Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 195 (2009) 729–743
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

A Guided Tabu Search for the Vehicle Routing Problem with two-dimensional loading constraints

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

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

Received 12 January 2007; accepted 31 May 2007
Available online 21 November 2007

Abstract

We present a metaheuristic methodology for the Capacitated Vehicle Routing Problem with two-dimensional loading constraints (2L-CVRP). 2L-CVRP is a generalisation of the Capacitated Vehicle Routing Problem, in which customer demand is formed by a set of two-dimensional, rectangular, weighted items. The purpose of this problem is to produce the minimum cost routes, starting and terminating at a central depot, to satisfy the customer demand. Furthermore, the transported items must be feasibly packed into the loading surfaces of the vehicles. We propose a metaheuristic algorithm which incorporates the rationale of Tabu Search and Guided Local Search. The loading aspects of the problem are tackled using a collection of packing heuristics. To accelerate the search process, we reduce the neighbourhoods explored, and employ a memory structure to record the loading feasibility information. Extensive experiments were conducted to calibrate the algorithmic parameters. The effectiveness of the proposed metaheuristic algorithm was tested on benchmark instances and led to several new best solutions.

Keywords: Vehicle routing; Loading constraints; Tabu Search; Guided Local Search

1. Introduction

This paper addresses a variant of the standard version of the Vehicle Routing Problem (VRP) (Toth and Vigo, 2002), called the Capacitated Vehicle Routing Problem with two-dimensional loading constraints (2L-CVRP) (Iori et al., 2007; Iori, 2005; Gendreau et al., in press). It models the cases when the demand of customers consists of two-dimensional, rectangular and weighted items. Although 2L-CVRP is a practical problem, which often arises in the field of transportation logistics, only recently did researchers publish the first solution approaches for it.

The standard version of the CVRP (Rochat and Taillard, 1995; Prins, 2004; Tarantilis and Kiranoudis, 2002; Tarantilis, 2005; Mester and Bräysy, 2007) is an NP-hard optimisation problem which involves determining the optimal set of routes, for a homogenous fleet of vehicles, to service a set of customers with known demands. The designed routes must originate and terminate at the central depot, and totally satisfy customer demand. Each customer must be visited once by one vehicle only. The total demand of the customer set, covered by a route, must not exceed the maximum carrying load of the vehicles.

The problem examined in the present paper (2L-CVRP) is a generalisation of the CVRP, as it considers the demand of customers in the form of two-dimensional, rectangular and weighted items. The 2L-CVRP can be reduced to the CVRP by setting the dimensions of the items equal to zero and dealing only with their weight attribute. The aim of the 2L-CVRP is to generate a set of routes that respect the aforementioned CVRP constraints but also to guarantee the feasible loading/unloading of the items into/from the vehicles. We consider two versions of the 2L-CVRP: the *Unrestricted* one that only deals with the feasible loading

* Corresponding author. Fax: +30 2108816705.
E-mail address: tarantil@aueb.gr (C.D. Tarantilis).

0377-2217/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2007.05.058

of the items into the vehicles, and the *Sequential* one that considers both loading and unloading constraints, as it will be described in the next sections. Loading the two-dimensional items into the vehicles is closely related to the two-dimensional bin packing problem (2BPP). 2BPP is an NP-hard combinatorial optimisation problem which aims at packing a given set of rectangular items into the minimum number of identical rectangular bins (Baker et al., 1980; Chazelle, 1983; Martello and Vigo, 1998; Lodi et al., 1999; Burke et al., 2004; Fekete et al., 2007).

The 2L-CVRP is a particularly important problem. Its importance can be attributed to the fact that it is an interesting problem both from the theoretical and the practical points of view. Regarding the theoretical viewpoint, since 2L-CVRP is, in a sense, composed of two NP-hard optimisation problems (CVRP and 2BPP), it is also a challenging NP-hard problem of high complexity. As far as its practical importance is concerned, the 2L-CVRP has an obvious commercial value. A variety of real life applications in the distribution/collection management context involves the transportation of rectangular shaped items that cannot be stacked, due to their weight or fragility (household appliances, delicate pieces of furniture, etc.). Pallet distribution is another application example of 2L-CVRP.

To the best of our knowledge, only two algorithmic methodologies have been proposed for the 2L-CVRP. Iori et al. (2007) have developed an exact methodology which uses a branch-and-cut algorithm to deal with the routing characteristics of the problem and a branch-and-bound procedure to guarantee feasible loadings of the items into the vehicles. This exact solution methodology is applied to problem instances with no more than 30 customers and 90 items. Since, in practical conditions, the scale of the problem tends to be larger, Gendreau et al. (in press) focused their interest on metaheuristic approaches to solve larger 2L-CVRP instances. In particular, their algorithmic methodology employs Tabu Search for the routing aspects of the problem. Customers are relocated by means of the GENI heuristic (Gendreau et al., 1992). To guarantee the loading feasibility, they solve the two-dimensional strip packing problem (Martello et al., 2003). In their work, they provide 180 problem instances with diverse characteristics as far as the size of the customer set and the number of demanded items are concerned. Except for the CVRP with two-dimensional constraints, researchers have recently addressed routing problem extensions considering the transportation of three-dimensional items (Gendreau et al., 2006; Moura and Oliveira, 2007).

The purpose of this paper is to present an effective and efficient metaheuristic methodology designed for the 2L-CVRP called Guided Tabu Search (GTS). Regarding the routing aspects of the problem examined, our algorithm explores the solution space by employing a search strategy based on Tabu Search, guided by an objective function alteration mechanism. This guiding mechanism induces diversification, and helps the search process to cover vast areas of the solution space. To ensure the feasible loading of the items into the vehicles, we employ a bundle of packing heuristics that aim at producing diverse packing designs. To accelerate our methodology, the loading feasibility information obtained through the progress of the search is stored in a memory structure. The proposed solution approach was successfully tested on a 180 2L-CVRP benchmark data set introduced by Gendreau et al. (in press) and found several new best solutions.

The remainder of this paper is outlined as follows: in Section 2, a detailed description of the problem examined is provided. Section 3 presents the proposed algorithmic methodology, followed by the computational results in Section 4. Finally, Section 5 concludes the paper.

2. The problem

Following the description of Gendreau et al. (in press), the 2L-CVRP is defined as follows: let $G = (N, A)$ be an undirected graph, where N is the vertex set containing the central depot and n customers, and $A = \{(i,j): i,\ j \in N,\ i \neq j\}$ is the edge set. With each edge $(i,j) \in A$ is associated a cost c_{ij} that corresponds to the cost demanded for the transition from i to j. In the central depot, a fleet of vh homogenous vehicles is available. Every vehicle has a maximum carrying load equal to Q and a rectangular loading surface of length L and width W. The demand of each customer i consists of a set IT_i of m_i rectangular items of known weight. Each item I_{ik} in $IT_i (k = 1, 2, \ldots, m_i)$ has length l_{ik} and width w_{ik} (corresponding to height and width in the work of Gendreau et al. (in press)). Let a_i denote the total surface area of all the items in IT_i. The items must be placed on the loading surfaces without being rotated: their l- and w-edges must be parallel to the L- and W-edges of the vehicle surfaces, respectively. This constraint models the practical cases of automated, fixed orientation palette loading. The 2L-CVRP aims at determining the minimum cost set of routes that satisfy the following constraints: (a) the size of the generated route set does not exceed the number of available vehicles vh (at most one route per vehicle), (b) every route starts and ends at the central depot, (c) the demand of every customer is totally covered, (d) each customer is visited once, (e) the total weight of all items demanded by the set of customers covered by a route must not exceed the capacity of the vehicle Q and (f) there must be a non-overlapping loading of all items demanded by the set of customers covered by a route into the $L \times W$ loading surface of the vehicles.

As mentioned earlier, in this paper we study two versions of the 2L-CVRP: the *Unrestricted 2L-CVRP*, and the *Sequential 2L-CVRP* examined in the work of Iori et al. (2007). The *Unrestricted* version is modelled by the aforementioned six constraints (a–f), while for the *Sequential* version of the problem an additional constraint, namely *Sequence Constraint* is imposed: the loading of the items must ensure that whenever a customer i is visited, all items in the set IT_i can be unloaded by employing a sequence of straight movements (one per item) parallel to

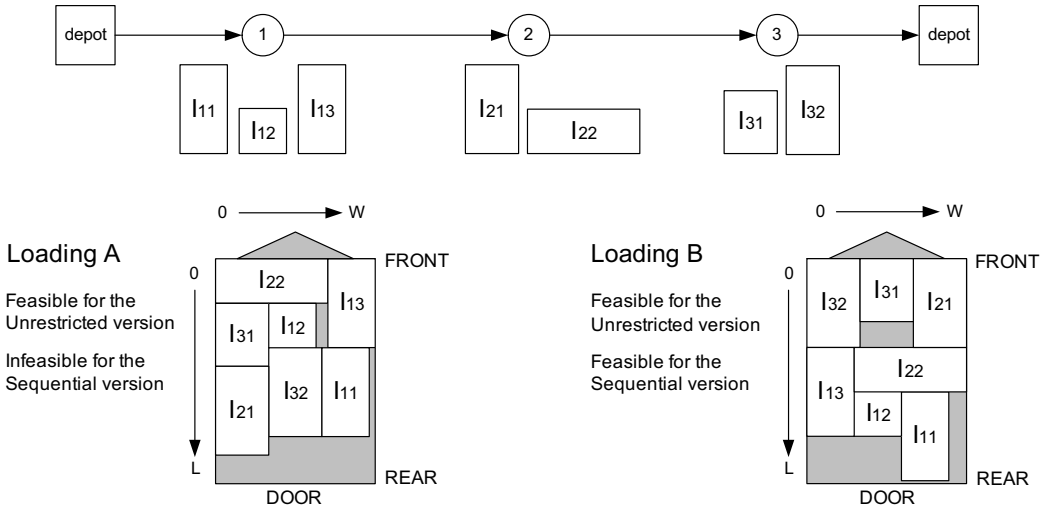

Fig. 1. The *Unrestricted* and *Sequential* loadings.

the length dimension of the vehicle surface. In other words, no item of customer j, visited after customer i, can be placed between items of customer i and the rear part (loading door) of the vehicle.

The sequence constraint arises in practice, when it is not feasible to move items inside the vehicle, due to their weight or fragility. In these cases, the unloading of every item must be accomplished without any repositioning of other items. Fig. 1 demonstrates the two versions of the 2L-CVRP. In particular, Fig. 1 demonstrates a route visiting three customers 1–3. Customer 1 demands items I_{11}, I_{12} and I_{13}, customer 2 demands items I_{21} and I_{22}, and customer 3 demands items I_{31} and I_{32}. Let $(0,0)$ correspond to the front left corner of the loading surface. The position of an item is expressed as the coordinate pair $(w_{\text{pos}}, l_{\text{pos}})$ where w_{pos} denotes the W-axis position and l_{pos} denotes the L-axis position of the item's left front corner. In Loading A, for example, item I_{22} is placed at $(0,0)$. Loading A is a feasible unrestricted packing but it violates the constraint posed for the *Sequential* version of the problem. To be precise, item I_{32} obstructs the unloading of items I_{12} and I_{22}, while item I_{31} obstructs the unloading of item I_{22}. On the other hand, Loading B is a feasible loading for both the *Unrestricted*, and the *Sequential* versions of the problem. Items I_{11}, I_{12} and I_{13} are the first to be unloaded, followed by I_{21} and I_{22}, and finally I_{31} and I_{32}.

3. The proposed algorithm

Due to the high complexity of both the VRP and the BPP variants, exact methodologies become inapplicable for solving large scale instances of the 2L-CVRP, that often arise in practice. Therefore, to solve such large scale problems, one's interest should be focused on metaheuristic methodologies that are able to produce near optimal solutions within reasonable computing time. Analytic surveys on the solution approaches designed for the VRP were published by Gendreau et al. (2002) and Cordeau et al. (2004). As mentioned in the first section, in terms of the routing aspects of the 2L-CVRP, our algorithmic framework explores the solution space by employing Tabu Search combined with a guiding mechanism that is based on the Guided Local Search principles (Voudouris and Tsang, 1996). This guiding strategy controls the objective function of the problem by penalising low-quality features present in the current solution. Regarding the loading constraints of the problem, we designed a bundle of packing heuristics that produce diverse packing structures in order to generate a feasible loading of the items onto the loading surface.

3.1. Packing heuristics bundle

To determine whether a route-sequence of customers-is feasible in terms of the loading constraints of the examined problem, we designed five packing heuristics. Given a customer sequence rt, and its corresponding set of items IS_{rt}, these heuristics try to generate diverse packing structures, in order to increase the probability of obtaining a feasible loading. If all of the packing heuristics fail to produce any feasible loading, the route examined is considered to violate the loading constraints of the problem. In particular, given a route, the packing procedure starts by generating two orderings $(\text{Ord}_{\text{Seq}}, \text{Ord}_{\text{Un}})$ of all items demanded by the set of customers covered by this route. Let the term *visit order* denote the position of a customer-and its corresponding items-within its route. For example, in Fig. 1, customer 3 is the third one to be visited by the vehicle; therefore the visit order of customer 3 and its demanded items I_{31} and I_{32} is equal to 3. The Ord_{Seq} ordering is mostly appropriate for the *Sequential* 2L-CVRP, and is produced by sorting all items transported by the route by decreasing visit order, breaking ties by decreasing surface area. The Ord_{Un} ordering is primarily suitable for the *Unrestricted* problem version and is generated by simply sorting the items by decreasing surface area. Using the first packing heuristic,

and starting from the first item ordering, items are selected successively to be inserted onto the loading surface. Let *posList* denote a list of available loading positions for the items. In the beginning, the only available loading position lies in the front left corner $(0,0)$ of the vehicle, so $posList = \{(0,0)\}$. Whenever an item is inserted, its loading position is erased from the *posList*, while at most four new loading positions are generated and added into the *posList*. In this way, the positions of holes that may be created between the placed items are stored into the *posList* and may be later filled by the subsequent items.

Fig. 2 schematically presents the mechanism of an item insertion: item E is selected to be inserted into the loading position (w_C, l_A). When E is inserted, the aforementioned position becomes unavailable and four new loading positions are created. The first one is created in the rear left corner of the inserted item $(w_C, l_A + l_E)$, the second one in the front right corner of the inserted item $(w_C + w_E, l_A)$, the third created loading position lies at the leftmost non-occupied point of the rear edge projection of the inserted item $(w_D, l_A + l_E)$, and the fourth available loading position is created at the minimum non-occupied l-axis point of the right edge projection of item $E(w_C + w_E, l_B)$. The list of available positions is updated as: $posList = (posList - \{(w_C, l_A)\}) \cup \{(w_C, l_A + l_E), \quad (w_C + w_E, l_A), \quad (w_D, l_A + l_E), (w_C + w_E, l_B)\}$. Note that any duplicate position entry is removed from *posList*.

The position for the placement of an item is selected from the list of available positions *posList* and must not lead to any loading constraint violation (overlapping or sequential constraint). As later explained, it is determined by the packing heuristic currently employed. If all items are packed onto the loading surface, the route is considered to be feasible in terms of the loading constraints of the problem. If, on the other hand, the insertion of an item into any available position leads to loading constraint violations, the method empties the loading surface, *posList* is set equal to $\{(0,0)\}$, and the next packing heuristic is employed from the beginning. If none of the five available packing heuristics manages to produce a feasible loading, the heuristic bundle is applied to the second ordering of the items. If again, no feasible loading is obtained, the examined route is considered to be infeasible regarding the loading constraints. Note, that although Ord_{Un} is primarily designed to deal with the *Unrestricted* problem version, it succeeds in producing feasible *Sequential* loading patterns in numerous cases when Ord_{Seq} fails. These cases usually involve loading few and voluminous items onto the vehicles.

As mentioned earlier, the loading position of an inserted item is determined by the packing heuristic currently in use. This position must be *feasible*, i.e., it must not lead to any overlaps (for both *Unrestricted* and *Sequential* problem versions), or sequence constraint violations (for the *Sequential* version). Each of the proposed five packing heuristics Heur_i $(i = 1\ldots5)$ employs a different criterion for selecting the loading position of an item:

Heur_1: *Bottom-Left Fill (W-axis)* (Chazelle, 1983)

From the *feasible* available loading positions of *posList*, the position selected is the one with the minimum W-axis coordinate, breaking ties by minimum L-axis coordinate. Using this heuristic, the packing tends to evolve in the form of strips parallel to the L-axis.

Heur_2: *Bottom-Left Fill (L-axis)* (Chazelle, 1983)

From the *feasible* available loading positions of *posList*, the position selected is the one with the minimum L-axis coordinate, breaking ties by minimum W-axis coordinate. Using this heuristic, the packing tends to evolve in the form of strips parallel to the W-axis.

Heur_3: *Max Touching Perimeter heuristic* (Lodi et al., 1999)

For each of the *feasible* available positions of *posList*, the total touching perimeter of the inserted item is calcu-

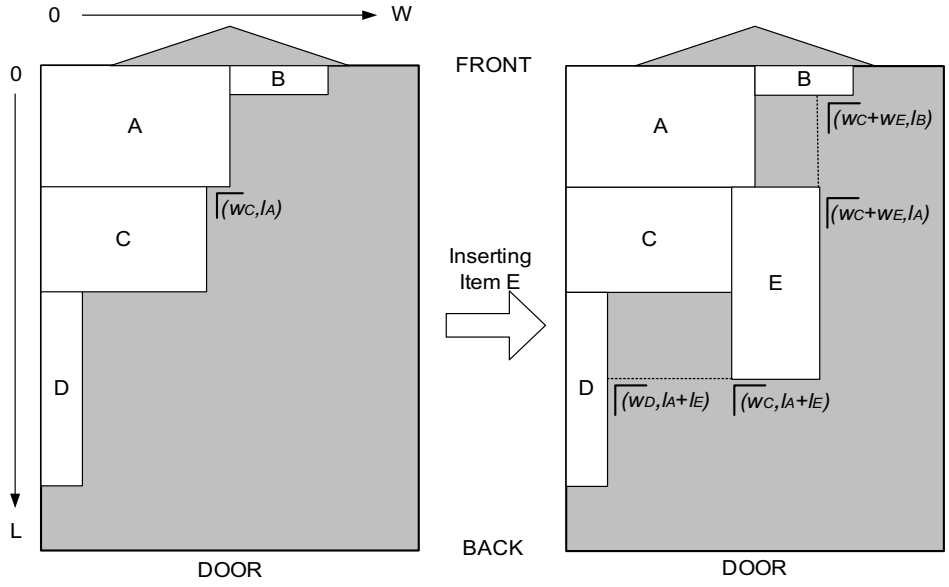

Fig. 2. Inserting an item into the loading surface.

lated. The total touching perimeter is evaluated as the sum of the common edges of the inserted item with the edges of the already inserted items, and the edges of the loading surface of the vehicle as seen in Fig. 3: the total touching perimeter of item C placed in position $(w_A, 0)$ is demonstrated by the bold dotted lines, and is equal to $l_C + w_C + (w_B - w_A)$. Term l_C corresponds to the common edges of items C and A, term w_C corresponds to the common edges of item C and the loading surface, and term $(w_B - w_A)$ corresponds to the common edges of items C and B. The item is placed into the loading position that maximises the value of touching perimeter. The *Max Touching Perimeter* heuristic tends to initially spread the items to the edges of the loading surface and later fill the inner parts of it.

Heur$_4$: *Max Touching Perimeter No Walls heuristic* (Lodi et al., 1999)

As in the case of the *Max Touching Perimeter* heuristic, for each of the *feasible* available positions of *posList*, the total touching perimeter of the inserted item is calculated. In this case, the total touching perimeter is evaluated as the sum of the common edges of the inserted item with the edges of the already inserted items. The common edges of the item and the loading surface are not taken into account. The evaluation of the touching perimeter is presented in Fig. 4: the total touching perimeter of item C placed in position $(w_A, 0)$ is demonstrated by the bold dotted lines and is equal to $l_C + (w_B - w_A)$. Term l_C corresponds to the common edges of items C and A and term $(w_B - w_A)$ corresponds to the common edges of items C and B. The item is inserted into the loading position that maximises the value of the touching perimeter. Following this rationale, the items initially tend to fill the inner part of the loading surface, and afterwards cover the edges of it.

Heur$_5$: *Min Area heuristic*

For each of the *feasible* available positions of *posList*, the area of its corresponding rectangular surface is calculated, as demonstrated in Fig. 5. The area of the rectangular surface determined by the loading position $(w_A, 0)$ is equal to $(W - w_A) \times l_A$, the area corresponding to loading

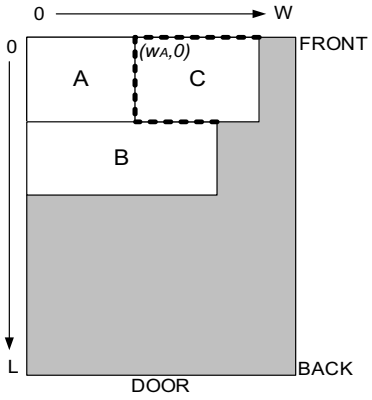

Fig. 3. Calculating the perimeter for the Max Touching Perimeter heuristic.

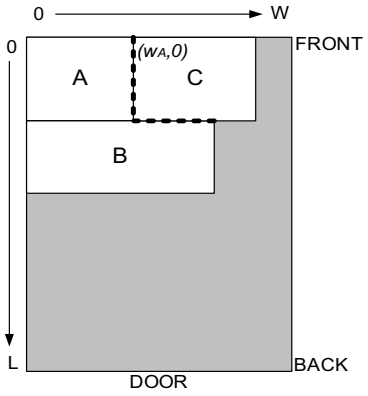

Fig. 4. Calculating the perimeter for the Max Touching Perimeter No Walls heuristic.

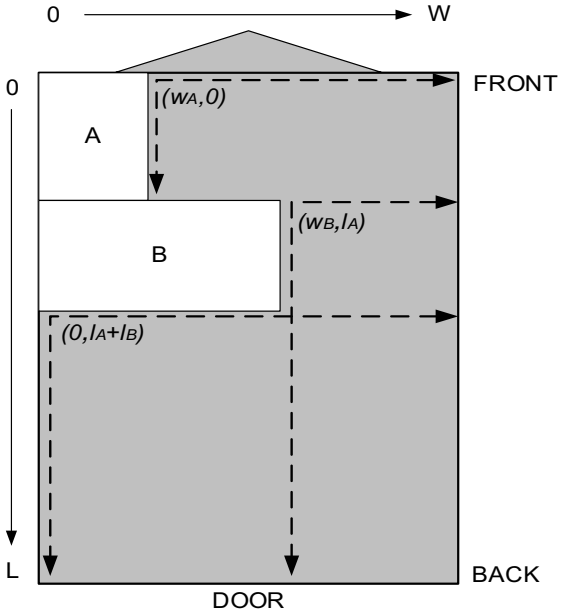

Fig. 5. Calculating the rectangular areas of the loading positions.

position (w_B, l_A) is equal to $(W - w_B) \times (L - l_A)$, and the area of the surface corresponding to position $(0, l_A + l_B)$ is equal to $W \times (L - l_A - l_B)$. The loading position selected is the one yielding the minimum surface area. This heuristic aims at achieving a high degree of utilisation of the available surfaces.

The proposed five heuristics are employed in the order presented. The simplest of the heuristics (*Bottom-Left Fill W-* and *L-*) are employed first. If they fail to produce a feasible packing, they are followed by the more complex and effective *Max Touching Perimeter*, *Max Touching Perimeter No Walls*, and *Min Area* heuristics, which require additional operations for calculating the touching perimeters, and the rectangular areas, respectively. Preliminary experiments indicated that applying the heuristics in ascending order of complexity, results into reduction of the overall computational time demanded by the packing heuristic

Table 1
Pseudocode for the collection of packing heuristics

```
bool is Feasible (Route rt, Version prVersion)
1  Orderings Ord1, Ord2, Heuristics Heur1, Heur2, Heur3, Heur4,Heur5
2  if (prVersion = Unrestricted)
3     Ord1 = Ord_Un (IS_rt), Ord2 = Ord_Seq (IS_rt)
4  else
5     Ord1 = Ord_Seq (IS_rt), Ord2 = Ord_Un (IS_rt)
6  end if
7  int ordInd = 1, heurInd = 1
8  if (ordInd < 3)
9        Empty Vehicle, posList = {(0,0)}
10       for each Item it of Ord_ordInd
11          Determine Position pos ∈ posList for it, according to
            Heur_heurInd
12          if (no feasible pos exists)
13             heurInd = heurInd + 1
14             if (heurInd = 6)
15                heurInd = 1, ordInd = ordInd + 1
16             end if
17             go to 9
18          end if
19          Place it in pos
20          Remove pos from posList, add new loading positions in
            posList
21       end for
22       return true
23       end if
24       return false
```

bundle. Table 1 provides a pseudocode for the proposed collection of packing heuristics.

3.2. Constructing the initial solution

The proposed methodology is based on the Best Fit Decreasing heuristic for the one-dimensional bin packing problem. It is initiated by constructing an initial feasible solution that is going to act as the starting point of the improvement metaheuristic described in the following subsections. To construct this initial solution, the following procedure is employed: all customers are sorted in decreasing value of a_i (total area of the surface of their items) and vh new routes (one for each available vehicle) are generated. Customers are selected successively to be inserted into the routes. The feasibility of inserting a customer i, into every position of every route is tested. If customer i can be feasibly inserted into a set of routes S_{feas}, for every route $p \in S_{\text{feas}}$ the quantity $freeArea_p - a_i$ is calculated, where $freeArea_p$ denotes the total non-occupied loading surface of the vehicle executing route p. The customer is assigned to the route that minimises the value of $freeArea_p - a_i$, and inserted into the feasible route point that leads to the minimum cost increase.

Due to the fact that the feasibility of a solution is mainly determined by the loading constraints of the problem, the proposed construction algorithm primarily aims at maximising the loading surface utilisation and secondarily at minimising the actual objective function of the problem. This strategy managed to successfully construct feasible initial solutions for all of the *Unrestricted* and *Sequential* 2L-CVRP problem instances.

3.3. Definition of neighbourhood structures

The proposed metaheuristic methodology explores the search space by performing moves for transiting from the current to the subsequent solution. We employ three move types, each of them defining a neighbourhood structure NS_i ($i = 1, 2, 3$). The size of these neighbourhoods is reduced by our accelerating strategy introduced in Section 3.5. In the following paragraphs a description of the three move types is provided.

3.3.1. Move Type 1 (NS_1) – customer relocation

This move type (Waters, 1987) relocates a customer from the route currently assigned to another route (Fig. 6). The move can be employed for every route pair and for every possible insertion position. When the move is performed within a single route, the customer changes its position within this route.

3.3.2. Move Type 2 (NS_2) – route exchange

This move type (Waters, 1987) exchanges the routes that cover a pair of customers (Fig. 7). It is employed for every route pair and for every customer pair visited by the routes involved in the move. When the move is performed within a single route, the customers swap their positions within this route.

3.3.3. Move Type 3 (NS_3) – route interchanging

The route interchanging move (2-opt) (Croes, 1958; Lin, 1965) can be employed within any single route and between

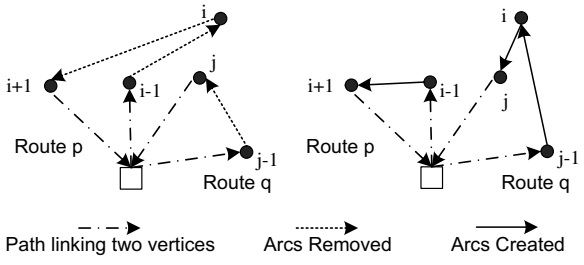

Fig. 6. Customer relocation move.

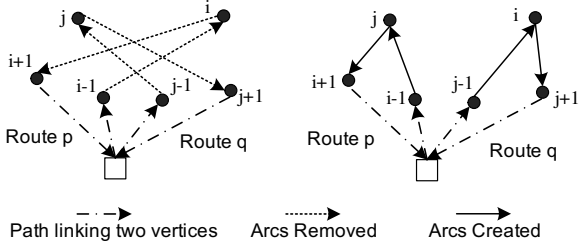

Fig. 7. Route exchange move.

any route pair. When employed within a route, two arcs of the route are removed, two new arcs are generated, and the section between the created arcs is reversed, as demonstrated in Fig. 8.

When the move is performed between a route pair, both routes involved are divided into their initial and terminating sections, by deleting a pair of arcs. The initial part of the first route is connected to the terminating part of the second one and vice versa. To connect the divided sections, two arcs are generated. These arcs link the last customers of the initial parts to the first customers of the terminating parts, as seen in Fig. 9.

The cost for implementing any move *mv*, involving the deletion of the edge set *ED* and the generation of the edge set *EG*, is evaluated as

$$Cost(mv) = \sum_{(i,j)\in EG} c_{ij} - \sum_{(i,j)\in ED} c_{ij}. \quad (1)$$

3.4. Guided Tabu Search

The initial solution produced by the construction algorithm is improved by the proposed metaheuristic methodology called Guided Tabu Search (GTS). This metaheuristic approach is based on the well-known Tabu Search (TS) methodology (Glover, 1986; Hertz, 2006) that has proven to be very effective for the VRP and its variants (Archetti et al., 2006; Brandão, 2006; El Fallahi et al., 2008; Gendreau et al., 1994, 1999). The proposed Tabu Search is controlled by a guiding mechanism that incorporates the Guided Local Search (Voudouris and Tsang, 1996) rationale and periodically modifies the objective function of the problem. The purpose of this guiding mechanism is twofold: it induces diversification, helping the search process to cover wider domains of the solution space, and aims at eliminating undesirable features from the final solution.

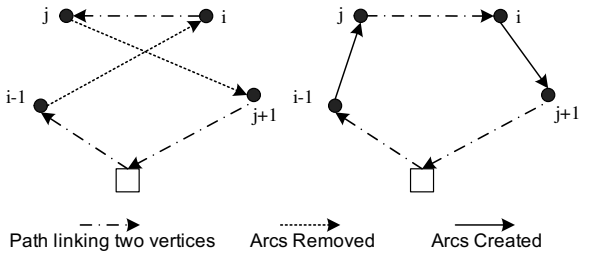

Fig. 8. Route interchange move (within a single route).

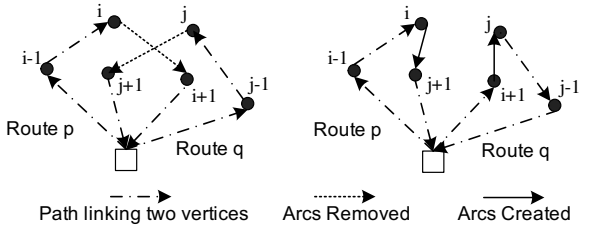

Fig. 9. Route interchange move (between a route pair).

The GTS algorithm starts off with the solution generated by the construction method of Section 3.2. At each iteration, one of the three neighbourhood structures (NS_1–NS_3) is stochastically selected. All structures share the same probability of being selected. The cost of performing every move defined by the selected neighbourhood structure is evaluated, according to (1). These moves must lead to feasible solutions regarding the loading and the capacity constraints of the problem, and must not be Tabu, unless they improve the best solution ever obtained through the search (*aspiration criterion*). The loading feasibility of the tentative solutions is checked using the packing heuristic bundle described in Section 3.1. The most economical of the moves examined is performed. This deterministic criterion may cause cycling phenomena to occur. To avoid these situations, whenever a move is performed, its reversal is considered Tabu for *tabuTenure* iterations. TS procedure is terminated after completion of *tabuIter* non-improving iterations.

Each time *guidFreq* iterations of TS have been completed, the proposed guiding mechanism is employed. This guiding mechanism embodies the rationale of the Guided Local Search into the Tabu Search process. Its principle is to locate and remove low quality features present in the candidate solution by augmenting the objective function through the use of penalisation terms. In the proposed methodology-as well as in the majority of routing problem approaches-long (expensive) edges of the solution are considered as undesirable features of a candidate solution. In particular, the arc maximising the following utility function (2) is selected to be penalised

$$U(i,j) = \frac{\frac{c_{ij}}{\mathrm{avg}_{ij}}}{1+p_{ij}}, \quad (2)$$

where avg_{ij} is the average cost of all edges starting from vertices i and j in the edge set A, and p_{ij} is the number of times that arc (i,j) has been selected to be penalised. By using the term avg_{ij}, our utility function (2) improves upon the most commonly used arc selection strategy that simply penalises the longest arc present in the solution (i.e. in the work of Mester and Bräysy (2007)). Term avg_{ij} reflects the positions of customers i and j relatively to their neighbouring customer positions, leading to a more balanced arc selection behaviour. Let *arcPen* denote the arc selected to be penalised. The cost c_{arcPen} of the penalised arc is doubled for the next *guidFreq*/2 iterations of the TS body. Using this aggressive penalisation policy, we force the algorithm to eliminate the *arcPen* from the candidate solution for at least a number of iterations equal to the penalisation horizon (*guidFreq*/2). If during these *guidFreq*/2 iterations, a tentative solution containing *arcPen* improves the best solution ever encountered, the penalisation term is

overridden. After the completion of *guidFreq*/2 iterations the cost of *arcPen* is restored to its original value.

To demonstrate both the effectiveness, and the diversification effect of the guiding mechanism employed in the GTS framework we executed the proposed methodology for the benchmark instance 18 – Class 2 (presented in Section 4.2) using and not using the guiding mechanism. Fig. 10 illustrates the indicative progress curves obtained. The grey line corresponds to the search process with the guiding mechanism, while the black line demonstrates the search process without the guiding mechanism. The guiding mechanism reduced the best solution cost by 4.26% (1068.82 with and 1116.40 without the guiding mechanism). Furthermore, the search was diversified, as the standard deviation of the candidate solutions with and without the use of the guiding mechanism was 44.57 and 40.53, respectively. Except for the proposed guiding mechanism which penalises a single arc of the solution, each time *guidFreq* GTS iterations have been performed, we also tested strategies involving the penalisation of greater arc sets. These strategies did not manage to achieve high quality results, as they did not let the search intensify into promising regions of the solution space. Table 2 provides a pseudocode for the proposed GTS metaheuristic methodology.

Table 2
GTS methodology pseudocode

Solution GTS (**Solution** *initSol*)

```
1  Solution s = initial, s* = initial, s'
2  int nonImp = 0, tabuIter = 7000, bestCost = Cost (initial)
3  int guidFreq = 20, guidCount = 0
4  Cost(s)// Returns the objective function value of solution s
5  Neighborhood NS
6  while (nonImp < tabuIter)
7    stochastically select NS from {NS1, NS2, NS3}, guidCount ++
8    if (guidCount = guidFreq)
9      select arc arcPen for penalisation
10       set c_arcPen = 2 c_arcPen, for (guidFreq/2) iterations,
         guidCount = 0
11   end if
12   Select a feasible solution s' ∈ NS(s) which minimises the total
     cost, and is not Tabu, or is Tabu and Cost(s') < bestCost
13   Implement move mv towards solution s', s = s'
14   Declare the reversal of mv Tabu for tabuTenure iterations
15   nonImp = nonImp + 1
16   if (Cost(s) < bestCost)
17     s* = s, bestCost = Cost(s)
18     nonImp = 0
19   end if
20 end while
21 return s*
```

3.5. Accelerating the search process

To accelerate the search process, we propose two strategies for limiting the computational effort required by the GTS algorithm. The first strategy reduces the size of the neighbourhoods explored by the search, at each GTS iteration. For every vertex i, the quantity avg_i is evaluated, where avg_i denotes the average cost of all arcs beginning at vertex i. With each vertex i is associated a set of neighbouring vertices NV_i, formed by all vertices j such that: $c_{ij} \leqslant \text{avg}_i$. When a neighbourhood is explored, only the moves leading to the connection of vertices i and j, such that $i \in NV_j$ and $j \in NV_i$, are taken into consideration. The aforementioned policy of reducing the neighbourhood size is overridden in the case of moves connecting the depot to any other vertex.

The second accelerating strategy aims at eliminating any unnecessary calls to the packing heuristic bundle by keeping track of the loading feasibility of the routes already examined. The loading constraints posed by the 2L-CVRP problem are handled by the application of the five packing heuristic algorithms, which drastically contribute to the overall computational effort required by the proposed methodology. As the GTS method moves from solution

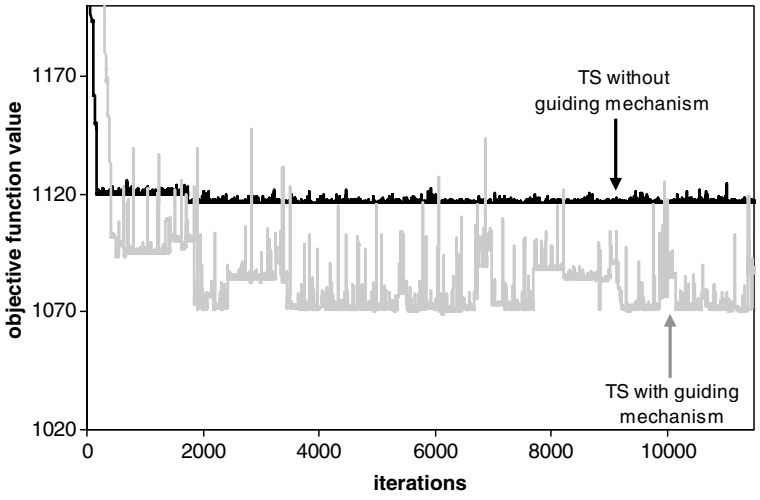

Fig. 10. The role of the guiding mechanism.

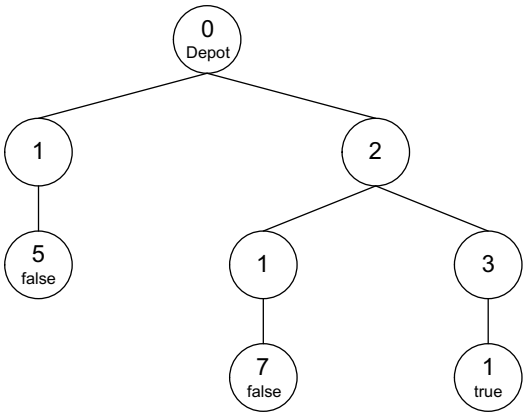

Fig. 11. Loading feasibility memory structure.

to solution, it needs to examine the feasibility of tentative routes, in terms of the loading constraints posed by the 2L-CVRP. Since the search has a local character, it may repeatedly encounter routes that have already been visited and checked for their loading feasibility by the application of the packing heuristic bundle of Section 3.1. For these routes, re-executing the packing heuristics is unnecessary; therefore, to avoid such duplicate and unnecessary calls to the packing heuristics procedure, whenever the loading feasibility of a route is checked for the first time, the corresponding result is recorded in a memory structure, so as to be available for future stages of the GTS search. To reduce the time needed for retrieving the stored feasibility information, this memory structure was designed in the form of a sorted tree, as demonstrated in Fig. 11. In particular Fig. 11 illustrates the case, in which the packing heuristics managed to generate a feasible loading for route 0-2-3-1-0, while routes 0-2-1-7-0 and 0-1-5-0 were determined to be infeasible regarding the loading constraints. As the search process evolves, if any of these three routes is revisited, its loading feasibility is going to be determined by accessing the memory structure, rather than reapplying the packing heuristic bundle. Using the loading feasibility memory resulted in a remarkable decrease of the demanded GTS computational time, ranging from 54% (for small scale 2L-CVRP instances) up to 33% (for larger instances). Note that, as the search process evolves, the physical memory required for storing the loading information may exceed the available memory limits (especially in the cases of larger 2L-CVRP). Therefore, the tree structure may need to be periodically destructed (and rebuilt from the beginning), after the completion of a number of GTS iterations that depends on the amount of physical memory available on the computer system in use. For our computer system with 1 GB RAM, setting the destruction period equal to 2000 GTS iterations did not lead to any memory overflows, even for the largest instances involving 255 customers.

4. Computational results

The proposed algorithmic framework was coded in Visual C#, executed on a Pentium IV 2.4 GHz with 1 GB of RAM under Windows XP. It was tested on 360 2L-CVRP test problems in total (180 for the *Sequential* and 180 for the *Unrestricted* version of the problem).

4.1. Benchmark instances

To test the proposed algorithm's performance and robustness we applied it to 180 2L-CVRP benchmark problems introduced by Iori et al. (2007) and Gendreau et al. (in press). These instances were derived from 36 CVRP instances, described by Toth and Vigo (2002), by expressing the customer demand as a set of two-dimensional, weighted and rectangular items. To generate the aforementioned item sets, five classes of the item demand characteristics are introduced (Iori et al., 2007; Iori, 2004):

- **Class 1**: with each customer is associated a single item of width and length equal to nil. The problems of Class 1 are in fact pure CVRP instances, as every customer sequence is feasible in terms of the loading constraints of the problem examined. They are used to test the algorithmic effectiveness in terms of the routing aspects of the problem examined.
- **Classes 2–5**: with each customer i, a set of m_i items is generated. m_i is uniformly distributed within a given range. Each item is classified into one of the three shape categories, namely *vertical*, *homogeneous* and *horizontal*, with equal probability. The dimensions (width and length) of an item are uniformly distributed into the ranges determined by this item's shape category. The m_i and the dimension ranges are provided in Table 3.

The length L and the width W of the loading surface are equal to 40 and 20, respectively. The costs of edges are

Table 3
Item set Classes 2–5

Class	m_i	Vertical		Homogeneous		Horizontal	
		Length	Width	Length	Width	Length	Width
2	[1, 2]	[0.4L, 0.9L]	[0.1W, 0.2W]	[0.2L, 0.5L]	[0.2W, 0.5W]	[0.1L, 0.2L]	[0.4W, 0.9W]
3	[1, 3]	[0.3L, 0.8L]	[0.1W, 0.2W]	[0.2L, 0.4L]	[0.2W, 0.4W]	[0.1L, 0.2L]	[0.3W, 0.8W]
4	[1, 4]	[0.2L, 0.7L]	[0.1W, 0.2W]	[0.1L, 0.4L]	[0.1W, 0.4W]	[0.1L, 0.2L]	[0.2W, 0.7W]
5	[1, 5]	[0.1L, 0.6L]	[0.1W, 0.2W]	[0.1L, 0.3L]	[0.1W, 0.3W]	[0.1L, 0.2L]	[0.1W, 0.6W]

Table 4
Characteristics of the 2L-CVRP benchmark instances

Inst	cus	Class 2			Class 3			Class 4			Class 5		
		it	*vh*	*r*%	*it*	*vh*	*r*%	*it*	*vh*	*r*%	*it*	*vh*	*r*%
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

cus: number of customers.
it: number of items.
vh: number of available vehicles.
r%: total item area/total loading area of the fleet available (%).

evaluated as the Euclidean distances between vertex pairs. Table 4 summarises the characteristics of the 36 2L-CVRP instances for Classes 2–5. For each of the 36 CVRP problems, five instances were created according to the five classes presented above (one instance per class), resulting in the 180 2L-CVRP instances. To guarantee the feasibility of the benchmark problems, the size of the available fleet of vehicles *vh* was determined by means of a packing heuristic. The GTS algorithm was applied to all 180 instances for both the *Unrestricted*, and the *Sequential* versions of the examined problem.

4.2. Parameter tuning

The proposed algorithmic framework, described in Section 3, contains three parameters, the setting of which was determined through an extensive calibration procedure involving both the *Unrestricted* and the *Sequential* versions of the examined problem. To test the robustness of the GTS algorithm, we applied it for solving 30 benchmark instances, for each problem version. In particular, to cover cases with diverse customer set sizes and item characteristics, we used 30 test problems (instances 6, 12, 18, 24, 30 and 36, Classes 1–5) to conduct the sensitivity analysis experiments. As there is no obvious correlation between the calibrated parameters, we valued each of them separately, within relative wide value ranges, and recorded the performance of the GTS methodology. The calibration procedure conducted is summarised in Table 5.

Parameter *tabuTenure* expresses the horizon of the memory employed by the TS procedure. To avoid cycling phenomena, the reversal of a performed move is declared

Table 5
Sensitivity analysis summary

Parameter	Description	Range	Value
tabuTenure	The number of iterations, for which the reversal of a performed move remains in the Tabu list	9–27	18
guidFreq	The frequency of employing the guiding mechanism, expressed in TS iterations	5–25	20
tabuIter	Maximum number of GTS non-improving iterations	5000–15,000	7000

Range: The value range used for the sensitivity analysis.
Value: The standard parameter setting.

Table 6
Calibration experiment results for the *tabuTenure* and *guidFreq* parameters

	tabuTenure						
	9	15	21	27	min	max	%Dev
Unrestr	1104.40	1101.28	1099.97	1104.31	1099.97	1104.40	0.40
Seq	1138.77	1136.18	1134.65	1141.07	1134.65	1141.07	0.57

	guidFreq							
	5	10	15	20	25			
Unrestr	1121.45	1109.67	1102.15	1097.85	1104.94	1097.85	1121.45	2.15
Seq	1157.06	1147.81	1141.63	1134.09	1141.11	1134.09	1157.06	2.03

Tabu for a certain number of iterations equal to *tabuTenure*. We varied the *tabuTenure* parameter from 3 to 9 TS iterations per move type. Since the GTS methodology explores three neighbourhood structures, the parameter examined was valued within [9, 27] with increments of 6. The algorithm was applied 200 times, for all 30 test problems, and for both problem versions. The average results obtained are reported in Table 6. The GTS methodology proved to be relatively insensitive to the *tabuTenure* value, as the percent deviation between the best and worst average scores achieved were 0.40% and 0.57%, for the *Unrestricted* and *Sequential* versions, respectively. The value of 21 yielded the highest quality solutions for five out of the six examined instances (6, 12, 24, 30 and 36), for the *Unrestricted* version. For the *Sequential* problem version, although the value of 21 reached the minimum average cost, 15 achieved the lowest solution costs for three (12, 18 and 36) out of the six examined instances, followed by 21 that minimised the average solution scores for instances 6 and 30. The percent gap between the average scores obtained by fixing *tabuTenure* at 15 and 21 were marginal (*Unrestricted*: 0.12% and *Sequential*: 0.14%). Therefore, to achieve a robust setting for both problem versions we set $tabuTenure = 18$, corresponding to 6 GTS iterations per move type.

Parameter *guidFreq* controls the frequency of applying the guiding mechanism that modifies the objective function of the problem. The smaller the value of *guidFreq*, the more diversified is the conducted search. To achieve a satisfactory interplay between the intensification and diversification character of the GTS search process, we performed the following calibration procedure for parameter *guidFreq*: we applied the GTS algorithm 200 times, to solve all 30 aforementioned test problems, for both problem versions, taking *guidFreq* values from {5, 10, 15, 20, 25}. The results are summarised in Table 6. The percent deviation between the best and worst average scores achieved were equal to 2.15% and 2.03%, for the *Unrestricted* and the *Sequential* versions, respectively, indicating that the GTS methodology is rather sensitive to the *guidFreq* parameter. In particular, setting the *guidFreq* to 20 resulted in obtaining the highest quality solutions for nine out of the twelve instances examined (for both *Unrestricted* and *Sequential* problem versions). Therefore, *guidFreq* was set equal to 20.

Regarding the termination condition of the GTS algorithm, we conducted tests taking the values of the *tabuIter* parameter from {5000, 7000, 10,000, 15,000}. For small instances, involving up to 100 customers, rarely did the search process reach an improving solution after the completion of 5000 non-improving iterations. For larger scale problems (instances 28–36), a significant decrease of the solution cost was occasionally observed after 10000 and 15,000 non-improving iterations, at the expense of excessive computational time. To bind the computational time required by the GTS methodology into reasonable limits, we fixed the value of *tabuIter* to 7000, for which a satisfactory balance between the solution quality and the computational effort is achieved.

4.3. Results on benchmark instances

To empirically test the effectiveness of the GTS algorithm, we applied it to the 180 benchmark problems of Iori et al. (2007) and Gendreau et al. (in press), for both the *Unrestricted*, and the *Sequential* problem versions. Since researchers have only recently addressed the 2L-CVRP,

Table 7
Computational results on the pure CVRP instances of Class 1

Inst	Class 1					
	s_{GTS}	s_{TS_G}	t_{GTS}	t_{TS_G}	tot_{GTS}	%gap
1	278.73	278.73	2.9	2.0	5.2	0.00
2	334.96	334.96	1.4	0.0	2.4	0.00
3	*358.40*	359.77	3.8	3.5	6.8	0.38
4	430.88	430.88	1.0	0.1	1.7	0.00
5	375.28	375.28	1.3	1.4	2.1	0.00
6	495.85	495.85	1.9	0.3	3.4	0.00
7	568.56	568.56	0.8	0.5	1.3	0.00
8	568.56	568.56	0.4	0.5	0.6	0.00
9	607.65	607.65	1.2	0.4	1.8	0.00
10	*535.80*	538.79	5.9	6.1	9.1	0.55
11	505.01	505.01	3.8	2.5	6.3	0.00
12	*610.00*	610.57	6.3	28.5	9.6	0.09
13	2006.34	2006.34	5.8	29.9	10.0	0.00
14	837.67	837.67	17.1	22.2	26.4	0.00
15	837.67	837.67	7.9	1.7	12.7	0.00
16	698.61	698.61	13.0	2.7	22.3	0.00
17	863.27	*862.62*	32.9	59.0	58.6	−0.08
18	730.85	*723.54*	47.1	81.9	77.0	−1.01
19	524.61	524.61	100.2	128.8	169.1	0.00
20	244.54	*241.97*	198.3	253.6	339.3	−1.06
21	*687.60*	688.18	221.5	325.0	359.1	0.08
22	740.66	740.66	662.9	2070.7	1160.9	0.00
23	*839.07*	860.47	1531.4	2210.1	2544.1	2.49
24	*1035.33*	1048.91	1012.7	866.9	1680.7	1.29
25	*829.45*	830.26	953.8	2371.0	1715.8	0.10
26	819.56	819.56	1031.7	3597.6	1743.1	0.00
27	*1097.63*	1099.95	871.2	355.9	1556.9	0.21
28	*1042.12*	1078.27	781.4	985.2	1383.4	3.35
29	1188.15	*1179.01*	1641.9	3080.0	2790.4	-0.78
30	*1037.05*	1061.55	873.3	1834.4	1393.1	2.31
31	*1421.20*	1464.04	631.4	288.8	1089.8	2.93
32	*1328.68*	1352.61	905.5	1780.8	1528.4	1.77
33	*1328.19*	1361.51	1708.6	2531.7	2628.1	2.45
34	*719.91*	858.94	834.1	1941.9	1495.3	16.19
35	*877.04*	992.86	907.2	766.7	1549.0	11.67
36	*594.10*	678.87	1492.6	1530.9	2299.3	12.49
avg						*1.54*

GTS: The proposed algorithmic methodology, TS_G: the algorithm of Gendreau et al. (in press).
s: The value of the best solution obtained, %gap: the percent improvement over the TS_G scores.
t: CPU time elapsed when the best solution was obtained (second).
tot: Total time required by the algorithm (second).
Italic characters correspond to higher quality solutions.
GTS: Visual C#, Pentium IV 2.4 GHz, 1 GB RAM, TS_G: C, Pentium IV 1.7 GHz computer.

Table 8
Computational results for the *Unrestricted* 2L-CVRP (Classes 2–5)

Inst	Classes 2–5					
	s_{GTS}	s_{TS_G}	t_{GTS}	t_{TS_G}	tot_{GTS}	%gap
1	295.74	*291.60*	2.2	4.2	3.5	−1.42
2	341.89	*341.02*	1.3	0.1	2.0	−0.26
3	384.49	*377.35*	0.7	1.6	1.2	−1.89
4	441.45	*437.45*	2.2	0.5	3.8	−0.91
5	382.22	*380.20*	4.7	5.0	7.5	−0.53
6	*499.47*	501.02	4.4	7.2	7.0	0.31
7	703.49	*700.34*	4.5	6.3	6.8	−0.45
8	705.59	*694.99*	6.4	11.2	10.2	−1.53
9	*615.65*	619.69	5.1	3.6	7.7	0.65
10	713.00	*700.39*	9.5	36.0	16.7	−1.80
11	740.40	*739.04*	18.1	55.7	31.4	−0.18
12	*616.83*	620.62	61.9	49.0	93.7	0.61
13	2599.40	*2598.20*	44.4	57.5	68.5	−0.05
14	*1036.77*	1047.72	167.4	375.8	299.1	1.05
15	*1197.83*	1201.38	86.1	156.7	138.3	0.30
16	702.29	*702.03*	78.3	20.5	132.4	−0.04
17	*864.26*	866.37	26.4	64.9	45.4	0.24
18	*1076.81*	1085.84	250.7	589.3	397.7	0.83
19	776.91	*772.25*	376.5	633.7	570.7	−0.60
20	*551.71*	564.67	518.7	954.5	839.7	2.30
21	*1050.43*	1066.21	129.0	460.1	203.8	1.48
22	*1076.11*	1087.46	941.1	1191.2	1584.6	1.04
23	*1091.18*	1104.72	1000.8	2032.4	1750.3	1.23
24	*1149.12*	1187.62	553.5	1454.1	842.4	3.24
25	*1418.87*	1436.09	635.9	1205.8	965.4	1.20
26	*1395.63*	1404.49	875.3	1173.9	1403.9	0.63
27	*1396.60*	1450.18	492.5	521.3	875.6	3.69
28	*2737.01*	2738.31	1079.1	2051.2	1836.4	0.05
29	*2315.20*	2474.33	1059.0	1406.5	1653.4	6.43
30	*1889.84*	1948.72	1711.2	1185.4	2852.4	3.02
31	*2413.45*	2506.99	2500.7	2375.8	4100.7	3.73
32	*2351.69*	2486.43	2240.1	1664.8	3597.8	5.42
33	*2455.79*	2504.00	2074.1	1843.2	3431.2	1.93
34	*1233.46*	1466.06	2549.7	1359.1	3843.3	15.87
35	*1498.78*	1765.30	2964.5	2061.7	4183.9	15.10
36	*1801.41*	1909.88	2680.3	2265.8	4004.3	5.68
avg						*1.84*

GTS: The proposed algorithmic methodology, TS_G: the algorithm of Gendreau et al. (in press).
s: The average value of the best solutions for instances of Classes 2–5.
%gap: The percent improvement over the TS_G scores.
t: CPU time elapsed when the best solution was obtained (second).
tot: Total time required by the algorithm (second).
Italic characters correspond to higher quality solutions.
GTS: Visual C#, Pentium IV 2.4 GHz, 1 GB RAM, TS_G: C, Pentium IV 1.7 GHz computer.

the aforementioned data set is the only one available in the literature.

The results obtained by the GTS algorithm are summarised and compared with those obtained by the Tabu Search of Gendreau et al. (in press) (denoted by TS_G) in Tables 7–9. Table 7 provides the results obtained on the pure CVRP instances (Class 1), while Tables 8 and 9 summarise the average results obtained on the benchmark instances of Classes 2–5, for the *Unrestricted* and the *Sequential* versions, respectively. For the pure CVRP instances the proposed methodology improves the solution values achieved by the TS_G method by 1.54%, on average. As seen in Tables 8 and 9, the GTS metaheuristic produces solutions with higher average quality compared to those obtained by TS_G. In particular, for the *Unrestricted* version of the problem, the GTS methodology achieved an average 1.84% improvement of the best solutions obtained for instances of Classes 2–5. For the *Sequential* version of the problem, the GTS algorithm achieved an average improvement of 1.62%. Regarding the computational effort demanded, the proposed algorithm obtained the final solutions within satisfactory time periods. The results obtained for all test problems are reported in Tables A1 and A2 of the Appendix.

Table 9
Computational results for the *Sequential* 2L-CVRP (Classes 2–5)

Inst	Classes 2–5					
	s_{GTS}	s_{TS_G}	t_{GTS}	t_{TS_G}	tot_{GTS}	%gap
1	304.22	*299.09*	2.9	2.6	4.4	−1.72
2	346.71	*345.23*	1.6	0.4	2.6	−0.43
3	393.35	*385.30*	2.3	3.8	4.0	−2.09
4	444.62	*443.42*	2.7	1.4	4.4	−0.27
5	396.36	*384.06*	6.7	4.1	11.4	−3.20
6	505.04	*502.78*	3.1	5.1	5.0	−0.45
7	723.83	*721.90*	18.1	15.5	31.9	−0.27
8	*715.72*	722.73	21.7	32.8	33.0	0.97
9	*622.20*	624.06	10.4	10.6	17.3	0.30
10	727.86	*714.90*	27.9	43.5	49.4	−1.81
11	*768.15*	773.45	54.0	99.0	94.8	0.69
12	*628.62*	631.85	81.4	58.8	131.1	0.51
13	*2679.34*	2687.03	49.2	49.0	87.8	0.29
14	*1092.78*	1101.49	104.1	146.0	173.2	0.79
15	*1234.21*	1240.89	41.3	165.4	74.2	0.54
16	707.56	*704.85*	38.5	28.0	58.1	−0.38
17	*865.20*	866.50	54.2	88.9	91.5	0.15
18	*1102.25*	1116.17	255.7	566.5	390.7	1.25
19	*800.94*	802.48	340.9	365.2	541.2	0.19
20	*576.58*	581.81	501.4	808.9	848.9	0.90
21	*1106.33*	1110.19	893.7	1702.2	1539.9	0.35
22	*1128.61*	1130.33	1314.7	1573.8	2208.3	0.15
23	*1151.24*	1186.36	766.9	675.8	1190.6	2.96
24	*1206.62*	1248.43	1376.5	2642.5	2200.3	3.35
25	*1479.03*	1480.63	2247.4	2336.5	3807.6	0.11
26	1475.96	*1471.74*	1136.5	1554.6	1984.8	−0.29
27	*1463.34*	1524.22	805.5	1308.2	1226.1	3.99
28	*2835.30*	2858.53	1731.4	2576.9	2932.7	0.81
29	*2460.74*	2575.28	1161.7	1162.5	2026.9	4.45
30	*2031.94*	2076.20	1318.5	2021.4	2127.5	2.13
31	*2554.37*	2592.17	1944.2	2102.2	3163.4	1.46
32	*2462.45*	2605.10	2933.7	2305.2	5009.8	5.48
33	*2565.41*	2610.55	2531.6	2221.2	4474.1	1.73
34	*1303.49*	1546.06	4384.1	2184.4	5371.4	15.69
35	*1675.35*	1985.44	3887.9	2223.1	5004.2	15.62
36	*1864.73*	1946.66	3158.3	2626.3	5340.4	4.21
avg						*1.62*

GTS: The proposed algorithmic methodology, TS_G: the algorithm of Gendreau et al. (in press).
s: The average value of the best solutions for instances of Classes 2–5.
%gap: The percent improvement over the TS_G scores.
t: CPU time elapsed when the best solution was obtained (second).
tot: Total time required by the algorithm (second).
Bold characters correspond to higher quality solutions.
GTS: Visual C#, Pentium IV 2.4 GHz, 1 GB RAM, TS_G: C, Pentium IV 1.7 GHz computer.

5. Conclusions

In the present paper, we study a generalisation of the VRP, in which the demand of customers consists of weighted, two-dimensional, rectangular items. This problem is called Vehicle Routing Problem with two-dimensional loading constraints (2L-CVRP) and aims at generating the minimum cost routes, and feasibly packing the transported items onto the loading surfaces of the vehicles. The 2L-CVRP is of particular theoretical interest as it combines two frequently studied combinatorial optimisation problems, namely the Vehicle Routing Problem, and the two-dimensional bin packing problem. Although 2L-CVRP has several real-life applications in the field of transportation logistics, only recently have the first solution approaches been published.

Regarding the packing features of the problem examined, our algorithm makes use of a bundle of packing heuristics, producing diverse packing structures in order to increase the probability of obtaining a feasible loading. The routing aspects of the problem are handled by a Tabu Search method that incorporates the rationale of Guided Local Search, employing a guiding mechanism that controls the objective function of the problem. This guiding mechanism drastically diversifies the search conducted by trying to eliminate low-quality features from the final solution. To limit the computational effort demanded by the proposed algorithm, we employ two accelerating strategies. Firstly, the size of the neighbourhoods explored is reduced, and secondly, we use a tree memory structure to record the loading feasibility of visited routes, in order to eliminate any unnecessary duplicate calls to the packing heuristic bundle.

Table A1
Results obtained for the *Unresticted* 2L-CVRP

Instance	Class 1	Class 2	Class 3	Class 4	Class 5
1	278.73	305.92	299.70	296.75	280.60
2	334.96	334.96	355.65	342.00	334.96
3	358.40	401.81	409.17	368.56	358.40
4	430.88	440.94	446.61	447.37	430.88
5	375.28	381.85	387.89	383.87	375.28
6	495.85	498.16	499.08	504.78	495.85
7	568.56	741.91	706.99	703.85	661.22
8	568.56	718.18	749.70	711.07	643.43
9	607.65	607.65	622.16	625.13	607.65
10	535.80	708.63	655.70	792.30	695.37
11	505.01	719.56	746.12	843.52	652.42
12	610.00	628.86	610.00	618.23	610.23
13	2006.34	2705.05	2542.86	2714.69	2434.99
14	837.67	1117.24	1092.10	994.66	943.08
15	837.67	1099.75	1186.61	1258.49	1246.46
16	698.61	702.70	698.61	709.27	698.61
17	863.27	870.86	861.79	861.79	862.62
18	730.85	1065.30	1124.54	1171.51	945.88
19	524.61	796.87	816.77	819.79	674.20
20	244.54	569.20	557.72	576.92	503.01
21	687.60	1076.24	1191.07	1019.74	914.68
22	740.66	1088.33	1110.73	1119.34	986.02
23	839.07	1124.60	1141.51	1123.17	975.42
24	1035.33	1234.03	1136.10	1160.92	1065.41
25	829.45	1500.07	1476.14	1486.54	1212.73
26	819.56	1387.30	1436.55	1491.00	1267.68
27	1097.63	1402.42	1476.73	1397.75	1309.50
28	1042.12	2856.93	2867.46	2770.05	2453.59
29	1188.15	2362.75	2249.80	2427.95	2220.32
30	1037.05	1929.93	2038.55	1965.45	1625.42
31	1421.20	2456.28	2478.94	2585.67	2132.92
32	1328.68	2465.17	2422.98	2432.49	2086.13
33	1328.19	2508.68	2595.41	2601.34	2117.72
34	719.91	1268.93	1298.48	1279.65	1086.79
35	877.04	1464.93	1570.67	1634.63	1324.89
36	594.10	1854.06	1965.46	1803.86	1582.25
avg	777.75	1205.45	1217.40	1223.45	1078.24

To test the effectiveness and the robustness of the proposed algorithm, we solved a wide variety of 2L-CVRP benchmark instances. Improvement over several solutions previously published, demonstrates the capabilities of our methodology.

In terms of future research directions, the problem examined should be extensively studied. Its formulation could be extended to cover more operational constraints such as heterogeneous fleet of vehicles and time windows. From the methodological point of view, new algorithms combining both metaheuristic and exact optimisation methods can be designed.

Acknowledgements

We are indebted to the anonymous referees, for extensively reviewing our paper, and offering constructive remarks and directions for the completion of our work.

Appendix

See Tables A1 and A2.

Table A2
Results obtained for the *Sequential* 2L-CVRP

Instance	Class 1	Class 2	Class 3	Class 4	Class 5
1	278.73	319.86	314.33	296.75	285.93
2	334.96	347.73	356.24	342.00	340.88
3	358.40	414.39	413.63	383.11	362.27
4	430.88	451.98	448.24	447.37	430.88
5	375.28	407.45	401.09	399.65	377.26
6	495.85	499.08	509.65	515.60	495.85
7	568.56	764.45	712.57	723.78	694.54
8	568.56	730.87	749.70	727.58	654.74
9	607.65	611.49	644.54	625.13	607.65
10	535.80	740.43	671.24	770.82	728.95
11	505.01	748.96	783.92	868.01	671.71
12	610.00	638.06	610.57	655.60	610.23
13	2006.34	2778.28	2660.74	2748.57	2529.77
14	837.67	1185.52	1139.74	1036.37	1009.49
15	837.67	1120.00	1188.63	1329.39	1298.80
16	698.61	704.58	708.83	718.21	698.61
17	863.27	870.86	861.79	864.58	863.58
18	730.85	1092.60	1133.13	1199.03	984.23
19	524.61	819.80	839.19	854.50	690.26
20	244.54	576.24	583.17	614.73	532.17
21	687.60	1137.92	1267.29	1070.05	950.06
22	740.66	1143.24	1168.11	1169.14	1033.96
23	839.07	1212.71	1193.57	1185.41	1013.28
24	1035.33	1325.92	1168.25	1238.44	1093.88
25	829.45	1542.71	1554.00	1544.76	1274.63
26	819.56	1406.07	1499.53	1674.14	1324.11
27	1097.63	1512.41	1518.69	1448.80	1373.48
28	1042.12	2822.69	2954.63	2928.88	2635.00
29	1188.15	2518.99	2318.45	2590.22	2415.33
30	1037.05	2002.71	2304.98	2139.16	1680.91
31	1421.20	2542.41	2644.98	2759.44	2270.65
32	1328.68	2537.87	2521.68	2603.47	2186.76
33	1328.19	2572.98	2677.29	2811.12	2200.25
34	719.91	1302.54	1379.90	1374.86	1156.65
35	877.04	1564.85	1671.31	2072.36	1392.88
36	594.10	1914.94	2004.45	1871.24	1668.30
avg	777.75	1246.77	1266.06	1294.51	1126.05

References

Archetti, C., Speranza, M.G., Hertz, A., 2006. A Tabu Search algorithm for the split delivery Vehicle Routing Problem. Transportation Science 40 (1), 64–73.

Baker, B.S., Coffman, E.G., Rivest, R.L., 1980. Orthogonal packings in two dimensions 1980. SIAM Journal on Computing 9 (4), 846–855.

Brandão, J., 2006. A new Tabu Search algorithm for the Vehicle Routing Problem with backhauls. European Journal of Operational Research 173 (2), 540–555.

Burke, B.S., Kendall, G., Whitwell, G., 2004. A new placement heuristic for the orthogonal stock-cutting problem. Operations Research 52, 655–671.

Chazelle, B., 1983. The bottom-left bin packing heuristic: An efficient implementation. IEEE Transactions on Computers C-32, 697–707.

Cordeau, J.F., Gendreau, M., Hertz, A., Laporte, G., Sormany, J.S., 2004. New heuristics for the Vehicle Routing Problem. In: Langevin, A., Riopel, D. (Eds.), Logistics Systems: Design and Optimisation. Springer, New York.

Croes, G., 1958. A method for solving traveling salesman problems. Operations Research 6, 791–812.

El Fallahi, A., Prins, C., Calvo, R.W., 2008. A memetic algorithm and a Tabu Search for the multi-compartment Vehicle Routing Problem. Computers and Operations Research 35 (5), 1725–1741.

Fekete, S.P., Schepers, J., van der Veen, J., 2007. An exact algorithm for higher-dimensional orthogonal packing. Operations Research 55 (3), 569–587.

Gendreau, M., Hertz, A., Laporte, G., 1994. A Tabu Search heuristic for the Vehicle Routing Problem. Management Science 40, 1276–1290.

Gendreau, M., Iori, M., Laporte, G., Martello, S., in press. A Tabu Search heuristic for the Vehicle Routing Problem with two-dimensional loading constraints. Networks, doi:10.1002/net.20192.

Gendreau, M., Iori, M., Laporte, G., Martello, S., 2006. A Tabu Search algorithm for a routing and container loading problem. Transportation Science 40 (3), 342–350.

Gendreau, M., Laporte, G., Potvin, J.-Y., 2002. Metaheuristics for the VRP. In: Toth, P., Vigo, D. (Eds.), The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia.

Gendreau, M., Guertin, F., Potvin, J.-Y., Taillard, E., 1999. Parallel Tabu Search for real-time vehicle routing and dispatching. Transportation Science 33 (4), 381–390.

Gendreau, M., Hertz, A., Laporte, G., 1992. New insertion and post-optimization procedures for the traveling salesman problem. Operations Research 40, 1086–1094.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research 13, 533–549.

Hertz, A., 2006. Anniversary focused issue of Computers and Operations Research on Tabu Search. Computers and Operations Research 33, 2447–2448.

Iori, M., 2004. Metaheuristic algorithms for combinatorial optimization problems. PhD Thesis, DEIS, University of Bologna.

Iori, M., 2005. Metaheuristic algorithms for combinatorial optimization problems. 4OR 3, 163–166.

Iori, M., Salazar Gonzalez, J.J., Vigo, D., 2007. An exact approach for the Vehicle Routing Problem with two dimensional loading constraints. Transportation Science 41 (2), 253–264.

Lin, S., 1965. Computer solutions of the traveling salesman problem. Bell System Technical Journal 44, 2245–2269.

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.

Martello, S., Monacci, M., Vigo, D., 2003. An exact approach to the strip packing problem. INFORMS Journal on Computing 15, 310–319.

Martello, S., Vigo, D., 1998. Exact solution of the two-dimensional finite bin packing problem. Management Science 44, 388–399.

Mester, D., Bräysy, O., 2007. Active-guided evolution strategies for large-scale Capacitated Vehicle Routing Problems. Computers and Operations Research 34 (10), 2964–2975.

Moura, A., Oliveira, J.F., An integrated approach to the vehicle routing and container loading problems. Working Paper No. 2/2007, Institute for Systems and Computers Engineering at Coimbra (INESCC), University of Coimbra, Portugal.

Prins, C., 2004. A simple and effective evolutionary algorithm for the Vehicle Routing Problem. Computers and Operations Research 31 (12), 1985–2002.

Rochat, Y., Taillard, E.D., 1995. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1, 147–167.

Tarantilis, C.D., Kiranoudis, C.T., 2002. BoneRoute: An effective memory-based method for effective fleet management. Annals of Operations Research 115, 227–241.

Tarantilis, C.D., 2005. Solving the Vehicle Routing Problem with adaptive memory programming methodology. Computers and Operations Research 32, 2309–2327.

Toth, P., Vigo, D., 2002. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia.

Voudouris, C., Tsang, E., 1996. Partial constraint satisfaction problems and Guided Local Search. In: Proceedings of the Second International Conference on Practical Application of Constraint Technology (PACT'96), London, pp. 337–356.

Waters, C.D.J., 1987. A solution procedure for the vehicle scheduling problem based on iterative route improvement. Journal of Operations Research Society 38, 833–839.