Metaheuristics for the Vehicle Routing Problem with Loading Constraints

Karl F. Doerner[1], Guenther Fuellerer[1], Manfred Gronalt[2],
Richard F. Hartl[1], Manuel Iori[3]

(1) Institute for Management Science, University of Vienna, Bruenner
Strasse 72, 1210 Vienna, Austria
{Karl.Doerner, Guenther.Fuellerer, Richard.Hartl}@univie.ac.at
(2) Institute for Production Management and Logistics, University of
Natural Resources and Applied Life Sciences, Feistmantelstrasse, 1190
Vienna, Austria
Manfred.Gronalt@boku.ac.at
(3) Dipartimento di Elettronica, Informatica e Sistemistica (DEIS),
University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy

miori@deis.unibo.it

Abstract

We consider a combination of the capacitated vehicle routing prob-
lem and a class of additional loading constraints involving a parallel
machine scheduling problem. The work is motivated by a real-world
transportation problem occurring to a wood-products retailer, which
delivers its products to a number of customers in a specific region.
We solve the problem by means of two different metaheuristics algo-
rithms: a Tabu Search and an Ant Colony Optimization. Extensive
computational results are given for both algorithms, on instances de-
rived from the vehicle routing literature and on real-world instances.

Keywords: Vehicle routing, Ant Colony Optimization, Tabu Search.

1

1 Introduction

We discuss a combinatorial optimization problem that combines packing and routing aspects, and which is directly derived from a real-world transportation problem occurring at a large Austrian wood-products retailer. The company operates in Eastern Austria and delivers different types of wood-products for further use in the building industry, as supporting or construction material, or in the production of furniture. The customers are mainly building sites, do-it-yourself stores, furniture producers and similar.

Although the company delivers different products, including logs and timber for furniture companies, their main interest concerns the optimization of the deliveries of chipboards. The chipboards are delivered daily to a large set of customers by means of special vehicles. The minimization of the driving distance is of particular interest to the company. We can distinguish between different board types like chipboards or fibre boards. For planning issues they can be grouped into four main types:

- *long chipboards*: the most common ones, used in the building sites as construction material, or cut to produce short chipboards;

- *short chipboards*: used mainly in the furniture production;

- *chipboards for doors*: used in the construction of doors or similar products;

- *heavy-use chipboards*: used mainly in the building sites as supporting material.

Loading/unloading operations are normally performed by means of forklift trucks. For this reason the chipboards of the same type are grouped together and placed on a pallet. Long chipboards are as long as three pallets, while heavy-use chipboards have the same length as one pallet. The other chipboards have smaller dimensions: two chipboards for doors or three short chipboards, respectively, can be placed together side by side on a single pallet. All chipboards have the same width of the pallet, but they have in general different heights. In Figure 1–(a) we illustrate the heights and lengths of the four types of chipboards and of the pallet. We use the term *item* to define a group of chipboards of the same type with a pallet requested by a customer. Each item has the width and length of a pallet and a height given by the sum of the height of the pallet and the height obtained by loading together the chipboards of the same type.

It is important to group chipboards of the same type into a unique item, because this allows both the use of a single pallet, saving space in the vehicle, and the use of a single trip of the forklift trucks from the vehicle to the customer's warehouse. Only in the case in which the height of an item would exceed the height of the vehicle, the chipboards are placed into two (or more) items, the first of which is as high as possible. In Figure 1–(b) we show an item with eight chipboards for doors and an item with 12 short chipboards.

The resulting optimization problem consists thus in delivering a set of items to a set of customers, with the aim of satisfying the requests of all the customers with minimum routing cost. An illustrative example is depicted in Figure 2, in which two vehicles serve five customers demanding a total of 18 items.

long chipboard

heavy use chipboard

chipboards for doors
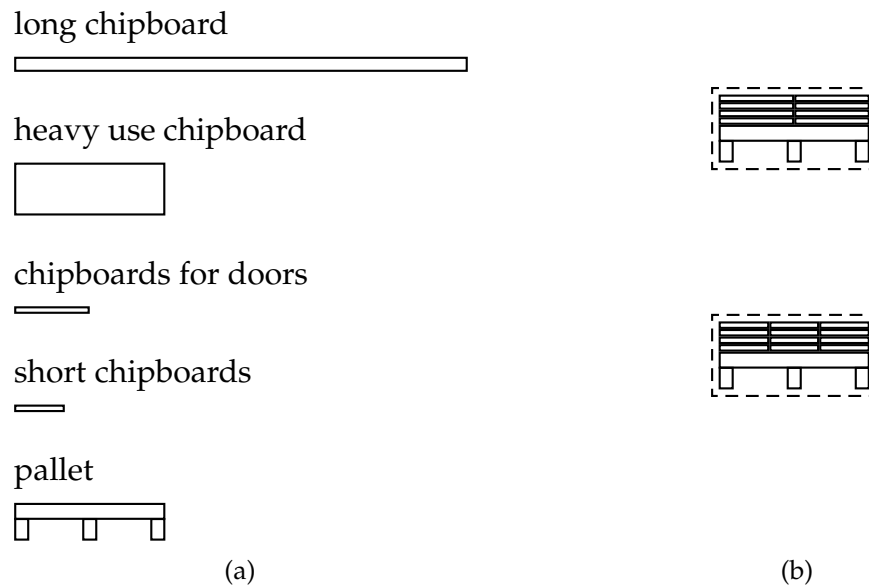
short chipboards

pallet

(a)

(b)

Figure 1: (a) Dimensions of the types of chipboards and of the pallet. (b) Examples of chipboards and pallets grouped together to form items.

To deliver the items, a fleet of suitable vehicles is available. These vehicles are identical, and have an opening through the widest part of the truck, that can make easier the loading/unloading operations (i.e., all the length of the vehicle can be used by forklift trucks for accessing items). Further, a vehicle has the same width as a pallet and can contain up to three pallets along its length. Because of this particular configuration, the vehicles can be loaded by forming up to three different *piles* of items, (i.e., up to three different sequences of items placed on top of each other). Long items use all the three piles at the same level, while the other types of items use a single pile.

Hence, the original three-dimensional loading problem can be reduced to a suitably defined one-dimensional problem. An example is given in Figure 3, were we present a feasible loading for the items associated to the route with customers 1,2 and 3 of Figure 2.
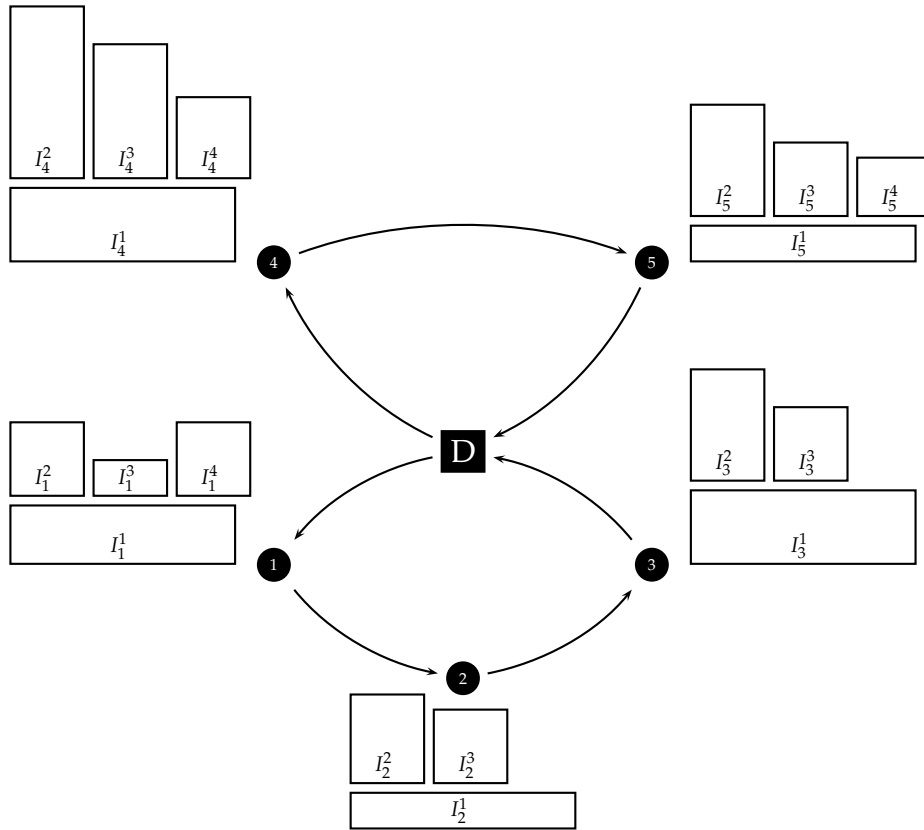
Figure 2: An example of the multi-pile-vehicle routing problem. I_i^k is the $k-$th item of the $i-$th customer.

We are interested in the practical case in which the unloading operations can be performed without having to move items of customers that will be visited later along the route. This requirement is usually referred to *sequential loading* of the items and is frequently encountered in real-world transportation. In Figure 3 we can see how the unloading operations of customers 1,2 and 3 can be performed by picking the items from the top of the piles.

Because of this sequential loading requirement, we may be forced to leave unused space between the items. We can note this aspect in Figure 3-(a), where the dashed area shows the unused space which is left between the items of customer 3 and those of customer 2. This situation occurs
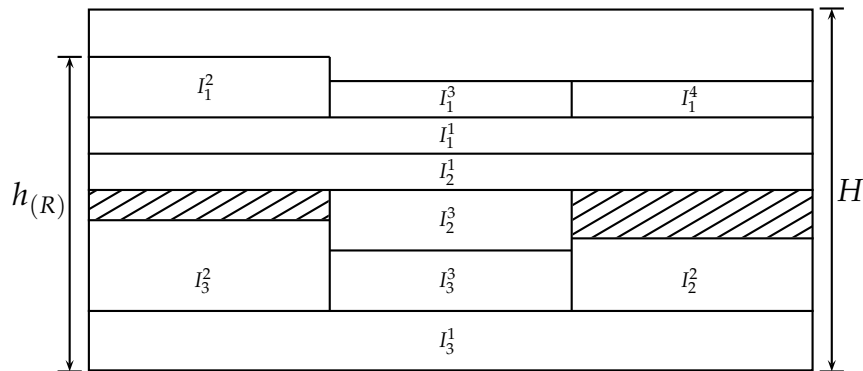
Figure 3: A feasible loading for route (1, 2, 3) of Figure 2.

frequently, since long items produce "cuts" between the loaded items, i.e., they divide all the piles into two separated parts. To allow the stability of the upper part, some bulk material is needed. This support can be created through other pieces of wood, or iron beams or other padding devices. Once the supported items have been delivered, these devices are removed and do not represent an obstacle for the successive unloading operations. In Figure 3 we also note that the area between the items of customer 1 and the top of the vehicle is simply left empty.

We disregard the constraint on the weight capacity of the vehicle, since it is never active in the real transportation problem. Indeed, the heaviest cargo (that can be obtained by loading only long-chipboards) would have a weight of 15 tons, which does not represent a problem since the vehicle weight capacity is 17 tons. For similar reasons also the weight stability of the cargo in the vehicle is disregarded.

In this work we will refer to this particular routing and loading problem as to the *Multi-Pile Vehicle Routing Problem* (MP-VRP). The MP-VRP is naturally an NP-hard problem since it generalizes the Capacitated Vehicle Routing Problem (CVRP). Indeed the two problems are equivalent if

a single pile is available in the vehicles. The MP-VRP is also particularly difficult because of the loading requirements. For this reason, and also for the interest of the company in having a fast algorithm, we found it convenient to resort to metaheuristic algorithms. We thus developed and tested two different approaches: Tabu Search and Ant Colony Optimization.

The MP-VRP combines vehicle routing problems with packing and scheduling problems. Many exact and heuristic algorithms have been developed for the Vehicle Routing Problem (VRP). Concerning exact approaches we refer to the surveys by Laporte and Nobert [24] and Toth and Vigo [35]. In the book edited by Toth and Vigo [37], the chapters of Toth and Vigo [36], Naddef and Rinaldi [29], and Bramel and Simchi-Levi [3] cover, respectively, branch-and-bound, branch-and-cut and set covering approaches. Recent results were provided, e.g., by Baldacci et al. [2], with an algorithm based on a two-commodity formulation, by Letchford and Salazar [26] through branch-and-cut, and by Fukasawa et al. [16] through branch-and-cut-and-price. From the heuristic point of view good results were obtained, e.g., by Gendreau et al. [18] and by Toth and Vigo [38] through Tabu Search, by Prins [31] and Mester and Bräysy [28] through evolution strategies, and by Reimann et al. [32] with an Ant Colony Optimization heuristic. Recent surveys devoted to heuristics for the VRP are the ones by Laporte and Semet [25], Gendreau et al. [21] and Cordeau and Laporte [10].

The problem of loading the items into a vehicle is strictly related to two of the most well known problems in combinatorial optimization: the *Bin Packing Problem* (BPP), and the *parallel processor scheduling problem* (denoted as the $P||C_{\max}$ problem in the three-field notation by Graham et al. [22]).

The BPP calls for the packing of items of a given weight into the minimum number of bins with a limited weight capacity. Exact procedures were proposed by Martello and Toth [27] and Scholl et al. [34] through branch-and-bound, and by Vanderbeck [39] through column generation. For a survey on heuristic and approximation algorithms we refer to Coffman et al. [8]. More recent heuristic results were presented, e.g., by Fleszar and Hindi [14] through variable neighborhood search, by Alvim et al. [1] through tabu search, and by Brugger et al. [4] by means of an Ant Colony Optimization procedure.

For the $P||C_{\max}$ problem a branch-and-bound algorithm was presented by Dell'Amico and Martello [12], while a multi-exchange neighborhood search algorithm was provided by Frangioni et al. [15].

This is not the first time in which the two optimization areas of routing and packing are studied together. Iori et al. [23] presented a particular VRP in which the demands of the customers were composed by rectangular weighted items, which had to be loaded on vehicles having a two-dimensional surface, and delivered with minimum routing cost. The problem, defined as the *Two-Dimensional Loading CVRP* (2L-CVRP), was solved by means of a branch-and-cut algorithm, iteratively calling an inner branch-and-bound for the solution of the loading subproblem. The 2L-CVRP was later addressed by Gendreau et al. [20] through heuristics and tabu search. Finally, the generalization to the three-dimensional case was studied by Gendreau et al. [19], who proposed a Tabu Search for the routing aspect, with a nested Tabu Search for the loading part, and used it to solve real-world instances.

Although the MP-VRP is motivated by a real-world application, it can

be seen as a very general loading and routing problem. For this reason, in the following sections we describe both the problem and the algorithm in the most general way. We then return to the original application in Section 6.3, where we address specific real-world instances.

The remainder of the paper is organized as follows. In Section 2 we formally present the transportation problem addressed. The solution approaches are discussed in Section 3, where we present heuristic and dynamic programming approaches for determining a feasible loading for one vehicle, and in Section 4 and Section 5, where we describe a Tabu Search algorithm and an Ant Colony Optimization procedure for the combined routing and loading problem. Extensive computational results are given in Section 6, both on instances derived from the CVRP literature and on real-world instances. Finally in Section 7 we draw some conclusions.

2   Problem Description

In the MP-VRP we are given a complete undirected graph $G = (V_0, E)$, where $V_0 = V \bigcup \{0\}$, $V = \{1, \ldots, n\}$ is the set of vertices corresponding to customers i and 0 is the vertex corresponding to the depot. Each edge (i, j) has an associated routing cost c_{ij}, for $(i, j) \in E$. We are given a fleet of identical vehicles, having a maximum height H and p piles for the loading of the items.

Each customer i has a demand consisting of m_i items. We denote by I_i^k the $k-$th item demanded by the $i-$th customer ($i = 1, \ldots, n; k = 1, \ldots, m_i$). The height of I_i^k is denoted by h_i^k and is a positive integer value. The length of I_i^k (i.e., the number of piles needed for loading the items in the vehicle) is denoted by l_i^k. For the long items $l_i^k = p$, while for all the other items

$l_i^k = 1$. The set of items demanded by a given customer i is defined by $I(i) = \{I_i^k : k = 1, \ldots, m_i\}$. Without loss of generality we suppose that the items in $I(i)$ are sorted by decreasing length, breaking ties by decreasing height (i.e., the first item is the long one, if demanded, and then the other items are the short ones, sorted by decreasing value of height). We also define $M = \sum_{i=1}^n m_i$ as the total number of items in an instance.

Finally we define a route $r = (r_1, r_2, \ldots, r_t)$ as an ordered sequence of customers and $I(r) = \bigcup_{r_i \in r} I(r_i)$ as the total set of items to be loaded in the vehicle traveling along the route r. For each route r we have to determine if a feasible loading of the items in $I(r)$ into a single vehicle exists. This subproblem arises when looking for the set or routes of lowest cost and deserves a formal definition. We define the *One Vehicle Loading Problem* (1-VLP) as follows: given a route r, and a corresponding set of items $I(r)$, determine a loading of the items into a single vehicle such that the following conditions are respected:

a) the items do not overlap;

b) the items are completely contained into up to p piles;

c) when visiting a customer, all his items must be unloaded without having to move items of customers visited later on along the route;

d) the height of the resulting loading is minimum.

By defining $h(r)$ as the solution of the 1-VLP (i.e., the height of the loading associated to the route r), we can determine if r is feasible by checking $h(r)$ against the vehicle height H (see, e.g., Figure 3). Note that conditions a) and b) derive by the loading requirements. Condition c) derives instead

by the sequential loading constraint, and imposes that, for each pile, the items must be sorted by increasing order of visit from the top to the bottom.

We can now formalize the complete routing and loading problem. The MP-VRP calls for the deliveries of the items $I(i)$ demanded by each customer i ($i = 1, \ldots, n$) through a set s of routes r with the aim of minimizing the total routing cost

$$z(s) = \sum_{r \in s} c(r) \tag{1}$$

where $c(r)$ is the routing cost of route r. The routes in the solution have to be 1-VLP feasible and with height $h(r) \leq H$.

3   Solution of the 1-VLP

3.1   Complexity of the 1-VLP

We first note that the 1-VLP is a difficult problem since it generalizes the $P||C_{\max}$ scheduling problem. Consider the case in which all the items have length $l_i^k = 1$: determining the minimum height $h(r)$ in a 1-VLP instance is equivalent to finding the minimum makespan in a $P||C_{\max}$ instance. The most relevant difference between the two problems lies in the presence of the long items. These items produce a cut of the loading in the piles and, because of the sequential loading constraint, divide the items of the customers according to the sequence of visit along the route. This interesting property reduces the complexity of the problem (see Lemma 1 below) and can be used to produce a fast heuristic (see Subsection 3.2 below) and a dynamic programming (see Subsection 3.3 below).

We thus initially focus on the case in which every customer asks for a long item (i.e., $l_i^1 = p$ for $i = 1, \ldots, n$) and define this problem as the 1-VLP(ℓ). We use the term *pair* to denote a couple of consecutive customers in a route, say r_i and r_{i+1}, such that the short items of r_i are loaded on top of his long item, and the short items of r_{i+1} are loaded at the bottom of his long item. A good loading for this pair is the one for which the combined height of the short items of the two customers is a minimum, and can be obtained in a preprocessing phase for each couple of customers (see Section 3.2 below). A *loading configuration*, i.e., a loading pattern of the items into the vehicle, can be defined by considering all the different pairs that can be formed with consecutive customers in route r.

Let us consider the number f_n of these possible loading configurations for the 1-VLP(ℓ). An upper bound is always given by $f_n \leq 2^{n-2}$, since the bottom customer should have the long item at the bottom and the customer on the top should have the long item on the top. However only some of these loadings are reasonable, e.g., it would not make sense that several subsequent customers have the long items on the top or on the bottom.

Lemma 1 *Complexity of the 1-VLP(ℓ): the number of reasonable loading configurations, f_n, of n customer demands can be computed by the recursion*

$$f_{n+1} = f_{n-2} + f_{n-1} \tag{2}$$

with starting values $f_1 = 1$, $f_2 = 1$.

Proof. Clearly, $f_1 = 1$, $f_2 = 1$, $f_3 = 2$, and $f_4 = 2$, since with 3 customers the customer in the middle has 2 options to be packed and with 4 customers either 2 pairs are formed or the 2 intermediate customers form a

pair. Now a recursion for f_n can be derived by induction. Given $f_1, \ldots, f_n$ we now compute f_{n+1}. There are two possibilities. Case 1: customer r_{n+1} is not combined with customer r_n. This is only reasonable if customers r_n and r_{n-1} form a pair. Hence, there are f_{n-2} possibilities for arranging the first $n-2$ customers (see also the left hand side of Figure 4). Case 2: customer r_{n+1} is combined with customer r_n. Then there remain f_{n-1} possibilities for the first $n-1$ customers (see also the right hand side of Figure 4). Hence, in total there are $f_{n+1} = f_{n-2} + f_{n-1}$ possible loadings. ∎
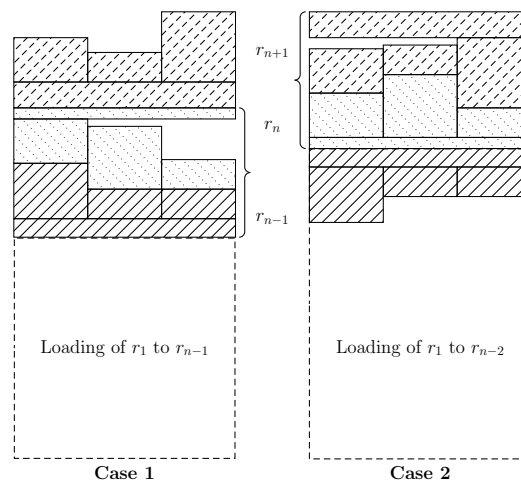

Figure 4: Possible combinations of customers.

Together with the above initial conditions we obtain

n	5	10	15	20	25	30
f_n	3	12	49	200	816	3329

which could, indeed, be enumerated. However, since the loading routine is called very often during the metaheuristic approaches, it is essential to compute a loading pattern in a much shorter time. This can be done either by means of simple heuristics or using dynamic programming.

3.2 A simple heuristic for the 1-VLP(ℓ)

Denote $\overline{h}_i$ as the minimum height of the loading of customer i alone on the p piles. Denote also $\overline{h}_{i,j}$ as the minimum height of a pair, obtained by loading *together* the items of customers i and j, computed through the following algorithm: 1) load the long item demanded by customer i, if any; 2) load the short items demanded by customers i and j, if any, through a branch-and-bound procedure derived from the one proposed by Dell'Amico and Martello [12]; 3) load the long item demanded by customer j, if any. These values are computed in a pre-processing phase. The $\overline{h}_i$ values are computed in a similar way.

The values obtained are then used in a heuristic algorithm that we denote by HL. Consider a route r of length t, HL gives a heuristic $h(r)$ value by computing

$$h(r) = \sum_{i=1}^{t/2} \overline{h}_{r_{2i-1}, r_{2i}} \tag{3}$$

if t is even, or

$$h(r) = \min \left\{ \overline{h}_{r_1} + \sum_{i=1}^{(t-1)/2} \overline{h}_{r_{2i}, r_{2i+1}}; \sum_{i=1}^{(t-1)/2} \overline{h}_{r_{2i-1}, r_{2i}} + \overline{h}_{r_t} \right\} \tag{4}$$

if t is odd. Equation (4) takes into consideration the minimum value that can be produced by loading separately either the first or the last customer in r. The HL approach gives very quickly an approximated value for the loading of a route and proved to be very useful in the metaheuristic search process (see Sections 4 and 5 below).

For evaluating the HL behavior, let us define the *worst case performance ratio* of a heuristic algorithm A as the minimum value $WCP(A)$ such that $WCP(A) \geq UB(I)/z(I)$ for any instance I of a problem, where $z(I)$ is the

optimal solution of the instance and $UB(I)$ is the heuristic solution found by the algorithm A.

Lemma 2 *Worst case performance of HL: for the 1-VLP(ℓ), a tight upper bound on the worst case performance ratio of HL is $WCP(HL) \leq 2$.*

Proof. First, note that the heuristic solution value UB found by HL is not greater than the sum of the heights $\overline{h}_{r_i}$ of all the customers, thus

$$UB(I) \leq \sum_{i=1}^{2p} \overline{h}_{r_i}. \tag{5}$$

Second, a valid lower bound can be found by considering the following relaxation. Suppose the number of customers is even (if it is odd, one can always add a dummy customer with $\overline{h}_i = 0$). Since each customer asks for a long item, at most two customers can be placed side by side at the same level. When loading two customers i and j together, the resulting height is at least equal to the maximum height of the two customers, thus

$$\overline{h}_{i,j} \geq \max \left\{ \overline{h}_i, \overline{h}_j \right\} \tag{6}$$

for $i = 1, \ldots, n$, $j = 1, \ldots, n$, $j \neq i$. Relax this constraint too and suppose that (6) is always satisfied with equality. In this case the best possible situation arises when the highest customer is placed together with the second highest customer, the third one with the fourth one and so on. Now drop the constraint on the sequential loading, sort the customers by decreasing values of $\overline{h}_{r_i}$ and store the corresponding indices in $o(i)$ (i.e., $\overline{h}_{o(1)}$ is the highest height of a customer). The value

$$LB(I) = \overline{h}_{o(1)} + \overline{h}_{o(3)} + \ldots \overline{h}_{o(2p-1)} = \sum_{i=1}^{p} \overline{h}_{o(2i-1)} \tag{7}$$

is thus a valid lower bound for the problem. Since $\overline{h}_{o(i)} \geq \overline{h}_{o(i+1)}$ for $i = 1, \ldots, n-1$, we can also limit the value of the lower bound by

$$LB(I) \geq 1/2 \sum_{i=1}^{2p} \overline{h}_{o(i)} = 1/2 \sum_{i=1}^{2p} \overline{h}_{r_i} \qquad (8)$$

and consequently $WCP(HL) = 2$ is a valid worst case performance ratio for HL for the 1-VPP(ℓ).

We finally note that this value is tight. Consider indeed an instance with four customers. Each customer demands a long item of height δ. Moreover the first and the fourth customer demand a short item of height ε and the second and third customer demand a short item of height 1. For small values of ε, the optimal solution consists in loading together the second and third customers, obtaining $h(r) = 1 + 2\varepsilon + 4\delta$. HL loads instead the first and second customers together, and the third and fourth customers, obtaining $h(r) = 2 + 4\delta$. Thus $2 + 4\delta/(1 + 2\varepsilon + 4\delta) \rightarrow 2$ for $\varepsilon \rightarrow 0, \delta \rightarrow 0$ and the performance is tight. ∎

3.3 Solving the 1-VLP(ℓ) using dynamic programming

If a better solution of the 1-VLP(ℓ) is needed, it can be computed using a dynamic programming approach (DP), that exploits the natural step structure within the problem.

Proposition 3 *The optimal loading for a route with n customers can be computed in linear time with effort*

$$2(n-2) * additions + (n-2) * comparisons$$

Proof. This is done by induction. Let $L(r_1, \ldots, r_n)$ be the optimal loading height of the partial route $(r_1, \ldots, r_n)$. We start with $L(r_1) = \overline{h}_{r_1}$ and

$L(r_1, r_2) = \overline{h}_{r_1, r_2}$. With 3 customers, the middle customer is either combined with the top or the bottom customer, i.e., $L(r_1, r_2, r_3) = \min\{L(r_1) + \overline{h}_{r_2, r_3}, L(r_1, r_2) + \overline{h}_{r_3}\}$. For extending the route to customer r_{n+1}, we again have to consider the 2 cases from Figure 4. Either the loading of the first n customers remains unchanged and customer r_{n+1} is added on top (not combined) leading to total height $L(r_1, \ldots, r_n) + \overline{h}_{r_{n+1}}$, or customers r_n and r_{n+1} are combined to form a pair giving total height $L(r_1, \ldots, r_{n-1}) + \overline{h}_{r_n, r_{n+1}}$. Taking the lower value of these two heights gives

$$L(r_1, \ldots, r_n) = \min\{L(r_1, \ldots, r_{n-2}) + \overline{h}_{r_{n-1}, r_n}, \; L(r_1, \ldots, r_{n-1}) + \overline{h}_{r_n}\}.$$

Summing up, for each of the customers from r_3 to r_n, two additions and one comparison have to be performed. ∎

Although the DP algorithm is capable of of finding better loadings than HL, it will turn out in Section 6 that the improvement in the solution quality is only negligible.

3.4   Solving the 1-VLP

When we apply the HL or the DP algorithm (designed for the 1-VLP(ℓ)) from the previous subsections to the general 1-VLP, where not all customers order large items, the worst case performance deteriorates:

Lemma 4 *Worst case performance of HL and DP:* *If not all customers order large items, a tight upper bound on the worst case performance ratio for both algorithms, HL and DP, is WCP $\leq p$.*

Proof. We first note that the worst case performance ratio WCP(A) of every heuristic algorithm A for the 1-VLP (and for the $P||C_{\max}$ problem), is

limited by p. Indeed, consider an instance in which the loading is perfect (i.e., all the piles are used completely) and forms a height of value z. The worst heuristic solution consists in assigning all the items to a single pile, obtaining $UB(I) = pz$. Thus $WCP(A) \leq p$ is valid for every heuristic algorithm A. This value is tight for HL. Indeed consider an instance with $2p$ customers none of which order large items, in which the customers in odd positions along the route $(r_1, r_3, \ldots, r_{2p-1})$ demand for a single short item of height ε, and the customers in even positions $(r_2, r_4, \ldots, r_{2p})$ demand for a single short item of height 1. For small values of ε, the optimal solution consists in loading every couple of items in order into a single pile, obtaining height $z = 1 + \varepsilon$. HL and DP instead obtain a sequence of p pairs of value $\overline{h}_{i,j} = 1$, in which each item of height ε is loaded at the same level of the following item of height 1. The resulting solution has $h(r) = p$ and thus $WCP(HL) = p$ is tight for $\varepsilon \to 0$. ∎

Although this value is arbitrarily bad for $p \to \infty$, the average performance is quite satisfactory, as shown in Section 6 below.

With additional effort, a better solution for the 1-VLP can be computed by exploiting the following idea: assume for the moment, that route r contains only one sequence of consecutive customers $(r_i, \ldots, r_j)$ not ordering large items. Apply to this sequence the exact algorithm already used for computing $\overline{h}_{i,j}$ (see Section 3.2), and obtain the loading height $L(r_{i-1}, r_i, \ldots, r_j, r_{j+1})$, i.e., the minimum height obtained when packing first the long item of r_{i-1} then the short items of $r_{i-1}, \ldots, r_{j+1}$ and then the long item of r_{j+1}. Then, 1) decompose the route in three parts, 2) compute the heights of $r_1, \ldots, r_{i-2}$ and $r_{j+2}, \ldots, r_t$ with the DP algorithm of Section

3.3, and 3) add the height $L\left(r_{i-1}, \ldots, r_{j+1}\right)$ to obtain a valid upper bound on the loading height.

This idea can be easily extended to the case in which more sequences of customers not ordering long items are present. For each of these sequence evaluate two options: compute the height as in a normal pairing process using heuristic HL, or compute the height with the procedure described above. Then use the DP algorithm to both compute the heights of the remaining parts of the route (i.e., those parts for which all customers demand for long items), and to select the best of the two options for each sequence with no long items.

The resulting algorithm, defined as HL2, can provide better loading solutions than HL, but with a remarkable increase in the computational effort required.

4   A Tabu Search Approach

We developed a Tabu Search approach that is focused at the minimization of a modified objective function. Denoting s as the current solution, we modify the objective function in Equation (1) by adding a penalty term. The modified objective function can be expressed as

$$z'(s) = z(s) + \alpha e(s), \tag{9}$$

$$e(s) = \sum_{r \in s} \max\{h(r) - H, 0\}, \tag{10}$$

where, for all the routes r in the solution s, we consider both the routing cost $c(r)$ and the excess of loading height $(h(r) - H)$. If the excess is equal to 0, then $z'(s)$ is equal to the routing cost and we have reached a feasible

solution. Otherwise the excess is penalized by a given parameter α. During the search process, according to the difficulty of the particular instance, α is updated in order to give more or less importance to the loading penalization. In particular, if total excess height $e(s) > 0$ then $\alpha = \alpha(1 + \delta)$, otherwise $\alpha = \alpha/(1 + \delta)$, with δ being a given parameter greater than 0.

The idea of accepting but penalizing infeasible solutions in a tabu-search framework was first applied to the CVRP by Gendreau et al. [18], leading to interesting results. The approach was later generalized for the periodic and multi-depot CVRP by Cordeau et al. [9], and to the two- and three-dimensional loading CVRP by Gendreau et al. [20, 19].

A starting heuristic solution is found by adapting to the MP-VRP the Clarke and Wright [7] savings algorithm for the CVRP. In this algorithm the initial solution consists of the assignment of each customer to a separate route. Successively, for each pair of customers i and j the following savings measure is calculated:

$$s_{ij} = c_{i0} + c_{0j} - c_{ij} \tag{11}$$

(recall that c_{ij} denotes the cost of edge (i, j) and 0 denotes the depot). Thus, the values s_{ij} contain the savings of combining two customers i and j on one route as opposed to serving them on two different routes. We accept mergings of two routes into a unique route r only if this leads to an height $h(r) \leq H$. In this heuristic and during the following iterations of the search process, the $h(r)$ values are computed through algorithm HL of Section 3.

Let us consider the $\pi(i)$ vertices that are closest to a given vertex i (i.e., those vertices connected to i by arcs $(i, \pi(i))$ having minimum cost among the arcs leaving i). A move consists in removing a customer i from his

current route and re-inserting it into another route containing at least one of his $\pi(i)$ neighbors. Both routes are re-optimized by means of the 4-opt insertion procedure described in Gendreau et al. [17] for the Traveling Salesman Problem.

At each iteration all the possible moves, obtained by removing each customer and re-inserting it into all the possible routes, are computed. The search is directed towards less explored regions by means of an additional penalization factor. By denoting as ϕ the frequency in which a customer has been assigned to a given route, a penalty term $\beta\phi$ is added to $z(s)$, with β being a parameter greater than 0. Among all the moves, the one leading to the lowest value of $z'(s) + \beta\phi$ is selected and used to update the current solution.

When a move is performed, reinserting the corresponding customer into his former route is declared tabu for the next θ iterations, unless this leads to an improvement in the incumbent solution. Finally, a simple tool is adopted as intensification: each time a new incumbent solution is found, during the next iteration the size of the neighborhood is doubled for each customer i.

In order to set the parameters in the best possible way, the algorithm was run in different configurations on the instances that will be presented in Section 6.2 below. The Tabu Search proved to be very robust with respect to the initial value of the parameter α, finally set to $10\bar{c}/H$, with $\bar{c}$ being the average cost of the arcs. Other values tested were 1, $\bar{c}/H$ and $20\bar{c}/H$. The parameter δ was fixed to the value 1, which led to better computational results than 0.01, 0.1, 0.5 and 2.

Concerning the size of the neighborhood, $\pi(i)$ was set equal to a value

Π for each $i = 1, \ldots, n$, where $\Pi = \min(25, n/5)$. Also the values $\min(20, n/6)$ and $\min(30, n/4)$ were tested leading to worse results. Other values independent from n were tested (namely 5, 7, 10, 15, 20, 25) but were disregarded. A good choice for β proved to be the value n, while n^2, $\sqrt{n}$, and other values depending from M proved to be less efficient. The number of iterations for which a move is declared tabu was set to $\theta = n/4$, more efficient than n^2, n, $n/2$ and $n/8$.

Two versions of the algorithm were tested, a single start and a multi-start. The multi-start version stops the Tabu Search after a given number of iterations γ, and re-iterates the whole process starting with a different heuristic solution (obtained by randomizing the initial heuristic). In contradiction with what obtained in Gendreau et al. [19], for the MP-VRP the best choice turned out to be the multi-start version. In this case the algorithm proved to be sensitive to the variations of γ, which was finally set to 50 000, after having tested 200 000, 100 000, 10 000 and other values depending from n.

Finally, we decided to halt the algorithm after 250 000 iterations (i.e., 5 different starting points in the multi-start approach) or after a CPU time limit of 2 hours, as this proved to be the best compromise between elapsed time and quality of the solutions found.

5 Savings based Ant Colony Optimization

Based on the observation of real ants' foraging behavior, the *Ant Colony Optimization* (ACO) was developed as a graph-based, iterative, constructive metaheuristic by Dorigo et al. [13]. The main idea of ACO is that a population of artificial ants repeatedly builds and improves solutions to a given

instance of a combinatorial optimization problem. From one generation to the next a joint memory is updated to guide the search of the successive populations. The memory update is based on the solutions found by the ants and more or less biased by their associated quality. The Savings based ACO algorithm mainly consists of the iteration of three steps:

- generation of solutions by ants according to heuristic and pheromone information;

- application of local search to each solution;

- update of the pheromone information.

In our approach the implementation of these three steps is based on the framework presented by Reimann et al.[33], and is described in Section 5.1. Then, in Section 5.2, we see how the ACO approach can be tailored for the MP-VRP.

5.1 Standard Savings based Ant Colony Optimization

The solutions are constructed according to the well known savings algorithm of Clarke and Wright [7]. The computed savings values are sorted in decreasing order and stored in a list. In the iterative phase, partial routes are combined by sequentially choosing feasible entries from this list. In our case a combination is feasible if it does not violate the weight capacity of the vehicle and if it leads to an height $h(r)$ (computed through algorithm HL of Section 3) not greater than H.

The decision making about combining customers is based on a probabilistic rule that takes into account both the above mentioned savings values and the pheromone information. Let τ_{ij} denote the pheromone concen-

tration on the edge connecting customers i and j, representing how good the combination of these two customers was in the previous iterations.

In each decision step of an ant, we consider the Π best combinations still available, where Π is a parameter that represents the size of the neighborhood. Let Ω_Π denote the set of Π neighbors, i.e., the Π feasible combinations (i, j) yielding the largest savings considered in a given decision step, then the probability of choosing to combine customers i and j in one route is given by (12) and (13):

$$
\mathcal{P}_{ij} = \begin{cases} \dfrac{\xi_{ij}}{\sum_{(h,l) \in \Omega_\Pi} \xi_{hl}} & \text{if } (i,j) \in \Omega_\Pi \\[2mm] 0 & \text{otherwise,} \end{cases}
\tag{12}
$$

$$
\xi_{ij} = s_{ij}^{\beta} \tau_{ij}^{\alpha}
\tag{13}
$$

where α and β bias the relative influence of the pheromone trails and the savings values, respectively. Once no more feasible savings values are available, the algorithm results in a (sub-)optimal set of routes connecting all customers.

A solution obtained through this procedure is then subjected to a local search in order to ensure local optimality. In our algorithm we sequentially apply the *move* and *swap* neighborhood (see Osman [30]) between routes to improve the clustering and the 2-opt algorithm (see Croes [11]) within routes to improve the routing.

During the iterations, the pheromone is updated according to the rule proposed by Bullnheimer et al. [5]. Its pheromone management centers around two concepts borrowed from Genetic Algorithms, namely ranking and elitism. Let $0 \leq \rho \leq 1$ be the trail persistence and F the number of

elitists (i.e., those ants leading to the best current solutions). Then, the pheromone update scheme can be formally written as

$$\tau_{ij} = \rho\tau_{ij} + \sum_{q=1}^{F-1} \Delta\tau_{ij}^{q} + F\Delta\tau_{ij}^{*} \,. \qquad (14)$$

First, the best solution found by the ants up to the current iteration is updated as if F ants had traversed it. The amount of pheromone laid by the elitists is $\Delta\tau_{ij}^{*} = \iota$, where ι is 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 r such that the q-th best ant lays $\Delta\tau_{ij}^{q} = (F - q) \cdot \iota$. Edges belonging to neither of those solutions just face a pheromone decay at the rate $(1 - \rho)$, which constitutes the trail evaporation.

5.2 Adaptation of the Savings-Based ACO to the MP-VRP

The savings-based ACO described above was adapted to solve the MP-VRP by modifying three elements. First, a second heuristic measure was introduced to consider the loading. Second, in combination with this heuristic measure an additional pheromone matrix for the loading (denoted as the loading pheromone matrix in the following) was also used. Third, a pheromone update mechanism for the loading pheromone matrix was adapted.

In order to take into account the loading information, we combine two different measures. The first measure is given by the height $\overline{h}_{i,j}$ of two combined customers i and j. We prefer large values of $\overline{h}_{i,j}$, so as to load first the customers with large demands (as done in the well known First Fit Decreasing heuristic for the BPP). The idea for this measure is to prefer

customer pairs which require more space on the truck than pairs that need less. Thus, two combined orders with a larger height are more likely to be chosen than two items with a smaller height. In addition to that measure, we consider the amount of bulk material needed to load together two customers. It is reasonable to load customers i and j consecutively on the same vehicle when few bulk material is required and therefore few loading space is wasted. The most preferable customer pairs are those whose combination results in a high loading height and simultaneously in a low unused capacity (i.e., a low usage of bulk material). The value γ_{ij} stands for the amount of unused capacity when combining customer i and j (i.e., the value of the required bulk material when combining customer i and customer j). We then define the second measure for the loading as in Equation (15):

$$\gamma'_{ij} = \gamma^{\max} - \gamma_{ij} \tag{15}$$

where $\gamma^{\max}$ denotes the maximum amount of bulk material required between two customers in the current problem instance. (The $\gamma^{\max}$ value is computed in a preprocessing phase.)

We finally get the heuristic information p_{ij} for the loading by multiplying the two heuristic measures (see Equation 16):

$$p_{ij} = \gamma'_{ij} \cdot \overline{h}_{ij} \tag{16}$$

We found out that a multiplication of the two heuristic values provides better results than an additive combination of them. We deal only with feasible solutions, therefore a pair of two customers can only be added if its inclusion does not violate the capacity of the vehicle.

The weighted attractiveness value of $\xi_{i,j}$ is modified in such a way that

also the lost capacity and the total height of the combination of the two customers is taken into account by integrating the value p_{ij} and the corresponding pheromone information τ_{ij}^p (see Equation (17)). The pheromone value τ_{ij}^p represents the pheromone information for the loading part of the problem and is the second modification of the standard ACO: high values τ_{ij}^p for each couple of customers i and j represent the fact that it is reasonable to combine them together, because this lead to good results during the previous iterations.

Hence, the probability of choosing to combine customers i and j in one route is given again by (12), but with the following modification:

$$\xi_{i,j} = \delta \left[(s_{i,j})^\beta (\tau_{i,j}^r)^\alpha \right] + (1 - \delta) \left[(p_{i,j})^\beta (\tau_{i,j}^p)^\alpha \right] \tag{17}$$

where, by introducing the parameter $\delta \in [0, 1]$, it is possible to put more weight on the routing information and less on the loading information or vice versa.

For the update of the two pheromone matrices we use two objectives. The first objective is the original one (see (1)), which is used to update the routing pheromone $\tau_{i,j}^r$ as described in (14). For the update of the packing pheromone $\tau_{i,j}^p$ we use the total packing height

$$\sum_{r \in S} h(r) \tag{18}$$

as the objective function.

The third modification affects the pheromone update. We modified the pheromone update for our MP-VRP in the following way. We determine not only the F best solutions of the current population for the current iteration with respect to the routing, but additionally we consider also the F

best solutions concerning the loading. In addition to that, the elitist ants for the loading are only allowed to update the pheromone matrices when the solution found has the same number of vehicles as the current best solution found so far.

The settings of the parameters presented by Bullnheimer et al. [5] for the rank based Ant System in general, and by Reimann et al. [33]) for the Savings based Ant System in particular prove to be a good choice also for the MP-VRP. We use a neighborhood size (Π) equal to $n/4$, and 6 elitist ants ($F = 6$). The population size of the ants is set to $n/2$. The values α and β are set to $\alpha = \beta = 5$. The initial pheromone value is set to 2. The value ι for the pheromone update is set to $\iota = 0.0005$. Finally, we use a trail persistence rate of $\rho = 0.95$.

6 Computational Results

We describe the computational tests which we performed in order to compare the solution quality and performance of the two approaches described in Sections 4 and 5. Both algorithms were coded in C and and run on a Pentium IV 2600 MHz.

6.1 Test Setting

The metaheuristic algorithms have been tested both on the a test set obtained by modifying instances from the CVRP literature and on a real-world test set. The instances of the first test set can be downloaded from the internet at http://www.univie.ac.at/bwl/prod/.

We generated random problem instances in the following way (see Tables 1 and 2). The number of customers and the graph (V, E) are taken

from the first seven VRP instances given by Christofides et al. [6], and are denoted as CMT01-CMT07. The instances $8 - 14$ in [6] have the same structure than the first seven and differ only in an additional route length constraint, and therefore have not been considered in this work.

We developed a problem generator for the different demands. We introduced three different customer types: a minimum demand customer, who orders a small quantity of the different products (e.g., a handyman), a mean demand customer, who orders a reasonable quantity of the different products, and a maximum demand customer (e.g., a do-it-yourself store), who orders a large quantity of the different products, respectively. The amounts of different chipboards for the different customer types are given in Table 1 (e.g., a minimum demand customer orders between 0 and 2 long chipboards). The ordered amounts are drawn according to a uniform distribution in the intervals given, and reflect real-world typical demands. We considered the following heights: the height of a pallet is 5, the height of the long (respectively short, doors and heavy-use) chipboards is 5 (respectively 1, 1 and 3), and the loading height of a truck is 200.

Table 1: Demands for the three different types of customers.

	min. demand	mean demand	max. demand
long chipboards	$0 \leq d < 2$	$4 \leq d < 6$	$7 \leq d \leq 11$
short chipboards	$0 \leq d < 9$	$9 \leq d < 15$	$21 \leq d \leq 33$
chipboards for doors	$0 \leq d < 8$	$8 \leq d < 12$	$16 \leq d \leq 22$
heavy use chipboards	$0 \leq d < 2$	$4 \leq d < 6$	$8 \leq d \leq 11$

For each original instance we generated three different order combinations by considering different percentages in the assignment of a customer to a type of order (see Table 2). Each combination is denoted as a class (and referred to as *cl* in the following tables). Note that in all the instances we

Table 2: Three different configurations of customer orders.

Class	min. demand	mean demand	max. demand
1	40%	10%	50%
2	33%	34%	33%
3	10%	80%	10%

created, each client demands for a number of short items which is lower or equal to the number of piles, because this is the most usual condition in the real-world situation.

In addition, we also compared the approaches by applying them to real-world data. The real-world data were provided by a large Austrian wood products retailer located 100 kilometers north of Vienna. For the locations of the customers we used the real customer location data of a typical week in the rural regions around Vienna (depicted in Figure 5). The real demands of the customers were slightly modified for privacy reasons. We generated five problem instances by modifying the demands and re-assigning them randomly to the different customers.

We generated the routing costs c_{ij} between each pair of customers as the Euclidean distances between their coordinates. We note that both algorithms work for CVRPs with general cost matrices and not only for the Euclidean CVRP.

6.2 Results on randomly created instances

In Table 3 we present the results obtained by running our algorithms on instances derived from the CVRP literature. The first columns give the name of the original instance, the number of customers, the class and the total number of items, respectively. For the Tabu Search algorithm we report the solution quality z_{TS}, the required runtime in seconds when the best solu-
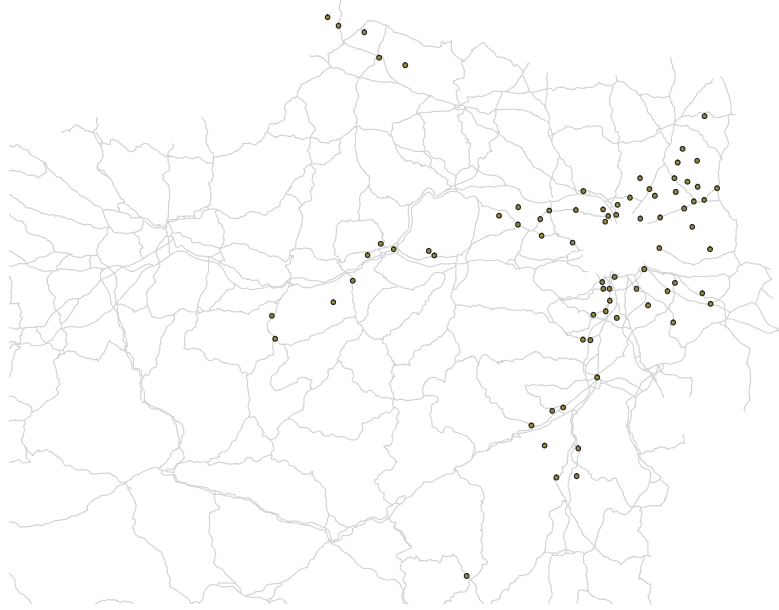
Figure 5: Customer locations for the real-world instances.

tion was found (sec_{inc}) and the total runtime of the algorithm sec_{tot}. We recall that the Tabu Search algorithm was halted after two hours of runtime or 250 000 iterations. Concerning the ACO, since it is a randomized algorithm we performed 10 repetitions for each instance. We report the minimum, average and maximum solution value (z_{ACO}^{min}, $\overline{z}_{ACO}$ and z_{ACO}^{max}, respectively), the minimum, average and maximum runtime in seconds when the best solution was found (sec_{inc}^{min}, $\overline{sec}_{inc}$ and sec_{inc}^{max}, respectively) and the minimum, average and maximum total runtime (sec_{tot}^{min}, $\overline{sec}_{tot}$ and sec_{tot}^{max}, respectively). The ACO algorithm was halted after two hours of runtime or 100 000 iterations (note that the maximum time limit of two hours is reached only for the instances with 199 customers).

We can see that for the small problems the Tabu Search algorithm outperforms the ACO algorithm, whereas for the larger problems the ACO algorithm finds better results. The ACO algorithm outperforms the Tabu

Table 3: Performance of the Tabu Search and Ant Colony Optimization algorithms on instances from the CVRP literature.

Instance				Tabu Search			Ant Colony Optimization								
(V,E)	n	cl	M	z_{TS}	sec_{inc}	sec_{tot}	z_{ACO}^{min}	$\overline{z}_{ACO}$	z_{ACO}^{max}	sec_{inc}^{min}	$\overline{sec}_{inc}$	sec_{inc}^{max}	sec_{tot}^{min}	$\overline{sec}_{tot}$	sec_{tot}^{max}
CMT01	50	1	170	**594.06**	12.4	2966.9	594.06	594.56	596.98	1.2	4.9	6.4	11.9	12.3	12.9
		2	180	**620.91**	312.7	2229.2	622.64	622.82	623.08	4.3	5.9	7.6	11.2	11.5	12.0
		3	193	**636.95**	1261.8	1809.0	637.41	638.97	640.93	3.7	4.9	6.0	11.1	11.4	11.8
CMT02	75	1	278	990.51	119.1	2599.8	978.66	**981.07**	985.77	28.0	31.8	37.6	69.2	73.7	76.5
		2	271	**912.62**	3409.6	3594.9	912.66	915.34	917.55	25.4	30.6	33.4	67.5	72.7	78.9
		3	293	920.61	336.6	2694.8	916.48	**917.84**	922.86	22.0	29.6	37.3	67.0	72.4	75.1
CMT03	100	1	355	1209.46	3481.8	4885.6	1194.66	**1208.72**	1224.53	121.2	144.6	182.0	314.8	364.3	395.9
		2	380	1247.54	2740.1	3585.3	1234.95	**1242.87**	1248.16	120.1	139.4	176.3	306.9	349.2	381.8
		3	387	1196.15	3576.6	3994.0	1185.72	**1187.49**	1189.26	104.9	130.8	149.7	307.1	366.8	386.9
CMT04	150	1	529	1672.70	2660.6	7200.0	1648.39	**1660.55**	1676.81	1073.3	1599.5	3891.7	3139.3	3978.3	4316.6
		2	548	1603.09	4925.4	7200.0	1566.90	**1575.28**	1580.15	1078.1	1513.0	3317.1	3122.7	3998.5	4351.7
		3	580	1592.68	4902.5	7092.1	1578.06	**1583.65**	1589.18	1046.7	1269.7	1721.5	2881.9	3940.5	4233.7
CMT05	199	1	712	2107.49	1717.4	7200.1	2077.57	**2085.68**	2092.32	4518.1	5501.5	7200.0	7200.0	7200.0	7200.0
		2	707	1879.00	6611.1	7200.0	1853.98	**1863.42**	1872.34	4680.2	5416.6	7200.0	7200.0	7200.0	7200.0
		3	773	2042.28	4282.6	7200.0	1988.83	**1999.74**	2014.74	3823.0	5090.5	7200.0	7200.0	7200.0	7200.0
CMT06	120	1	421	2292.03	522.9	6406.4	2260.46	**2269.56**	2285.63	631.3	837.0	1237.1	1222.6	1466.8	1628.9
		2	447	2122.34	1784.2	7200.0	2087.84	**2107.66**	2120.44	550.8	719.5	938.5	1113.4	1327.1	1533.4
		3	467	2237.86	2855.0	4927.7	2186.59	**2195.66**	2203.58	468.8	666.5	928.3	1150.6	1368.7	1491.4
CMT07	100	1	346	1154.31	1944.3	5918.0	1142.78	**1153.45**	1161.34	128.7	150.2	234.2	278.0	302.9	331.9
		2	375	**1237.43**	148.6	4088.1	1239.84	1248.83	1259.35	109.3	157.5	215.9	260.7	303.5	326.4
		3	388	1183.18	3859.6	4052.3	1181.84	**1182.92**	1184.09	95.6	109.8	147.4	245.5	274.8	289.7
Average				1402.53	2450.7	4954.5	1385.25	**1392.19**	1399.48	887.4	1121.6	1660.4	1722.9	1899.8	1977.9

Search algorithm by 0.7 % on all the 21 problem instances with respect to solution quality, and in 16 out of the 21 problem instances it provides better results. Finally, the ACO algorithm is generally faster than the Tabu Search: it needs 1121.6 seconds (against 2450.7 seconds) to find the best solution, and 1899.8 seconds (against 4954.5 seconds) to run to completion.

In Table 4 we present the results for each group of instances derived from the same graph of the CVRP literature. On the small problem instances the Tabu Search outperforms the ACO algorithm and provides a solution quality of 0.2 % lower than the ACO algorithm. The Tabu Search algorithm provides also better results for the 100 customer instances, while for all the other instances the ACO algorithm finds better solutions.

Table 4: Average values per (V, E) instance (three instances per line).

Instance		Tabu Search			Ant Colony Optimization			
(V, E)	n	z_{TS}	sec_{inc}	sec_{tot}	z_{ACO}^{min}	$\overline{z}_{ACO}$	$\overline{sec}_{inc}$	$\overline{sec}_{tot}$
CMT01	50	**617.30**	529.0	2335.0	618.04	618.78	5.2	11.7
CMT02	75	941.24	1288.4	2963.2	935.93	**938.08**	30.6	72.9
CMT03	100	1217.72	3266.2	4155.0	1205.11	**1213.03**	138.3	360.1
CMT04	150	1622.82	4162.8	7164.0	1597.78	**1606.49**	1460.8	3972.4
CMT05	199	2009.59	4203.7	7200.0	1973.46	**1982.95**	5336.2	7200.0
CMT06	120	2217.41	1720.7	6178.1	2178.30	**2190.96**	741.0	1387.5
CMT07	100	**1191.64**	1984.2	4686.1	1188.15	1195.07	139.2	293.7
Average		1402.53	2450.7	4954.5	1385.25	**1392.19**	1121.6	1899.8

In Table 5 we present the average performances for each class used for the generation of the demands. We can see how the average solution value found by the ACO always outperforms the one obtained by the Tabu Search.

The above results were obtained by using the *HL* heuristic for solving the 1-VLP. We have also recomputed the solutions to some instances by solving the 1-VLP with the exact dynamic programming method of Sec-

Table 5: Average values per class (seven instances per line).

Instance		Tabu Search			Ant Colony Optimization			
cl	$\overline{M}$	z_{TS}	sec_{inc}	sec_{tot}	z_{ACO}^{min}	$\overline{z}_{ACO}$	$\overline{sec}_{inc}$	$\overline{sec}_{tot}$
1	2503	1431.51	1494.1	5311.0	1413.80	**1421.94**	1181.4	1914.0
2	2314	1374.70	2847.4	5013.9	1359.83	**1368.03**	1140.4	1894.6
3	2203	1401.39	3010.7	4538.6	1382.13	**1386.61**	1043.1	1890.7
Average		1402.53	2450.7	4954.5	1385.25	**1392.19**	1121.6	1899.8

tion 3.3. We found out that the computation time was increased by about 100% while the solution quality was improved just by 0.18%, on average. Also using algorithm HL2 of Section 3.4 proved to be less efficient, yielding to a worsening of 3% in the average solution value. This was mainly due to a substantial increase in the CPU time needed to compute the loading patterns, and consequently to a reduction in the CPU time spent on the routing part.

6.3 Results on real-world instances

In Table 6 we give the results obtained on instances derived from the real-world transportation problem. In this case too we can note a difference in the performance of the two algorithms. The ACO algorithm provides the best solution value for four instances, but the difference in the average solution quality in comparison with the Tabu Search is only 0.3 %. The ACO is also much faster, since it needs on average a CPU time which is 1/100 of the CPU time needed by the Tabu Search.

This project was intended to show how an automated system of packing and routing could work. The management was quite impressed by the efficient routing and packing achieved but at the moment they did not want to devote any significant resources (with respect to money and time)

in order to develop such an automated packing and routing system. Also they did not want to provide the routes actually used by the trucks. However the management is now confident that such a system could be useful and they will consider this as one of the next projects.

Table 6: Performance of the Tabu Search and Ant Colony Optimization algorithms on real-world instances.

Instance			Tabu Search			Ant Colony Optimization			
(V, E)	n	M	z_{TS}	sec_{inc}	sec_{tot}	z_{ACO}^{min}	$\overline{z}_{ACO}$	$\overline{sec}_{inc}$	$\overline{sec}_{tot}$
WOOD01	76	142	1616.68	2694	7200.1	1594.88	**1599.71**	36.9	68.0
WOOD02	76	141	**1483.94**	7098.4	7200.0	1481.28	1491.98	49.8	76.8
WOOD03	76	142	1389.88	1234.8	7200.0	1384.73	**1388.41**	37.7	68.0
WOOD04	76	144	1485.71	6380.8	7141.6	1469.97	**1477.19**	41.0	72.1
WOOD05	76	184	1494.18	4190.4	6323.4	1484.82	**1490.08**	40.0	70.5
Average			1494.07	4319.78	7013.0	1483.14	**1489.47**	41.1	71.1

6.4 On the minimization of the number of vehicles

We finally present some results concerning the combined minimization of the routing cost and of the number of vehicles for the ACO algorithm.

In Table 7 we give the results for different δ values (see Equation 17) in the objective function of the ACO algorithm for the instances derived from the CVRP literature. When only the routing is considered (i.e., $\delta = 1$), on average the solution value is 1392.19 and the number of vehicles used is 17.9. When we also consider the loading aspect in the solution construction phase (e.g., we put a weight of 10 % on the loading by considering $\delta = 0.9$) we can reduce the number of the required vehicles to 17.6, in spite of an increase in the routing cost to 1488.67. If the weight on the loading is increased further (20 %), we can reduce the required number of vehicles to 17.5, but having an increase in the routing cost to 1582.25 on average.

Table 7: Sensitivity analysis of different δ values in the ACO algorithm

		0.8		0.85		0.9		0.95		1	
	cl	$\overline{z}_{ACO}$	#	$\overline{z}_{ACO}$	#	$\overline{z}_{ACO}$	#	$\overline{z}_{ACO}$	#	$\overline{z}_{ACO}$	#
CMT01	1	603.86	7	599.91	7	598.90	7	595.93	7	594.56	7
	2	749.27	7	716.74	7	684.29	7	652.88	7	622.82	7
	3	645.62	8	645.00	8	642.71	8	641.04	8	638.97	8
CMT02	1	1038.94	13	1015.62	13	1002.00	13	988.67	13	981.07	13
	2	1153.95	11	1121.80	11	1069.67	11	968.32	11.6	915.34	12
	3	965.51	11.9	941.32	12	945.44	11.9	933.29	12	917.84	12
CMT03	1	1398.19	16	1375.17	16	1325.27	16	1251.93	16.8	1208.72	16.8
	2	1401.42	17	1366.85	17	1331.62	17	1290.92	17	1242.87	17
	3	1242.78	15.9	1240.03	15.9	1214.28	15.9	1201.69	15.9	1187.49	16
CMT04	1	1961.36	23.9	1840.52	24	1791.73	24	1717.85	24	1660.55	24.2
	2	2049.09	22.4	1876.16	22.8	1784.74	23	1670.35	23	1575.28	23
	3	1649.65	23	1638.20	23	1623.85	23	1597.02	23	1583.65	23
CMT05	1	2462.47	32	2342.48	32	2303.33	32	2171.84	32.2	2085.68	32.7
	2	2418.68	27	2282.98	27	2037.21	27.8	1949.83	28	1863.42	28
	3	2045.16	31	2055.26	31	2040.05	31	2018.12	31	1999.74	31.1
CMT06	1	2645.22	18	2606.11	18	2513.42	18	2433.85	18.4	2269.56	19
	2	2429.27	18	2389.46	18	2333.19	18	2262.99	18	2107.66	18.1
	3	2373.26	18.2	2292.42	18.6	2227.36	18.9	2202.02	18.9	2195.66	19
CMT07	1	1389.25	15	1350.00	15	1284.26	15	1213.65	15	1153.45	15.5
	2	1355.52	16	1331.63	16	1295.07	16	1266.51	16	1248.83	16.5
	3	1248.88	16	1233.66	16	1213.72	16	1198.37	16	1182.92	16
Average		1582.25	17.5	1536.25	17.5	1488.67	17.6	1439.38	17.7	1392.19	17.9

In Table 8 we give the same results for the real-world instances. When we consider only the routing we have an average solution cost of 1489.47 and an average number of vehicles equal to 10. For $\delta = 0.9$ the number of required vehicles reduces to 9.7 but the routing cost increases to 1615.52. Moreover, when $\delta = 0.8$ the number of required vehicles reduces to 9.4 but the routing cost increases to 1746.33, on average.

7 Conclusion and Future Research

We presented a new combinatorial optimization problem deriving from a real-world transportation case in which items have to be loaded on vehicles and then delivered to customers with minimum total cost. The prob-

Table 8: Sensitivity analysis of different δ values in the ACO algorithm

	0.8		0.85		0.9		0.95		1	
	$\overline{z}_{ACO}$	#	$\overline{z}_{ACO}$	#	$\overline{z}_{ACO}$	#	$\overline{z}_{ACO}$	#	$\overline{z}_{ACO}$	#
WOOD01	1843.00	10	1786.53	10	1734.28	10	1668.05	10	1599.71	10
WOOD02	1773.28	9	1698.44	9	1649.77	9	1581.83	9	1491.98	10
WOOD03	1698.64	9	1647.41	9	1522.57	9.8	1453.55	10	1388.41	10
WOOD04	1790.20	9.2	1676.78	9.6	1608.61	9.7	1554.60	10	1477.19	10
WOOD05	1626.51	10	1598.39	10	1562.36	10	1541.98	10	1490.08	10
Average	1746.33	9.4	1681.51	9.5	1615.52	9.7	1560.00	9.8	1489.47	10

lem combines together the vehicle routing problem and the parallel processor scheduling problem. We developed heuristics and implemented a dynamic programming algorithm for solving the loading of a single vehicle. These algorithms were integrated within two different metaheuristic approaches, based on a Tabu Search and an Ant Colony Optimization scheme, providing interesting results. We extended the standard Savings based Ant System by using an additional memory for the optimization of the loading of the vehicles. We performed a sensitivity analysis with the Ant System and showed that an optimized packing reduces the number of required vehicles by 6 %, but leads, for the real-world instances, to an increase of 17 % in the route lengths. Both heuristics were tested on instances derived from the literature and on instances derived from the real-world problem.

We note that considering the more general case in which the length of an item I_i^k could be $1 \leq l_i^k \leq p$ would lead to a more complex two-dimensional rectangular loading problem, similar to problem 2L addressed by Iori et al. [23] for the 2L-CVRP. Since ACO provides good average results for the MP-VRP, as future research we will extend this approach and study possible applications to the 2L-CVRP. As future research, we will

also investigate better heuristics and exact algorithms for the problem of loading the items into a vehicle.

Acknowledgements

References

[1] A.C.F. Alvim, F. Glover, C.C. Ribeiro, and D.J. Aloise. A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics*, 10(2):205–229, 2004.

[2] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52(5):723–738, 2004.

[3] J. Bramel and D. Simchi-Levi. Set-covering-based algorithms for the capacitated vrp. In *The vehicle routing problem*, pages 85–108. Society for Industrial and Applied Mathematics, 2001.

[4] B. Brugger, K. F. Doerner, R. F. Hartl, and M. Reimann. AntPacking - An Ant Colony Optimization Approach for the One-Dimensional Bin Packing Problem. In *Lecture Notes in Computer Science 3004*, pages 41–50. Springer, 2004.

[5] B. Bullnheimer, R. F. Hartl, and Ch. Strauss. A new rank based version of the ant system: a computational study. *Central European Journal of Operations Research*, 7(1):25–38, 1999.

[6] N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In *Combinatorial Optimization*, pages 315–338. Wiley, 1979.

[7] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[8] Jr. Coffman, E. G., G. Galambos, S. Martello, and D. Vigo. Bin packing approximation algorithms: Combinatorial analysis. In *Handbook of Combinatorial Optimization*. Kluwer, Boston, 1998.

[9] J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997.

[10] J.-F. Cordeau and G. Laporte. Tabu search heuristics for the vehicle routing problem. In *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, pages 145–163. Kluwer, Boston, 2004.

[11] G. A. Croes. A method for solving traveling salesman problems. *Operations Research*, 6(6):791–801, 1958.

[12] M. Dell'Amico and S. Martello. Optimal scheduling of tasks on identical parallel processors. *ORSA Journal on Computing*, 7(2):191–200, 1995.

[13] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics*, 26(1):29–41, 1996.

[14] K. Fleszar and K. S. Hindi. New heuristics for one-dimensional bin-packing. *Computers & Operations Research*, 29(7):821–839, 2002.

[15] A. Frangioni, E. Necciari, and M. G. Scutellá . A multi-exchange neighborhood for minimum makespan parallel machine scheduling problems. *Journal of Combinatorial Optimization*, 8(2):195–220, 2004.

[16] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, and R. F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006.

[17] M. Gendreau, A. Hertz, and G. Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094, 1992.

[18] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, 1994.

[19] M. Gendreau, M. Iori, G. Laporte, and S. Martello. A heuristic algorithm for a routing and container loading problem. *Transportation Science*, 2006 (to appear).

[20] M. Gendreau, M. Iori, G. Laporte, and S. Martello. A tabu search approach to vehicle routing problems with two-dimensional loading constraints. *Networks*, 2006 (to appear).

[21] M. Gendreau, G. Laporte, and J. Y. Potvin. Metaheuristics for the capacitated vrp. In *The vehicle routing problem*, pages 129–154. Society for Industrial and Applied Mathematics, 2001.

[22] R. L. Graham, E .L. Lawler, J. K. Lenstra, and A. H. G Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[23] M. Iori, J. J. Salazar González, and D. Vigo. An exact approach for the vehicle routing problem with two dimensional loading constraints. *Transportation Science*, 2006 (to appear).

[24] G. Laporte and Y. Nobert. Exact algorithms for the vehicle routing problem. *Annals of Discrete Mathematics*, 31:147–184, 1987.

[25] G. Laporte and F. Semet. Classical heuristics for the capacitated vrp. In *The vehicle routing problem*, pages 109–128. Society for Industrial and Applied Mathematics, 2001.

[26] A. N. Letchford and J. J. Salazar González. Projection results for vehicle routing. *Mathematical Programming*, 105(2 - 3):251–274, 2006.

[27] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Chichester, 1990.

[28] D. Mester and O. Bräysy. Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research*, 2006 (to appear).

[29] D. Naddef and G. Rinaldi. Branch-and-cut algorithms for the capacitated vrp. In *The vehicle routing problem*, pages 53–84. Society for Industrial and Applied Mathematics, 2001.

[30] I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41(4):421–451, 1993.

[31] D. Pisinger. Heuristics for the container loading problem. *European Journal of Operational Research*, 141(2):382–392, 2002.

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

[33] M. Reimann, M. Stummer, and K. F. Doerner. A savings based ant system for the vehicle routing problem. In *Proceedings of the Genetic and Evolutionary Computation Conference 2002*, pages 1317–1325. Morgan Kaufmann, 2002.

[34] A. Scholl, R. Klein, and C. Juergens. Bison: a fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers & Operations Research*, 24(7):627–645, 1997.

[35] P. Toth and D. Vigo. Exact algorithms for vehicle routing. In *Fleet Management and Logistics*. Kluwer, Boston, 1998.

[36] P. Toth and D. Vigo. Branch-and-bound algorithms for the capacitated vrp. In *The vehicle routing problem*, pages 29–51. Society for Industrial and Applied Mathematics, 2001.

[37] P. Toth and D. Vigo. *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications, Philadelphia, 2001.

[38] P. Toth and D. Vigo. The granular tabu search and its application to the vehicle routing problem. *INFORMS Journal on Computing*, 15(4):333–346, 2003.

[39] F. Vanderbeck. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, 86(3):565–594, 1999.