach is spent executing the proposed two-dimensional packing heuristic. To accelerate the algorithm, we use memory mechanisms for storing loading feasibility information obtained through the search process, and thus eliminate any redundant duplicate executions of the packing heuristic. This strategy of filtering-out unnecessary feasibility examinations is performed in two hierarchical layers: local-search move layer and tentative route layer.

In terms of the local search move layer, SMD instances are designed to contain the loading feasibility status of the encoded local search move. To store the packing feasibility status of a particular local search move, the corresponding SMD instance includes two attributes: the first one is a binary flag (*true*/*false*) which shows whether the encoded move satisfies the loading requirements, whereas the second attribute corresponds to the algorithmic iteration when this feasibility information was last determined. At any stage of the search process, if the loading feasibility of an SMD instance has to be investigated, the method checks if the routes involved in the encoded move have been modified since the recorded SMD loading feasibility was last obtained. If the routes have remained unmodified, the loading feasibility is directly retrieved by the feasibility flag of the SMD instance. If, on the contrary, the routes have been affected, the SMD feasibility has to be determined again by examining the loading feasibility of the new tentative routes.

When the loading feasibility of tentative routes must be examined, the method employs the second feasibility memory layer. More specifically, it checks whether the packing heuristic procedure has already been applied to the routes involved. If this is the case, instead of re-applying the packing method, it retrieves the relative information from two memory structures, responsible for storing obtained route feasibility information. These two structures, called *rht* and *rhtLight*, have been implemented as

Table 4
Comparative results for the 2|UO|L version (averaged over Classes 2–5).

Inst.	EGTS + LBFH		ACO (3 hours)		GRASPxELS		BAS	PRMP		
	z	*t*	*z*	*t*	*z*	*t*		*z*	*t*	*gap%*
1	291.97	3.2	285.77	2.9	282.65	0.9	282.65	**281.23**	0.4	0.50
2	341.89	1.7	341.02	0.2	**339.26**	0.1	339.26	**339.26**	0.3	0.00
3	379.41	3.5	**376.32**	1.2	**376.32**	0.5	376.32	**376.32**	0.4	0.00
4	440.85	2.8	438.66	1.4	**435.01**	0.2	435.01	**435.01**	0.3	0.00
5	382.22	3.1	**379.03**	6.4	**379.03**	0.1	379.03	**379.03**	1.1	0.00
6	498.96	3.4	497.28	2.0	**497.04**	0.4	497.04	**497.04**	0.3	0.00
7	699.29	7.7	696.91	4.8	691.11	1.4	691.11	**690.67**	1.6	0.06
8	701.77	9.9	691.14	10.0	**678.84**	0.8	678.84	**678.84**	2.6	0.00
9	614.67	4.8	612.02	2.1	**612.01**	0.6	612.01	**612.01**	1.6	0.00
10	705.04	27.3	682.53	30.1	**675.79**	15.1	675.79	676.75	26.9	−0.14
11	731.41	20.3	721.83	23.6	705.95	11.3	705.95	**703.22**	27.2	0.39
12	617.47	7.5	**611.26**	3.8	**611.26**	16.9	611.26	**611.26**	1.4	0.00
13	2581.41	23.5	2520.73	39.1	**2490.62**	78.0	2490.62	2491.18	52.7	−0.02
14	1030.49	34.3	991.26	96.5	984.42	79.9	984.42	**975.88**	164.3	0.87
15	1194.71	55.4	1167.28	68.9	1144.69	257.7	1144.69	**1132.91**	20.1	1.03
16	702.46	11.9	699.80	6.2	**699.79**	6.0	699.79	**699.79**	4.1	0.00
17	**862.62**	16.3	864.06	4.0	864.05	21.6	862.62	864.05	2.4	−0.17
18	1065.94	45.7	1043.31	162.7	**1029.71**	413.5	1029.71	1031.95	33.0	−0.22
19	771.57	43.8	752.05	60.5	**739.19**	268.5	739.19	741.78	24.3	−0.35
20	544.57	140.4	528.17	448.3	522.68	1658.3	522.68	**515.44**	552.2	1.39
21	1038.82	133.3	1015.05	405.1	994.58	1450.8	994.58	**992.78**	241.5	0.18
22	1061.73	169.4	1044.50	331.2	**1021.45**	965.0	1021.45	1023.01	166.6	−0.15
23	1065.70	128.0	1047.23	518.1	1038.16	1373.6	1038.16	**1032.36**	336.8	0.56
24	1119.11	173.1	1122.20	164.2	1107.93	480.3	1107.93	**1104.64**	319.6	0.30
25	1397.98	365.4	1365.77	864.2	1345.08	2967.7	1345.08	**1341.26**	921.7	0.28
26	1373.99	312.6	1346.22	1033.6	1317.41	2299.2	1317.41	**1311.79**	403.5	0.43
27	1377.29	262.2	1342.29	713.1	1323.54	2716.6	1323.54	**1318.04**	438.2	0.42
28	2673.04	769.6	2606.88	9276.3	2560.06	5065.5	2560.06	**2530.46**	3701.9	1.16
29	2272.00	1023.6	2232.09	9580.5	2191.46	4128.6	2191.46	**2173.02**	1835.7	0.84
30	1862.15	1141.9	1792.71	9222.1	1775.44	4753.7	1775.44	**1760.59**	2151.8	0.84
31	2378.20	3388.7	2292.41	10098.6	2282.28	4988.2	2282.28	**2244.13**	2927.4	1.67
32	2307.68	2503.3	2232.73	10350.2	2233.27	4900.6	2232.73	**2196.85**	3713.8	1.61
33	2422.95	2289.3	2305.73	10350.9	2284.82	4988.9	2284.82	**2261.68**	1964.8	1.01
34	1213.33	5350.2	1185.49	10349.5	1191.13	5244.5	1185.49	**1157.22**	3551.7	2.38
35	1479.40	5956.6	1432.04	10581.5	1435.22	5015.5	1432.04	**1401.17**	2756.5	2.16
36	1762.02	5648.7	1733.21	10795.0	1729.79	4874.0	1729.79	**1669.44**	4245.6	3.49
Avg										*0.57*

Each row provides the average values for the four problems of Classes 2–5.
EGTS + LBFH: The algorithm of Leung et al. (2011) – Intel Core 2 Duo 2.0 GHz, Visual C++.
ACO: The Ant Colony Optimization method of Fuellerer et al. (2009) – Pentium IV 3.2 GHz, C++, (Results reported in Doerner et al. (2007a)).
GRASPxELS: The method of Duhamel et al. (2011) – Opteron 2.1 GHz, C++.
PRMP: The proposed methodology – Intel Core 2 Duo E6600 2.4 GHz, Visual C#.
z: highest solution scores achieved.
t: CPU time (in seconds) elapsed when the best quality solutions were identified.
BAS: Best algorithmic solution (among EGTS + LBFH, ACO and GRASPxELS).
Bold entries correspond to higher quality solutions.
gap%: Percent gap between our solution scores and BAS ($gap\% = 100(\text{BAS} - z_{PRMP})/\text{BAS}$).

Table 5
Results on the 144 instances of the 2|UO|L version.

Inst	Class 2			Class 3			Class 4			Class 5		
	BKS	*z*	*gap%*	BKS	*z*	*gap%*	BKS	*z*	*gap%*	BKS	*z*	*gap%*
1	**278.73**	**278.73**	0.00	**284.52**	**284.52**	0.00	**282.95**	**282.95**	0.00	**278.73**	**278.73**	0.00
2	**334.96**	**334.96**	0.00	**352.16**	**352.16**	0.00	**334.96**	**334.96**	0.00	**334.96**	**334.96**	0.00
3	**387.70**	**387.70**	0.00	**390.55**	394.72	−1.07	**362.41**	364.45	−0.56	**358.40**	**358.40**	0.00
4	**430.88**	**430.88**	0.00	**430.88**	**430.88**	0.00	**447.37**	**447.37**	0.00	**430.88**	**430.88**	0.00
5	**375.28**	**375.28**	0.00	**381.69**	**381.69**	0.00	**383.87**	**383.87**	0.00	**375.28**	**375.28**	0.00
6	**495.85**	**495.85**	0.00	**497.17**	498.16	−0.20	**498.32**	**498.32**	0.00	**495.75**[a]	495.85	−0.02
7	**725.46**	**725.46**	0.00	**678.75**	**678.75**	0.00	**700.72**	**700.72**	0.00	**657.77**	**657.77**	0.00
8	**674.55**	**674.55**	0.00	**738.43**	**738.43**	0.00	**692.47**	**692.47**	0.00	**609.90**	**609.90**	0.00
9	**607.65**	**607.65**	0.00	**607.65**	**607.65**	0.00	**621.23**	625.10	−0.62	**607.65**	**607.65**	0.00
10	**689.68**	**689.68**	0.00	**615.68**	620.33	−0.75	**710.87**	**710.87**	0.00	686.78	**686.12**	0.10
11	**693.45**	694.47	−0.15	**706.73**	**706.73**	0.00	**786.85**	**786.85**	0.00	636.77	**624.82**	1.88
12	**610.57**	**610.57**	0.00	**610.00**	**610.00**	0.00	**614.23**	**614.23**	0.00	**610.00**	610.23	−0.04
13	**2585.72**	**2585.72**	0.00	**2436.56**	**2436.56**	0.00	**2587.63**	2607.66	−0.77	**2334.78**	**2334.78**	0.00
14	**1038.09**	1038.20	−0.01	**996.11**	1003.52	−0.74	981.90	**981.00**	0.09	921.45	**880.80**	4.41
15	**1013.29**	1017.95	−0.46	**1154.66**	1166.17	−1.00	1234.14	**1187.30**	3.80	1176.68	**1160.20**	1.40
16	**698.61**	**698.61**	0.00	**698.61**	**698.61**	0.00	**703.35**	**703.35**	0.00	**698.61**	**698.61**	0.00
17	**863.66**	870.86	−0.83	**861.79**	**861.79**	0.00	**861.79**	**861.79**	0.00	**861.79**	**861.79**	0.00
18	**1004.99**	**1004.99**	0.00	**1069.45**	1081.98	−1.17	**1118.71**	1119.55	−0.07	925.72	**921.29**	0.48
19	**754.53**	762.58	−1.07	**771.74**	772.20	−0.06	**778.35**	780.39	−0.26	652.15	**651.97**	0.03
20	534.24	**525.75**	1.59	524.81	**522.01**	0.53	547.95	**540.58**	1.35	480.10	**473.41**	1.39
21	**992.83**	997.94	−0.51	1121.84	**1119.88**	0.17	978.82	**974.08**	0.48	884.84	**879.21**	0.64
22	1036.11	**1035.81**	0.03	**1052.98**	1058.87	−0.56	**1045.91**	1062.78	−1.61	950.79	**934.60**	1.70
23	1041.04	**1035.18**	0.56	1081.48	**1079.03**	0.23	1080.02	**1075.36**	0.43	950.09	**939.88**	1.07
24	1178.39	**1178.07**	0.03	1083.14	**1080.88**	0.21	**1111.27**	**1111.27**	0.00	**1028.04**	1048.33	−1.97
25	1419.42	**1412.72**	0.47	1374.68	**1370.27**	0.32	**1405.65**	1409.98	−0.31	1180.57	**1172.08**	0.72
26	1285.01	**1281.51**	0.27	1344.66	**1344.10**	0.04	1405.57	**1400.38**	0.37	1234.39	**1221.16**	1.07
27	1327.06	**1322.34**	0.36	1378.01	**1377.99**	0.00	1326.16	**1319.87**	0.47	1262.93	**1251.95**	0.87
28	2587.23	**2565.73**	0.83	2629.38	**2608.27**	0.80	2641.09	**2627.03**	0.53	2361.89	**2320.81**	1.74
29	**2208.57**	2211.01	−0.11	2107.87	**2094.60**	0.63	2270.44	**2253.91**	0.73	2163.51	**2132.55**	1.43
30	**1816.05**	1817.77	−0.09	1850.78	**1835.79**	0.81	1856.54	**1837.31**	1.04	1562.30	**1551.50**	0.69
31	2291.41	**2279.39**	0.52	2305.51	**2287.11**	0.80	2436.42	**2394.00**	1.74	2060.50	**2016.03**	2.16
32	2282.15	**2272.93**	0.40	2267.82	**2246.74**	0.93	2308.40	**2282.76**	1.11	2017.15	**1984.95**	1.60
33	2285.94	**2259.29**	1.17	2390.58	**2371.86**	0.78	2416.77	**2408.76**	0.33	2029.22	**2006.82**	1.10
34	**1136.13**	1179.76	−3.84	**1179.87**	1204.87	−2.12	**1183.71**	1209.13	−2.15	**1031.74**	1035.13	−0.33
35	1392.07	**1383.93**	0.58	1477.05	**1446.20**	2.09	1538.30	**1511.99**	1.71	1281.65	**1262.56**	1.49
36	1736.62	**1711.12**	1.47	1824.85	**1781.75**	2.36	1728.69	**1672.44**	3.25	1549.28	**1512.46**	2.38
Avg			*0.03*			*0.08*			*0.31*			*0.72*

BKS: Cost of the best known solution (reported in Doerner et al. (2007a), Fuellerer et al. (2009), Leung et al. (2011), Duhamel et al. (2011), Strodl et al. (2010)).
z: Cost of the highest quality solution obtained by the PRMP method.
Bold entries correspond to higher quality solutions.
gap%: Percent gap between our solution values and the best known solution scores ($gap\% = 100(\text{BKS} - z)/\text{BKS}$).
[a] This value reported by Leung et al. (2011) appears to be lower than the solution score for Problem 6-Class 1, thus it might be mistakenly reported.

hashtables. The *rht* structure stores feasibility information obtained through "strong" packing heuristic applications, with $\mu = 1000$ (for cost-improving moves), whereas *rhtLight* keeps route feasibility information defined by "light", single-attempt ($\mu = 1$) applications of the packing method (for cost-deteriorating moves). Each entry of both hashtables contains a key that encodes a particular route, together with a binary value (*true*/*false*) corresponding to the loading feasibility status of this route. We note that the hashtable structures have been implemented using the *System.Collections.HashTable* class of *.NET Framework 2.0*.

Pseudocode 1. Strategy for eliminating unnecessary calls to the packing heuristics.

```
bool LoadingFeasibility (SMD $smd$, int $it$)
    Route $r1 = r1_{smd}$; Route $r2 = r2_{smd}$;
    //Local Search Layer
    if ($it_{r1} < it_{smd}$ AND $it_{r2} < it_{smd}$) return $f_{smd}$;
    //Tentative Route Layer
    $r1 = r1'_{smd}$; $r2 = r2'_{smd}$;
    bool $f1 = null$; bool $f2 = null$;
    if (value ($rht$, key ($r1$)) $\neq null$)
        $f1$ = value ($rht$, key ($r1$))
        if ($f1 == false$) $it_{smd} = it$; $f_{smd} = false$; return $false$;
    end if
    if (value ($rht$, key ($r2$)) $\neq null$)
        $f2$ = value ($rht$, key ($r2$))
        if ($f2 == false$) $it_{smd} = it$; $f_{smd} = false$; return $false$;
    end if
    if ($ct_{smd} < 0$)
        if ($f1 == null$)
            $f1$ = pack ($r1$, 1000); push ($rht$, key ($r1$), $f1$);
            if ($f1 == false$) $it_{smd} = it$; $f_{smd} = false$; return $false$;
        end if
        if($f2 == null$)
            $f2$ = pack ($r2$, 1000); push ($rht$, key ($r2$), $f2$);
            if ($f2 == false$) $it_{smd} = it$; $f_{smd} = false$; return $false$;
        end if
    else
        if ($f1 == null$ AND value ($rhtLight$, key ($r1$)) $\neq null$)
            $f1$ = value ($rhtLight$, key ($r1$));
            if ($f1 == false$) $it_{smd} = it$; $f_{smd} = false$;return $false$;
        end if
        if ($f2 == null$ AND value ($rhtLight$, key ($r2$)) $\neq null$)
            $f2$ = value ($rhtLight$, key ($r2$));
            if ($f2 == false$) $it_{smd} = it$; $f_{smd} = false$;return $false$;
        end if
```

```
      if (f1 == null)
        f1 = pack (r1, 1); push (rhtLight, key (r1), f1);
        if (f1 == false) $it_{smd}$ = it; $f_{smd}$ = false; return false;
      end if
      if(f2 == null)
        f2 = pack (r2, 1); push (rhtLight, key (r2), f2);
        if (f2 == false) $it_{smd}$ = it; $f_{smd}$ = false; return false;
      end if
    end if
    if(value (rhtLight, key (r1)) == false) remove (rhtLight, key (r1)); end if
    if(value (rhtLight, key (r2)) == false) remove (rhtLight, key (r2)); end if
    if(value (rhtLight, key (r1)) == null) push (rht, key (r1), true); end if
    if(value (rhtLight, key (r2)) == null) push (rht, key (r2), true); end if
    if(value (rht, key (r1)) == null) push (rht, key (r1), true); end if
    if(value (rht, key (r2)) == null) push (rht, key (r2), true); end if
    $it_{smd}$ = it; $f_{smd}$ = true;return true;
```

To thoroughly present the rationale used for investigating the loading feasibility of a local-search move, we provide Pseudocode 1. It presents a method which takes as input the SMD instance encoding the investigated local-search move (*smd*) and the current iteration number (*it*). Without loss of generality, assume that the SMD considered corresponds to an inter-route move, and thus involves two routes. The routine returns *true* if the move is found to be feasible and *false* otherwise. Pseudocode 1 makes use of the following notation:

ct_{smd}: the cost tag of SMD instance *smd* (objective function change incurred by applying *smd*).
f_{smd}: the loading feasibility flag of SMD instance *smd* (equal to *true*/*false*).
it_{smd}: the algorithmic iteration when the feasibility of SMD instance *smd* was last determined.
$r1_{smd}$, $r2_{smd}$: the routes involved in the move encoded by SMD instance *smd*, before the move application.
$r1'_{smd}$, $r2'_{smd}$: the modified routes, after applying the move encoded by SMD instance *smd*.
it_r: the algorithmic iteration, when route *r* was modified for the last time.
pack (r, μ): executes the packing heuristic (Section 4.3) for route

Table 6
Results for the 2|SO|L version of 2L-CVRP.

Inst	Class 2				Class 3				Class 4				Class 5			
	z_{bst}	z_{avg}	*gap%*	*t*	z_{bst}	z_{avg}	*gap%*	*t*	z_{bst}	z_{avg}	*gap%*	*t*	z_{bst}	z_{avg}	*gap%*	*t*
1	290.84	290.84	0.00	0.5	284.52	284.52	0.00	0.5	294.25	294.25	0.00	1.8	278.73	278.73	0.00	2.7
2	347.73	347.73	0.00	0.5	352.16	352.16	0.00	0.5	342.00	342.00	0.00	1.4	334.96	334.96	0.00	1.7
3	403.93	403.93	0.00	0.6	394.72	394.72	0.00	0.7	368.56	368.56	0.00	1.5	358.40	358.40	0.00	2.2
4	440.94	440.94	0.00	0.6	440.68	440.68	0.00	0.7	447.37	447.37	0.00	2.8	430.88	430.88	0.00	2.4
5	388.72	388.72	0.00	2.3	381.69	381.69	0.00	2.7	383.87	383.87	0.00	2.5	375.28	375.28	0.00	2.7
6	499.08	499.08	0.00	4.3	504.68	504.68	0.00	5.2	498.32	498.32	0.00	5.2	495.85	495.85	0.00	7.6
7	734.65	734.65	0.00	4.8	709.72	709.72	0.00	5.8	703.49	703.49	0.00	4.5	661.22	661.22	0.00	6.0
8	725.91	725.91	0.00	7.1	741.12	741.12	0.00	7.9	697.92	697.92	0.00	6.1	633.23	633.23	0.00	6.9
9	611.49	611.49	0.00	6.2	619.48	619.48	0.00	5.0	625.10	625.10	0.00	7.0	607.65	607.65	0.00	6.4
10	700.20	700.20	0.00	52.8	646.87	646.87	0.00	66.6	715.82	716.42	0.08	50.8	691.04	691.04	0.00	49.9
11	721.54	721.54	0.00	86.8	720.45	720.45	0.00	68.4	815.68	815.68	0.00	59.1	645.65	645.65	0.00	87.0
12	619.63	619.63	0.00	5.1	610.00	612.16	0.35	4.6	618.23	618.23	0.00	8.6	610.23	610.91	0.11	10.2
13	2669.39	2669.39	0.00	103.0	2504.78	2504.78	0.00	98.6	2610.57	2610.57	0.00	106.6	2434.99	2434.99	0.00	170.2
14	1101.61	1113.81	1.10	558.3	1093.08	1093.82	0.07	624.1	989.60	989.60	0.00	582.0	925.21	925.21	0.00	783.8
15	1099.91	1104.05	0.37	64.5	1181.68	1181.90	0.02	65.6	1247.69	1250.77	0.25	78.2	1230.60	1238.81	0.66	64.4
16	698.61	698.61	0.00	9.4	698.61	698.61	0.00	12.3	708.20	708.20	0.00	14.1	698.61	698.61	0.00	20.8
17	870.86	870.86	0.00	38.6	861.79	862.34	0.06	46.9	861.79	866.32	0.52	37.3	861.79	861.79	0.00	40.6
18	1059.44	1059.46	0.00	88.7	1106.33	1114.53	0.74	82.2	1137.34	1139.06	0.15	91.8	928.88	932.11	0.35	117.8
19	794.15	794.15	0.00	175.3	802.76	802.76	0.00	181.3	807.46	808.51	0.13	202.0	659.84	663.09	0.49	194.7
20	549.57	552.19	0.48	1647.4	546.63	546.66	0.01	2021.0	555.59	556.76	0.21	1591.9	487.68	487.94	0.05	1383.1
21	1070.66	1073.03	0.22	398.7	1168.94	1169.39	0.04	415.3	1014.17	1014.71	0.05	372.5	913.32	915.35	0.22	494.1
22	1086.25	1087.39	0.10	510.1	1117.20	1120.15	0.26	493.9	1101.93	1104.34	0.22	518.7	960.87	961.52	0.07	574.3
23	1113.50	1124.13	0.95	594.5	1121.51	1123.93	0.21	501.4	1105.73	1105.85	0.01	428.2	964.02	968.44	0.46	553.9
24	1222.43	1232.00	0.78	894.4	1128.30	1130.19	0.17	947.7	1150.24	1151.99	0.15	1087.9	1055.51	1061.33	0.55	1327.2
25	1476.34	1480.74	0.30	1630.8	1452.95	1454.77	0.13	2026.2	1467.34	1467.92	0.04	2178.3	1207.36	1209.88	0.21	3442.6
26	1330.94	1331.40	0.03	1308.8	1411.53	1411.96	0.03	1004.2	1468.43	1475.36	0.47	1340.9	1272.21	1279.25	0.55	2310.7
27	1367.87	1374.21	0.46	2495.3	1449.27	1453.32	0.28	4183.0	1371.88	1372.96	0.08	4606.0	1300.94	1312.02	0.84	5371.0
28	2717.14	2725.69	0.31	9839.9	2800.34	2808.39	0.29	9709.9	2731.04	2735.18	0.15	8251.6	2427.73	2441.58	0.57	6759.0
29	2309.35	2325.06	0.68	5045.0	2201.64	2208.47	0.31	4535.6	2340.96	2347.79	0.29	4991.1	2203.11	2211.69	0.39	7365.5
30	1915.54	1918.68	0.16	4024.2	1947.51	1958.95	0.58	4302.7	1946.53	1948.86	0.12	4190.1	1602.51	1608.26	0.36	6190.5
31	2389.26	2406.01	0.70	7505.3	2413.28	2419.04	0.24	5587.2	2523.98	2534.42	0.41	5636.8	2106.52	2123.60	0.80	4652.2
32	2413.19	2424.74	0.48	9575.2	2386.89	2399.14	0.51	10835.0	2410.15	2412.25	0.09	8457.3	2080.61	2092.97	0.59	8865.4
33	2415.80	2427.81	0.49	4772.8	2535.42	2535.91	0.02	5618.5	2532.13	2543.04	0.43	5175.3	2093.93	2111.31	0.82	7083.5
34	1253.52	1256.21	0.21	10237.5	1278.94	1280.38	0.11	12919.9	1279.67	1290.75	0.86	14958.1	1090.00	1100.24	0.93	14451.7
35	1491.44	1499.90	0.56	7247.7	1566.63	1569.03	0.15	9090.7	1599.02	1603.94	0.31	10615.4	1320.18	1331.02	0.81	9004.5
36	1810.07	1824.84	0.81	9874.5	1889.21	1895.29	0.32	7456.8	1768.88	1785.06	0.91	9344.3	1580.52	1594.56	0.88	13562.9
Avg			*0.26*				*0.14*				*0.16*				*0.30*	

z_{bst}: Cost of highest quality solution identified over the ten runs.
z_{avg}: Average solution score achieved over the ten runs.
gap%: The percent gap between the best and average solution scores over the ten runs ($gap\% = 100(z_{avg} - z_{bst})/z_{avg}$).
t: Average CPU time elapsed when the best solutions of the ten runs were identified.

r and for μ attempts. Returns *true* if r is found to be feasible, *false* otherwise.
key (r): returns the hash key for a route r (hashing function).
value (h,k): Returns the value stored in hashtable h for a key k. If no such value exists, it returns *null*.
push (h,f,k): Inserts the key k – value f pair into hashtable h.
remove (h,k): Removes the hashtable h entry which corresponds to key k.

The above-described scheme for eliminating unnecessary calls to the packing heuristic method plays a critical role in the overall design of the 2L-CVRP algorithm. Experimental runs without their use, especially for instances of more than 50 customers, required impractical CPU times. More specifically and for some 2L-CVRP benchmark instances, the speed-up factors observed, when using the proposed memory structures, were up to two orders of magnitude.

5. Computational results

To assess the effectiveness of the proposed algorithm, thereafter referred to as PRMP (Promise Routing-Memory Packing), we have solved the 2L-CVRP instances generated by Gendreau et al. (2008). Two problem versions were tackled, depending on the loading constraint configuration under consideration: 2|UO|L version (no sequence constraint, fixed orientation) and 2|SO|L version (sequence constraint, fixed orientation). Our methodology was implemented in Visual C# and executed on a single core of an Intel E6600 processor (2.4 GHz). The analytical solutions obtained are available at http://users.ntua.gr/ezach/.

5.1. The 2L-CVRP benchmark instances

Gendreau et al. (2008) have introduced a set of 2L-CVRP benchmark instances, by extending well-known classical CVRP test problems. With each customer of the original CVRP problem, a set of demanded two-dimensional items has been associated. In addition, the number of available vehicles (k) and the dimensions of the loading surfaces (W,L) were also specified. More specifically, Gendreau et al. (2008) have used 36 CVRP instances containing from 15 to 255 customers. For each of these CVRP problems, they have created five 2L-CVRP instances by considering five classes of item characteristics (Class 1–5). Problems of Class 1 can be regarded as pure CVRP problems, involving items of unit dimensions. They are used to evaluate the performance of algorithms, solely in

Table 7
Comparative results for the 2|SO|L version (averaged over Classes 2–5).

Inst.	EGTS + LBFH		ACO (3 hours)		BAS	PRMP		
	z	*t*	*z*	*t*		*z*	*t*	*gap%*
1	303.40	3.2	294.48	6.9	294.48	**287.08**	1.37	2.51
2	345.23	1.3	345.23	0.3	345.23	**344.21**	1.04	0.29
3	387.89	5.0	**381.40**	2.9	381.40	**381.40**	1.26	0.00
4	443.25	2.4	441.11	2.9	441.11	**439.97**	1.62	0.26
5	387.60	4.6	**382.39**	12.1	382.39	**382.39**	2.55	0.00
6	502.25	3.5	**499.48**	4.1	499.48	**499.48**	5.55	0.00
7	715.54	8.3	**702.27**	11.4	702.27	**702.27**	5.27	0.00
8	716.36	8.3	711.65	11.9	711.65	**699.54**	7.02	1.70
9	621.23	4.3	**614.54**	4.8	614.54	615.93	6.17	−0.23
10	731.69	23.3	697.20	59.2	697.20	**688.48**	55.03	1.25
11	762.83	38.0	728.61	68.8	728.61	**725.83**	75.31	0.38
12	622.35	7.9	615.77	7.7	615.77	**614.52**	7.13	0.20
13	2647.88	30.8	2591.76	73.2	2591.76	**2554.93**	119.61	1.42
14	1075.04	49.0	1042.34	192.6	1042.34	**1027.38**	637.04	1.44
15	1223.19	65.8	1212.94	166.7	1212.94	**1189.97**	68.14	1.89
16	703.74	13.0	701.28	7.8	701.28	**701.00**	14.16	0.04
17	869.93	16.1	864.06	4.3	864.06	**864.05**	40.85	0.00
18	1096.57	67.8	1068.14	354.1	1068.14	**1058.00**	95.15	0.95
19	798.20	61.7	774.09	146.3	774.09	**766.05**	188.31	1.04
20	559.17	232.5	541.66	1307.1	541.66	**534.87**	1660.87	1.25
21	1084.98	179.0	1049.25	1277.5	1049.25	**1041.77**	420.17	0.71
22	1113.64	152.3	1076.58	894.0	1076.58	**1066.56**	524.22	0.93
23	1130.13	215.3	1088.05	1482.8	1088.05	**1076.19**	519.52	1.09
24	1177.28	132.7	1150.80	378.4	1150.80	**1139.12**	1064.30	1.01
25	1470.11	373.2	1413.29	2988.1	1413.29	**1401.00**	2319.48	0.87
26	1431.32	499.0	1407.01	2919.7	1407.01	**1370.78**	1491.15	2.57
27	1445.64	371.4	1389.46	1873.1	1389.46	**1372.49**	4163.84	1.22
28	2808.10	979.8	2693.61	10370.6	2693.61	**2669.07**	8640.09	0.91
29	2396.78	1150.4	2306.16	10016.1	2306.16	**2263.76**	5484.32	1.84
30	1983.48	1699.0	1886.05	10254.4	1886.05	**1853.02**	4676.89	1.75
31	2497.25	4368.2	2392.08	10508.8	2392.08	**2358.26**	5845.41	1.41
32	2438.65	2445.8	2364.92	10728.2	2364.92	**2322.71**	9433.23	1.78
33	2543.24	2053.7	2435.22	10668.7	2435.22	**2394.32**	5662.52	1.68
34	1276.27	3443.5	1251.62	10649.9	1251.62	**1225.54**	13141.80	2.08
35	1606.38	4560.8	1582.32	10657.0	1582.32	**1494.32**	8989.57	5.56
36	1850.50	3667.1	1828.37	10798.8	1828.37	**1762.17**	10059.61	3.62
Avg								*1.21*

Each row provides the average values for the four problems of Classes 2–5.
EGTS + LBFH: The algorithm of Leung et al. (2011) – Intel Core 2 Duo 2.0 GHz, Visual C++.
ACO: The Ant Colony Optimization method of Fuellerer et al. (2009) – Pentium IV 3.2 GHz, C++, (Results reported in Doerner et al. (2007a)).
PRMP: The proposed methodology – Intel Core 2 Duo E6600 2.4 GHz, Visual C#.
z: Highest solution scores achieved.
t: CPU time (in seconds) elapsed when the best quality solutions were identified.
BAS: Best algorithmic solution (among EGTS + LBFH and ACO).
Bold entries correspond to higher quality solutions.
gap%: Percent gap between our solution scores and BAS ($gap\% = 100(\text{BAS} - z_{PRMP})/\text{BAS}$).

terms of the routing aspects of the problem. Problems of Classes 2–5 correspond to actual 2L-CVRP problems, for which feasible loading patterns must be identified. Table 1 presents the characteristics of the 144 2L-CVRP benchmark instances of Classes 2–5.

5.2. Results on the pure CVRP benchmark instances

The proposed method was applied ten times on each of the pure CVRP instances of Class 1. The best results obtained are provided in Table 2. The routing component of the proposed methodology proved to be rather effective, as it managed to produce better solutions compared to the ones obtained by previously published 2L-CVRP approaches for 8 test problems, while it robustly matched the best solution scores for 26 instances. For benchmark instances 10 and 28, PRMP generated solutions increasing the best objective scores by 0.01% and 0.14%, respectively. The average solution cost improvement over the previous best algorithmic scores is 0.07%. Regarding the CPU times, we do not intend to perform analytic comparisons, because different computers, programming languages and operating systems have been used. However, the routing component of PRMP appears to be faster than the compared methods, as the average CPU time required for obtaining the best solutions ranged up to 434.1 s.

5.3. Results for the 2|UO|L problem version

As far as the 2|UO|L problem version is concerned, the algorithm has been executed ten times on each of the 144 2L-CVRP benchmark instances of Classes 2–5. In Table 3, the obtained results are provided. We observe that the proposed method is rather stable, as the percentage gaps between the average and best solution scores achieved over the ten runs were limited to 0.37%, 0.30%, 0.18% and 0.28%, for the problems of Classes 2, 3, 4 and 5, respectively. The CPU time required for obtaining the best solutions ranged up to 4977.5 s (Instance 28 – Class 5), which is deemed reasonable, taking into account both the complexity of the 2L-CVRP model and the high quality of the obtained results.

Table 4 compares the best solution scores averaged over the four instances (Classes 2–5) of each test problem (problems 1–36) against previous solution approaches proposed for the 2|UO|L version of the 2L-CVRP. We note that Table 4 does not contain the scores presented in the work of Strodl et al. (2010), because the methods presented in this work did not obtain feasible solutions for all 144 instances, thus a direct comparison would be problematic. Our results are compared to those obtained by EGTS + LBFH (Leung et al., 2011), ACO-3 CPU hours (Fuellerer et al., 2009) and GRASPxELS (Duhamel et al., 2011). Note that for

Table 8
Results on the 144 instances of the 2|SO|L version.

Inst	Class 2			Class 3			Class 4			Class 5		
	BKS	*z*	*gap%*	BKS	*z*	*gap%*	BKS	*z*	*gap%*	BKS	*z*	*gap%*
1	**290.84**	**290.84**	0.00	304.41	**284.52**	6.53	296.75	**294.25**	0.84	285.93	**278.73**	2.52
2	**347.73**	**347.73**	0.00	356.24	**352.16**	1.15	**342.00**	**342.00**	0.00	**334.96**	334.96	0.00
3	**403.93**	**403.93**	0.00	**394.72**	**394.72**	0.00	**368.56**	**368.56**	0.00	**358.40**	**358.40**	0.00
4	**440.94**	**440.94**	0.00	445.25	**440.68**	1.03	**447.37**	**447.37**	0.00	**430.88**	430.88	0.00
5	**388.72**	**388.72**	0.00	**381.69**	**381.69**	0.00	383.88	**383.87**	0.00	**375.28**	375.28	0.00
6	**499.08**	**499.08**	0.00	**504.68**	**504.68**	0.00	**498.32**	**498.32**	0.00	**495.85**	**495.85**	0.00
7	**734.65**	**734.65**	0.00	**709.72**	**709.72**	0.00	**703.49**	**703.49**	0.00	**661.22**	**661.22**	0.00
8	**725.91**	**725.91**	0.00	**741.12**	**741.12**	0.00	727.42	**697.92**	4.06	646.46	**633.23**	2.05
9	**611.49**	**611.49**	0.00	**613.90**	619.48	−0.91	625.13	**625.10**	0.01	**607.65**	**607.65**	0.00
10	**700.20**	**700.20**	0.00	**628.93**	646.87	−2.85	760.61	**715.82**	5.89	699.05	**691.04**	1.15
11	**721.54**	**721.54**	0.00	**718.09**	720.45	−0.33	816.10	**815.68**	0.05	658.71	**645.65**	1.98
12	**619.63**	**619.63**	0.00	**610.00**	**610.00**	0.00	623.20	**618.23**	0.80	**610.23**	**610.23**	0.00
13	**2669.39**	**2669.39**	0.00	**2497.42**	2504.78	−0.29	2689.59	**2610.57**	2.94	2508.22	**2434.99**	2.92
14	1135.93	**1101.61**	3.02	1093.63	**1093.08**	0.05	993.47	**989.60**	0.39	946.31	**925.21**	2.23
15	1109.10	**1099.91**	0.83	1189.09	**1181.68**	0.62	1276.85	**1247.69**	2.28	1273.60	**1230.60**	3.38
16	**698.61**	**698.61**	0.00	**698.61**	**698.61**	0.00	709.27	**708.20**	0.15	**698.61**	**698.61**	0.00
17	**870.86**	**870.86**	0.00	**861.79**	**861.79**	0.00	**861.79**	**861.79**	0.00	**861.79**	**861.79**	0.00
18	**1059.44**	**1059.44**	0.00	1120.55	**1106.33**	1.27	1153.37	**1137.34**	1.39	939.20	**928.88**	1.10
19	794.75	**794.15**	0.08	**801.13**	802.76	−0.20	824.15	**807.46**	2.03	676.32	**659.84**	2.44
20	553.12	**549.57**	0.64	550.28	**546.63**	0.66	567.72	**555.59**	2.14	495.51	**487.68**	1.58
21	1076.95	**1070.66**	0.58	**1162.07**	1168.94	−0.59	1032.04	**1014.17**	1.73	925.94	**913.32**	1.36
22	1086.67	**1086.25**	0.04	1123.10	**1117.20**	0.53	1114.22	**1101.93**	1.10	982.34	**960.87**	2.19
23	1116.36	**1113.50**	0.26	1141.01	**1121.51**	1.71	1110.62	**1105.73**	0.44	984.19	**964.02**	2.05
24	1237.65	**1222.43**	1.23	**1126.33**	1128.30	−0.17	1160.59	**1150.24**	0.89	1078.62	**1055.51**	2.14
25	1478.83	**1476.34**	0.17	1459.58	**1452.95**	0.45	1480.83	**1467.34**	0.91	1233.93	**1207.36**	2.15
26	1332.40	**1330.94**	0.11	**1409.10**	1411.53	−0.17	1572.23	**1468.43**	6.60	1304.02	**1272.21**	2.44
27	1376.49	**1367.87**	0.63	1450.35	**1449.27**	0.07	1404.41	**1371.88**	2.32	1326.58	**1300.94**	1.93
28	2726.26	**2717.14**	0.33	**2796.83**	2800.34	−0.13	2765.90	**2731.04**	1.26	2485.45	**2427.73**	2.32
29	2328.50	**2309.35**	0.82	2231.33	**2201.64**	1.33	2390.83	**2340.96**	2.09	2273.96	**2203.11**	3.12
30	**1899.53**	1915.54	−0.84	1980.40	**1947.51**	1.66	2020.59	**1946.53**	3.67	1643.68	**1602.51**	2.50
31	**2382.29**	2389.26	−0.29	2425.91	**2413.28**	0.52	2611.01	**2523.98**	3.33	2149.12	**2106.52**	1.98
32	2419.54	**2413.19**	0.26	2400.24	**2386.89**	0.56	2518.80	**2410.15**	4.31	2121.11	**2080.61**	1.91
33	2431.53	**2415.80**	0.65	**2522.99**	2535.42	−0.49	2627.36	**2532.13**	3.62	2159.01	**2093.93**	3.01
34	1265.01	**1253.52**	0.91	1302.24	**1278.94**	1.79	1324.63	**1279.67**	3.39	1114.58	**1090.00**	2.21
35	1504.61	**1491.44**	0.88	1576.22	**1566.63**	0.61	1890.84	**1599.02**	15.43	1357.62	**1320.18**	2.76
36	1858.33	**1810.07**	2.60	1952.47	**1889.21**	3.24	1843.85	**1768.88**	4.07	1651.33	**1580.52**	4.29
Avg			*0.36*			*0.49*			*2.17*			*1.66*

BKS: Cost of the best known solution (reported in Doerner et al. (2007a), Fuellerer et al. (2009), Leung et al. (2011)).
z: Cost of the highest quality solution obtained by the proposed method.
Bold entries correspond to higher quality solutions.
gap%: Percent gap between our solution values and the best known solution scores (*gap%* = 100(BKS − *z*)/BKS).

the ACO method, detailed solution values were found at the report of Doerner et al. (2007a). Our approach managed to improve the average solution scores for 21 out of 36 benchmark problems, whereas for 9 test problems, it matched the best solution values. The PRMP solution cost improvement varied up to a significant 3.49% for the largest-scale problem 36, averaging at 0.57%.

Individual results for the 144 problems of Classes 2–5, under the 2|UO|L version are reported in Table 5, where PRMP solution values are compared to the best known solution scores. We observe that our method improved 13, 15, 15 and 21 previously best known solutions, for Classes 2, 3, 4 and 5, respectively. The average solution improvements for Classes 2, 3, 4 and 5 were equal to 0.03%, 0.08%, 0.31% and 0.72%, respectively, indicating that our method performs better for problems which involve many items per vehicle (Class 5). This can be attributed to the proposed memory mechanism used for item insertions which is capable of testing diverse item orderings for building feasible packing patterns.

5.4. Results for the 2|SO|L problem version

PRMP was applied ten times on each of the 144 benchmark instances of Classes 2–5, considering the 2|SO|L problem version. The obtained results are reported in Table 6. Again, the method proves to be stable, as the gaps between the best and average solution scores obtained are limited to 0.26%, 0.14%, 0.16% and 0.30% for problems of Classes 2, 3, 4 and 5, respectively. Another observation is related to the CPU time required for obtaining the final solutions which is significantly higher (almost 3 times) compared to the CPU time required by the 2|UO|L version. This is because the number of necessary route feasibility investigations is greater: for the 2|UO|L version, one feasibility test is necessary for every customer combination, whereas for the 2|SO|L version, where customer ordering is taken into account, one feasibility test must be applied for every customer permutation.

Table 7 compares the best solution scores averaged over the four instances (Classes 2–5) of each test problem (instances 1–36) against previous solution approaches proposed for the 2|SO|L version of 2L-CVRP. These approaches are the EGTS + LBFH algorithm (Leung et al., 2011) and the ACO metaheuristic (Fuellerer et al., 2009). We observe that the proposed method consistently improves or matches the best algorithmic scores for 35 out of the 36 problems. More specifically, for 30 instances, PRMP improves the average solution scores obtained by EGTS + LBFH and ACO, whereas for 5 instances, it matches their solution values. The average improvement is equal to 1.21%. In terms of the computational effort, the CPU time required by PRMP and ACO methods are comparable, whereas EGTS + LBFH appears to be much faster, however it consistently produces solutions of poorer quality.

Individual results for the 144 problems of Classes 2–5, under the 2|SO|L version are provided in Table 8, where the PRMP scores are compared to the best known solution values. We see that the proposed method improves 18, 18, 29 and 26 best known solution scores for Classes 2, 3, 4 and 5, respectively. For the aforementioned Classes of problems, the improvement over the best known solution scores averages at 0.36%, 0.49%, 2.17% and 1.66%, respectively.

6. Conclusions

In the present paper, we have introduced a new metaheuristic methodology for the vehicle routing problem with two-dimensional loading constraints (2L-CVRP). The proposed solution framework optimizes the routing characteristics by applying an efficient and compact-structured local-search method which contains only one search parameter. The loading constraints are tackled via a two-dimensional packing heuristic which repetitively attempts to obtain feasible loadings for the transported products. These attempts are effectively coordinated with the use of an innovative memory mechanism. The behavior of the proposed packing heuristic is also controlled by a single parameter. Several additional memory structures are employed for drastically accelerating the overall solution framework.

The proposed algorithm is tested on 2L-CVRP benchmark instances, for two distinct configurations of the problem's loading constraints. It consistently produces high-quality results improving or matching most of the previously best known solution scores.

Acknowledgment

The authors are indebted to the anonymous referees for extensively reviewing their paper and offering constructive remarks and directions for the completion of their work.

References

Bortfeldt, A., 2012. A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. Computers & Operations Research 39, 2248–2257.
Crainic, T.G., Perboli, G., Tadei, R., 2008. Extreme point-based heuristics for three-dimensional bin packing. INFORMS Journal on Computing 20, 368–384.
Derigs, U., Kaiser, R., 2007. Applying the attribute based hill climber heuristic to the vehicle routing problem. European Journal of Operational Research 177, 719–732.
Doerner, K., Fuellerer, G., Hartl, R., Iori, M., 2007a. Ant colony optimization for the two-dimensional loading vehicle routing problem. Technical Report, POM, University of Vienna, Austria. <www.univie.ac.at/bwl/prod/research/VRPandBPP>.
Doerner, K., Fuellerer, G., Hartl, R., Gronalt, M., Iori, M., 2007b. 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.
Fredman, M., Tarjan, R., 1987. Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM 34, 596–615.
Fuellerer, G., Doerner, K., Hartl, R., Iori, M., 2009. Ant colony optimization for the two-dimensional loading vehicle routing problem. Computers & Operations Research 36, 655–673.
Fuellerer, G., Doerner, K., Hartl, R., 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.
Glover, F., Laguna, M., 1997. Tabu Search. Kluwer Academic Publishers, Boston.
Iori, M., Salazar-Gonzãlez, J.-J., Vigo, D., 2007. An exact approach for the vehicle routing problem with two-dimensional loading constraints. Transportation Science 41, 253–264.
Iori, M., Martello, S., 2010. Routing problems with loading constraints. TOP 18, 4–27.
Laporte, G., 2009. Fifty years of vehicle routing. Transportation Science 43, 408–416.
Leung, S.C.H., Zhou, X., Zhang, D., Zheng, J., 2011. Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem. Computers & Operations Research 38, 205–215.
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.
Malapert, A., Guéret, C., Jussien, N., Langevin, A., Rousseau, L.-M., 2008. Two-dimensional Pickup and Delivery Routing Problem with Loading Constraints. 1st Workshop on Bin Packing and Placement Constraints (BPPC'08), Paris.
Moura, A., Oliveira, J.F., 2009. An integrated approach to the vehicle routing and container loading problems. OR Spectrum 31, 775–800.
Ruan, Q., Zhang, Z., Miao, L., Shen, H., 2011. A hybrid approach for the vehicle routing problem with three-dimensional loading constraints. Computers & Operations Research, doi: j.cor.2011.11.013.
Strodl, J., Doerner, K., Tricoire, F., Hartl, R., 2010. On Index Structures in Hybrid Metaheuristics for Routing Problems with Hard Feasibility Checks: An Application to the 2-Dimensional Loading. In: Blesa, M., Blum, C., Raidl, G., Roli, A., Sampels, M. (Eds.), Hybrid Metaheuristics, Lecture Notes in Computer Science, vol. 6373. Springer, Berlin/Heidelberg, pp. 160–173.
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.

Tricoire, F., Doerner, K., Hartl, R., Iori, M., 2011. Heuristic and exact algorithms for the multi-pile vehicle routing problem. OR Spectrum 33, 931–959.
Whittley, I.M., Smith, G.D., 2004. The attribute based hill climber. Journal of Mathematical Modelling and Algorithms 3, 167–178.
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., Kiranoudis, C.T., 2010. A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. Computers & Operations Research 37, 2089–2105.
Zachariadis, E.E., Kiranoudis, C.T., 2012. An effective local search approach for the Vehicle Routing Problem with Backhauls. Expert Systems with Applications 39, 3174–3184.
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.