Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, Manufacturing and Logistics

Metaheuristics for vehicle routing problems with three-dimensional loading constraints

Guenther Fuellerer [a], Karl F. Doerner [a,*], Richard F. Hartl [a], Manuel Iori [b]

[a] Department of Business Administration, University of Vienna, Bruenner Strasse 72, 1210 Vienna, Austria
[b] DISMI, University of Modena and Reggio Emilia, Via Amendola 2, 42100 Reggio Emilia, Italy

ARTICLE INFO

Article history:
Received 3 December 2007
Accepted 28 March 2009
Available online 5 April 2009

Keywords:
Vehicle routing
Ant Colony Optimization
Three-dimensional packing

ABSTRACT

This paper addresses an important combination of three-dimensional loading and vehicle routing, known as the Three-Dimensional Loading Capacitated Vehicle Routing Problem. The problem calls for the combined optimization of the loading of freight into vehicles and the routing of vehicles along a road network, with the aim of serving customers with minimum traveling cost. Despite its clear practical relevance in freight distribution, the literature on this problem is very limited. This is because of its high combinatorial complexity.

We solve the problem by means of an Ant Colony Optimization algorithm, which makes use of fast packing heuristics for the loading. The algorithm combines two different heuristic information measures, one for routing and one for packing. In numerical tests all publicly available test instances are solved, and for almost all instances new best solutions are found.

1. Introduction

The *Three-Dimensional Loading Capacitated Vehicle Routing Problem* (3L-CVRP) is a highly complex problem combining three-dimensional loading and vehicle routing. The problem was introduced in Gendreau et al. [20] and calls for the determination of the routes travelled by a vehicle fleet for delivering items to customers, while minimizing the total travel cost. Items consist of rectangular boxes of given size and weight, and must be feasibly loaded within the vehicles before they are shipped.

The problem is of practical interest in freight distribution, because of its many real-world transportation applications. It is particularly relevant for those cases where shippers have to deal with large items and loading is not trivial. Examples are the distributions of household appliances, kitchen components, mechanical components and others. The first paper on the 3L-CVRP was indeed motivated by a furniture distribution problem [20].

The problem is also of theoretical interest because it generalizes two of the most well known problems in combinatorial optimization: the *Capacitated Vehicle Routing Problem* (CVRP), and the *Three-dimensional Bin Packing Problem* (3BPP).

The CVRP calls for the determination of the set of routes of minimal cost, to be travelled by a set of vehicles so as to deliver a given quantity of freight to each customer. The loading is not taken explicitly into account, except checking that, for each vehicle, the total weight of the loaded freight does not exceed the given vehicle weight capacity. The literature on the problem is very wide. For recent surveys we refer the reader to Toth and Vigo [31] and Cordeau et al. [10]. Other contributions especially focused on heuristics and metaheuristics are Cordeau and Laporte [9] and Cordeau et al. [7].

The 3BPP calls for packing a given set of three-dimensional rectangular items into the minimum number of three-dimensional rectangular boxes (bins), while ensuring that items do not overlap and are completely contained by the bins. The 3BPP is particularly difficult and, despite some decades of research, several instances with less than 50 items cannot be solved to optimality (see Martello et al. [25,26]). For recent and very effective metaheuristics on the 3BPP we refer the reader to Faroe et al. [16] and Crainic et al. [11,12]. All these approaches solve the most common version of the 3BPP, in which orthogonal packing is required (items must be packed with their edges parallel to the edges of the bin).

Concerning the practical relevance of the 3L-CVRP, the seminal paper [20] was motivated by a particularly difficult loading problem, with items of high cost, high risk of being damaged during transportation and very different shapes (e.g., chests of drawers, night tables, beds, wardrobes and accessories). Other loading and routing problems have been addressed very recently. Doerner et al. [14] studied the delivery of timber chipboards. The chipboards had to be placed on three different piles contained in particular vehicles, where the opening for the loading/unloading operations was placed on one side. The problem was solved by

* Corresponding author. Tel.: +43 1 4277 38113; fax: +43 1 4277 38094.
E-mail addresses: guenther@fuellerer.info (G. Fuellerer), Karl.Doerner@univie.ac.at (K.F. Doerner), Richard.Hartl@univie.ac.at (R.F. Hartl), manuel.iori@ unimore.it (M. Iori).

an Ant Colony Optimization (ACO) algorithm and by Tabu Search (TS), both using fast approximation algorithms for the loading. A work in progress related to the 3L-CVRP is presented by Pisinger (http://www.diku.dk/~pisinger/), who addresses the problem of loading and delivering sofas. In this problem the packing may result in complicated structures due to the sofas having non-convex shapes.

The 3L-CVRP generalizes another loading and routing problem known as the *Two-Dimensional Loading Capacitated Vehicle Routing Problem* (2L-CVRP). In the 2L-CVRP items have a two-dimensional rectangular shape, and vehicle have a two-dimensional rectangular space for loading items. The problem is of interest for optimizing the distribution of those items that may not be stacked one over the other (e.g., refrigerators, pallets as high as the vehicle, big and heavy boxes). The problem was introduced by Iori et al. [22], who solved it in an exact way by means of a branch and cut algorithm, making use of a nested branch and bound algorithm for the loading subproblem. It was then addressed with metaheuristics by Gendreau et al. [21] and Zachariadis et al. [32], through TS, and by Fuellerer et al. [18], through ACO. This paper generalizes the results obtained in [18] for the 2L-CVRP to the case of 3L-CVRP, where loading is much more complex and challenging, and real-world constraints have to be carefully handled.

Some first papers combining loading with routing or pickup and delivery have been presented very recently. In the *Traveling Salesman Problem with Pickup and Delivery and LIFO Loading* (TSPPDL) a single vehicle has to first pickup and deliver freight to customers, by ensuring that a *Last-In-Fist-Out* (LIFO) policy is respected in loading and unloading of cargo. This implies that the vehicle may be seen as a single stack, being rear-loaded and rear-unloaded. The TSPPDL was addressed by using variable neighborhood search by Carrabs et al. [4] and by means of a branch and cut algorithm by Cordeau et al. [8]. A more complex problem has been presented by Pettersen [27] and defined as the double TSP with multiple stacks. In this case the vehicle is formed by three stacks and again a LIFO policy must be fulfilled on each of the stacks. In [27] a mathematical model and two metaheuristics, based on TS and simulated annealing paradigms, are described.

The problem of loading a three-dimensional container is, from a practical point of view, more complex than the standard 3BPP. Additional constraints may arise from the nature of the items, from the cargo stability and from practical considerations in transportation (see, e.g., Bortfeldt and Gehring [2] and Pisinger [28] for a discussion on container loading problems). In the 3L-CVRP the total weight of the items must not exceed the vehicle weight capacity and all items required by a customer must be placed on the same vehicle (no split deliveries). In addition to the orthogonal loading, items have a fixed top, while 90° rotations are allowed on the horizontal plane. Some of the items may be fragile, and in this case it is requested that no non-fragile item is placed over a fragile one. When an item is placed on top of other items, its base area must be supported totally or partially by the other items. Finally, the loading of each vehicle should allow an easy unloading, i.e., it should be possible to unload the items of a customer without shifting in the cargo items requested by other customers.

A simple example of a 3L-CVRP instance is given in Fig. 1, in which eight customers demand for a total of 15 weighted items, and must be served through vehicles based at a central depot. In the example, fragile items are depicted in gray color. The sum of the weights of the items demanded by each customer (indicated by d_i for a given customer i) should not exceed the maximum weight vehicle capacity $D = 100$. A possible solution formed by three routes is also depicted. In Fig. 2 the feasible loadings associated to each route of Fig. 1 are shown. No non-fragile item is placed over a fragile item, and each item basis is supported, partially or totally, by other items or by the vehicle surface. It is also easy to check how unloading operations may be performed for each customer along the l axis.

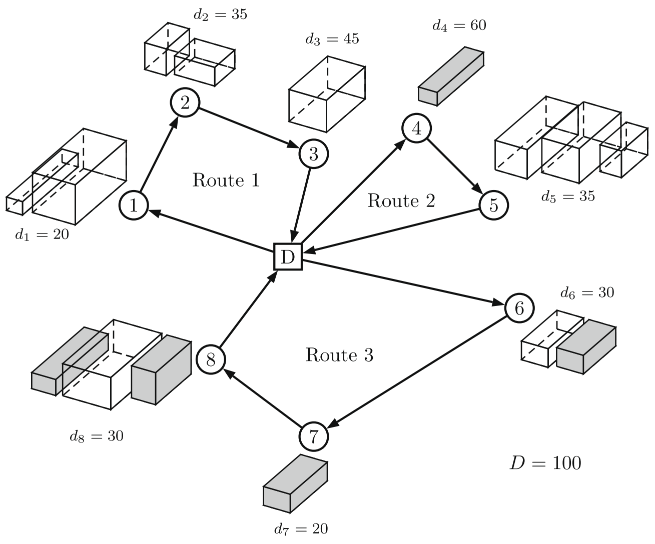

Fig. 1. A simple 3L-CVRP instance.

In this paper we propose an ACO algorithm for the 3L-CVRP. This is motivated by the excellent results that this metaheuristic paradigm obtained on both classical routing problems (see, e.g., Reimann et al. [29,30]) and loading and routing problems (see Doerner et al. [14] and Fuellerer et al. [18]). Our aim is to provide high quality solutions with reasonable computational effort, hence developing an algorithm capable of dealing with large-size instances, and being possibly useful for real world distribution. We do this by carefully merging and tailoring to the problem techniques from the literature. The contribution of this paper is twofold. First, we provide very good results on all publicly available test instances improving previous approaches on average by more than 6%, and in some cases even by more than 17%. Second, we show that these results can be obtained using very simple but fast packing heuristics. It seems to be better to consider some simple packing information in the construction of the solutions than using highly complicated packing heuristics. This is in contrast to our results in the two-dimensional case [18], where apart from the heuristics also a truncated branch and bound algorithm is used.

The paper is organized as follows. A more formal description of the 3L-CVRP and of its loading constraints is given in Section 2. The ACO algorithm is described in Section 3 and is computationally evaluated in Section 4. Some conclusions are finally drawn in Section 6.

2. Problem description

Let $G = (V, E)$ be a complete graph, where $V = \{0, 1, \ldots, n\}$ is a set of $n + 1$ vertices and E the complete set of edges connecting each vertex pair. Vertex 0 corresponds to the depot, while vertices $\{1, \ldots, n\}$ are the n customers to be served. Each edge is denoted by (i,j) and has an associated routing cost c_{ij} $(i,j = 0, \ldots, n)$. It is also given a fleet of v identical vehicles, each of which has a weight capacity D and a three-dimensional box-shaped loading space of width W, height H and length L. Each vehicle has an opening on the rear for the loading/unloading operations. We suppose the opening to be as large as the vehicle $(W \times H)$.

The demand of customer i consists of a set of m_i items whose total weight is d_i $(i = 1, \ldots, n)$. Each item k of customer i is denoted by I_{ik} and is a three-dimensional cuboid, having width w_{ik}, height h_{ik} and length l_{ik} $(i = 1, \ldots, n,\ k = 1, \ldots, m_i)$. The total volume

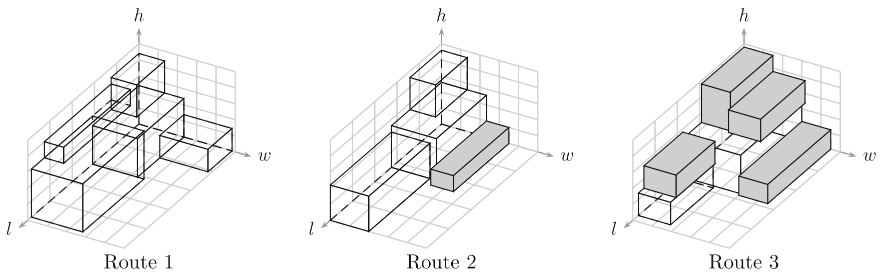

Fig. 2. Feasible items loadings for the routes of Fig. 1.

demanded by a customer i is denoted by $vol_i = \sum_{k=1}^{m_i} w_{ik}h_{ik}l_{ik}$ $(i = 1, \dots, n)$. The notation is based on Gendreau et al. [20].

The 3L-CVRP calls for finding a set of at most v routes, such that each route starts and ends at the depot, each customer is visited once (no split deliveries), a three-dimensional feasible loading is ensured for each vehicle and the total cost is minimized.

A three-dimensional loading is feasible if it does not exceed the vehicle weight capacity D and if there exists a placement of the items in the vehicle volume that satisfies both the classical 3BPP constraints (items do not overlap and are completely contained by the bins/vehicles) and a series of operational constraints. These additional constraints are derived from the nature of the items, the cargo stability, as well as practical policies in transportation and are as follows:

- The loading must be orthogonal, i.e., items must be loaded with their edges parallel to the sides of the vehicle. Items have a fixed orientation with respect to the height (i.e., as usual in transportation they have a fixed top), but they can be rotated by 90° on the w–l plane (see Fig. 2).
- Items are divided into two groups: fragile and non-fragile. A *fragility* flag f_{ik} is used, being equal to 1 if I_{ik} is fragile, and 0 otherwise $(i = 1, \dots, n,\ k = 1, \dots, m_i)$. Non-fragile items cannot be placed on top of fragile ones, although fragile items can be stacked on top of each other. Also non-fragile items can be stacked on top of each other.
- When an item I_{ik} is placed on top of other items, its base must be supported by a minimum *supporting area*. This means that the items that are placed under I_{ik}, and with their top touching directly the bottom of I_{ik}, should form a cumulative area $\overline{A} \geqslant a w_{ik} l_{ik}$, where $0 \leqslant a \leqslant 1$ is a given parameter representing the minimum fraction of the area of I_{ik} to be supported. Clearly, when an item is placed directly on the basis of the vehicle the minimum supporting area constraint is always satisfied.
- When a customer i is visited, it must be possible to unload all his/her items I_{ik} through a sequence of straight movements (one per item) parallel to the L-edge. In other words, no item demanded by a customer visited later may be placed on top of I_{ik} or between I_{ik} and the rear of the vehicle. This LIFO policy is a usual request in transportation and, following the definition introduced in Iori et al. [22], is denoted in the following as *sequential loading* constraint. Recalling the loading of route 3 in Fig. 3, all items of customer 6 can be unloaded without rearranging items from customer 7 or 8. After delivering the goods of customer 6 the item of customer 7 can be unloaded without rearranging any item of customer 8.

In the standard version of the 3L-CVRP, as denoted in Gendreau et al. [20], all these constraints must be fulfilled. By removing one or more constraints at a time, different loading and routing problems can be obtained. All these problems are of interest, since they represent different practical problems in transportation. Also, it is of interest to evaluate the solution value differences between one

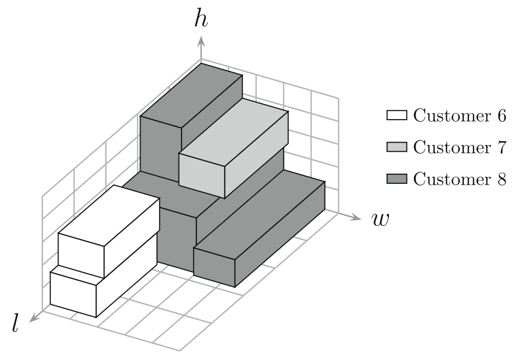

Fig. 3. Loading of route 3 from Fig. 1 with different colors for different customers.

loading configuration and another, so as to understand the cost implications of different constraints. Finally, note that other operational constraints, in addition to the ones described above, could be imposed on the vehicles, but this depends directly on the nature of the goods to be transported and on the policy of the shippers. For example, an additional restriction, not studied so far in the 3L-CVRP literature, is the one requiring a balanced weight distribution of the load into the container (see, e.g., Davies and Bischoff [13] and Eley [15]).

3. Solution methods

Since both, the three-dimensional loading problem and the CVRP are NP-hard problems, this is clearly also the case for the combined problem 3L-CVRP under consideration here. Hence, it cannot be expected that an exact algorithm will be able to solve this problem in reasonable time for realistic problem dimensions and heuristic solutions methods must be employed. In this paper, the routing is done by an ant based algorithm that reuses some successful features of the Savings-based ACO algorithm for the standard CVRP (see Reimann et al. [29,30]). The modifications of this Savings-based ACO are based on introducing an additional visibility measure related to packing. Besides using this extended visibility measure, another important difference to the standard Savings-based ACO algorithm is that the feasibility check is much more difficult now, because an NP-hard loading problem must be solved. In Section 3.1 we present the algorithms used for determining a feasible loading of a vehicle, iteratively invoked by the ACO during the search process. In Section 3.2 we briefly describe the classical Savings-based ACO for the CVRP. Finally in Section 3.3 we discuss how these algorithms have been adapted and merged, so as to solve the 3L-CVRP.

3.1. Algorithms for computing the loading of a vehicle

Given a route, we are interested in determining whether all the items demanded by the customers in the route can be loaded into a single vehicle, while satisfying the operational constraints defined in Section 2. The algorithms used to solve this problem must be flexible, since they should be adaptable to take into account all loading constraints or just a subset of them. They must also be fast, since they are called very often from the ACO algorithm.

To compute the potential infeasibility of a route, we first easily check if the weight constraint is not violated, and then apply the lower bounds for the 3BPP by Martello et al. [25]. If the lower bounds do not prove the infeasibility, then we repeatedly apply the two greedy heuristics developed by Gendreau et al. [20]. These heuristics place the items according to a given sequence, one at a time, into a container of width W, height H and infinite length. Their aim is to find a feasible loading of minimum length, which does not exceed the vehicle length L. By changing the sequence of the items and again invoking the heuristics one can obtain new solutions. This process is iterated θ times or until a feasible solution has been found.

If sequential loading is required, the first sequence is obtained by listing the customers in inverse order of visits, and then, for each customer, sorting the items by increasing value of the fragility flag (i.e., the fragile items, if any, are the last ones), breaking ties by decreasing volume. If sequential loading is not required, the complete set of items belonging to all customers in the route is sorted by increasing value of the fragility flag, breaking ties by decreasing volume. The sequence is then changed by switching the order of two items. In the case of sequential loading the switchings may occur only for items belonging to the same customer, otherwise all switchings are considered. The first switching is applied to the first two items in the sequence. Assuming an initial item list of (1–2–3–4–5) this would result in (2–1–3–4–5). The second switch is performed between the first and the third item yielding (3–2–1–4–5), the forth switch gives (4–2–3–1–5) and so on. The algorithm accepts a switch as soon as this improves the solution value.

The first heuristic generalizes the classical *bottom-left-fill* algorithm (see Baker et al. [1]) and packs the current item into the *normal position* (see Christofides and Whitlock [5], i.e., with its bottom edge touching either the bottom of the bin or the top edge of another item, with its left edge touching either the left edge of the bin or the right edge of another item and with its front edge touching either the front edge of the bin or the right front of another item), which has lowest width, breaking ties by lowest height, breaking ties by lowest length. For each position, both feasible orientations on the w–l plane are considered. The first packing satisfying the loading conditions is selected and the process is iterated until all items are packed or no feasible packing exists for an item.

The second heuristic generalizes the *touching perimeter* algorithm (see Lodi et al. [24]), and, among all the possible normal positions for the current item, selects the one maximizing the percentage of the item surface touching the container and other items already packed. In the following we denote this algorithm as the *touching-area* heuristic. In this case too, the process is iterated until all items are packed or no feasible packing exists for an item.

The TS heuristic used by Gendreau et al. [20] is not used here, since in our computational experiments this lead to only very limited improvements at the cost of higher run times. On the basis of computational evidence, the number of calls to the two heuristics was set to $\theta = 5\tilde{n}$, where $\tilde{n}$ is the number of customers in the route being checked. More considerations on the effectiveness and speed of the loading algorithms we implemented are given in details in Section 4.

We finally note that, if a balanced weight distribution is required, then one could try to modify the solution obtained by the heuristics above by shifting the items and modifying their positions in the container. A similar idea was indeed proposed by Davies and Bischoff [13] and by Eley [15] for the three-dimensional container loading problem. The heuristic procedures in [13] operate in a two-stage fashion: first, by using blocks of items they construct a solution satisfying all constraints but the weight distribution; then, they attempt to obtain a balanced distribution (i.e., a distribution whose center of gravity is as close as possible to the geometrical center of the container basis) through a mixture of blocks re-orderings and blocks reflections. The heuristics in [15] provide a simple yet effective generalization of the ones in [13], obtained by using a tree search phase.

3.2. Standard Savings-based Ant Colony Optimization

Our route building algorithm is based on the Savings-based ACO developed by Reimann et al. [29,30], which is explained in this subsection. It looks for high-quality feasible solutions by iteratively applying a randomized version of the classical Clarke and Wright savings algorithm (Clarke and Wright [6]) on a modified cost matrix and two simple local search procedures. The classical savings algorithm initially assigns each customer to a separate route, and then, for each pair (i,j) of customers, it evaluates the cost savings s_{ij} ($s_{ij} = c_{i0} + c_{0j} - c_{ij}$) by combining the two customers on the same route. The best feasible combination is chosen, and the procedure is re-iterated by merging the partial routes obtained at the previous step, until no more feasible cost savings exist.

The Savings-based ACO is initialized by a population of P ants. Each ant starts with the out and back (single customer) routes as initial solution, and then looks for improvements by sequentially choosing feasible entries from an available list Ω_{Π} of savings values. Such a list is composed of the Π feasible combinations (i,j) yielding the largest savings values. An entry in the list is chosen by applying a probabilistic rule, taking into account both the savings values and the pheromone information. For each edge (i,j) we keep a pheromone concentration τ_{ij}, representing how good the combination of these two customers was in the previous iterations. The pheromone values are all set to τ_0 in the first iteration, and are then modified according to the values of the solutions found by the P ants. The probability $\mathscr{P}_{ij}$ of choosing to combine customers i and j in one route is evaluated by defining:

$$\xi_{ij} = (\tau_{ij})^{\alpha}(s_{ij})^{\beta} \tag{1}$$

and

$$\mathscr{P}_{ij} = \frac{\xi_{ij}}{\sum_{(h,l)\in\Omega_{\Pi}} \xi_{hl}} \tag{2}$$

for each $(i,j) \in \Omega_{\Pi}$. Parameters α and β represent the relative influence of pheromone trails and savings values, respectively. A combination (i,j) is realized by using the probability in (2) and the two routes containing i and j are concatenated. After that, the Ω_{Π} list of best feasible savings is updated and the procedure is re-executed. When no more feasible savings exist, the iterative procedure is halted and the heuristic solution found is returned. A solution obtained through this procedure is then post-optimized through the *move* and *swap* neighborhoods (Kindervater and Savelsbergh [23]), where single customers are moved and swapped, respectively.

The process is performed for each of the P ants in the population. This corresponds to a complete iteration of the ACO. At the end of each complete iteration, the incumbent solution and the pheromone is updated according to the rule proposed by Bullnheimer et al. [3]: let $0 \leqslant \rho \leqslant 1$ be the trail persistence and F the number of ants leading to the best current solutions (elitist ants), the pheromone information is update as:

$$\tau_{ij} = \rho\tau_{ij} + \sum_{q=1}^{F-1} \Delta\tau_{ij}^{q} + \Delta\tau_{ij}^{*} \tag{3}$$

for $i,j = 0,\ldots,n$. The rule in (3) first updates the pheromone on the arcs belonging to the incumbent solution: the amount of pheromone laid by the best ant is $\Delta\tau_{ij}^{*} = F\varepsilon$ if arc (i,j) belongs to the incumbent solution, $\Delta\tau_{ij}^{*} = 0$ otherwise, with ε being a small constant. Second, the $F-1$ best ants of the current iteration are allowed to lay pheromone on the edges they traversed. The quantity laid by these ants depends on their rank q, such that the qth best ant lays $\Delta\tau_{ij}^{q} = (F-q)\varepsilon$ if arc (i,j) belongs to the solution found by the ant, $\Delta\tau_{ij}^{q} = 0$ otherwise. Edges belonging to neither of these solutions just face a pheromone decay at the rate $(1-\rho)$, which constitutes the trail evaporation. The overall procedure described is re-iterated for T times (i.e., T complete iterations of the ACO population).

3.3. Adaptation of the Savings-based ACO to the 3L-CVRP

The Savings-based ACO described above has been considerably modified so as to solve the 3L-CVRP. While in the standard Savings-based ACO only the savings value was considered as heuristic information (visibility), we now consider a second visibility measure related to packing. Simply speaking it is the density of the packing obtained. In other words a combination of routes where the items can be arranged in a packing with few and small holes contributes to high visibility of this combination. This idea is an extension of the two-dimensional approach in Fuellerer et al. [18].

The algorithms presented in Section 3.1 are used at each stage of the ACO algorithm to guarantee that the ants move within the space of feasible solutions. The determination of the potential customers in the neighborhood Ω_{Π} is modified by replacing ξ_{ij} from (1) and (2) by:

$$\xi'_{ij} = (\tau_{ij})^{\alpha}(s_{ij})^{\beta}\left(\frac{\sum_{k\in S} vol_k}{\widetilde{vol}_S}\right)^{\varphi} \tag{4}$$

for $i,j = 1,\dots,n$. Subset S is formed by the customers already belonging to the two partial routes containing i and j. The term vol_k expresses the volume of all the items demanded by customer k, as defined in Section 2. For a given loading, we define $\widetilde{vol}_S$ as the area generated by a *virtual cuboid*, i.e., the "smallest cuboid" (corresponding to the feasible loading found) contained in the vehicle loading space and containing all the items in S. Hence the last term in parentheses is the utilization of the virtual cuboid. This quantity is taken to the power of φ.

Note that, giving the loading of the items in the vehicle, it is straightforward to determine the virtual cuboid. The probability of choosing and merging the partial route with end customer i with the partial route with starting customer j is again given by (2) with ξ_{ij} replaced by ξ'_{ij} from (4).

The additional factor in (4) is introduced so that combinations obtaining good filling of the vehicles receive a higher ξ'_{ij} value, and thus a higher probability of being chosen. Therefore, the current neighborhood is also evaluated according to how well two partial routes fit together with respect to the overall loading of the customers' items.

Since in the 3L-CVRP the number of vehicles is limited to v, only solutions with at most v vehicles are allowed to update the pheromone information. However, the solutions generated by the ants may have more routes than vehicles available. Permitting infeasible solutions is often useful in vehicle routing (see, e.g., Gendreau et al. [19]). In this case the local search (single customer move and swap as in the standard Savings-based ACO) must try to reduce the number of routes. This is done by adding a penalty $u(s)$ to the objective function for evaluation during local search (to be explained below). Another important adjustment of the objective function is done for tightly constrained problems, where the total weight of all items is almost as high as the total weight capacity of all vehicles. In this case the ants are permitted to overload the vehicles a little and during local search this overload is penalized by a penalty term $q(s)$. More precisely, we use the modified objective function:

$$z'(s) = z(s) + \gamma q(s) + \delta u(s), \tag{5}$$

where s denotes a solution. The terms in (5) are evaluated through:

$$z(s) = \sum_{k=1}^{\tilde{v}} c_r(k),$$

$$q(s) = \sum_{k=1}^{\tilde{v}} \max\left\{\sum_{i\in S(k)} d_i - D, 0\right\},$$

$$u(s) = \begin{cases} \dfrac{vWHLD}{\sum_{k=1}^{\tilde{v}}\left(\sum_{i\in S(k)} d_i\right)\left(\sum_{i\in S(k)} vol_i\right)} & \text{if } \tilde{v} > v, \\ 0 & \text{otherwise,} \end{cases}$$

where $\tilde{v}$ is the number of routes in s and $c_r(k)$ is the total cost of route k. The term $q(s)$ represents the weight excess in the vehicles. Finally, the term $u(s)$ is a penalty term for the capacity/volume utilization, and is used only when the number of vehicles exceeds v. Note that $q(s)$ turned out to be useful for those instances for which capacity constraints are very tight, and is thus used only when $(\sum_{i=1}^{n} d_i)/(vD) \geqslant 0.9$. The value $u(s)$ allows the local search to accept solutions with worse $z(s)$ value but better capacity/volume utilization, so as to be able to reduce the number of vehicles $\tilde{v}$. The values γ and δ are positive parameters. γ is kept constant at 100, whereas δ is dynamically adapted from iteration to iteration based on the number of feasible solutions within one ant population.

4. Computational results

The ACO algorithm was coded in ANSI C++ and compiled using the Linux g++ compiler. The algorithm was tested by computational experiments on a Pentium IV with 3.2 GHz and 2 GB of RAM, running under a Linux operative system.

The ACO algorithm was tested on the set of instances proposed in [20], that can be downloaded from http://www.or.deis.unibo.it/research.html. These instances are the only 3L-CVRP instances available on the web. They provide an interesting test bed since heuristic solutions are available for comparison. In the instances, the graphs, the weights demanded by the customers and the vehicle weight capacities are taken from 27 Euclidean CVRP instances (see Toth and Vigo [31] for a detailed description of CVRP test bed instances). The arc costs are determined as the Euclidean (not rounded) distances between customers coordinates. The loading volume has dimensions $W = 25$, $H = 30$ and $L = 60$. For each customer the number of requested items is randomly generated according to a uniform distribution between 1 and 3. Each item dimension is randomly generated according to a uniform distribution in the interval between 20% and 60% of the corresponding vehicle dimension.

In Table 2 we present the results of the ACO on the test bed and compare them with the TS in [20]. Since we use a random seed, the ACO was run 10 times on each instance. The TS in [20] was run on a Pentium IV 3 GHz with 512 MB of RAM, under Windows XP operation system. Being deterministic, it was run a single time on each instance. It was allowed a CPU time limit of 1800 seconds for instances 1–9, 3600 for instances 10–18 and 7200 seconds for instances 19–27. The ACO is halted instead when $\min\{2n, 100\}$ iterations have been performed, or 3 CPU hours have been elapsed.

The parameter setting for the ACO is given in Table 1. These Parameters have been heavily tested in Reimann et al. [29,30] and in Doerner et al. [14]. The same parameter setting has also been used very successfully to solve the 2L-CVRP in Fuellerer et al. [17].

The first five columns in Table 2 report the index I of the instance, the number of customers n, the total number of items M

Table 1
Best parameter setting for the ACO ($\tilde{n} = \max\{50, n\}$).

Parameter	Description	Value	Parameter	Description	Value
P	No. of ants	$\tilde{n}/2$	α	Pheromone weight	5
F	No. of elitists	6	β	Savings weight	5
T	No. of ACO iterations	$2\tilde{n}$	γ	Capacity penalty weight	100
Π	Neighborhood size	$\tilde{n}/4$	ε	Update constant	0.0005
τ_0	Initial pheromone	2	ρ	Trail persistency	0.95

($M = \sum_{i=1}^{n} m_i$), the total routing cost z_{ts} found by the TS and the time in seconds in which this solution was found (sec_h). For the ACO we report the best, average and worst total routing costs found over the 10 runs z_{min}, z_{avg} and z_{max}. The time in seconds in which the best solution was found is also averaged over the 10 runs and reported in column $\overline{sec}_h$. The time required by the ACO algorithm to run to completion is evaluated in the same manner and reported in $\overline{sec}_{tot}$. To make the comparison easier, we also give the percentage gaps between the solution values, evaluated as $\%g_{min} = 100((z_{min} - z_{ts})/z_{ts})$, $\%g_{avg} = 100((z_{avg} - z_{ts})/z_{ts})$ and $\%g_{max} = 100((z_{max} - z_{ts})/z_{ts})$. For each column the row AVG gives the average values on the 27 instances. The loading constraints to be satisfied on each vehicle are the standard 3L-CVRP requirements: sequential loading imposed, a minimum supporting of 75% of each item basis ($a = 0.75$), fragility constraints to be handled, and rotation allowed only on the basis.

In terms of solution quality the ACO algorithm is clearly superior to the TS. The average solution value found by the ACO is worse than the one found by the TS in just one case (instance 11), while in all other cases it is better. This leads to an average improvement of 6.43%. Considering the best solutions found by the ACO on each of the 10 runs, the improvement increases to 6.98%. Even the worst solutions show improvements of 5.89% on average. The times required by the algorithms to find the incumbent solutions are similar, 1747 CPU seconds required on average by the ACO against 2059 of the TS. The ACO algorithm runs to completion in an average time of 1793 CPU seconds, against the 4200 of the TS. Although the computer on which the TS was run is a bit slower than the one used for the ACO (3 GHz vs 3.2 GHz), we can conclude that the ACO is slightly faster than the TS.

The good behavior of the ACO in terms of solution quality is obtained even allowing it a shorter maximum time limit of only 3600 CPU seconds (against the 7200 of the TS, although on a slower computer). Indeed also in this configuration the ACO results clearly better than the TS, producing an improvement of 5.77% for the average solutions, of 6.32% for the best solutions, and of 5.24% for the worst solutions.

For what concerns memory usage, the ACO requires more memory than the TS. Nevertheless, 2 GB of RAM was always sufficient to allow the ACO to run to completion.

For what concerns the robustness with respect to the parameters configurations, we note that the ACO algorithm proved to be very robust with respect to the general parameters introduced in Table 1. On the other hand, it showed more sensitivity to variations of the algorithms used for solving the loading subproblem. We checked several configurations of these algorithms, particularly focusing on the *touching-area* heuristic introduced in Section 3.1. The outcome of these tests is given in Table 3, where we describe the average effect (over all the set of 27 instances) of three new binary parameters. Parameter *bt* takes value 0 if the loading

Table 2
Comparisons of the ACO algorithm with the TS by Gendreau et al. [20] on 3L-CVRP instances from the literature. All loading constraints imposed.

I	n	M	TS		ACO					ACO/TS		
			z_{ts}	sec_h	z_{min}	z_{avg}	z_{max}	$\overline{sec}_h$	$\overline{sec}_{tot}$	$\%g_{min}$	$\%g_{avg}$	$\%g_{max}$
1	15	32	316.32	129.5	304.13	305.35	307.18	11.2	12.0	−3.85	−3.47	−2.89
2	15	26	350.58	5.3	334.96	334.96	334.96	0.1	0.6	−4.46	−4.46	−4.46
3	20	37	447.73	461.1	399.68	409.79	415.87	88.5	121.8	−10.73	−8.47	−7.12
4	20	36	448.48	181.1	440.68	440.68	440.68	3.9	5.4	−1.74	−1.74	−1.74
5	21	45	464.24	75.8	450.93	453.19	454.14	22.7	30.9	−2.87	−2.38	−2.18
6	21	40	504.46	1167.9	498.32	501.47	504.39	17.5	18.4	−1.22	−0.59	−0.01
7	22	46	831.66	181.1	792.13	797.47	801.73	51.4	67.4	−4.75	−4.11	−3.60
8	22	43	871.77	156.1	820.67	820.67	820.67	56.2	78.6	−5.86	−5.86	−5.86
9	25	50	666.10	1468.5	635.50	635.50	635.50	15.3	16.3	−4.59	−4.59	−4.59
10	29	62	911.16	714.0	840.75	841.12	842.75	241.2	246.7	−7.73	−7.69	−7.51
11	29	58	819.36	396.4	818.87	821.04	829.84	172.4	199.8	−0.06	0.21	1.28
12	30	63	651.58	268.1	626.37	629.07	634.65	46.2	48.2	−3.87	−3.45	−2.60
13	32	61	2928.34	1639.1	2739.80	2739.80	2739.80	235.4	308.8	−6.44	−6.44	−6.44
14	32	72	1559.64	3451.6	1466.84	1472.26	1481.07	623.8	642.8	−5.95	−5.60	−5.04
15	32	68	1452.34	2327.4	1367.58	1405.48	1426.51	621.0	656.8	−5.84	−3.23	−1.78
16	35	63	707.85	2550.3	698.92	698.92	698.92	12.8	14.8	−1.26	−1.26	−1.26
17	40	79	920.87	2142.5	868.59	870.33	874.24	11.8	14.9	−5.68	−5.49	−5.06
18	44	94	1400.52	1452.9	1255.64	1261.07	1262.88	2122.2	2209.8	−10.34	−9.96	−9.83
19	50	99	871.29	1822.3	777.18	781.29	784.77	614.3	623.6	−10.80	−10.33	−9.93
20	71	147	732.12	790.0	604.28	611.26	617.13	3762.3	3901.0	−17.46	−16.51	−15.71
21	75	155	1275.20	2370.3	1110.09	1124.55	1136.59	5140.0	5180.6	−12.95	−11.81	−10.87
22	75	146	1277.94	1611.3	1194.18	1197.43	1207.22	2233.6	2290.3	−6.55	−6.30	−5.53
23	75	150	1258.16	6725.6	1158.51	1171.77	1178.66	3693.4	3727.6	−7.92	−6.87	−6.32
24	75	143	1307.09	6619.3	1136.80	1148.70	1160.00	1762.8	1791.5	−13.03	−12.12	−11.25
25	100	193	1570.72	5630.9	1429.64	1436.32	1444.02	8619.7	8817.1	−8.98	−8.56	−8.07
26	100	199	1847.95	4123.7	1611.78	1616.99	1624.01	6651.2	6904.3	−12.78	−12.50	−12.12
27	100	198	1747.52	7127.2	1560.70	1573.50	1600.30	10325.8	10483.9	−10.69	−9.96	−8.42
AVG			1042.26	2058.9	960.87	966.66	972.54	1746.6	1793.1	−6.98	−6.43	−5.89

Table 3
Performance of the ACO algorithm under different parameters settings (aggregate results for the complete set of 27 instances).

TS		ACO								ACO/TS		
z_{ts}	sec_h	bt	st	φ	z_{min}	z_{avg}	z_{max}	$\overline{sec}_h$	$\overline{sec}_{tot}$	$\%g_{min}$	$\%g_{avg}$	$\%g_{max}$
1042.26	2058.9	0	0	1	960.87	966.66	972.54	1746.6	1793.1	−6.98	−6.43	−5.89
1042.26	2058.9	1	0	1	965.15	972.48	980.17	1798.6	1838.8	−6.59	−5.97	−5.29
1042.26	2058.9	0	1	1	961.94	967.56	973.41	1729.7	1767.5	−6.91	−6.41	−5.87
1042.26	2058.9	0	0	0	959.94	967.87	973.91	1794.0	1833.1	−7.03	−6.35	−5.81
1042.26	2058.9	0	0	5	961.62	967.16	973.45	1557.0	1605.6	−6.91	−6.41	−5.80

heuristic does not take into consideration the bottom surface of the vehicle for the evaluation of the touching area. It takes value 1 if, instead, the bottom surface is considered (as done in Gendreau et al. [20]). Parameter *st* takes value 1 if the touching area between two items on the *w*–*l* plane counts double in the evaluation of the overall touching area, 0 if it counts single. These measures favor the stacking of items one on top of the other and introduce a simple improvement with respect to the loading algorithms presented by Gendreau et al. [20]. Finally we recall that φ is the parameter used to weight the *virtual cuboid* in (4). The first line in the table gives the values obtained by our best configuration (i.e., the one used in Table 2), whereas the successive lines give the values obtained by changing one parameter at a time. We note that not considering the bottom surface ($bt = 0$) and/or not multiplying the touching areas on the *w*–*l* plane ($st = 0$) allows us to obtain better results. The best value for φ is one.

5. Evaluation of loading restrictions

In the previous section we have shown the overall good performance of our algorithm compared to the benchmark approach, when all loading restrictions are present. In this section we will focus on the influence of each loading restriction on the overall costs. First we will demonstrate that for all loading configurations that are also considered by the benchmark approach our algorithm outperforms this benchmark. Afterwards we will provide a more detailed analysis of the cost effects of all loading restrictions and focus on managerial implications.

In Table 4 we examine the effect of the loading constraints discussed in Section 2, namely fragility, supporting area and sequential loading (LIFO) policy. The ACO algorithm was run 10 times for each loading configuration. The pairs of columns in the table give the average solution value z_{avg} and the average time to find the incumbent $\overline{sec}_h$, computed as in Table 2. Columns two and three refer to the configuration in which all constraints are imposed (i.e., they give the same values of Table 2). The next pairs of columns report the results for the loading configurations without the fragility constraint, without the LIFO constraint, without the supporting area constraint and with none of these constraints.

The table highlights the percentage differences (%*g*) between the average solution values found for each loading configuration and the ones found for the standard configuration. The qualitative information is consistent with what found in [20]. A strong reduction in the solution value is obtained by removing the supporting area constraint and the LIFO constraint. Removing the fragility constraint leads to the lowest reduction. The configuration with no operational constraints has by far better solution values, with cost reductions close to 10%. Also the average CPU times required to

Table 4
Performance of the ACO algorithm for different loading configurations.

I	All constraints		No fragility		No LIFO		No support		3D loading only	
	z_{avg}	$\overline{sec}_h$	z_{avg}	$\overline{sec}_h$	z_{avg}	$\overline{sec}_h$	z_{avg}	$\overline{sec}_h$	z_{avg}	$\overline{sec}_h$
1	305.35	11.2	310.42	9.1	301.74	3.5	302.17	4.8	297.65	1.0
2	334.96	0.1	334.96	0.1	334.96	0.1	334.96	0.1	334.96	0.1
3	409.79	88.5	380.87	50.1	362.27	20.0	391.18	38.2	362.27	16.2
4	440.68	3.9	440.68	3.9	430.89	1.9	440.91	2.4	430.89	0.5
5	453.19	22.7	440.33	27.2	436.73	21.2	441.87	19.4	406.50	9.6
6	501.47	17.5	495.89	8.6	498.38	6.7	495.85	7.3	495.85	1.2
7	797.47	51.4	763.33	48.0	747.30	16.6	771.07	50.9	732.52	18.1
8	820.67	56.2	816.33	47.3	778.88	6.9	803.77	42.0	735.14	13.3
9	635.50	15.3	632.70	13.1	634.51	4.9	631.09	12.2	630.13	3.7
10	841.12	241.2	833.79	188.2	801.62	97.3	780.25	213.6	711.45	92.6
11	821.04	172.4	801.98	152.6	769.94	94.1	750.81	159.2	718.25	81.9
12	629.07	46.2	614.31	26.9	613.99	11.6	614.52	19.3	612.63	7.5
13	2739.80	235.4	2670.54	269.9	2674.69	139.9	2593.43	271.5	2391.77	174.5
14	1472.26	623.8	1450.03	537.5	1346.25	553.7	1363.92	748.2	1222.17	425.9
15	1405.48	621.0	1351.57	682.6	1315.69	525.1	1325.27	672.2	1182.86	645.0
16	698.92	12.8	698.92	11.1	698.61	3.9	698.92	7.9	698.61	2.8
17	870.33	11.8	871.75	9.8	871.39	5.3	870.02	6.9	862.18	3.1
18	1261.07	2122.2	1205.29	1681.4	1197.00	1043.8	1204.26	2200.7	1112.18	1484.6
19	781.29	614.3	758.40	527.0	736.46	318.4	739.96	558.7	671.60	414.4
20	611.26	3762.3	576.26	3299.2	562.24	1428.2	566.47	3153.2	515.39	1436.7
21	1124.55	5140.0	1092.61	3728.1	1045.60	1738.0	1056.78	4198.2	951.87	2105.7
22	1197.43	2233.6	1176.67	1956.1	1112.46	1176.1	1123.47	1980.0	1030.12	1218.4
23	1171.77	3693.4	1125.75	2385.9	1062.85	1196.6	1074.77	2346.8	971.05	1231.7
24	1148.70	1762.8	1131.25	902.8	1088.79	349.8	1100.47	754.0	1057.39	184.7
25	1436.32	8619.7	1418.26	7581.3	1308.84	4282.0	1340.36	7979.1	1207.97	3986.1
26	1616.99	6651.2	1608.30	6395.5	1550.31	2712.0	1557.75	5829.1	1453.39	2843.6
27	1573.50	10325.8	1514.96	5556.4	1456.28	2510.8	1457.25	4903.6	1333.16	2208.3
AVG	966.66	1746.6	945.04	1337.0	916.25	676.6	919.69	1340.0	856.67	689.3
%*g*			−2.09		−4.79		−4.12		−9.68	

Table 5
Summarized comparison between the ACO and the TS in [20] on different loading configurations.

Configuration	TS		ACO					ACO/TS		
	z_{ts}	sec_h	z_{min}	z_{avg}	z_{max}	$\overline{sec}_h$	$\overline{sec}_{tot}$	$\%g_{min}$	$\%g_{avg}$	$\%g_{max}$
All constraints	1042.26	2058.9	960.87	966.66	972.54	1746.6	1793.1	−6.98	−6.43	−5.89
No fragility	1014.49	2410.8	940.30	945.04	950.76	1337.0	1375.1	−6.49	−6.15	−5.68
No LIFO	951.19	1709.8	910.34	916.25	921.23	676.6	716.8	−3.86	−3.41	−2.99
No support	939.53	1882.5	914.72	919.69	926.02	1340.0	1376.0	−2.14	−1.75	−1.16
3D only	876.31	1567.4	854.39	856.67	858.86	689.3	741.5	−1.97	−1.75	−1.55

find the incumbent solutions decrease when removing constraints, since solving the loading subproblems gets easier.

Also in these additional tests on different loading configurations the behavior of the ACO algorithm remains better than that of the TS. In Table 5 we present the average values for all the configurations. The values in the columns have the same meanings of the ones in Table 2, but are now average values over the complete set of 27 instances. The ACO finds better average solution values and has always negative $\%g_{min}$, $\%g_{avg}$ and $\%g_{max}$.

In Table 5 we note that the average CPU times in which the best solution is found by the ACO ($\overline{sec}_h$) may change consistently from one loading configuration to another. This fact is further analyzed in Fig. 4, where we present the evolution of the average cost over the 27 instances for increasing CPU times. In the figure we report z_{avg} on the y axis and the time limit given to the ACO, in CPU seconds, on the x axis. Each line denotes the evolution for a different loading configuration. Apparently 5400 CPU seconds are more than enough to converge to the best solution values for the cases in which the loading is easier (3D only, no support and no LIFO), whereas more time is required for the other configurations.

We now provide a more in depth analysis of the various loading constraints by analyzing loading configurations that were not solved by the benchmark approach. Due to space restrictions we do not provide the results in additional tables but only report these results in condensed form.

First, rather than having only the options support or no support as in Table 5, we also evaluated *other minimum supporting areas*. For a reduced minimum support area of 50% we obtained cost savings of −3.07%/−3.14%/−3.14% compared to the 75% case w.r.t. the best/average/worst solution. Reducing the minimum support area even further to 25% we obtained cost savings of −4.00%/−3.92%/−3.97% compared to the 75% case w.r.t. the best/average/worst solution. Abandoning the minimum support restriction completely amounted to cost savings of −5.46%/−5.61%/−5.65% compared to the 75% case w.r.t. the best/average/worst solution. Hence, the first reduction from 75% to 50% yields a larger destruction than the further reduction from 50% to 0%. Since packing becomes easier if the support restriction is relaxed one could also observe faster computation times for smaller minimum support areas.

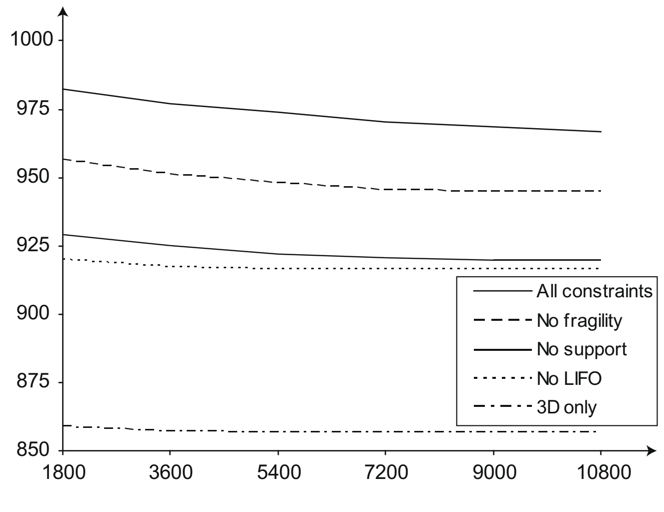

Fig. 4. Evolution over the time of the solution value for each tested loading configuration.

Second, we also evaluated the effect of imposing a *fixed orientation* of the items instead of allowing 90° rotations on the basis. When all other loading restrictions remained binding, this resulted in an average worsening of the best solution by 0.91%, of the average solution by 1.02% and of the worst solution by 1.09%. The CPU times are slightly smaller, since the loading heuristics become faster: 1464 instead of 1747 for sec_h, and 1501 instead of 1793 for sec_{tot}. Also if we added a fixed orientation in any of the other loading configurations we obtained a cost increase of around 1%. More precisely, if none of the other loading restrictions (fragility, support, LIFO) were imposed, requiring a fixed orientation resulted in smaller cost increases of 0.98% on average. This is because relaxing all other loading constraints leaves enough flexibility in loading so that the no-rotation constraint does not really hurt. The highest cost increase (of 1.40% on average) by the no-rotation constraint was observed when fragility and support (75%) constraints were imposed but LIFO-loading was not required. The smallest cost increase (of 0.61% on average if 75% support was required and 0.73% on average if no support was required) by the no-rotation constraint was observed when LIFO-loading was required but fragility was not imposed.Finally, while for all other loading configurations the stacking of items should be favored by setting the parameter $bt = 1$ (see Table 3), in case of fixed orientation setting $bt = 0$ was slightly better. However the parameters are quite robust so that the difference in the objective for $bt = 1$ and $bt = 0$ are only marginal.

6. Conclusions

In this paper, we reconsidered the Three-Dimensional Loading Capacitated Vehicle Routing Problem (3L-CVRP). This problem combines two difficult combinatorial optimization problems, namely the capacitated vehicle routing problem and the three-dimensional loading problem. While the practical relevance is evident, so far only very few papers have been devoted to the combination of packing and routing, because of its complicated nature.

No exact algorithm has been developed so far for the 3L-CVRP and it is hard to imagine that reasonably sized problems can ever be solved to optimality. Therefore, as in the previous literature, the problem is tackled using a heuristic. Our heuristic combines, a modified Savings-based ACO algorithm, originally developed for the CVRP in Reimann et al. [29,30], with fast and simple loading heuristics.

Numerical studies using all publicly available test instances show a very good performance of our algorithm compared to a recent TS approach. We were able to improve the total routing cost on average by 6.43%. The analysis of different loading configurations (fragility, LIFO, support) confirmed similar results previously found in the literature, namely considerable cost reductions when relaxing these constraints. For instance, relaxing all these constraints and permitting any three-dimensional loading lead to average cost reductions close to 10%.

In [18] we proposed a related approach for the two-dimensional version, i.e. the 2L-CVRP, also yielding the currently best results for the test instances available. Some findings here were similar to the two-dimensional case. First, the reason to the superior performance is the introduction of a "packing visibility" in the ant construction process in addition to the savings value. Simply applying the Savings-based ACO algorithm and checking feasibility using packing heuristics gave much worse results. Second, for tightly constrained problems (w.r.t. vehicle capacity or vehicle number) it is important to allow for small infeasibilities that are penalized in the evaluation function for local search.

An interesting result that is contrary to the findings for the two-dimensional case is related to the use of the packing heuristics. In the 2L-CVRP our best results were obtained with lower bounds, simple packing heuristics and a truncated branch and bound algorithm, while in the 3L-CVRP we only use lower bounds and simple packing heuristics to gain the reported improvements. It did not pay to apply more sophisticated packing routines.

While the touching parameter (2L)-heuristic can be transferred to the touching surface (3L)-heuristic, for best results one should apply some bias that favors stacking of items rather than simply evaluating the touching surface which would lead to filling the ground surface of the vehicle too early.

While in the two-dimensional case there are only the loading configurations LIFO/non-LIFO and oriented/non-oriented loading, in the three-dimensional case also fragility and support restrictions and the various combinations of these can be investigated. We have provided a detailed analysis of the cost effects of imposing/relaxing these various restrictions. As expected, the cost savings did not simply add up if all were relaxed, but still each relaxation yielded some significant cost reduction.

Acknowledgements

Financial support from the Oesterreichische Nationalbank (OENB) under Grant #11984, and from the Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) is gratefully acknowledged.

References

[1] B.S. Baker, E.G. Coffman Jr., R.L. Rivest, Orthogonal packing in two dimensions, SIAM Journal on Computing 9 (1980) 846–855.

[2] D. Bortfeldt, H. Gehring, A hybrid genetic algorithm for the container loading problem, European Journal of Operational Research 131 (2001) 143–161.

[3] B. Bullnheimer, R.F. Hartl, C. Strauss, A new rank based version of the ant system: A computational study, Central European Journal of Operations Research 7 (1999) 25–38.

[4] F. Carrabs, J.-F. Cordeau, G. Laporte, Variable neighbourhood search for the pickup and delivery traveling salesman problem with LIFO loading, INFORMS Journal on Computing 19 (2007) 618–632.

[5] N. Christofides, C. Whitlock, An algorithm for two-dimensional cutting problems, Operations Research 25 (1977) 30–44.

[6] G. Clarke, J.V. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Operations Research 12 (1964) 568–581.

[7] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, J.-S. Sormany, New heuristics for the vehicle routing problem, in: A. Langevin, D. Riopel (Eds.), Logistics Systems: Design and Optimization, Kluwer, Boston, 2005, pp. 279–298.

[8] J.-F. Cordeau, M. Iori, G. Laporte, J.J. Salazar-González, Branch-and-cut for the pickup and delivery traveling salesman problem with LIFO loading, Networks (in press).

[9] J.-F. Cordeau, G. Laporte, Tabu search heuristics for the vehicle routing problem, in: C. Rego, B. Alidaee (Eds.), Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search, Kluwer, Boston, 2004, pp. 145–163.

[10] J.F. Cordeau, G. Laporte, M.W.P. Savelsbergh, D. Vigo, Vehicle routing, in: Transportation, Handbooks in Operations Research and Management Science, vol. 14, 2007, pp. 367–417.

[11] T.G. Crainic, G. Perboli, R. Tadei, Extreme point-based heuristics for the three-dimensional bin packing, INFORMS Journal on Computing 20 (2008) 368–384.

[12] T.G. Crainic, G. Perboli, R. Tadei, TS^2 PACK: A two-level tabu search for the three-dimensional bin packing problem, European Journal of Operational Research 195 (2009) 744–760.

[13] A.P. Davies, E.E. Bischoff, Weight distribution considerations in container loading, European Journal of Operational Research 114 (1999) 509–527.

[14] K.F. Doerner, G. Fuellerer, M. Gronalt, R. Hartl, M. Iori, Metaheuristics for vehicle routing problems with loading constraints, Networks 49 (4) (2007) 294–307.

[15] M. Eley, Solving container loading problems by block arrangement, European Journal of Operational Research 141 (2002) 393–409.

[16] O. Faroe, D. Pisinger, M. Zachariasen, Guided local search for the three-dimensional bin packing problem, INFORMS Journal on Computing 15 (2003) 267–283.

[17] G. Fuellerer, K.F. Doerner, R.F. Hartl, M. Iori, Ant colony optimization for the two-dimensional loading vehicle routing problem: Detailed results, Technical Report, POM, University of Vienna, 2007, <http://www.univie.ac.at/bwl/prod/research/VRPandBPP>.

[18] M. Fuellerer, K.F. Doerner, R. Hartl, M. Iori, Ant colony optimization for the two-dimensional loading vehicle routing problem, Computers & Operations Research 36 (2009) 655–673.

[19] M. Gendreau, A. Hertz, G. Laporte, A tabu search heuristic for the vehicle routing problem, Management Science 40 (1994) 1276–1290.

[20] M. Gendreau, M. Iori, G. Laporte, S. Martello, A tabu search algorithm for a routing and container loading problem, Transportation Science 40 (2006) 342–350.

[21] M. Gendreau, M. Iori, G. Laporte, S. Martello, A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints, Networks 51 (2008) 4–18.

[22] M. Iori, J.J. Salazar-González, D. Vigo, An exact approach for the vehicle routing problem with two-dimensional loading constraints, Transportation Science 40 (2006) 342–350.

[23] G.A.P. Kindervater, M.W.P. Savelsbergh, Vehicle routing: Handling edges exchanges windows, in: E. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, John Wiley & Sons Ltd., Chichester, 1997, pp. 337–360.

[24] A. Lodi, S. Martello, D. Vigo, Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems, INFORMS Journal on Computing 11 (1999) 345–357.

[25] S. Martello, D. Pisinger, D. Vigo, The three-dimensional bin packing problem, Operations Research 48 (2000) 256–267.

[26] S. Martello, D. Pisinger, D. Vigo, E. den Boef, J. Korst, Algorithm 864: Algorithms for general and robot-packable variants of the three-dimensional bin packing problem, ACM Transactions on Mathematical Software 33 (1) 7, March 2007, Article 7, 12p.

[27] H.L. Petersen, O.B.G. Madsen, The double travelling salesman problem with multiple stacks – Formulation and heuristic solution approaches, European Journal of Operational Research 198 (2009) 139–147.

[28] D. Pisinger, Heuristics for the container loading problem, European Journal of Operational Research 141 (2002) 382–392.

[29] M. Reimann, K.F. Doerner, R.F. Hartl, D-ants: Savings based ants divide and conquer the vehicle routing problem, Computers & Operations Research 31 (4) (2004) 563–591.

[30] M. Reimann, M. Stummer, K.F. Doerner, A savings based ant system for the vehicle routing problem, in: Proceedings of the Genetic and Evolutionary Computation Conference 2002, Morgan Kaufman, 2002, pp. 1317–1325.

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

[32] E.E. Zachariadis, C.D. Tarantilis, C.T. Kiranoudis, A guided tabu search for the vehicle routing problem with two-dimensional loading constraints, European Journal of Operational Research 195 (2009) 729–743.