INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 21 (2014) 375–398
DOI: 10.1111/itor.12070

A biased-randomized algorithm for the two-dimensional vehicle routing problem with and without item rotations

Oscar Dominguez[a], Angel A. Juan[b] and Javier Faulin[c]

[a]*Institute of Intelligent Systems and Numerical Applications in Engineering (SIANI), University of Las Palmas de Gran Canaria, 35017 Las Palmas de Gran Canaria, Spain*
[b]*Department of Computer Science, Multimedia, and Telecommunication, IN3 – Open University of Catalonia, 08018 Barcelona, Spain*
[c]*Department of Statistics and Operations Research, Public University of Navarre, 31006 Pamplona, Spain*
E-mail: oscar@opein.com [Dominguez]; ajuanp@uoc.edu [Juan]; javier.faulin@unavarra.es [Faulin]

Received 10 August 2012; received in revised form 16 October 2013; accepted 28 November 2013

Abstract

This paper proposes an efficient algorithm, with a reduced number of parameters, for solving the two-dimensional loading-capacitated vehicle routing problem (2L-CVRP). This problem combines two of the most important issues in logistics, that is, vehicle routing and packing problems. Our approach contemplates unrestricted loading including the possibility of applying 90° rotations to each rectangular-shaped item while loading it into the vehicle, which is a realistic assumption seldom considered in the existing literature. The algorithm uses a multistart approach that is designed to avoid local minima and also to make the algorithm an easily parallelizable one. At each restart, a biased randomization of a savings-based routing algorithm is combined with an enhanced version of a classical packing heuristic to produce feasible good solutions for the 2L-CVRP. The proposed algorithm has been compared with the classical benchmarks for two different 2L-CVRP variants, that is, with and without item rotations. Experimental results show that our approach outperforms several best-known solutions from previous work, both in terms of quality and the computational time needed to obtain them.

Keywords: vehicle routing problem; vehicle packing; multistart algorithms; biased randomization; heuristics

1. Introduction

The capacitated vehicle routing problem (CVRP) is a well-known combinatorial optimization problem in which a fleet of vehicles departing from a central depot has to satisfy some customers' demands at the lowest possible cost. It is assumed that each vehicle has a maximum loading capacity and also each customer has to be served by a single vehicle. Additional assumptions, such as distance-based or time-based constraints for any given route, or the existence of serving costs, are sometimes

considered. The CVRP has been extensively studied in the literature due to its potential applications to solve real-world problems (Golden et al., 2008; Toth and Vigo, 2002) as well as due to the fact that it constitutes a rich environment to develop and try new approaches, either of exact or approximate nature (Cordeau et al., 2005; Laporte, 1992).

In this paper, we are considering a relatively new variant of the CVRP, the so-called two-dimensional CVRP, or 2L-CVRP (Iori, 2005). In this variant, the customers' demands consist of lots of rectangular weighted items that cannot be stacked because of their fragility, weight, or dimensions. Note that the way these items are assigned to vehicles and packed into them might have a significant influence over the distribution costs. Thus, in addition to the routing issue, decision makers might face also a two-dimensional packing problem (Riff et al., 2009). Some real-life examples of the 2L-CVRP can be found, for example, in the transportation of large items, such as kitchen appliances, furniture, or heavy machinery in edification. In all these cases, it is necessary to consider not only the weight of the load but also the different dimensions of the rectangular-shaped products so that the corresponding items are packed in the assigned vehicle without overlapping. Note that, in order to satisfy the CVRP assumptions, all items requested by a single customer have to be loaded in the same vehicle—otherwise customers would be visited more than once by different vehicles.

From the point of view of the loading problem, there are different constraints that could be considered. Depending on these constraints, it is possible to distinguish among the following loading settings: (a) oriented loading (OL), where rotation of items is not allowed—that is, it is assumed that all items have a fixed orientation given as an input of the problem; (b) nonoriented loading (RL), where it is allowed to rotate items by 90° during the packing process; (c) sequential loading (SL), where items are always loaded in a reverse order to the order in which customers are visited but rearrangements of items inside the vehicle are not allowed once the route has started; and (d) unrestricted loading (UL), where items are allowed to be rearranged during the distribution process. According to the aforementioned loading settings, Fuellerer et al. (2009) propose the following classification for 2L-CVRPs: (a) two-dimensional sequential oriented loading (2|SO|L); (b) two-dimensional unrestricted oriented loading (2|UO|L); (c) two-dimensional sequential nonoriented (rotated) loading (2|SR|L); and (d) two-dimensional unrestricted nonoriented (rotated) loading (2|UR|L)

Figure 1 provides a vehicle packing and routing solution for a small-sized 2L-CVRP instance with 3 vehicles and 15 nodes. This solution was obtained with the approach presented in this paper, and it corresponds to one of the instances analyzed in the numerical section—instance 1, class 2, without items rotation

To the best of our knowledge, Fuellerer et al. (2009) are the first and only authors so far to deal with the 2L-CVRP with nonoriented loading. In our opinion, allowing items rotation during the packaging process is a realistic assumption that has not received enough attention in the existing literature. For that reason, nonoriented loading is precisely one of the variants studied in this paper. The oriented version is considered too in order to compare our approach with other existing approaches in the literature. Therefore, this paper presents an algorithm for solving both the two-dimensional unrestricted oriented (2|UO|L), as well as the two-dimensional unrestricted nonoriented (2|UR|L) loading cases. Moreover, in order to avoid complex and time-consuming fine-tuning processes, we want our algorithm to be easily parallelizable and to have a reduced number of parameters. To that end, a multistart probabilistic algorithm with few and robust parameters

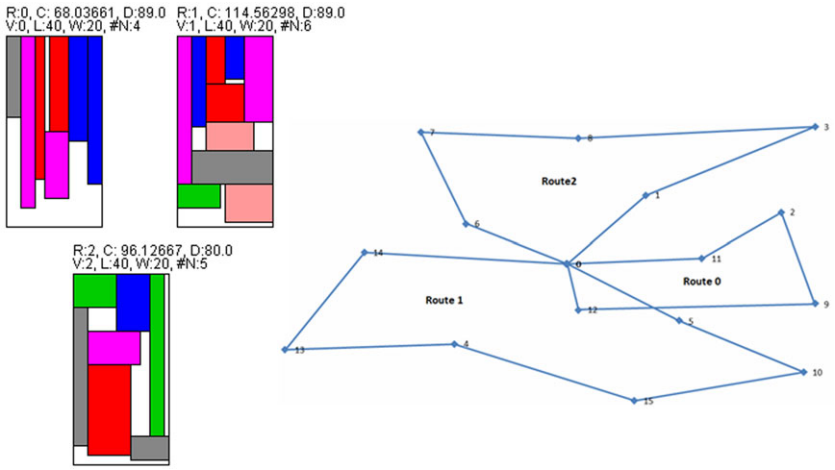

Fig. 1. Example of an integrated packing and routing solution for the 2L-CVRP.

is proposed. This algorithm makes use of biased randomization of classical heuristics, which transforms a deterministic procedure into a multistart probabilistic algorithm without losing the logic behind the heuristic (Juan et al., 2011a). In particular, the routing part of our approach employs an enhanced and biased-randomized version of the well-known savings heuristic (Clarke and Wright, 1964). This randomized version makes use of a pseudogeometric distribution as initially proposed in Juan et al. (2010). Moreover, it also uses some memory-based and splitting strategies as described in Juan et al. (2011b). During the solution-construction process, the algorithm verifies two feasibility conditions: (a) the total item weight is lower than the vehicle maximum capacity (vehicle-capacity constraint) and (b) the items can be loaded or packed into the assigned vehicle without overlapping (packing constraint). When verifying this latter constraint, the algorithm employs an adapted version of the enhanced best-fit packing heuristic developed by Burke et al. (2004). The algorithm is also encapsulated into a multistart process with randomly selected starting points, which facilitates it to escape from local minima.

This paper is structured as follows. Section 2 describes the 2L-CVRP and presents the corresponding mathematical model. Section 3 reviews some related work. Section 4 gives an overview of our approach, including an explanation of our biased-randomized version of the Clarke and Wright Savings (CWS) heuristic, as well as an explanation of how the algorithm integrates the best-fit method. Section 5 provides and discusses the pseudocode associated with our algorithm. Section 6 describes some numerical experiments that contribute to illustrate and validate our approach. Finally, Section 7 summarizes the main contributions and results of this work.

2. Problem formulation

Since the 2L-CVRP combines a vehicle routing problem with a vehicle packing problem, its formulation will include constraints from both research topics. Therefore, as a CVRP, the 2L-CVRP assumes the existence of a homogeneous fleet of vehicles initially located in a central depot. These

vehicles must distribute products in order to satisfy customers' demands. Then, the main goal is to find a feasible and pseudo-optimal routing plan that minimizes distribution costs while not violating any of the routing constraints, namely (a) each customer is visited by just one vehicle, which satisfies its total demand; (b) each vehicle starts and ends its route at the depot (closed-routes assumption); and (c) the total demand covered by any vehicle does not exceed its maximum weight-loading capacity. In the extensive CVRP literature, it is possible to find different mathematical models, see for instance Toth and Vigo (2002) and Golden et al. (2008). Next, we present an integer linear programming (ILP) adaptation from models presented in the aforementioned works. Consider a complete undirected graph $G = (N, E)$, where

- $N = \{0, 1, \ldots, n\}$ is a set of $n + 1$ nodes, representing the central depot (node 0) and the n customers to be supplied (nodes 1 to n).
- $E = \{(i, j) \mid i, j \in N \text{ with } i \neq j\}$ is the set of edges connecting nodes i and j.

Moreover, consider also the following problem inputs:

- For all $i, j \in N$ with $i \neq j$, $c_{ij} = c_{ji} > 0$ are the traveling cost associated with traveling from one node to another. Note that these costs are assumed to be symmetric.
- For each customer i ($i \in N$), $m_i \geq 0$ is the number of items requested by node i. It is assumed that the depot has no demand, that is, $m_0 = 0$.
- For each customer's item l ($1 \leq l \leq m_i$), d_{il} represents the weight of item l in customer i. Thus, the total weight (demand) associated with each customer i can be expressed as d_i, where $d_i = \sum_{l=1}^{m_i} d_{il}$.
- The integer number $K > 1$ of identical vehicles in the fleet.
- The real number $D > 0$, representing the maximum weight-loading capacity of each vehicle.
- The binary variable z_{ijk}, which takes the value 1 if vehicle k is used to go from node i to node j, while it takes the value 0 otherwise.

Then, the routing part of the problem can be expressed as
Objective function:

$$\text{Minimize} \sum_{(i,j)\in E} c_{ij} \sum_{k=1}^{K} z_{ijk}$$

Subject to

$$\sum_{k=1}^{K} \sum_{\substack{j\in N\setminus\{0\} \\ j\neq i}} z_{ijk} = 1 \quad \forall i \in N\setminus\{0\}, \tag{1}$$

$$\sum_{h\neq i\in N} z_{ihk} = \sum_{\substack{j\in N \\ j\neq h}} z_{hjk} \quad \forall h \in N, \forall k \in \{1, 2, \ldots, K\}, \tag{2}$$

$$\sum_{i\in N} d_i \sum_{\substack{j\in N \\ j\neq i}} z_{ijk} \leq D \quad \forall k \in \{1, 2, \ldots, K\}, \tag{3}$$

$$\sum_{k=1}^{K} \sum_{j \in N \setminus \{0\}} z_{0jk} \leq K, \tag{4}$$

$$z_{ijk} \in \{0, 1\} \quad \forall (i, j) \in E, \forall k \in \{1, 2, \ldots, K\} \tag{5}$$

Constraints (1) require that each customer is visited just once. Constraints (2) ensure that each node h has the same number of incoming arcs as of outgoing arcs. Constraints (3) control that the weight-loading capacity of each vehicle is never exceeded. Constraint (4) guarantees that there are enough vehicles in the fleet. Finally, constraints (5) set the z_{ijk} variables as binary ones.

Additionally, when modeling the 2L-CVRP, some vehicle packing assumptions must be taken into account. These assumptions will be related to the feasible packing of rectangular-shaped items inside the vehicle surface. As in previous works, in this paper it is assumed that each item must be loaded with its edges parallel to the edges of the vehicles (orthogonal packing). Also, the items need to be loaded without overlapping over the vehicle surface, which has some predefined dimensions. Thus, the following problem inputs arise:

- For each customer's item l ($1 \leq l \leq m_i$), consider the natural numbers h_{il} and w_{il}. These numbers are, respectively, the height and width of item l in customer i.
- The natural numbers H and W, which are the height and width, respectively, are associated with the surface of a vehicle.

Therefore, the surface of a vehicle can be seen as a $W \times H$ matrix with indexes x in $\{1, 2, \ldots, W\}$ and y in $\{1, 2, \ldots, H\}$. Thus, the coordinates (x_{ilk}, y_{ilk}) represent, for vehicle k delivering product l to customer i, the position where the bottom-left corner of item l is located inside the $W \times H$ matrix. Finally, the binary variable Ω_{il} takes the value 1 if the rectangular-shaped item l in customer i is considered as initially given—that is, without rotation—and 0 if it is rotated by 90°—that is, in the item characteristics w_{il} becomes h_{il} and *vice versa*. Using these packing-related inputs and variables, we have adapted some of the constraints introduced by Iori et al. (2007) as follows:

$$\left(W + 1 - h_{il}\right)\left(1 - \Omega_{il}\right) + \left(W + 1 - w_{il}\right)\Omega_{il} \geq x_{ilk} \geq 1 \tag{6a}$$

$$\left(H + 1 - w_{il}\right)\left(1 - \Omega_{il}\right) + \left(H + 1 - h_{il}\right)\Omega_{il} \geq y_{ilk} \geq 1 \tag{6b}$$

$$\forall\, k \in \{1, 2, \ldots, K\}, \forall i, j \in N \setminus \{0\} : z_{ijk} = 1, \forall l \in \left\{1, 2, \ldots, m_i\right\}$$

$$x_{jl'k} + w_{jl'}\Omega_{jl'} + h_{jl'}\left(1 - \Omega_{jl'}\right) \leq x_{jl''k} \leq W\text{ory}_{jl'k} + h_{jl'}\Omega_{jl'} + w_{jl'}\left(1 - \Omega_{jl'}\right) \leq y_{jl''k} \leq H \tag{7a}$$

$$x_{ilk} + w_{jl}\Omega_{jl} + h_{il}\left(1 - \Omega_{il}\right) \leq x_{jl'k} \leq W\text{ory}_{ilk} + h_{il}\Omega_{il} + w_{il}\left(1 - \Omega_{il}\right) \leq y_{jl'k} \leq H$$

$$\forall k \in \{1, 2, \ldots, K\}, \forall i, j \in N \setminus \{0\} : z_{ijk} = 1, \forall l \in \left\{1, 2, \ldots, m_i\right\}, l' \neq l'' \in \left\{1, 2, \ldots, m_j\right\} \tag{7b}$$

On the one hand, constraints (6a) and (6b) ensure that each item, say l, will not exceed the loading surface, that is, they guarantee that there will be enough space in the vehicle to load the rectangular-shaped item. Note that whenever Ω_{il} equals 0 (i.e., the item is rotated by 90°) only the second term in the constraints is considered, meaning that the height of the item will become now its width and *vice versa*. On the other hand, constraints (7a) and (7b) avoid overlapping of items when they are located over the surface of the vehicle. Thus, the bottom-left corner of a new item can only be set into a (x, y) position if this position has not been previously occupied by any other item. Note also that when all items are 1×1 squares, loading constraints become trivial and the problem is reduced to a standard CVRP.

3. Literature review

In recent years, the 2L-CVRP has attracted an increasing interest from the research community. Several variants of the problem have been considered, mainly due to the use of different vehicle routing and packing assumptions. Wang et al. (2009) present a survey on vehicle routing problems with loading constraints. Also, a systematic review on the 2L-CVRP can be found in Iori and Martello (2010). As a generalization of the CVRP, it is clear that the 2L-CVRP is NP-complete and therefore, difficult to solve in practice. The first approach to this problem was made by Iori et al. (2007), using exact methods to solve the two-dimensional sequential oriented loading variant, 2|SO|L. This exact approach employs a branch-and-cut technique to solve the routing problem, whereas to solve the packing problem uses lower bounds, a constructive heuristic, and finally, if both fail, a branch-and-bound technique is used. Although exact solutions are obtained up to instances with 35 customers and 114 items, nevertheless there are smaller instances that remain unsolved. Thus, for example, Iori and Martello (2010) identify an instance with just 29 customers and 43 items, which has not been solved yet with exact methods.

Since these limits are not reasonable for real-life practice, several heuristics and metaheuristics have also been proposed for 2L-CVRP. Initially, a tabu search (TS) algorithm was used by Gendreau et al. (2008), for both sequential and unrestricted oriented loading, 2|SO|L and 2|UO|L, respectively. They increase the number of solved instances to 36, including 18 new larger instances (with a maximum of 255 customers and 786 items). Likewise, a guided TS (GTS) algorithm was proposed by Zachariadis et al. (2009) for solving the two oriented loading variants, 2|SO|L and 2|UO|L. For checking feasibility of the loading configuration, they made use of five heuristics with different selection criteria for the position of the items into the vehicle loading surface. These heuristics were executed sequentially, starting with the simplest (bottom-left fill) and ending with the most complex and effective (min area heuristic).

Similarly, Fuellerer et al. (2009) was the first study to develop two new variants to the 2L-CVRP, which consider the option of rotating 90° the items to load (nonoriented loading), considering sequential and unrestricted loading (2|SR|L and 2|UR|L variants). Moreover, Fuellerer et al. (2009) tried to solve the problem using an algorithm based on ant colony optimization (ACO), adapted to 2L-CVRP. The results obtained by these authors for the variants 2|UO|L and 2|SO|L of the oriented problems show that ACO achieves better results than TS, having a performance average greater than 3%, but involving much more

computational time. For this kind of problems, ACO is especially good with cases with loads of higher complexity.

Additionally, Leung et al. (2010) published the first paper that makes use of simulated annealing (SA) as a metaheuristics to tackle the 2L-CVRP in its traditional variants 2|UO|L and 2|SO|L. The obtained results, considering the average quality of the solution, are 3% better than the TS results (Gendreau et al., 2008), and superior in a 2% for the GTS methods (Zachariadis et al., 2009). Generally speaking, the improvement reached by the SA methods is better in the most complex cases. Furthermore, Leung et al. (2011), following the aforementioned work of Zachariadis et al. (2009), used the metaheuristics called extended GTS (EGTS) for the main routing problem and adds a new heuristics for the load configuration checking. Besides, Zachariadis et al. (2009) employed six different heuristics described in the paper to solve the secondary problem in a sequential way; and having the purpose of not losing good solutions due to penalties, they used an aspiration criterion obtaining a good performance with the application of those combined methods.

Finally, Duhamel et al. (2009, 2011) presented an innovative methodology to solve the 2L-CVRP in the 2|UO|L version, based on the transformation of the 2L load problem in a resource constrained project scheduling problem (RCPSP), in order to solve the resultant RCPSP-CVRP using the metaheuristics GRASP × ELS (greedy randomized adaptive search procedure hybridized with evolutionary local search; Prins, 2009). Later, the obtained solution for the RCPSP-CVRP is transformed into a feasible solution for the 2L-CVRP, achieving very promising results.

4. Overview of the solution approach

This section provides an overview of our algorithm and discusses some of its main design properties, such as (a) the parameters it employs and (b) how it combines different heuristics and local search processes from the vehicle routing and packing literature. In our approach, we consider two loading configurations, in both loading configurations items are allowed to be rearranged during the distribution process (unrestricted case). These loading configurations use the following realistic assumptions: (a) any item is allowed to be moved inside the truck while other items are being unloaded (unrestricted-order assumption); (b) in the case of 2|UR|L, items can be rotated while being loaded in order to look for a better packing solution (nonoriented loading assumption); and (c) in the case of 2|UO|L, items cannot be rotated during truck loadings operations (oriented loading assumption). Figure 2 shows a flowchart diagram offering a high-level view of our algorithm.

First of all, note that while other approaches use a two-stage schema—one for solving the vehicle packing problem and then another for solving the vehicle routing problem—our approach integrates the packing issue as part of the routing-construction process. That is, the proposed algorithm will generate routing solutions implicitly taking into account the loading constraints as well.

The algorithm starts by computing a dummy but feasible solution. For each customer, this initial solution assigns one round-trip route from the depot to it. The algorithm also computes the savings

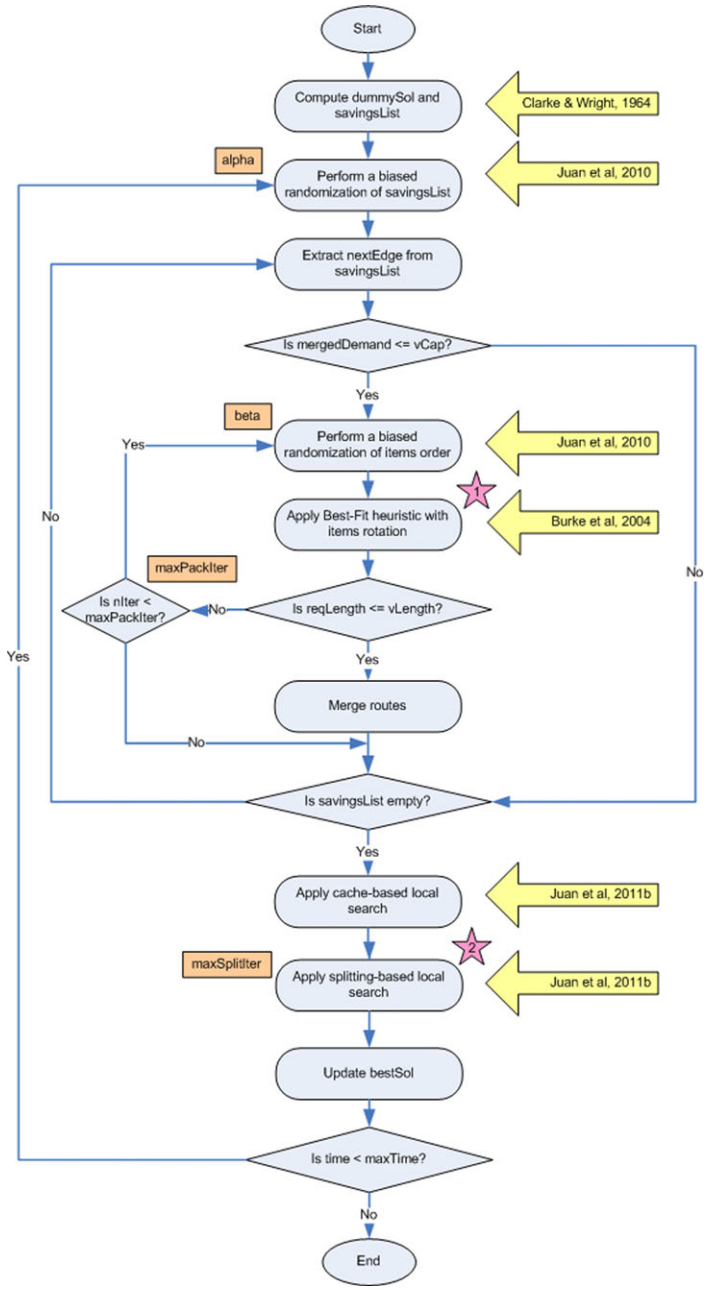

Fig. 2. Flowchart of our approach.

associated with each edge as proposed in Clarke and Wright (1964). These edges are then listed from highest to lowest savings. At this point, a multistart process is initiated. At each iteration of this process, the savings list of edges is randomized using a biased probability distribution. In our case, a geometric distribution is employed to induce this biased randomization behavior. This distribution uses one single parameter, α ($0 < \alpha < 1$). As discussed in Juan et al. (2010), by performing a biased randomization of the savings list, edges are selected in a different order at each iteration of the multistart process while, at the same time, the logic behind the savings-based heuristic is maintained, that is, edges with higher savings are more likely to be selected than those with lower savings. Then, until the savings list gets empty, an iterative process begins in which the edge at the top of the biased-randomized list is extracted. This edge connects two different routes. If—and only if—these two routes can be merged without violating any constraint—both in terms of capacity and loading—then the route merging is carried out. In order to check feasibility of items packing in the truck, the algorithm employs a multistart biased-randomized version of the best-fit heuristic with items rotation (Burke et al., 2004). This biased-randomization process is similar to the previous one. In this case, however, the biased randomization applies over the list of items to be loaded. Again, we propose to use a geometric distribution with one single parameter, β ($0 < \beta < 1$). A new parameter, *maxPackIter*, controls the maximum number of times that the randomized best-fit heuristic will be run before assuming that the items cannot be fitted into a single vehicle—that is, before assuming that the merge is not possible since the loading constraint would be transgressed.

At this point, a new feasible solution is ready. The routing component of this solution can be improved further by using the cache and splitting techniques described in Juan et al. (2011b). The cache technique is a fast local search process that adds "memory" to the algorithm, allowing it to achieve a faster convergence to a pseudooptimal solution. The splitting technique constitutes an even more interesting local search process. This divide-and-conquer strategy performs the following basic sequential steps: (a) it splits the new feasible solution into different disjoint (and feasible) parts according to some geometric properties; (b) for each of these parts, it dissolves the associated routes and starts an iterative process in which the aforementioned methodology is repeated during *maxSplitIter* iterations to obtain an improved "local" solution to the current part, which, by construction, represents a smaller size 2L-CVRP; and (c) reunifies all "local" solutions into an enhanced "global" solution. Note that the use of a multistart process with randomized savings lists contributes to escape from local minimum and also facilitates parallelization of the algorithm.

5. Pseudocode of the algorithm

This section presents a lower level description of the proposed algorithm. Figure 3 shows the pseudocode associated with "MultiStart-BiasedRand" (MS-BR), the main procedure of our approach. This procedure requires the four aforementioned parameters: (a) the one associated with the geometric distribution employed during the biased randomization process of the edges list (alpha), and the one associated with the geometric distribution employed during the biased randomization process of the items list (beta); and (b) the two associated with the maximum number of packing iterations (*maxPackIter*), and with the maximum number of splitting iterations (*maxSplitIter*). The

```
procedure MultiStart-BiasedRand(inputs, alpha, beta, maxPackIter, maxSplitIter)
01 dummySol <- calcDummySol(inputs) % generate the dummy feasible sol
02 savings <- calcSortedSavingsList(inputs) % compute the sorted savings list
03 cwsSol <- packAndRoute(dummySol, savings, beta, maxPackIter, inputs) % reference sol
04 bestSol <- cwsSol
05 while {ending condition is not met} do % time- or iteration-based condition
06   randSavings <- biasedRand(savings, alpha) % biased randomization of savings list
07   newSol <- packAndRoute(dummySol, randSavings, beta, maxPackIter, inputs) % new random sol
08   newSol <- cache(newSol) % use the fast cache-based local search
09   if {newSol is a 'promising sol'} then % e.g. cost(newSol) < cost(cwsSol)
       % use the splitting-based local search
10     newSol <- splitting(newSol, alpha, inputs, maxSplitIter)
11   end if
12   if {cost(newSol) < cost(bestSol)} then
13     bestSol <- newSol
14   end if
15 end while
16 return bestSol
end procedure
```

Fig. 3. Main procedure of the multistart biased-randomized algorithm.

procedure also needs the instance inputs: Euclidean coordinates of each node—or, alternatively, the traveling costs matrix—customers' demands, vehicle maximum capacity, vehicle dimensions, and dimensions of each rectangular-shaped product. First, following the savings heuristic, the procedure generates a savings list and also an initial dummy solution (Fig. 3, lines 1 and 2). This dummy solution is obtained by assigning, for each customer, a round-trip route from the depot to it. Then, a reference solution is constructed using the "packAndRoute" procedure (Fig. 3, line 3), which basically combines the classical savings-based heuristic with our randomized version of the well-known best-fit heuristic. After that, a multistart process is started (Fig. 3, lines 5–15). As explained before, this multistart process is particularly useful for two reasons: (a) it allows the algorithm to escape from local minima and (b) it makes it very easy to parallelize the randomized algorithm. In fact, the algorithm can be parallelized by just running multiple instances of it, each one using a different seed for the pseudorandom number generator. These multiple instances can be executed over several threads, cores, or computers. At each iteration of the multistart process, a new feasible solution is obtained (Fig. 3, line 7). This new solution is provided by the "packAndRoute" procedure after performing a biased randomization of the savings list (Fig. 3, line 6). Next, a fast memory-based local search process is applied to the new solution (Fig. 3, line 8). This local search tries to update each route in the new solution by a better route covering the same set of nodes and obtained in previous iterations. Moreover, if the solution can be considered a "promising" one, we apply another local search procedure, which is described later in more detail. This procedure is based on splitting strategies (Fig. 3, lines 9–11). If the new solution improves the best solution so far, then the former is updated (Fig. 3, lines 12–14). Eventually, the algorithm will return the best solution produced in the entire multiround process (Fig. 3, line 15).

One of the key steps in our approach is the generation of new random solutions that aim at minimizing total costs while satisfying the capacity and packing constraints. Details of this constructive process can be found at the pseudocode included in Fig. 4. Using the initial dummy solution as starting point, a route-merging constructive process starts (Fig. 4, lines 2–16). In this process, new edges are selected from the randomly sorted savings list and their corresponding routes

```
procedure packAndRoute(dummySol, randSavings, beta, maxPackIter, inputs)
01 newSol <- dummySol
02 while {savings list is not empty} do
03   nextEdge <- extractNextEdge(randSavings)
04   iR <- getRoute(origin(nextEdge))
05   jR <- getRoute(end(nextEdge))
06   newRoute <- merge(iR, jR)
07   demand <- calcDemand(newRoute) % here demand is measured in terms of weight
08   if {demand <= vehicleCapacity} then
09     if {routeRectanglesArea < vehicleLoadingSurface} then
         % Apply Best-Fit with rotation of items
10       reqLength <- bestFit(routeRectangles, beta, maxPackIter, vehWidth, vehLength)
11       if {reqLength <= vehLength} then
12         newSol <- updateRoute(iR, jR, newRoute)
13       end if
14     end if
15   end if
16 end while
17 return newSol
end procedure
```

Fig. 4. Procedure for generating new feasible random solutions.

are merged, if possible (Fig. 4, lines 8–15)—that is, if no constraints are violated, including the packing constraint. Note that any efficient packing algorithm could be used in line 10 (Fig. 4) to solve the packing problem in order to test loading constrains feasibility (Fig. 4, lines 11–13). As a result, at the end of each constructive process, a random feasible solution is obtained (Fig. 4, line 17). Of course, the quality of the random solution will depend on the way edges have been randomly selected from the list. There is where the biased geometric probability distributions can be very helpful. In other words, if a uniform probability distribution were used instead of a geometric one, the resulting solutions would not be competitive at all, since the logic behind the heuristic would be lost.

The splitting procedure (Fig. 5) is also a key component in our approach. Roughly speaking, given a new random—and feasible—solution, it splits the solution into several disjoint—but still feasible—subsolutions (sets of routes). Then, it applies a biased-randomized version of the savings heuristic over each of these subsolutions. In other words, it divides the entire problem into several feasible subproblems and then focuses on improving the current solution for each of them. Note that this approach greatly benefits from two facts: (a) due to their combinatorial nature, smaller subproblems are much easier to solve than the entire problem and (b) improving any of the disjoint subsolutions will improve the global solution. In particular, as discussed in Juan et al. (2011b), many different geometry-based policies can be used to divide the global solution into disjoint subsolutions. Note that if distances inside a group of routes are relatively small, then some of the nodes in these routes could be transferred from one route to another without a significant increment in routing costs. Consequently, the main idea is simple: use the current feasible solution to group routes that are located "close" together in the Euclidean plane. Thus, for example, a North–South splitting policy could be summarized as (a) consider the geometric center of the problem, (x_0, y_0), as well as the geometric center associated with each route i, (x_i, y_i); and then (b) divide the routes into two

```
procedure splitting(newSol, alpha, inputs, maxSplitIter)
01 globalSol <- empty
02 subSols <- split(newSol) % split the newSol into disjoint subSols
03 for each {sol in subSols} do
04   bestSol <- sol
05   inputs <- getInputs(sol)
06   dummySol <- calcDummySol(inputs) % generate the dummy feasible sol
07   savings <- calcSortedSavingsList(inputs) % compute the sorted savings list
08   iter <- 1
09   while {iter <= maxSplitIter} do
10     randSavings <- biasedRand(savings, alpha) % biased randomization of savings list
11     newSol <- packAndRoute(dummySol, randSavings, inputs) % new random sol
12     newSol <- cache(newSol) % use the fast cache-based local search
13     if {cost(newSol) < cost(bestSol)} then
14       bestSol <- newSol
15     end if
16     iter <- iter + 1
17   end while
18   globalSol <- add(bestSol, globalSol)
19 end for
20 return globalSol
end procedure
```

Fig. 5. Procedure for splitting local search.

groups, those with $x_i \geq x_0$ (North routes) and those with $x_i < x_0$ (South routes). In a similar way, other policies can be defined, for example, East–West, NE–NW–SE–SW, etc.

6. Numerical experiments

The algorithm described in this paper has been implemented as a Java application. At the core of this implementation, we included the SSJ library provided in L'Ecuyer et al. (2002) and, in particular, the LFSR113 pseudorandom number generator. An Intel® Core™ 2 Duo at 2.4 GHz and 4 GB RAM was used to perform all tests, which were run directly on the Netbeans platform for Java over Windows 7.

In order to compare the efficiency of our approach for the two unrestricted configurations, 2|UO|L and 2|UR|L, some classical benchmark instances for the 2L-CVRP were selected from the web site www.or.deis.unibo.it/research.html. This site contains detailed information on a large number of benchmark instances for different CVRPs. In the case of 2|UR|L, all instances and classes were tested. In the case of 2|UO|L, a subset of instances was randomly selected from the entire set. These benchmark instances consider the existence of a fleet with a limited number of vehicles, each of them having a loading surface of 40×20 squared units and a maximum capacity of weight. Each customer has a demand consisting of a set of two-dimensional rectangular items with given sizes and weight. Moreover, for each of these instances, five different classes are considered. As described in Iori et al. (2007) and Gendreau et al. (2008), these classes differ in the number and size of their items. Note that class 1 is a pure CVRP instance and it is not affected by the loading constraints. For each of the chosen benchmark instances, our algorithm was tested in all five classes. For each

instance–class combination, 10 independent iterations (replicas) were run using a different seed for the pseudorandom number generator.

6.1. Scenario A: items rotation is allowed (nonoriented loading)

In the case of two-dimensional unrestricted nonoriented (rotated) loading, 2|UR|L, both the best solution found (BEST10) as well as the average value of the generated solutions (AVG10), were registered. These BEST10 and AVG10 values were compared with the results obtained with the ACO algorithm developed by Fuellerer et al. (2009), who also completed 10 runs (replicas) per instance and class using a maximum CPU time of three hours for each replica. In the case of our algorithm, however, each replica was run for a maximum time of 500 seconds, that is, each instance–class combination was run for a total maximum time of 83 minutes (1.4 hours). The ACO algorithm was coded in C++, and all their runs were performed on a Pentium 4 with 3.2 GHz under the Linux operating system. The web site http://prolog.univie.ac.at/research/VRPandBPP/ contains 10 different solutions (runs) generated by this ACO algorithm for each of the five classes of the classical 2L-CVRP instances. The comparisons between this ACO algorithm and our MS-BR algorithm are summarized in Tables 1–5.

The comparison in Tables 1–5 shows that our approach seems to be quite competitive, showing negative gaps for most instances, with respect to the results published in Fuellerer et al. (2009). It is worthy to remember that, to the best of our knowledge, these authors are the only ones analyzing the two-dimensional unrestricted and nonoriented loading problem, or 2|UR|L. Figure 6 shows, for each class, a box-plot of the gaps between the BEST10 value given by the ACO algorithm and the BEST10 value obtained with our MS-BR approach. The negative gaps show that our approach performs slightly better for all classes on the average—and also for most individual instances—especially for classes 3, 4, and 2.

6.2. Scenario B: items rotation is not allowed (oriented loading)

As discussed in the introduction, several approaches exist for solving the two-dimensional unrestricted oriented loading variant, 2|UO|L, where no items rotation is allowed. In order to test the efficiency of our approach in this scenario, we compare the results obtained using our MS-BR algorithm with those published in the literature. In particular, the following algorithms were selected: (a) the EGTS developed by Leung et al. (2011), (b) the SA developed by Leung et al. (2010), (c) the ACO developed by Fuellerer et al. (2009), and (d) the GRASP $\times$ ELS developed by Duhamel et al. (2011). For each class, the best results obtained with each of the previous approaches are compared with our best results in Tables 6–10. At this point, it is worthy to note that all these algorithms are quite recent, so they were run in similar CPUs (between 2.0 and 3.0 GHz). Also, all of them were run for similar maximum time periods (between one and 1.5 hours for each replica). Algorithms (1)–(4) were developed in C/C++, whereas our algorithm was developed in Java. Being an interpreted language, Java-based programs do not execute as fast as C/C++ compiled programs, the performance of which can be optimized in several ways. However, Java permits a rapid, platform-independent, development of object-oriented prototypes that can be used to test the potential of

Table 1
Comparison between the ACO algorithm and our MS-BR algorithm for 2|UR|L class 1

Class 1							
	BEST10				AVG10		
Instance	ACO	MS-BR	MS-BR Time (seconds)	Gap (%)	ACO	MS-BR	Gap (%)
1	278.73	278.73	0	0.00	278.73	278.73	0.00
2	334.96	334.96	0	0.00	334.96	334.96	0.00
3	358.40	358.40	1	0.00	358.40	358.40	0.00
4	430.88	430.88	0	0.00	430.89	430.88	0.00
5	375.28	375.28	3	0.00	375.28	375.28	0.00
6	495.85	495.85	0	0.00	495.85	495.85	0.00
7	568.56	568.56	0	0.00	568.56	568.56	0.00
8	568.56	568.56	0	0.00	568.56	568.56	0.00
9	607.65	607.65	1	0.00	607.65	607.65	0.00
10	535.80	535.80	61	0.00	535.80	537.90	0.39
11	505.01	505.01	1	0.00	505.01	505.01	0.00
12	610.00	610.00	2	0.00	611.22	610.00	−0.20
13	2006.34	2006.34	47	0.00	2006.34	2006.34	0.00
14	837.67	837.67	48	0.00	837.67	837.67	0.00
15	837.67	837.67	8	0.00	837.67	837.67	0.00
16	698.61	698.61	25	0.00	698.60	698.61	0.00
17	861.79	861.79	20	0.00	862.37	861.96	−0.05
18	723.54	723.54	92	0.00	725.88	723.89	−0.27
19	524.61	524.61	40	0.00	527.29	524.61	−0.51
20	241.97	241.97	31	0.00	242.02	242.00	−0.01
21	690.20	687.60	14	−0.38	691.94	687.60	−0.63
22	742.91	741.90	90	−0.14	746.45	742.01	−0.60
23	845.34	838.60	12	−0.80	853.09	838.60	−1.70
24	1030.25	1028.38	315	−0.18	1042.26	1030.50	−1.13
25	830.82	830.14	482	−0.08	834.96	833.06	−0.23
26	819.56	819.56	17	0.00	819.56	819.56	0.00
27	1100.22	1091.09	290	−0.83	1104.42	1095.70	−0.79
28	1062.23	1043.89	69	−1.73	1151.05	1044.15	−9.29
29	1168.13	1166.73	375	−0.12	1212.40	1167.39	−3.71
30	1041.05	1042.38	303	0.13	1047.66	1049.82	0.21
31	1341.89	1329.11	453	−0.95	1372.20	1356.49	−1.14
32	1334.26	1309.03	407	−1.89	1359.58	1318.34	−3.03
33	1331.69	1308.39	243	−1.75	1356.86	1316.70	−2.96
34	712.32	711.71	386	−0.09	715.72	714.69	−0.14
35	868.12	883.90	247	1.82	879.56	888.52	1.02
36	616.69	613.00	495	−0.60	632.68	619.39	−2.10
Average				−0.21			−0.75

Table 2
Comparison between the ACO algorithm and our MS-BR algorithm 2|UR|L class 2

Class 2							
	BEST10				AVG10		
Instance	ACO	MS-BR	MS-BR Time (seconds)	Gap (%)	ACO	MS-BR	Gap (%)
1	278.73	278.73	2	0.00	278.73	278.73	0.00
2	334.96	334.96	0	0.00	334.96	334.96	0.00
3	380.35	380.35	67	0.00	383.10	383.56	0.12
4	430.88	430.88	2	0.00	430.89	430.88	0.00
5	375.28	375.28	14	0.00	375.28	375.28	0.00
6	495.85	495.85	1	0.00	495.85	495.85	0.00
7	715.02	715.02	10	0.00	715.02	715.02	0.00
8	674.19	665.17	4	− 1.34	674.20	665.17	− 1.34
9	607.65	607.65	2	0.00	607.65	607.65	0.00
10	684.42	667.42	250	− 2.48	684.42	667.42	− 2.48
11	678.93	664.48	231	− 2.13	693.93	667.87	− 3.76
12	610.00	610.00	11	0.00	611.23	611.85	0.10
13	2504.53	2502.65	106	− 0.08	2504.53	2505.17	0.03
14	1032.01	1029.34	112	− 0.26	1033.53	1030.44	− 0.30
15	1008.56	1001.51	137	− 0.70	1008.69	1003.54	− 0.51
16	698.61	698.61	48	0.00	698.60	698.61	0.00
17	863.27	861.79	21	− 0.17	863.68	861.94	− 0.20
18	988.91	988.61	142	− 0.03	988.91	988.90	0.00
19	730.16	726.51	123	− 0.50	730.16	726.96	− 0.44
20	492.91	489.23	129	− 0.75	495.73	489.29	− 1.30
21	978.07	964.49	443	− 1.39	985.32	969.21	− 1.63
22	988.15	976.70	154	− 1.16	988.67	978.49	− 1.03
23	1005.94	985.18	400	− 2.06	1011.20	987.94	− 2.30
24	1160.48	1152.35	272	− 0.70	1164.41	1153.71	− 0.92
25	1360.72	1356.24	250	− 0.33	1368.24	1360.05	− 0.60
26	1267.04	1262.43	430	− 0.36	1267.86	1264.03	− 0.30
27	1283.66	1285.24	284	0.12	1289.58	1287.96	− 0.13
28	2528.64	2517.25	430	− 0.45	2552.85	2519.87	− 1.29
29	2184.59	2151.68	315	− 1.51	2191.29	2157.40	− 1.55
30	1780.54	1755.89	367	− 1.38	1791.33	1760.03	− 1.75
31	2232.71	2171.60	449	− 2.74	2240.35	2196.94	− 1.94
32	2221.66	2191.58	292	− 1.35	2236.19	2199.60	− 1.64
33	2205.34	2175.85	408	− 1.34	2218.42	2188.53	− 1.35
34	1150.81	1140.83	468	− 0.87	1159.73	1148.94	− 0.93
35	1350.91	1340.41	470	− 0.78	1356.18	1352.92	− 0.24
36	1702.33	1679.27	298	− 1.35	1713.10	1681.37	− 1.85
Average				− 0.72			− 0.82

Table 3
Comparison between the ACO algorithm and our MS-BR algorithm for 2|UR|L class 3

Class 3							
	BEST10				AVG10		
Instance	ACO	MS-BR	MS-BR Time (seconds)	Gap (%)	ACO	MS-BR	Gap (%)
1	284.23	282.95	2	– 0.45	284.46	284.13	– 0.12
2	352.16	352.16	2	0.00	352.16	352.16	0.00
3	390.55	385.32	27	– 1.34	393.47	385.59	– 2.00
4	430.88	430.88	1	0.00	430.89	430.88	0.00
5	379.94	379.94	36	0.00	379.94	379.94	0.00
6	498.16	498.16	8	0.00	498.34	498.16	– 0.04
7	678.75	664.96	16	– 2.03	678.75	664.96	– 2.03
8	738.43	738.43	9	0.00	739.77	739.33	– 0.06
9	607.65	607.65	14	0.00	607.65	607.65	0.00
10	615.68	615.36	243	– 0.05	618.47	615.36	– 0.50
11	706.94	699.35	59	– 1.07	706.94	699.35	– 1.07
12	610	610.00	17	0.00	613.05	610.00	– 0.50
13	2450.19	2377.39	196	– 2.97	2450.41	2379.83	– 2.88
14	996.11	988.79	43	– 0.73	996.26	988.79	– 0.75
15	1145.04	1120.75	115	– 2.12	1157.62	1131.85	– 2.23
16	698.61	698.61	17	0.00	698.60	698.61	0.00
17	862.62	861.79	8	– 0.10	862.62	861.87	– 0.09
18	1025.35	986.30	205	– 3.81	1028.90	999.27	– 2.88
19	753.66	752.06	199	– 0.21	753.84	752.54	– 0.17
20	517.61	511.46	132	– 1.19	518.13	511.49	– 1.28
21	1114.16	1089.75	235	– 2.19	1116.61	1092.95	– 2.12
22	1046.71	1031.79	373	– 1.43	1051.15	1035.09	– 1.53
23	1068.63	1056.56	491	– 1.13	1080.73	1059.81	– 1.94
24	1082.3	1073.01	310	– 0.86	1087.93	1075.63	– 1.13
25	1355.61	1353.90	419	– 0.13	1367.77	1358.46	– 0.68
26	1344.32	1335.80	189	– 0.63	1356.53	1338.33	– 1.34
27	1376.34	1354.76	279	– 1.57	1381.89	1358.21	– 1.71
28	2604.08	2587.25	214	– 0.65	2668.02	2596.13	– 2.69
29	2090.56	2067.69	346	– 1.09	2108.62	2071.84	– 1.74
30	1811.22	1812.72	338	0.08	1830.30	1817.13	– 0.72
31	2276.01	2246.54	496	– 1.29	2285.01	2262.95	– 0.97
32	2247.06	2219.26	438	– 1.24	2262.68	2231.10	– 1.40
33	2355.08	2325.36	492	– 1.26	2367.58	2335.06	– 1.37
34	1204.31	1176.71	469	– 2.29	1210.49	1190.25	– 1.67
35	1439.13	1437.30	479	– 0.13	1445.87	1443.10	– 0.19
36	1791.54	1739.36	226	– 2.91	1801.23	1759.92	– 2.29
Average				– 0.97			– 1.11

Table 4
Comparison between the ACO algorithm and our MS-BR algorithm for 2|UR|L class 4

Class 4							
	BEST10				AVG10		
Instance	ACO	MS-BR	MS-BR Time (seconds)	Gap (%)	ACO	MS-BR	Gap (%)
1	282.95	282.95	3	0.00	282.95	282.95	0.00
2	342.00	334.96	1	– 2.06	342.00	334.96	– 2.06
3	362.41	358.40	25	– 1.11	362.41	358.80	– 1.00
4	447.37	447.37	3	0.00	447.37	447.37	0.00
5	383.88	383.87	18	0.00	383.88	383.87	0.00
6	498.32	498.32	6	0.00	498.35	498.32	– 0.01
7	702.45	686.26	16	– 2.31	702.45	686.26	– 2.31
8	692.47	688.32	24	– 0.60	692.47	691.64	– 0.12
9	625.13	625.10	110	– 0.01	625.80	625.10	– 0.11
10	703.63	703.64	90	0.00	704.37	703.64	– 0.10
11	782.31	773.58	143	– 1.12	784.13	777.14	– 0.89
12	614.24	614.23	10	0.00	614.73	614.23	– 0.08
13	2583.27	2533.79	150	– 1.92	2584.24	2539.44	– 1.73
14	981.90	981.00	335	– 0.09	981.90	981.30	– 0.06
15	1216.14	1164.77	358	– 4.22	1237.80	1168.60	– 5.59
16	703.35	703.35	36	0.00	703.35	703.35	0.00
17	861.79	861.79	10	0.00	862.45	861.79	– 0.08
18	1110.48	1100.66	230	– 0.88	1111.38	1107.94	– 0.31
19	772.05	765.51	364	– 0.85	784.70	770.67	– 1.79
20	546.91	534.14	367	– 2.34	547.69	541.98	– 1.04
21	976.48	967.85	370	– 0.88	983.11	974.82	– 0.84
22	1057.15	1052.60	180	– 0.43	1062.19	1054.61	– 0.71
23	1079.63	1064.76	359	– 1.38	1083.51	1070.09	– 1.24
24	1103.28	1099.40	94	– 0.35	1108.19	1101.66	– 0.59
25	1408.64	1402.08	398	– 0.47	1415.11	1404.41	– 0.76
26	1414.28	1391.02	311	– 1.64	1420.62	1404.28	– 1.15
27	1318.93	1318.45	437	– 0.04	1329.98	1321.84	– 0.61
28	2638.07	2647.15	476	0.34	2667.34	2657.68	– 0.36
29	2267.37	2274.09	530	0.30	2286.84	2276.78	– 0.44
30	1834.68	1851.15	291	0.90	1851.25	1855.20	0.21
31	2385.63	2387.72	479	0.09	2405.59	2399.87	– 0.24
32	2268.67	2267.57	419	– 0.05	2282.81	2283.86	0.05
33	2393.01	2387.22	445	– 0.24	2407.72	2394.90	– 0.53
34	1208.19	1210.66	475	0.20	1215.12	1215.72	0.05
35	1503.42	1519.28	456	1.05	1507.66	1528.16	1.36
36	1683.25	1670.84	492	– 0.74	1691.06	1678.19	– 0.76
Average				– 0.58			– 0.66

Table 5
Comparison between the ACO algorithm and our MS-BR algorithm for 2|UR|L class 5

Class 5							
	BEST10				AVG10		
Instance	ACO	MS-BR	MS-BR Time (seconds)	Gap (%)	ACO	MS-BR	Gap (%)
1	278.73	278.73	6	0.00	278.91	278.76	− 0.06
2	334.96	334.96	0	0.00	334.96	334.96	0.00
3	358.4	358.40	20	0.00	358.40	358.40	0.00
4	430.88	430.88	2	0.00	430.89	430.88	0.00
5	375.28	375.28	52	0.00	375.28	375.28	0.00
6	495.85	495.85	0	0.00	495.85	495.85	0.00
7	657.77	657.77	39	0.00	657.77	657.77	0.00
8	609.9	609.90	31	0.00	609.90	609.90	0.00
9	607.65	607.65	3	0.00	607.65	607.65	0.00
10	680.26	684.17	145	0.57	685.53	688.31	0.41
11	624.82	624.82	153	0.00	624.82	628.41	0.57
12	610.23	610.00	23	− 0.04	613.43	610.09	− 0.54
13	2334.78	2334.78	6	0.00	2334.78	2360.91	1.12
14	889.2	875.07	216	− 1.59	911.42	903.05	− 0.92
15	1160.2	1160.96	124	0.07	1168.15	1188.52	1.74
16	698.61	698.61	5	0.00	698.60	698.61	0.00
17	861.79	861.79	11	0.00	862.02	861.79	− 0.03
18	924.04	921.29	153	− 0.30	924.07	923.50	− 0.06
19	651.97	644.59	389	− 1.13	651.97	647.34	− 0.71
20	477.32	472.77	214	− 0.95	478.17	476.45	− 0.36
21	888.26	886.04	347	− 0.25	891.58	896.26	0.52
22	942.06	945.92	482	0.41	945.25	947.88	0.28
23	942.8	938.25	432	− 0.48	948.08	939.97	− 0.86
24	1048.33	1046.84	178	− 0.14	1051.74	1049.37	− 0.23
25	1170.38	1168.87	383	− 0.13	1174.13	1180.24	0.52
26	1231.72	1220.83	398	− 0.88	1237.27	1237.32	0.00
27	1260.11	1258.12	448	− 0.16	1263.79	1263.25	− 0.04
28	2336.45	2322.37	484	− 0.60	2358.04	2326.80	− 1.32
29	2158.78	2152.26	480	− 0.30	2168.62	2159.34	− 0.43
30	1542.14	1548.29	355	0.40	1551.63	1556.47	0.31
31	2016.59	2011.88	467	− 0.23	2030.79	2026.38	− 0.22
32	1983.34	1992.03	427	0.44	1993.23	2005.01	0.59
33	2002.72	2001.26	493	− 0.07	2013.71	2011.61	− 0.10
34	1036.16	1040.78	420	0.45	1043.92	1044.87	0.09
35	1256.34	1271.21	482	1.18	1263.76	1275.51	0.93
36	1505.54	1522.73	485	1.14	1522.62	1528.73	0.40
Average				− 0.07			0.04

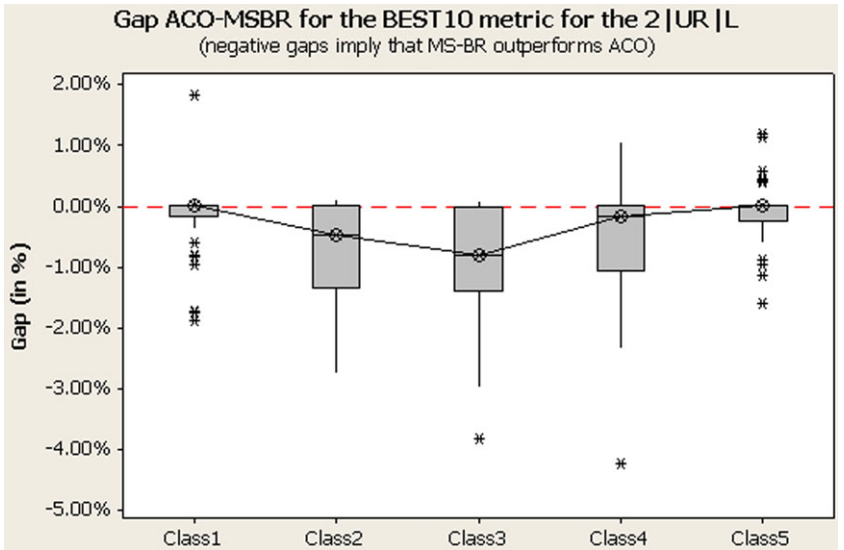

Fig. 6. Multiple boxplot comparing performance of ACO and MSBR algorithms for the 2/UR/L.

Table 6
Comparison among several approaches for 2|UO|L class 1

2|UO|L—class 1—best solutions												
	EGTS (1)		SA (2)		ACO (3)		GRASP × ELS (4)		MS-BR (5)			
Instance	Value	Gap (1)–(6) (%)	Value	Gap (2)–(6) (%)	Value	Gap (3)–(6) (%)	Value	Gap (4)–(6) (%)	Value	Time (seconds)	Gap (5) and (6) (%)	BKS (6)
1	278.73	0.00	278.73	0.00	278.73	0.00	278.73	0.00	278.73	0	0.00	278.73
2	334.96	0.00	334.96	0.00	334.96	0.00	334.96	0.00	334.96	0	0.00	334.96
8	568.56	0.00	568.56	0.00	568.56	0.00	568.56	0.00	568.56	0	0.00	568.56
12	610.00	0.00	610.00	0.00	610.00	0.00	610.00	0.00	610.00	4	0.00	610.00
14	837.67	0.00	837.67	0.00	837.67	0.00	837.67	0.00	837.67	1	0.00	837.67
17	861.79	0.00	861.79	0.00	861.79	0.00	861.79	0.00	861.79	39	0.00	861.79
21	687.80	0.03	693.56	0.87	690.20	0.38	687.60	0.00	687.60	11	0.00	687.60
26	819.56	0.00	819.56	0.00	819.56	0.00	819.56	0.00	819.56	1	0.00	819.56
36	586.58	0.00	603.69	2.92	616.69	5.13	592.87	1.07	612.99	469	4.50	586.58
Average		0.00		0.42		0.61		0.12			0.50	

Table 7
Comparison among several approaches for 2|UO|L class 2

2|UO|L—class 2—best solutions												
	EGTS (1)		SA (2)		ACO (3)		GRASP × ELS (4)		MS-BR (5)			
Instance	Value	Gap (1)–(6) (%)	Value	Gap (2)–(6) (%)	Value	Gap (3)–(6) (%)	Value	Gap (4)–(6) (%)	Value	Time (seconds)	Gap (5) and (6) (%)	BKS (6)
1	304.79	9.35	285.50	2.43	284.52	2.08	284.42	2.04	278.73	3	0.00	278.73
2	334.96	0.00	334.96	0.00	334.96	0.00	334.96	0.00	334.96	0	0.00	334.96
8	718.20	6.47	718.18	6.47	709.39	5.16	674.55	0.00	674.55	3	0.00	674.55
12	629.42	3.09	621.28	1.75	610.57	0.00	610.57	0.00	610.57	27	0.00	610.57
14	1116.43	7.55	1078.10	3.85	1038.68	0.06	1038.09	0.00	1038.77	140	0.07	1038.09
17	863.66	0.00	870.86	0.83	870.86	0.83	870.86	0.83	870.86	63	0.83	863.66
21	1060.04	6.77	1050.05	5.76	1013.49	2.08	992.83	0.00	998.78	58	0.60	992.83
26	1346.82	4.81	1330.00	3.50	1298.02	1.01	1285.01	0.00	1291.38	172	0.50	1285.01
36	1832.73	6.11	1776.71	2.87	1787.01	3.46	1782.99	3.23	1727.21	311	0.00	1727.21
Average		4.90		3.05		1.63		0.68			0.22	

Table 8
Comparison among several approaches for 2|UO|L class 3

2|UO|L—class 3—best solutions												
	EGTS (1)		SA (2)		ACO (3)		GRASP × ELS (4)		MS-BR (5)			
Instance	Value	Gap (1)–(6) (%)	Value	Gap (2)–(6) (%)	Value	Gap (3)–(6) (%)	Value	Gap (4)–(6) (%)	Value	Time (seconds)	Gap (5) and (6) (%)	BKS (6)
1	299.70	5.44	299.70	5.44	296.87	4.45	284.52	0.10	284.23	3	0.00	284.23
2	355.65	0.99	353.48	0.37	352.16	0.00	352.16	0.00	352.16	1	0.00	352.16
8	748.83	1.41	748.83	1.41	740.85	0.33	738.43	0.00	738.43	53	0.00	738.43
12	611.99	0.33	610.00	0.00	610.00	0.00	610.00	0.00	610.00	7	0.00	610.00
14	1087.16	9.31	1037.83	4.35	1018.75	2.43	996.25	0.16	994.61	119	0.00	994.61
17	861.79	0.00	861.79	0.00	861.79	0.00	861.79	0.00	861.79	39	0.00	861.79
21	1183.91	5.64	1174.10	4.77	1148.02	2.44	1121.84	0.10	1120.68	132	0.00	1120.68
26	1405.45	4.52	1411.41	4.96	1384.75	2.98	1344.66	0.00	1378.63	202	2.53	1344.66
36	1906.69	5.49	1906.05	5.45	1891.90	4.67	1834.97	1.52	1807.51	316	0.00	1807.51
Average		3.68		2.97		1.92		0.21			0.28	

Table 9
Comparison among several approaches for 2|UO|L class 4

2|UO|L—class 4—best solutions												
	EGTS (1)		SA (2)		ACO (3)		GRASP × ELS (4)		MS-BR (5)			
Instance	Value	Gap (1)–(6) (%)	Value	Gap (2)–(6) (%)	Value	Gap (3)–(6) (%)	Value	Gap (4)–(6) (%)	Value	Time (seconds)	Gap (5) and (6) (%)	(6) (6)
1	296.75	4.88	296.75	4.88	282.95	0.00	282.95	0.00	282.95	1	0.00	282.95
2	342.00	2.10	342.00	2.10	342.00	2.10	334.96	0.00	334.96	0	0.00	334.96
8	718.18	3.71	723.65	4.50	692.47	0.00	692.47	0.00	692.47	16	0.00	692.47
12	618.23	0.65	614.24	0.00	614.24	0.00	614.23	0.00	614.23	4	0.00	614.23
14	992.83	1.16	986.84	0.55	985.01	0.37	981.90	0.05	981.42	26	0.00	981.42
17	863.27	0.17	861.79	0.00	861.79	0.00	861.79	0.00	861.79	29	0.00	861.79
21	1008.15	3.51	1012.80	3.99	1001.14	2.79	978.82	0.50	973.96	151	0.00	973.96
26	1487.09	5.80	1468.02	4.44	1451.71	3.28	1405.57	0.00	1409.42	294	0.27	1405.57
36	1759.36	3.47	1764.13	3.75	1771.31	4.17	1728.69	1.67	1700.35	492	0.00	1700.35
Average		2.83		2.69		1.41		0.25			0.03	

Table 10
Comparison among several approaches for 2|UO|L class 5

2|UO|L—class 5—best solutions												
	EGTS (1)		SA (2)		ACO (3)		GRASP × ELS (4)		MS-BR (5)			
Instance	Value	Gap (1)–(6) (%)	Value	Gap (2)–(6) (%)	Value	Gap (3)–(6) (%)	Value	Gap (4)–(6) (%)	Value	Time (seconds)	Gap (5) and (6) (%)	BKS (6)
1	280.60	0.67	278.73	0.00	278.73	0.00	278.73	0.00	278.73	2	0.00	278.73
2	334.96	0.00	334.96	0.00	334.96	0.00	334.96	0.00	334.96	0	0.00	334.96
8	621.86	1.96	621.85	1.96	621.85	1.96	609.90	0.00	609.90	11	0.00	609.90
12	610.23	0.04	610.23	0.04	610.23	0.04	610.23	0.04	610.00	3	0.00	610.00
14	925.56	0.45	925.56	0.45	922.58	0.12	921.45	0.00	921.44	38	0.00	921.44
17	861.79	0.00	861.79	0.00	861.79	0.00	861.79	0.00	861.79	37	0.00	861.79
21	903.20	2.07	921.66	4.16	897.55	1.44	884.84	0.00	891.18	251	0.72	884.84
26	1256.59	1.80	1266.66	2.61	1250.41	1.30	1234.39	0.00	1234.39	270	0.00	1234.39
36	1549.28	0.66	1575.60	2.37	1570.81	2.06	1572.49	2.17	1539.05	495	0.00	1539.05
Average		0.85		1.29		0.77		0.25			0.08	

an algorithm. Moreover, as noticed by Luke (2011), Java facilitates duplicability of experimental results. As shown in Tables 6–10, our approach is able to provide competitive results, in particular for classes 2, 4, and 5, where it obtains the lowest average gaps with respect to the best-known solutions (BKS).

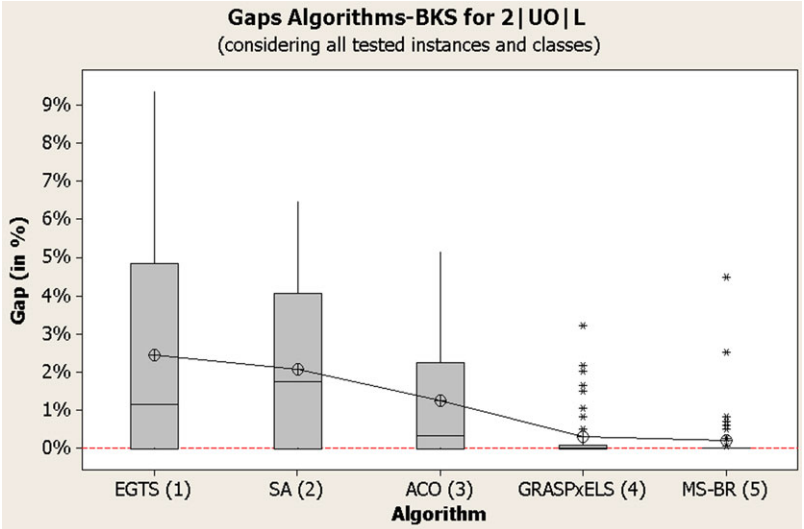

Fig. 7. Multiple box-plot comparing performance of different algorithms for the 2/UO/L.

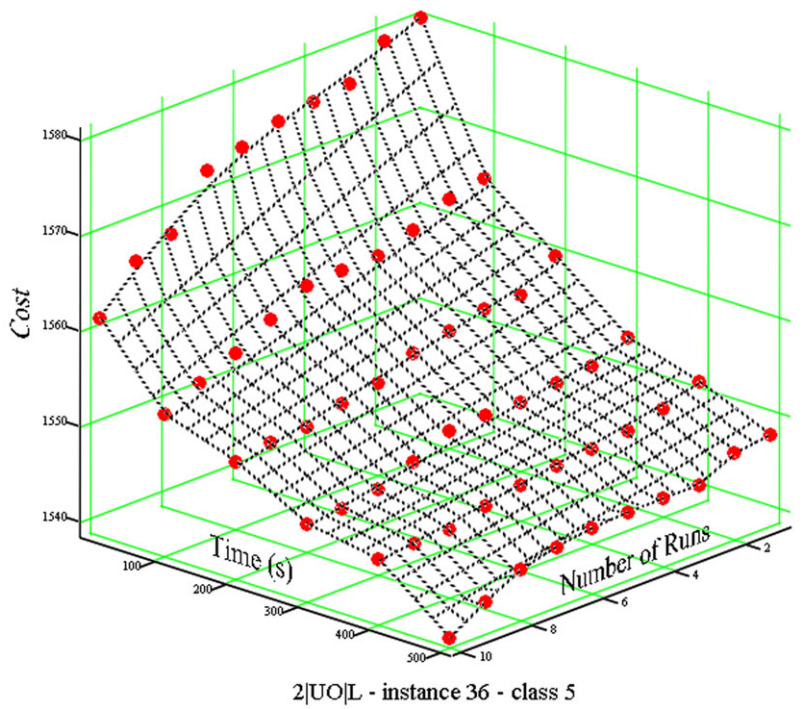

Fig. 8. Evolution of solutions quality as more time and runs are employed for 2|UO|L—instance 36–class 5.

Finally, Fig. 7 shows a comparison among the different algorithms considered in this section. All selected instances and classes have been considered in this plot. Note that our MS-BR algorithm is the one offering the best average performance and also the best results for most instances. The GRASP × ELS algorithm by Duhamel et al. (2011) offers very competitive results as well using

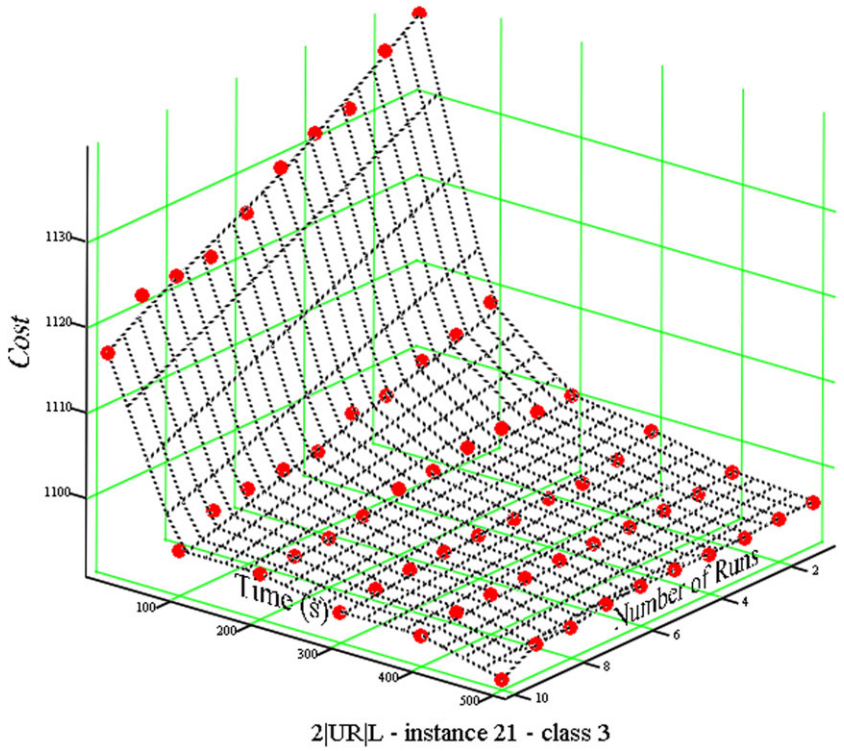

Fig. 9. Evolution of solutions quality as more time and runs are employed for 2|UR|L—instance 21–class 3.

similar CPUs and execution times—GRASP × ELS was run for a maximum of 1.5 hours for each replica in an Opteron at 2.1 GHz, while MS-BR was run for a maximum of 500 seconds for each replica in a notebook with an Intel Core 2 at 2.4 GHz.

6.3. Evolution of solutions quality with time and number of runs

Finally, Figs 8 and 9 show two examples of how the quality of the solution provided by our algorithm evolves as more time and number of runs are considered. These examples correspond to two different instances: (a) instance 36–class 5 of the 2|UO|L and (b) instance 21–class 3 of the 2|UR|L. Note that in both cases the quality of the solution improves with time, from 10 to 500 seconds, in a steady manner. Also, the quality of the solution improves as the number of runs increases, from 1 to 10. Even for small computing time (e.g., 10 seconds), it is possible to attain good-quality solutions by increasing the number of runs, which supports the idea that our approach can benefit from computing parallelization techniques to provide "good" solutions in "real-time."

7. Conclusions

This paper focuses on the 2L-CVRP, a relatively new and promising research area that combines vehicle routing and vehicle packing problems. The combination of these two classical problems is found in practical applications of some real-world transportation activities. Instead of offering a sequential two-stage approach to solve the 2L-CVRP, we develop an integrated approach that can consider both routing and packing costs simultaneously to better support the decision-making process. We also consider the possibility of rotating the items while they are being loaded into the truck, a realistic assumption that has been seldom considered in the existing literature.

The algorithm presented here is not only efficient but also flexible—modular and easily adaptable to new constraints—it employs just four parameters, and it is easily parallelizable. These properties make it an interesting alternative to other existing and more complex approaches. A set of standard benchmarks has been used to test our algorithm. The computational results obtained show that our approach is equal to and even outperforms other state-of-the-art approaches using similar CPUs and computing times, both for the scenario in which items rotation is allowed as well as for the scenario without this possibility.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Science and Innovation (TRA2010–21644-C03) and Ibero-American Programme for Science, Technology and Development (CYTED2010–511RT0419), in the context of the IN3-HAROSA Network (http://dpcs.uoc.edu). Similarly, we appreciate the financial support of the Sustainable TransMET Network funded by the Government of Navarre (Spain) inside the Jerónimo de Ayanz program.

References

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

Clarke, G., Wright, J.W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* 12, 568–581.

Cordeau, J.F., Gendreau, M., Hertz, A., Laporte, G., Sormany, J.S., 2005. New heuristics for the vehicle routing problem. In Langevin, A., Riopel, D. (eds) *Logistics Systems: Design and Optimization*. Kluwer, Boston, MA.

Duhamel, C., Lacomme, P., Quilliot, A., Toussaint, H., 2009. 2L-CVRP: a GRASP resolution scheme based on RCPSP. International Conference on Computers & Industrial Engineering, 2–9 July 2009, Conference Publications, pp. 1094–1099.

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, 3, 617–640.

Fuellerer, G., Doerner, K., Hartl, R., Iori, M., 2009. Ant colony optimization for the two-dimensional loading vehicle routing problem. *Computers and Operations Research* 36, 655–673.

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.

Golden, B., Raghavan, S., Wasil, E. (eds), 2008. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, New York.

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

Iori, M., Martello, S., 2010. Routing problems with loading constraints. *TOP* 18, 4–27.
Iori, M., Salazar, J.J., Vigo, D. 2007. An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science* 41, 2, 253–264.
Juan, A., Faulin, J., Ferrer, A., Lourenço, H., Barrios, B., 2011a. MIRHA: multi-start biased randomization of heuristics with adaptive local search for solving non-smooth routing problems. *TOP* 21, 1, 109–132.
Juan, A., Faulin, J., Jorba, J., Riera, D., Masip, D., Barrios, B., 2011b. On the use of Monte Carlo simulation, cache and splitting techniques to improve the Clarke and Wright saving heuristics. *Journal of the Operational Research Society* 62, 6, 1085–1097.
Juan, A., Faulin, J., Ruiz, R., Barrios, B., Caballé, S., 2010. The SR-GCWS hybrid algorithm for solving the capacitated vehicle routing. *Applied Soft Computing* 10, 215–224.
Laporte, G., 1992. The vehicle routing problem: an overview of exact and approximate algorithms. *European Journal of Operational Research* 59, 345–358.
L'Ecuyer, P., Meliani, L., Vaucher, J., 2002. SSJ: a framework for stochastic simulation in Java. Proceedings of the 2002 Winter Simulation Conference, ACM, New York, pp. 234–242.
Leung, S.C.H., Zheng, J., Zhang, D., Zhou, X., 2010. Simulated annealing for the vehicle routing problem with two-dimensional loading constraints. *Flexible Services and Manufacturing Journal* 22, 1–2, 61–82.
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.
Luke, S., 2011. *Essential of Metaheuristics*. Lulu, Raleigh, NC. Available at http://cs.gmu.edu/∼sean/book/metaheuristics/ (accessed 7 January 2014).
Prins, C., 2009. A GRASP × Evolutionary local search hybrid for the vehicle routing problem. In Pereira, F.B., Tavares, J. (ed.) *Bio-Inspired Algorithms for the Vehicle Routing Problem, Studies in Computational Intelligence*, Vol. 161. Springer, Berlin, pp. 35–53.
Riff, M.C., Bonnaire, X., Neveu, B., 2009. A revision of recent approaches for two-dimensional strip-packing problems. *Engineering Applications of Artificial Intelligence* 198, 823–827.
Toth, P., Vigo, D., 2002. *The Vehicle Routing Problem*. SIAM Publishers, Philadelphia, PA.
Wang, F., Tao, Y., Shi, N., 2009. A survey on vehicle routing problem with loading constraints. *International Joint Conference on Computational Sciences and Optimization* 2, 602–606.
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.