Erasmus University Rotterdam

Bachelor Thesis Econometrics & Operational Research

Quantitative Logistics

A Tabu Search Heuristic for the Vehicle Routing Problem with Two-Dimensional Loading Constraints

Author:
Jeroen Vester
Student number: 362010

Supervisor and first reader:
Dr. W van den Heuvel

Second reader:
Dr. T.A.B. Dollevoet

July 15, 2015

Abstract

This thesis evaluates the paper of Gendreau et al. (2008), where the well-known *Capacitated Vehicle Routing Problem* (CVRP) is addressed. They incorporate in this problem, which usually only depends on weight constraints, two-dimensional loading constraints. The CVRP minimizes the transportation costs resulting from delivering demanded goods to customers, carried out by a fleet of vehicles from a single depot. Since we now want a load of items to fit two-dimensionally in a vehicle, a feasibility check of the two-dimensional packing (2L) is executed for every vehicle. In the paper a Tabu Search heuristic is proposed for solving the combined problem, denoted as 2L-CVRP, since it is NP-hard and particularly hard to solve in practice. The effectiveness of the heuristic is evaluated with computational experiments.

Contents

1 Introduction

The *Capacitated Vehicle Routing Problem* (CVRP) is one of the most extensively studied combinatorial optimization problems, due to its relevance in delivery systems of which more and more arise. The CVRP seeks to service a set of customers with a fleet of vehicles, where the vehicles have limited carrying capacity of the goods that need to be delivered, expressed in weight. Despite of the relevance of this problem, applying this model to real-world problems is not always justifiable, since real-world problems often have additional constraints, such as the size of packages, two- or three-dimensional, or the time in which a delivery has to be made.

In Gendreau et al. (2008) one possible extension of the CVRP is discussed, namely adding two-dimensional loading constraints. This problem is referred to as 2L-CVRP (*Two-Dimensional Loading Capacitated Vehicle Routing Problem*). Gendreau et al. (2008) propose to solve this problem with a Tabu Search (TS) heuristic. The transportation business often handles rectangular packages that cannot be stacked upon each other, because of their weight or fragility, and then two-dimensional loading constraints apply. The goal of this thesis is to implement the TS heuristic for 2L-CVRP and compare the found solutions with the solutions of Gendreau et al. (2008).

Adding two-dimensional loading constraints to the CVRP makes it significantly more difficult to solve. This problem was first examined by Iori et al. (2007), where only *sequential loading* was considered. In Gendreau et al. (2008) also *unrestricted loading* is considered. With sequential loading the order of visits to customers is taken into account when the items are packed, with unrestricted loading it is not. Before the Tabu Search heuristic of Gendreau et al. (2008) was developed, only exact approaches for solving 2L-CVRP existed. The best performing approaches were branch-and-cut (-and-price) algorithms, which were only able to solve 2L-CVRP for relatively small instances. (Iori et al., 2007)

The *Two-Dimensional Bin Packing Problem* (2BPP) solves the problem of packing two-dimensional rectangular items in a rectangular space and is chosen by Gendreau et al. (2008) to be combined with CVRP for solving 2L-CVRP. The 2BPP minimizes the number of identical rectangular bins needed to pack a number of rectangularly shaped items. For both of the optimization problems, CVRP and 2BPP, several meta-heuristic approaches are developed, since both are NP-hard. Gendreau et al. (2008) justify the choice of developing a Tabu Search heuristic for 2L-CVRP by the fact that this approach was successful for both of these optimization problems separately. Furthermore they mention that since finding a feasible solution of 2L-CVRP is already strongly NP-hard, techniques based on populations of solutions, such as Genetic Algorithms and Scatter Search, would need to handle too much infeasible solutions.

2 Further literature

Since Gendreau et al. (2008) more heuristics for solving 2L-CVRP have been developed, which will be discussed here. In Zachariadis et al. (2009) the Tabu Search of Gendreau et al. (2008) is further extended. They developed the Guided Tabu Search (GTS), where the principles of Guided Local Search are incorporated. Here GTS is proven to slightly outperform the solutions of the TS heuristic. Then this heuristic was even further extended and improved by Leung et al. (2011), who present the Extended Guided Tabu Search (EGTS), which utilizes random moves and an aspiration criterion to improve the heuristic.

For solving 2L-CVRP also heuristics different from Tabu Search are found in literature regarding this subject. In Leung et al. (2010) Simulated Annealing (SA) is used to solve 2L-CVRP, introducing a new scoring-based heuristic in order to improve packing. Fuellerer et al. (2009) present the Ant Colony Optimization (ACO), an algorithm which explores and evaluates populations of solutions, with proven satisfactory results. Later the Greedy Randomized Adaptive Search Procedure combined with Evolutionary Local Search (GRASP $\times$ ELS) algorithm was developed by Duhamel et al. (2011), where the packing problem was addressed by solving a resource constrained project scheduling problem (RCPSP). Although the problem is only solved for unrestricted packing, the algorithm is proven to outperform all previous existing methods. Zachariadis et al. (2013) propose another meta-heuristic for solving 2L-CVRP, the Promise Routing-Memory Packing (PRMP), again outperforming previous methods. Recently, solving 2L-CVRP has been even more improved by Wei et al. (2015), who proposed a Variable Neighborhood Search heuristic.

Solving 3L-CVRP, similar to 2L-CVRP, but with three-dimensional loading constraints, has also been addressed by numerous researchers. In Gendreau et al. (2006) the problem is solved by utilizing a Tabu Search heuristic, in Tarantilis et al. (2009) GTS is used, in Fuellerer et al. (2010) ACO and in Lacomme et al. (2013) GRASP $\times$ ELS. Furthermore, Ma et al. (2011) developed a heuristic combining tabu search with local search and Bortfeldt (2012) introduces a hybrid algorithm, addressing the routing problem with tabu search and the loading problem with a tree search algorithm.

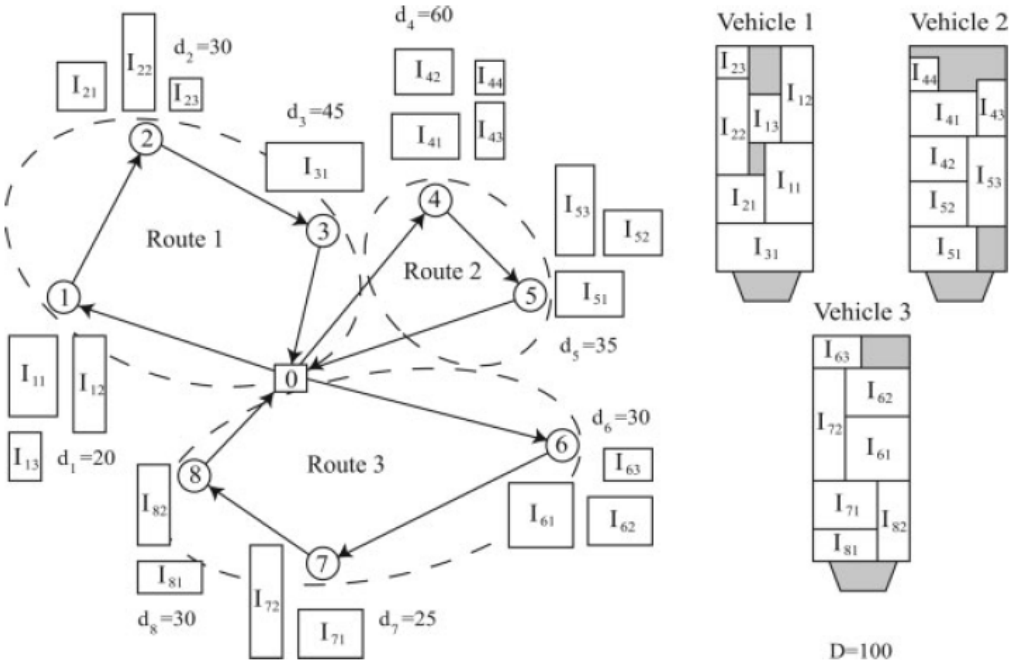


Figure 1: Example of 2L-CVRP

3 Problem Description

The problem is described as follows by Gendreau et al. (2008). Say we have a graph $G = (V, E)$, where V is a set of $n + 1$ vertices, consisting of a depot (vertex 0) and n customer vertices, and E is the set of edges (i, j) with a corresponding cost c_{ij} for every edge. We have v identical vehicles at our disposal, each having a weight capacity D and a rectangular loading area A of size $W \times H$, where W is the width of the vehicle and H is the length (in Gendreau et al. (2008) referred to as the *height* of the vehicle). Every client i $(i = 1, \ldots, n)$ has a demand of m_i items of total weight d_i and each item I_{il} $(l = 1, \ldots, m_i)$ has a width $w_{il} < W$ and a height $h_{il} < H$. Let $a_i = \sum_{l=1}^{m_i} w_{il} h_{il}$ be the total loading area of the demand of customer i.

Furthermore some assumptions are made. Firstly, every item is assumed to have a fixed orientation, meaning that it can be packed in only one single way, the w-edge of an item should be parallelled to the W-edge of the vehicle and the h-edge to the H-edge. This assumption is realistic, because for pallet loading, the loading fork usually has a fixed orientation it can load an item in. Secondly it is assumed that every customer should be serviced by only one single vehicle. The loading area of a vehicle is represented in the positive quadrant of a Cartesian coordinate system, with the coordinates $(0, 0)$ as its bottom left corner, and the W-edge and the H-edge parallelled to the x- and y-axis respectively. The rear of the vehicle will then be the line between $(0, H)$ and (W, H), the rear being the side of the vehicle where the tailgate is located. The position of an item I_{il} in a loading pattern is represented by the coordinates of its bottom left corner, (x_{il}, y_{il}).

A route assigned to vehicle k, visiting a set of customers $S(k) \subseteq \{1, \ldots, n\}$, must satisfy the following constraints:

***Weight constraint*:** the total weight $\sum_{i \in S(k)} d_i$ must not exceed the vehicle capacity D;

***Loading constraint*:** there must be a feasible (non-overlapping) loading of the items in

$$I(k) = \bigcup_{i \in S(k)} \bigcup_{l \in \{1, \ldots, m_i\}} I_{il} \tag{1}$$

into the $W \times H$ loading area.

An example of a 2L-CVRP solution, also given in Gendreau et al. (2008), is visually represented in Figure 1.

The objective is to find a partition of the clients in at most v subsets, such that for each subset $S(k)$ the constraints listed above hold. In Gendreau et al. (2008) two variations of this problem are considered, the *Unrestricted* 2L-CVRP and the *Sequential* 2L-CVRP. The first variation is the problem as was depicted above, for the second variation one constraint needs to be added for every vehicle k:

***Sequence constraint*:** a vehicle k must be packed such that when a client $i \in S(k)$ is visited, the items in the demand of this client $\bigcup_{l\in\{1,\ldots,m_i\}} I_{il}$ can be unloaded in a sequence of straight movements parallel to the H-edge of the vehicle, and thus no other items need to be replaced for unloading.

In other words, when visiting client i, no items of customers that will be visited later on the route may be loaded between the items of client i and the rear of the vehicle. Sequential loading might be useful in situations where items are very big, heavy or fragile, or have another characteristic that makes moving them complicated and time consuming, or even impossible.

4 Methodology

Gendreau et al. (2008) propose a Tabu Search Heuristic for solving the 2L-CVRP, considering sequential and unrestricted loading. We evaluate the TS heuristic for the unrestricted case in this research. Tabu Search is a meta-heuristic, based on the Steepest Descent method and declaring certain solutions tabu. In contrast to Steepest Descent, in every iteration Tabu Search can choose not only a better solution than the current one, but also a worse solution. By declaring recently visited solutions tabu, TS can escape from local optima.

TS starts from an initial, usually feasible, solution and from there on moves to the best solution in the *neighborhood* of the current solution. This could mean that the TS moves to a worse solution, if the current solution is better than all the solutions in the neighborhood. In order to make sure that the heuristic does not cycle in a local optimum, a visited solution is declared tabu for a predefined number of iterations. TS is often extended with a *diversification* phase and an *intensification* phase. The first one facilitates the TS to escape a poor or already extensively evaluated solution area, the second one enables the TS to more extensively investigate a promising solution area.

The TS used by Gendreau et al. (2008) is allowed to move to an infeasible solution. A tour always satisfies the sequence constraint, in case of the sequential 2L-CVRP, but can be *weight-infeasible* if the weight of a freight exceeds the capacity D, or *load-infeasible* if the needed loading area of a freight is larger than the loading area A of a vehicle. This means that the needed height is larger than the height H of the vehicle, since the packing algorithm does not allow for a packing that exceeds the width W of the vehicle. A *penalty* is assigned to moves that lead to an infeasible tour, proportional to the violation of the constraints.

Every move considered by the TS means a different distribution of items over the vehicles. For checking if the items $I(k)$ assigned to vehicle k (see equation (1)) violate the loading constraints, Gendreau et al. (2008) give the heuristics $\mathrm{LH_{2SL}}$ and $\mathrm{LH_{2UL}}$ for the sequential and unrestricted case, respectively. Within these heuristics a procedure $\mathrm{TP_{2SL}}$ or $\mathrm{TP_{2UL}}$ is called in which the items are placed onto a strip of width W in a predefined order, in such a way that for every item the percentage of the items perimeter touching the edges of the loading area or other items already packed, also known as the touching perimeter, is maximized. $\mathrm{LH_{2SL}}$ and $\mathrm{LH_{2UL}}$ iteratively use $\mathrm{TP_{2SL}}$ and $\mathrm{TP_{2UL}}$, respectively, to pack items in different orders, until a load-feasible packing is found, with a predefined maximum number of iterations. Each item is packed in the so-called *normal position*, described in Gendreau et al. (2008) as that the bottom edge of an item touches either the bottom of the strip or the top of another item and that the left edge of an item touches either the left edge of the strip or the right edge of another item.

4.1 Heuristics for checking the loading constraints

Since we initially evaluate the unrestricted case, we will only describe $\mathrm{LH_{2UL}}$ and $\mathrm{TP_{2UL}}$ in further detail. Both procedures combined check for load feasibility for one vehicle k. $\mathrm{TP_{2UL}}$, represented in Figure 2, is called by $\mathrm{LH_{2UL}}$ with a given order of items, sorted by non-increasing width, breaking ties by decreasing height. Sequential calls of the $\mathrm{TP_{2UL}}$ by $\mathrm{LH_{2UL}}$ are done with altered sequences of the items. Procedure $\mathrm{TP_{2UL}}$ can terminate in two possible ways, it finds a load-feasible packing within the loading area $W \times H$ or it finds a load-infeasible packing on a loading area of $W \times Hstrip$ with $Hstrip > H$, which is to be penalized in the Tabu Search.

Procedure $\mathrm{TP_{2UL}}$ is based on a packing algorithm introduced by Lodi et al. (1999) and works as follows. The first item of the sequence is packed at position (0,0), the following item, say I_{il} is packed at position (x, y) that maximizes the touching perimeter and satisfies that the loading area does not exceed the height of the vehicle. Maximizing the touching perimeter favors patterns that avoid "trapping" small areas that are hard to use for other items, according to Lodi et al. (1999). The overall time complexity of $\mathrm{TP_{2UL}}$ can be derived as follows. Let $\overline{m}$ be the total number of items required by the clients of the considered route. For each item, the number of candidate normal packing positions (set P), which needs to be redetermined for every item, is $O(\overline{m})$. The touching perimeter computation of line 7 requires $O(\overline{m})$ time, and thus the overall time complexity is $O(\overline{m}^2)$.

Input: an item sorting $I(k)$

procedure $\mathrm{TP}_{2\mathrm{UL}}(I(k))$
  pack the first item of $I(k)$ in position (0,0) and set $\overline{H}$ at its height;
  for each successive item, say I of size (w, h), of $I(k)$ **do**
    P :={set of normal packing positions (x, y) : (a),(b) below hold}
      (a) $x + w \leq W$;
      (b) the rectangle of sides $([(x, y), (x + w, y)], [(x, y), (x, y + h)])$ is empty;
    $\overline{P} := \{(x, y) \in P : y + h \leq H\}$;
    if $\overline{P} \neq \emptyset$ **then**
      pack I in the position $(x, y) \in \overline{P}$ producing maximum touching perimeter
    else pack I in the position $(x, y) \in P$ for which $y + h$ is a minimum;
    end if
    $\overline{H} := \max\{\overline{H}, y + h\}$
  end for
  return $\overline{H}$
end procedure

Figure 2: Inner heuristic for packing

After every placement of an item in the loading area, the set of normal packing positions needs to be adjusted. One normal packing positions is no longer available, obviously, since one of the packing positions is needed for placing the item in question. This implies that at least one packing position should be removed from the set of normal packing positions. However, this also implies that new packing positions become available that need to be added to the set of normal packing positions. In Figure 3 a variety of cases of adding and removing packing positions that may occur is visually represented. These are just examples and different situations may occur. The possible adjustments to the set of normal packing positions are, however, divisible in five categories:

I) one position removed, two positions added (e.g. in Figure 3a-3d)

II) one position removed, one position added (e.g. in Figure 3e-3h)

III) two positions removed, two positions added (e.g. in Figure 3i-3j)

IV) two positions removed, three positions added (e.g. in Figure 3k-3l)

V) one position removed, none added (e.g. when an item fits perfectly in a "trapped" area)

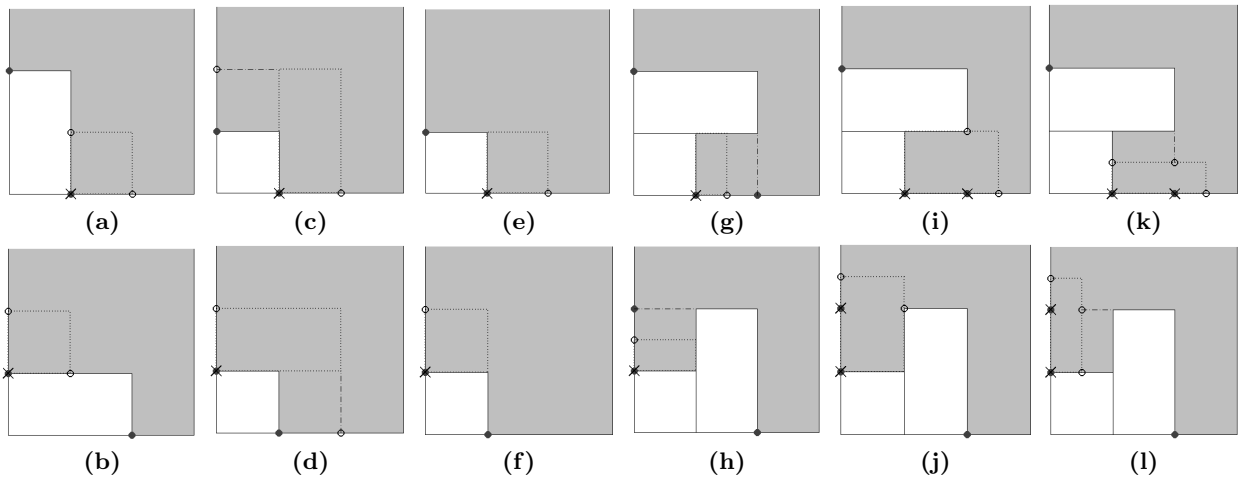


Figure 3: Adjustment situations for the set of normal packing positions; dotted lines denote the placement of a new item, dashed lines are perpendiculars for packing positions, a ● denotes an existing packing position, a ○ denotes a new packing position and a ✖ denotes an existing packing position that will be removed

Input: an item sorting $I(k)$ and a maximum number of iterations λ

procedure $\mathrm{LH}_{2\mathrm{UL}}(I(k), \lambda)$
  $Hstrip := \mathrm{TP}_{2\mathrm{UL}}(I(k))$, $it := 1$;
  while $Hstrip > H$ **and** $it \leq \lambda$ **do**
    randomly select two items and switch their positions;
    $\xi := \mathrm{TP}_{2\mathrm{UL}}(I(k))$;
    $Hstrip := \min\{Hstrip, \xi\}$;
    $it := it + 1$
  end while
  return $Hstrip$
end procedure

Figure 4: Heuristic algorithm for checking the loading constraints

Procedure $\mathrm{LH}_{2\mathrm{UL}}$ iteratively invokes $\mathrm{TP}_{2\mathrm{UL}}$ for different item sequences, as can be found in Figure 4 where the procedure is depicted. The procedure starts with an initial sequence of items in the order in which they need to be packed with $\mathrm{TP}_{2\mathrm{UL}}$, as was described above. Then, as long as the packing found with $\mathrm{TP}_{2\mathrm{UL}}$ exceeds the vehicle height H and the number of iterations does not exceed a predefined number λ, the sequence of items is iteratively altered by switching two randomly selected items in the sequence and a maximum touching perimeter packing is generated with $\mathrm{TP}_{2\mathrm{UL}}$. The result is a packing of height $Hstrip$, which does not necessarily need to fit within the vehicle height H.

4.2 Initial Solution

In Gendreau et al. (2008) two heuristic algorithms, each for both the sequential and the unrestricted case, are used for determining an initial solution in which the Tabu Search heuristic can start. The first algorithm, $\mathrm{IHG}_{2\mathrm{UL}}$, works for general 2L-CVRP instances where the travel distance between vertices is known. The second algorithm, $\mathrm{IHE}_{2\mathrm{UL}}$, is executed for the special case of Euclidean distances.

$\mathrm{IHG}_{2\mathrm{UL}}$ was obtained by embedding the loading constraints in the classical savings algorithm of Clarke and Wright (1964). Here the CVRP is solved by initially allocating a different vehicle to every customer, most likely exceeding the amount of available vehicles. Then, iteratively, the two routes that yield the largest saving of distance travelled if they would be merged are merged. A merge between two routes consists of connecting two vertices. Only merges that satisfy certain conditions can be executed. The conditions are:

I) The vertices that are to be connected are both connected to the depot (vertex 0)

II) The vertices that are to be connected are not already both on the same route

III) The combined weight of the items of the customers on the routes of the vertices that are to be connected does not exceed the weight capacity (weight-feasibility)

IV) The items of the customers on both of the routes of the vertices that are to be connected have a feasible packing, to be checked with $\mathrm{LH}_{2\mathrm{UL}}$ (load-feasibility)

The algorithm first only accepts feasible merges to reduce the number of vehicles to the amount that are available v. When the number of vehicles is still too large, but no more feasible merges are possible, first merges that lead to load-infeasible packings, but which are still weight-feasible, are accepted. If this still does not decrease the number of vehicles to the desired level, also weight-infeasible packings are accepted.

The Euclidean algorithm $\mathrm{IHE}_{2\mathrm{UL}}$ was derived from the algorithm of Cordeau et al. (1997) for solving periodic and multidepot *Vehicle Routing Problems* (VRP's). The loading constraints are embedded in this algorithm to suit our problem. $\mathrm{IHE}_{2\mathrm{UL}}$ starts with randomly selecting a radius emanating from the depot and constructs a feasible route containing the vertices closest to this radius, i.e. of which the angles between the radius and the segment connecting the vertices to the depot are smallest. The feasibility check is done with $\mathrm{LH}_{2\mathrm{UL}}$. For the first $v - 1$ vehicles only feasible packings are accepted, the remainder of customers is assigned to the last vehicle, whether that results in a feasible packing or not.

The instances used by Gendreau et al. (2008) only have coordinates of the vertices and do not give the travel distances between clients and the depot (see Section 5), but these can be approximated by taking the Euclidean distance between the vertices. With this approximation, usage of the $\mathrm{IHG}_{2\mathrm{UL}}$ algorithm is enabled and we choose to use this algorithm for determining the initial solutions.

4.3 Tabu Search

For the TS heuristic Gendreau et al. (2008) introduce an objective function in which infeasibilities are treated as penalties. Naturally the objective function needs to be minimized. Let s denote a solution, consisting of a set of $\tilde{v}$ routes, with $1 \leq \tilde{v} \leq v$, in which each client belongs to exactly one route. The objective function to be minimized is then defined as follows:

$$z'(s) = z(s) + \alpha q(s) + \beta h(s), \tag{2}$$

with

$$z(s) = \sum_{k=1}^{\tilde{v}} c(k), \tag{3}$$

$$q(s) = \sum_{k=1}^{\tilde{v}} \left[\sum_{i \in S(k)} d_i - D \right]^+, \tag{4}$$

$$h(s) = \sum_{k=1}^{\tilde{v}} \left[H_k - H \right]^+, \tag{5}$$

where $c(k)$ are the total costs of the edges in route k, H_k is the height of the two-dimensional loading of vehicle k and $[a]^+ = \max\{0, a\}$. Defined in equation (3), $z(s)$ represents the costs of the the edges that are in the routes of the solution s, $q(s)$, defined in equation (4), represents the violation of the weight constraint and $h(s)$ represents the violation of the height constraint, defined in equation (5). The search is regulated by three parameters, α and β in the objective function and λ in the $\mathrm{LH}_{2\mathrm{UL}}$ procedure. The weight-infeasibilites are controlled by α and the load-infeasibilities by β and λ. These parameters are adjusted depending on the violations of the constraints in the current solution. If a current solution is weight-infeasible (or not), we set $\alpha = \alpha(1+\delta)$ (resp.,$\alpha = \alpha/(1+\delta)$). Further, if a current solution is load-infeasible (or not), we set $\beta = \beta(1+\delta)$ and $\lambda = \min\{\lambda+1, 4\}$ (resp., $\beta = \beta/(1+\delta)$ and $\lambda = \max\{\lambda-1, 1\}$).

As was explained before, Tabu Search evaluates a selection of possible moves and chooses the best move. A move consists of selecting a client i that is currently in a route k, and allocating this client to another route k'. In Gendreau et al. (2008) the insertion procedure is according to the GENI (*Generalized Insertion Procedure*) heuristic developed by Gendreau et al. (1992) for the *Traveling Salesman Problem* (TSP). GENI considers a subset of possible 4-opt modifications of the two concerned tours, meaning a deletion of four arcs between vertices and adding five

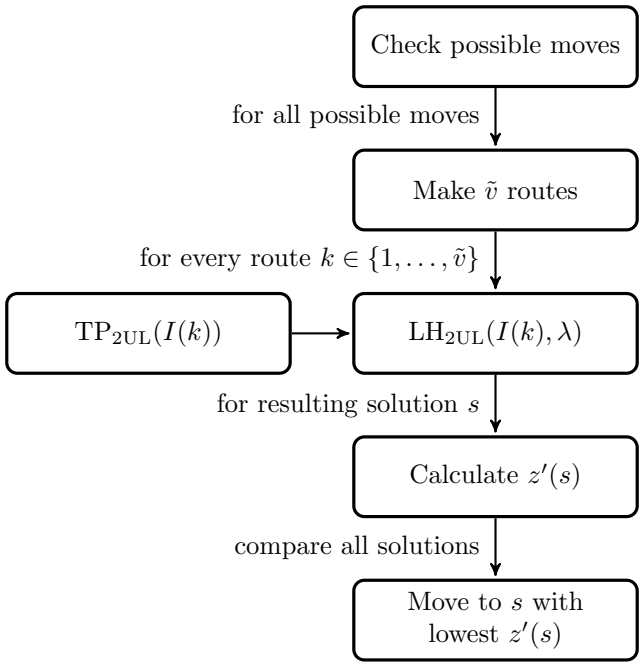


Figure 5: The steps that are iteratively executed in the Tabu Search heuristic

arcs in the tour where the client is inserted, and deleting five arcs and adding four in the tour where the client is deleted. For both tours the best modification is selected.

However for the particular instances tested over here (see Section 5), a vehicle does not visit a large amount of customers. This is due to the weight and loading constraints and the fact that the item characteristics are such that not many customers can be feasibly served with one vehicle. Therefore we choose to solve the Traveling Salesman Problem to exact optimality as insertion procedure when the amount of customers n is less than 10. We evaluated all possible routes, totalling to $n!$ evaluations, and choose the route with smallest travel distance. When the amount of customers a vehicle has to visit is larger, we used a Genetic Algorithm (GA) for solving TSP, because the number of possible routes then becomes too large to evaluate all. A GA is a search heuristic where first a randomly generated population of solutions is evaluated. Then, iteratively, a new generation is created by selecting a proportion of the population that has promising objective values and these are mutated into new solutions. This is done for a predefined number of iterations and the solution with the best objective value is chosen. We opt for GA because of its balance between simplicity and performance.

Let $N_p(i)$ be the set of p vertices closest to i (*neighbors* of client i). The p-neighborhood of client i is used when selecting a route to which it can be allocated in order to limit the options and therefore speed up the algorithm. Client i can only be allocated to an empty tour or a tour in which at least one client in $N_p(i)$ is visited. According to Gendreau et al. (2008) a good performing value for p is 10, balancing the computation time and the solution quality.

With every iteration of the Tabu Search, all possible moves are evaluated, i.e. for each client all possible allocations to other tours are considered, provided the allocation satisfies the neighborhood condition defined above. For every move the value of the objective function $z'(s)$, equation (2), is obtained. After a move has been performed, say removing client i from route k, reinserting client i in route k is declared tabu, and thus prohibited, for the next θ iterations. The value of θ was experimentally determined as $\theta = \log(nv)$. However, a tabu move may still be accepted. We define a $n \times v$ array M where the value of the best solution with client i assigned to route j is stored in the elements M_{ij}. If no such solution has yet been identified $M_{ij} = \infty$. If a tabu move would improve a current M_{ij}, then that move may be accepted as well.

In certain situations a *penalty term* π, defined in equation (6), is possibly added to the cost in order to diversify the search. Diversification is applied to Tabu Search in order to escape a poor or already extensively searched solution area. Let s_a be the current solution and s_b the solution obtained by removing client i from route k. If $z'(s_b) > z'(s_a)$ we add to $z'(s_b)$ a penalty π proportional to frequency of this move. Let ρ_{ik} denote the number of times client i has been removed from route k, divided by the total number of iterations. The penalty term is then defined as follows:

$$\pi = \rho_{ik}(z'(s_b) - z'(s_a))\gamma, \tag{6}$$

with γ set to $\sqrt{nv}$. Now when a certain poor solution area is extensively being searched, the solutions in this area will be increasingly penalized, thus escaping this poor local minimum. In other words, a step in the Tabu Search that leads to a worse solution compared to the current solution is penalized. This penalty depends on the number of times the customer in question has already been removed from the route it is currently in. The removal of client i out of route k will then yield increasingly worse objective values as the frequency of this move rises. This leads to another move turning out to be the best possible move. With declaring recent moves tabu, a part of the risk of cycling in the neighborhood of a local minimum is already repulsed, but with this diversification, neighborhoods of other neighboring solutions of local optima are also explored. Gendreau et al. (2008) propose an intensification phase as well, but we did not implement this procedure, due to a lack of time and the complexity of this algorithm. It namely requires a mixed integer problem to be solved. In Figure 5 the TS heuristic is schematically depicted.

Table 1: Classes 2-5

		Vertical		Homogeneous		Horizontal	
Class	m_i	h_{il}	w_{il}	h_{il}	w_{il}	h_{il}	w_{il}
2	[1,2]	$\left[\frac{4H}{10}, \frac{9H}{10}\right]$	$\left[\frac{W}{10}, \frac{2W}{10}\right]$	$\left[\frac{2H}{10}, \frac{5H}{10}\right]$	$\left[\frac{2W}{10}, \frac{5W}{10}\right]$	$\left[\frac{H}{10}, \frac{2H}{10}\right]$	$\left[\frac{4W}{10}, \frac{9W}{10}\right]$
3	[1,3]	$\left[\frac{3H}{10}, \frac{8H}{10}\right]$	$\left[\frac{W}{10}, \frac{2W}{10}\right]$	$\left[\frac{2H}{10}, \frac{4H}{10}\right]$	$\left[\frac{2W}{10}, \frac{4W}{10}\right]$	$\left[\frac{H}{10}, \frac{2H}{10}\right]$	$\left[\frac{3W}{10}, \frac{8W}{10}\right]$
4	[1,4]	$\left[\frac{2H}{10}, \frac{7H}{10}\right]$	$\left[\frac{W}{10}, \frac{2W}{10}\right]$	$\left[\frac{H}{10}, \frac{4H}{10}\right]$	$\left[\frac{W}{10}, \frac{4W}{10}\right]$	$\left[\frac{H}{10}, \frac{2H}{10}\right]$	$\left[\frac{2W}{10}, \frac{7W}{10}\right]$
5	[1,5]	$\left[\frac{H}{10}, \frac{6H}{10}\right]$	$\left[\frac{W}{10}, \frac{2W}{10}\right]$	$\left[\frac{H}{10}, \frac{3H}{10}\right]$	$\left[\frac{W}{10}, \frac{3W}{10}\right]$	$\left[\frac{H}{10}, \frac{2H}{10}\right]$	$\left[\frac{W}{10}, \frac{6W}{10}\right]$

5 Data

In order to be able to compare computational results with the computational research of Gendreau et al. (2008) we will use graphs and weights demanded by the customer obtained from the same CVRP instances. The number of items and their dimensions were created according to five classes as defined by Gendreau et al. (2008):

***Class 1*:** each customer i, for $i = 1, \ldots, n$, is assigned one item with unit width and height.

***Classes 2-5*:** an integer uniform distribution on a certain interval (see Table 1, column 2), is used to generate the number m_i of items for each customer i. Each item is then randomly assigned, with equal probability, one out of three possible shapes: *vertical* (the relative heights are greater than the relative widths), *homogeneous* (the relative heights and widths are generated in the same interval), or *horizontal* (the relative heights are smaller than the relative widths). Finally, the dimensions of the items are uniformly generated in a given interval (see Table 1, columns 3-8).

For the dimensions of the loading area of a vehicle the values $W = 20$ and $H = 40$ were chosen. Class 1 instances were pure CVRP and here the loading constraints were not even needed to be imposed, the instances of other classes, however, prove to be more difficult to solve due to differently sized items. Evidently, all input data are integer valued. This has, however, no consequences for the performance of the heuristic, since all data can be up- or downscaled with a certain factor, and the heuristic will yield the same solution. Integer data may thus be used without loss of generality. For 36 CVRP instances, combinations of number of customers n and number of vehicles v, an 2L-CVRP instance was created in all five classes, totalling to 180 2L-CVRP instances. We used the exact same 2L-CVRP instances as Gendreau et al. (2008). Both the CVRP instances and the 2L-CVRP instances can be downloaded from `http://www.or.deis.unibo.it/research.html`.

Table 2: Details on the instance generation

		Class 1			*Class 2*			*Class 3*			*Class 4*			*Class 5*		
Instance	n	M	v	LB	M	v	LB	M	v	LB	M	v	LB	M	v	LB
1. E016-03m	15	15	3	3	24	3	3	31	3	3	37	4	3	45	4	3
2. E016-05m	15	15	5	5	25	5	5	31	5	5	40	5	5	48	5	5
3. E021-04m	20	20	4	4	29	5	4	46	5	4	44	5	4	49	5	4
4. E021-06m	20	20	6	6	32	6	6	43	6	6	50	6	6	62	6	6
5. E022-04g	21	21	4	4	31	4	4	37	4	4	41	4	4	57	5	4
6. E022-06m	21	21	6	6	33	6	6	40	6	6	57	6	6	56	6	6
7. E023-03g	22	22	3	3	32	5	4	41	5	4	51	5	4	55	6	3
8. E023-05s	22	22	5	5	29	5	5	42	5	5	48	5	5	52	6	5
9. E026-08m	25	25	8	8	40	8	8	60	8	8	63	8	8	91	8	8
10. E030-03g	29	29	3	3	43	6	5	49	6	4	72	7	6	86	7	5
11. E030-04s	29	29	4	4	43	6	5	62	7	6	74	7	6	91	7	5
12. E031-09h	30	30	9	9	50	9	9	56	9	9	82	9	9	101	9	9
13. E033-03n	32	32	3	3	44	7	5	56	7	5	78	7	6	102	8	5
14. E033-04g	32	32	4	4	47	7	5	57	7	5	65	7	5	87	8	4
15. E033-05s	32	32	5	5	48	6	5	59	6	6	84	8	7	114	8	6
16. E036-11h	35	35	11	11	56	11	11	74	11	11	93	11	11	114	11	11
17. E041-14h	40	40	14	14	60	14	14	73	14	14	96	14	14	127	14	14
18. E045-04f	44	44	4	4	66	9	7	87	10	8	114	10	8	122	10	6
19. E051-05e	50	50	5	5	81	11	9	103	11	10	134	12	10	157	12	8
20. E072-04f	71	71	4	4	104	14	12	151	15	13	178	16	13	226	16	13
21. E076-07s	75	75	7	7	114	14	12	164	17	14	168	17	14	202	17	14
22. E076-08s	75	75	8	8	112	15	13	154	16	14	198	17	14	236	17	14
23. E076-10e	75	75	10	10	112	14	13	155	16	14	179	16	14	225	16	14
24. E076-14s	75	75	14	14	124	17	14	152	17	14	195	17	14	215	17	14
25. E101-08e	100	100	8	8	157	21	18	212	21	18	254	22	19	311	22	19
26. E101-10c	100	100	10	10	147	19	16	198	20	17	247	20	18	310	20	18
27. E101-14s	100	100	14	14	152	19	17	211	22	19	245	22	19	320	22	19
28. E121-07c	120	120	7	7	183	23	20	242	25	21	299	25	21	384	25	21
29. E135-07f	134	134	7	7	197	24	21	262	26	22	342	28	24	422	28	24
30. E151-12b	150	150	12	12	225	29	25	298	30	27	366	30	27	433	30	27
31. E200-16b	199	199	16	16	307	38	33	402	40	35	513	42	37	602	42	37
32. E200-17b	199	199	17	17	299	38	33	404	39	34	497	39	34	589	39	34
33. E200-17c	199	199	17	17	301	37	32	407	41	35	499	41	36	577	41	36
34. E241-22k	240	240	22	22	370	46	40	490	49	42	604	50	44	720	50	44
35. E253-27k	252	252	27	27	367	45	39	507	50	43	634	50	45	762	50	45
36. E256-14k	255	255	14	14	387	47	41	511	51	44	606	51	44	786	51	44

In summary, the 2L-CVRP instances were created from the CVRP instances by only generating a number of items, the amount per customer depending on the class, for every customer with certain heights and widths, also depending on the class. This implies that the five 2L-CVRP instances generated from one CVRP instance, one in every class, all have the same customer information. All five have the same amount of customers with the same coordinates and the same weight demand, only the number of items and the item measurements demanded by the customers differ between instances.

A more detailed insight in the different combinations can be found in Table 2. The original names of the CVRP instances can be found in the first column and for all 36 CVRP instances the number of customers (n) is given in the second column. In the other columns the total number of items ($M = \sum_{i=1}^{n} m_i$), the number of vehicles (v) and a lower bound (LB) on the number of vehicles are displayed for the five classes per CVRP instance. The lower bound was determined by Gendreau et al. (2008) as the maximum of two lower bounds: the number of available vehicles in the original CVRP instance and the value obtained by solving the two-dimensional bin packing problem (2BPP) for the items of the 2L-CVRP instance, with one bin defined as the loading area of a vehicle. 2BPP minimizes the number of bins all items can be packed in. This lower bound is given in order to show the reasonableness of the produced instances. As can be seen in Table 2, the values of v and LB are very close, in 73 cases they match exactly and the differ by at most two vehicles in 57 cases.

6 Results

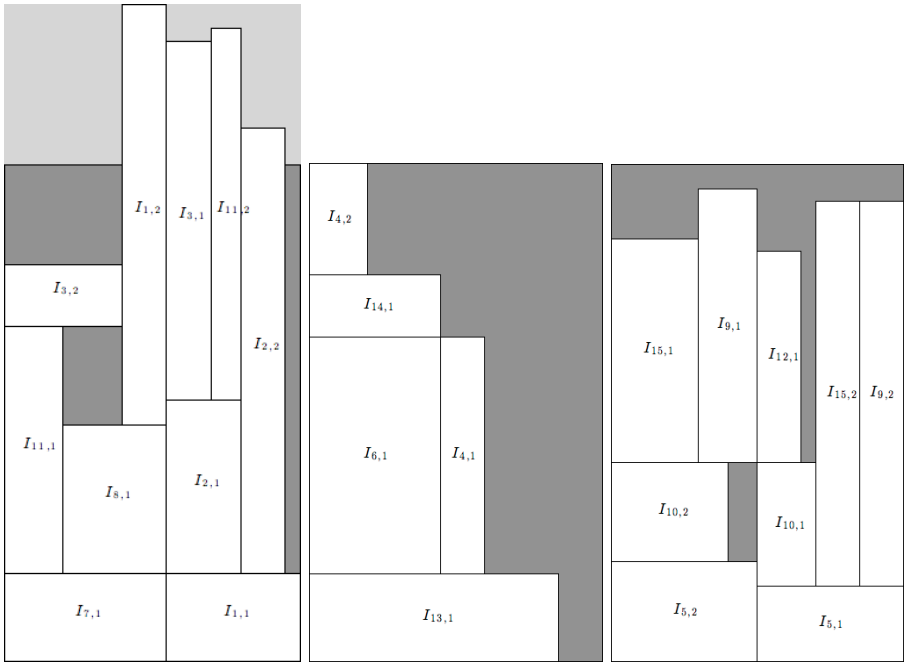
In this Section we discuss computational results of solving the 2L-CVRP instances described in Section 5. All procedures were implemented in MATLAB R2013a and results were obtained on a AMD Athlon™II X2 B28 Processor 3.00 GHz with 4.00 GB Installed RAM memory and Windows 7. First we discuss the initial solutions and thereafter the optimal solutions obtained through the Tabu Search heuristic.

6.1 Initial Solution

As was described in Section 4.2, Gendreau et al. (2008) make use of two different algorithms for determining an initial solution. The algorithm IHG_{2UL} is only usable in the case of known travel distances between vertices. The 2L-CVRP instances that are provided do not contain information about travel distances between vertices, only the coordinates of the vertices are given. However, as was described in Section 4.2, we choose to approximate these distances by taking the Euclidean distances between the vertices as the travel distances and make use of IHG_{2UL} for determining the initial solution.

Table 3: Computation times in seconds for initial solutions, colored cells indicate an initial solution is infeasible

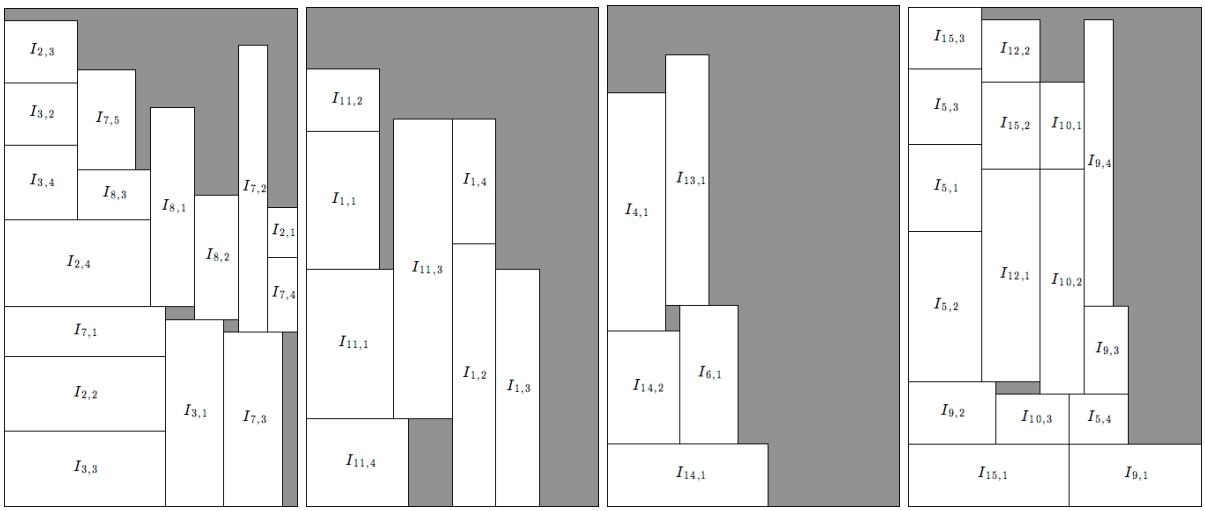
Instance	*Class 1*	*Class 2*	*Class 3*	*Class 4*	*Class 5*
1. E016-03m	2,50	3,53	4,37	4,69	4,56
2. E016-05m	2,13	2,61	3,70	3,12	3,78
3. E021-04m	6,21	9,19	15,40	9,47	9,31
4. E021-06m	5,96	6,89	7,75	8,79	9,74
5. E022-04g	7,54	9,36	10,42	10,15	11,20
6. E022-06m	7,47	8,81	8,95	12,22	10,61
7. E023-03g	9,90	16,49	17,71	22,20	22,34
8. E023-05s	9,79	13,38	20,34	21,26	18,13
9. E026-08m	14,13	15,50	19,32	19,67	21,09
10. E030-03g	26,04	43,42	46,44	67,28	70,83
11. E030-04s	25,97	42,87	54,98	74,77	78,00
12. E031-09h	28,15	29,77	31,98	36,32	39,49
13. E033-03n	41,55	64,10	70,39	89,12	102,85
14. E033-04g	40,08	59,01	57,82	64,87	71,46
15. E033-05s	36,77	51,53	70,03	90,60	106,09
16. E036-11h	47,44	50,90	54,53	60,23	65,11
17. E041-14h	80,86	89,75	90,66	95,74	98,94
18. E045-04f	121,55	190,74	230,18	281,70	255,69
19. E051-05e	208,23	359,48	393,74	492,03	466,32
20. E072-04f	819,24	1180,47	1381,36	1585,03	2109,54
21. E076-07s	1401,78	1652,92	1996,85	1697,36	1731,21
22. E076-08s	978,29	1364,08	1592,58	1809,49	2008,11
23. E076-10e	1075,03	1305,87	1507,95	1642,86	1556,59
24. E076-14s	964,72	1327,82	1250,76	1386,27	1305,63
25. E101-08e	3029,59	7910,53	9095,79	9968,06	9452,81



(a) Vehicle 1, $Hstrip = 53$, $\sum_{i \in S(1)} d_i = 114$ **(b)** Vehicle 2, $Hstrip = 40$, $\sum_{i \in S(2)} d_i = 68$ **(c)** Vehicle 3, $Hstrip = 38$, $\sum_{i \in S(3)} d_i = 76$

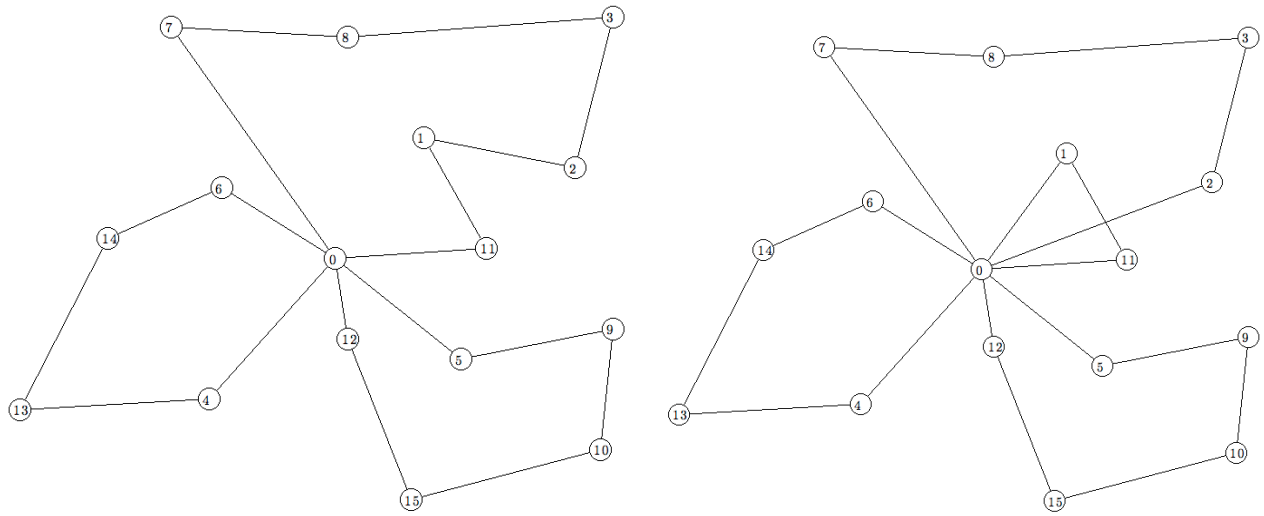
Figure 6: The packings for the initial solution of instance 1, class 2, $v = 3$, $n = 15$, weight capacity $D = 90$

Since the 2L-CVRP instances increase in number of customers and vehicles, as can be seen in Table 2, the number of possible merges the IHG_{2UL} algorithm has to consider every iteration increases drastically. Therefore the computation times of the initial solution increase exponentially as well. Since finding an initial solution for larger instances takes too much time, we decided to leave the largest instances for later research. We found initial solutions for the smallest 125 instances and the corresponding computation times can be found in Table 3. Here colored cells indicate a solution is infeasible, which was the case for 100 of the 125 instances. In 56 cases the initial solution was only load-infeasible, in 17 cases it was only weight-infeasible and in 27 cases it was both. The computation times for finding optimal solutions were given in Gendreau et al. (2008) and these were already significantly smaller than our computation times for finding an initial solution. This may be attributed to the choice of program in which the algorithms were implemented. Unfortunately the choice of program is not given by Gendreau et al. (2008), but likely is that they made use of a more efficient program.



(a) Vehicle 1, $Hstrip = 39$, $\sum_{i \in S(1)} d_i = 88$ **(b)** Vehicle 2, $Hstrip = 35$, $\sum_{i \in S(2)} d_i = 26$ **(c)** Vehicle 3, $Hstrip = 36$, $\sum_{i \in S(3)} d_i = 68$ **(d)** Vehicle 4, $Hstrip = 40$, $\sum_{i \in S(4)} d_i = 76$

Figure 7: The packings for the initial solution of instance 1, class 5, $v = 4$, $n = 15$, weight capacity $D = 90$



(a) Class 2 (infeasible solution)

(b) Class 5 (feasible solution)

Figure 8: The routes for the initial solutions of instance 1

In Figures 6-8 two initial solutions are visually represented. These are the solutions of the first CVRP instance (`E016-03m`) for class 2 and 5. One important difference between the two instances is that for class 2, three vehicles are available, and for class 5, four vehicles are available. This is due to the fact that the instance for class 5 has more items, 45 in total, that need to be packed than the instance for class 2, which has 24 items in total. The initial solution for class 5 is a feasible solution, as can be derived from Figure 7. All four vehicles have a packing that does not exceed the weight capacity or the loading capacity. The initial solution for class 2 however, is infeasible. The packing of the first vehicle exceeds the weight and loading capacity, as can be seen in Figure 6. The routes for the vehicles are given in Figure 8.

6.2 Tabu Search

Since the computation times for finding initial solutions were already significantly larger than the computation times of the Tabu Search heuristic of Gendreau et al. (2008), we decided to use only the 20 smallest CVRP instances (100 2L-CVRP instances) for our Tabu Search. For these, we found infeasible initial solutions in 79 cases, of which 36 were only load-infeasible, 16 were only weight-infeasible, and 27 were both. In Tables 4 and 5 detailed information on the solutions obtained for the first 17 CVRP instances is displayed. Only the first 17 are displayed here, because we want to compare our computational results to those of Gendreau et al. (2008) and they only provide detailed information on the first 17 CVRP instances. Gendreau et al. (2008) added extra restrictions to the problem here, namely that every vehicle should be used and that no single-customer-routes were allowed. These restrictions were added in order to be able to compare their results to the results that were obtained through another method. Solutions violating these restrictions were made infeasible through penalties. However, we did not embed these restrictions in our Tabu Search, because they have no further purpose in solving the problem. For obtaining the objective, the Euclidean distances, rounded down, were used as costs between vertices. We used a slightly different policy for running the Tabu Search than Gendreau et al. (2008), namely that the Tabu Search is halted after $2n^2v$ iterations (the same) or after two hours of CPU time (twice as long).

For every instance the objective of the best found solution is given for both Tabu Searches, the one of Gendreau et al. (2008) (z_{TSgen}) and the one that was implemented with this thesis (z_{TSves}). An asterisk indicates a proven optimal solution and two asterisks indicate that no feasible solution was found. For both also the elapsed CPU time in seconds is given until the best solution was found (CPU_{opt}) and until the Tabu Search was halted (CPU). Also the $\%gap$ is given, with $\%gap = 100(z_{TSves} - z_{TSgen})/z_{TSgen}$, to evaluate the quality of the solutions compared to the solutions of Gendreau et al. (2008). Since we did not embed the extra restrictions discussed before, sometimes a proven optimal solution can be beaten by our solution. Our solutions are allowed to leave vehicles unused or let a vehicle visit only one customer. To illustrate this, the number of vehicles used in the best solution ($\tilde{v}$) and the number of vehicles available (v) can also be found in Tables 4 and 5 for every instance. Naturally, $\tilde{v}$ is only given for the solutions of this thesis, since the solutions of Gendreau et al. (2008) always use all vehicles available. Most

Table 4: Performance of our Tabu Search heuristic compared to the Tabu Search heuristic of Gendreau et al. (2008)

Instance		Tabu Search (Gendreau et al., 2008)			Tabu Search (Vester)					
No.	Class	z_{TSgen}	CPU_{opt}	CPU	z_{TSves}	$\tilde{v}$	v	CPU_{opt}	CPU	%gap
1	1	273*	0.0	2.4	273	3	3	32.0	548.7	0.00
	2	285*	0.2	4.6	**$\diamond$	**	3	**	4262.9	**
	3	280*	1.3	7.8	309	3	3	1414.4	1464.0	10.36
	4	290*	0.3	7.5	290	4	4	3.3	1944.1	0.00
	5	279*	2.3	15.7	273	3	4	68.9	2544.3	-2.15
2	1	329*	0.1	1.4	344$\diamond$	5	5	221.6	844.2	4.56
	2	342*	1.1	2.0	329	5	5	24.5	827.0	-3.80
	3	350	0.2	3.2	**$\diamond$	**	5	**	5367.4	**
	4	336*	0.1	6.7	341$\diamond$	5	5	109.3	4102.2	1.49
	5	329*	0.3	6.1	**$\diamond$	**	5	**	2998.1	**
3	1	351*	0.2	8.4	357	4	4	2769.4	7249.1	1.71
	2	407	8.9	9.8	381	5	5	971.4	2928.0	-6.39
	3	387*	1.0	20.0	387	5	5	546.7	5425.1	0.00
	4	374*	14.3	21.9	357	4	5	1175.3	7202.0	-4.55
	5	369*	1.0	29.9	351	4	5	479.3	7201.1	-4.88
4	1	423*	0.2	5.7	**$\diamond$	**	6	**	3203.4	**
	2	434*	0.3	7.4	452	6	6	977.4	3081.1	4.15
	3	438	5.0	12.8	437	6	6	86.4	5188,8	-0.23
	4	451	0.8	16.9	454	6	6	692.6	5539,3	0.67
	5	423*	1.2	27.1	445	6	6	233.0	4692.7	5.20
5	1	367*	2.9	12.8	376	4	4	631.4	7201.4	2.45
	2	396	4.0	19.7	380	4	5	1075.1	7265.6	-4.04
	3	377	0.6	20.5	377	4	5	4114.6	7204.9	0.00
	4	406	5.9	33.0	380	4	5	83.9	7205.1	-6.40
	5	389*	5.7	57.8	367	4	5	155.0	7200.5	-5.66
6	1	488*	0.1	10.3	488	6	6	1220.6	2351.5	0.00
	2	498	1.8	14.0	497	6	6	448.5	4171.8	-0.20
	3	496*	11.2	16.9	500	6	6	1379.8	5318.9	0.81
	4	503	0.1	40.2	497	6	6	4346.2	7200.5	-1.19
	5	488*	0.7	34.2	488	6	6	2534.0	5354.9	0.00
7	1	558*	0.3	22.0	558	3	3	307.9	7315.9	0.00
	2	752	0.7	18.9	715$\diamond$	5	5	12789.8	14480.5	-4.92
	3	704	19.9	29.8	**$\diamond$	**	5	**	14503.1	**
	4	742	26.5	50.4	782$\diamond$	5	5	0.0	14439.0	5.39
	5	743	16.1	75.1	**$\diamond$	**	6	**	14472.2	**
8	1	657*	7.0	31.3	558	3	5	857.7	7283.2	-15.07
	2	720*	3.1	20.1	744$\diamond$	5	5	0.0	14422.2	3.33
	3	752	20.5	33.4	**$\diamond$	**	5	**	14466.3	**
	4	722	11.2	50.0	**$\diamond$	**	5	**	14528.6	**
	5	736	12.0	90.2	699	6	6	0.0	7255.9	-5.03
9	1	609*	1.9	11.9	**$\diamond$	**	8	**	9123.6	**
	2	612*	2.9	15.4	618	8	8	293.2	7200.4	0.98
	3	626	7.9	38.4	**$\diamond$	**	8	**	14401.8	**
	4	627	5.9	43.5	**$\diamond$	**	8	**	14401.3	**
	5	609*	8.5	81.9	**$\diamond$	**	8	**	14401.6	**

of the times when our Tabu Search finds a better objective value than the Tabu Search of Gendreau et al. (2008), its solution violates the extra restrictions of the minimum number of customers each vehicle should visit. However, for some instances, e.g. instance 4, class 3, our Tabu Search finds a better objective value than the Tabu Search of Gendreau et al. (2008) without violating these restrictions.

Unfortunately, the Tabu Search that was implemented with this paper does not perform as well as the Tabu Search of Gendreau et al. (2008). Their heuristic manages to find solutions for all instances, whereas we only find solutions in 47 of the 100 instances. This may probably be attributed to the significantly larger computation times, but it is also possible that the parameters that the Tabu Search is dependent on play a role here. Our computation times are on average approximately 400 times larger, given that the Tabu Search runs all the predefined number of iterations. Mostly, the Tabu Search does not run all of the predefined number of iterations, but is halted by the limit of the CPU time, which was set to two hours. This means that our Tabu Search runs over fewer iterations and therefore simply does not reach a feasible solution within the set time limit. The Tabu Search does find a better solution than the initial solution for all instances where no feasible solution was found, but these were still infeasible. However, it should be noted that a "better" infeasible solution, means that it has a lower objective value.

Table 5: Performance of our Tabu Search heuristic compared to the Tabu Search heuristic of Gendreau et al. (2008)

Instance		Tabu Search (Gendreau et al., 2008)			Tabu Search (Vester)					
No.	Class	z_{TSgen}	CPU_{opt}	CPU	z_{TSves}	$\tilde{v}$	v	CPU_{opt}	CPU	%gap
10	1	544	19.9	97.3	**$\diamond$	**	3	**	14840.4	**
	2	703	3.8	72.2	692	6	6	5325.6	7238.3	-1.56
	3	676	47.9	118.7	**$\diamond$	**	6	**	14405.0	**
	4	773	47.3	156.9	**$\diamond$	**	7	**	14421.5	**
	5	724	84.5	308.9	691	6	7	24.4	7245.5	-4.56
11	1	500*	0.8	107.8	494	4	4	3489.7	7448.7	-1.20
	2	734	4.7	72.2	**$\diamond$	**	6	**	14401.7	**
	3	785	72.2	101.7	748$\diamond$	7	7	14206.8	14404.1	-4.71
	4	877	196.4	209.5	**$\diamond$	**	7	**	14440.5	**
	5	696	279.2	387.0	661$\diamond$	7	7	12068.6	14403.6	-5.03
12	1	598	19.4	33.8	**$\diamond$	**	9	**	14400.4	**
	2	628	9.6	42.9	**$\diamond$	**	9	**	14400.9	**
	3	597	12.9	50.7	**$\diamond$	**	9	**	14400.6	**
	4	640	96.0	120.6	**$\diamond$	**	9	**	14406.6	**
	5	597	103.5	188.2	617	9	9	315.8	7201.0	3.35
13	1	1991*	47.4	218.7	1991	3	3	2229.9	7775.4	0.00
	2	2775	43.7	123.1	2857$\diamond$	7	7	0.0	14704.8	2.95
	3	2696	121.5	170.3	**$\diamond$	**	7	**	14544.3	**
	4	2743	29.3	277.0	**$\diamond$	**	7	**	14564.1	**
	5	2737	237.5	691.9	2701	8	8	0.0	7515.9	-1.32
14	1	823	21.8	145.0	826	4	4	3381.5	7238.6	0.36
	2	1266	52.3	144.4	1279$\diamond$	7	7	0.0	14434.5	1.03
	3	1204	137.9	207.1	1148	6	7	92.1	7210.2	-4.65
	4	1187	108.5	324.9	**$\diamond$	**	7	**	14402.0	**
	5	1309	103.2	895.0	927	5	8	3137.4	7474.2	-29.18
15	1	907*	51.9	196.4	823	4	5	6847.2	7428.4	-9.26
	2	1135	94.7	133.9	**	**	6	**	7408.8	**
	3	1183	37.1	205.9	**	**	6	**	7424.8	**
	4	1372	268.2	332.2	**	**	8	**	7252.8	**
	5	1361	651.4	671.7	**	**	8	**	7228.7	**
16	1	682*	56.1	96.6	687	11	11	27.2	7200.2	0.73
	2	682*	20.5	91.7	691	11	11	239.0	7200.9	1.32
	3	682*	15.3	138.1	727	11	11	3984.8	7200.9	6.60
	4	704	21.7	206.1	716	11	11	1165.2	7202.9	1.70
	5	682*	62.8	276.6	687	11	11	90.6	7200.2	0.73
17	1	842	84.4	158.6	**	**	14	**	7200.3	**
	2	851	56.6	132.5	**	**	14	**	7200.1	**
	3	842	38.8	200.6	**	**	14	**	7200.7	**
	4	845	7.4	279.7	**	**	14	**	7202.2	**
	5	842	44.7	402.2	**	**	14	**	7201.6	**

This implies that the total travelled distance is smaller, but does not account for the extent of violations of the constraints. In short, a "better" infeasible solution has a smaller overall travel distance, but might also violate the constraints more.

In some cases, where the initial solution was already feasible, no better solution was found through Tabu Search. These cases can be found in the Tables 4 and 5 where the value for CPU_{opt} of our Tabu Search is equal to 0.0. This could either be because the initial solution already was a good feasible solution or because the Tabu Search failed to escape a local minimum. The latter should be made improbable by declaring latest visited solutions tabu and the diversification through penalties, but our Tabu Search has not been tested for cycling in local minimum solutions and thus we cannot know with certainty that the Tabu Search does not cycle in local optima.

For the first 14 CVRP instances we tested whether altering the parameters of which the Tabu Search is dependent would lead to feasible solutions for instances where initially no feasible solution was found or no better feasible solution was found than the initial solution. There are some cases where the initial solution is already feasible and a better solution than the one found by Gendreau et al. (2008), for these cases we did not alter the parameters. The solutions z_{TSves} for which we altered the parameters are indicated with a $\diamond$ in Tables 4 and 5, a total of 33 instances. In order to widen the search for a solution, the alteration of parameters was as follows. We extended the maximum value of λ from 4 to 10. This means that checking for load-feasibility will take longer, but also might find packings that are feasible, which were not found before. We also squared the value of θ, declaring moves tabu for more iterations. Furthermore we doubled the maximum number of iterations and maximum CPU time. As can be

Table 6: Summary of the solutions that are obtained compared to the solutions obtained by Gendreau et al. (2008)

	CVRP (Gendreau et al., 2008)			CVRP (Vester)			2L-CVRP (Gendreau et al., 2008)			2L-CVRP (Vester)			
Instance	z_1	CPU_{opt}	CPU	z_1	CPU_{opt}	CPU	$\bar{z}_{2\text{-}5}$	CPU_{opt}	CPU	$\bar{z}_{2\text{-}5}$	*solved*	CPU_{opt}	CPU
1	278.73	2.0	2.2	278.98	32.0	548.7	291.60	4.2	9.7	296.60	[3,4,5]	495.5	2553.8
2	334.96	0.0	1.4	349.63	221.6	844.2	341.02	0.1	3.7	340.96	[2,4]	66.9	3323.7
3	359.77	3.5	8.4	364.45	2769.4	7249.1	377.35	1.6	19.6	376.32	[2,3,4,5]	793.2	5689.1
4	430.88	0.1	5.7	**	**	3203.4	437.45	0.5	14.7	455.40	[2,3,4,5]	410.0	3743.8
5	375.28	1.4	12.8	384.06	631.4	7201.4	380.20	5.0	25.2	383.89	[2,3,4,5]	1357.2	7219.0
6	495.85	0.3	8.7	496.07	1220.6	2351.5	501.02	7.2	18.8	503.66	[2,3,4,5]	2177.1	5511.5
7	568.56	0.5	22.6	568.56	307.9	7315.9	700.34	6.3	48.4	759.36	[2,4]	6394.9	14473.7
8	568.56	0.5	36.2	568.56	857.7	7283.2	694.99	11.2	61.3	731.07	[2,5]	0.0	12668.4
9	607.65	0.4	13.5	**	**	9123.6	619.69	3.6	41.2	630.21	[2]	293.2	11701.4
10	538.79	6.1	81.7	**	**	14840.4	700.39	36.0	192.0	704.98	[2,5]	2675.0	10827.6
11	505.01	2.5	98.9	505.68	3498.7	7448.7	739.04	55.7	207.6	718.67	[3,5]	13137.7	14412.5
12	610.57	28.5	32.5	**	**	14400.4	620.62	49.0	82.7	632.27	[5]	315.8	12602.3
13	2006.34	29.9	161.6	2006.68	2229.8	7775.4	2598.20	57.5	332.0	2796.39	[2,5]	0.0	12833.3
14	837.67	22.2	152.1	841.35	3381.5	7238.6	1047.72	375.8	565.6	1134.90	[2,3,5]	1076.5	10880.2
15	837.67	1.7	182.5	837.97	6847.2	7428.4	1201.38	156.7	375.9	**	**	**	7328.7
16	698.61	2.7	99.0	703.17	27.2	7200.2	702.03	20.5	160.1	722.18	[2,3,4,5]	1369.9	7201.2
17	862.62	59.0	162.8	**	*	7200.3	866.37	64.9	218.2	**	**	**	7280.8
18	723.54	81.9	587.3	748.54	3702.8	7410.5	1085.84	589.3	1318.0	**	**	**	7329.6
19	524.61	128.8	641.9	**	*	7756.3	772.25	633.7	1524.5	833.44	[4]	7154.0	7357.5
20	241.97	253.6	1005.2	252.75	3905.3	7404.0	564.67	954.5	3237.3	565.84	[4,5]	3281.4	3624.2
Average	*620.38*	*31.3*	*165.9*	*636.18*	*2116.7*	*7061.2*	*762.11*	*151.7*	*422.8*	*684.77*	*2.1*	*2411.7*	*8607.0*

seen in Tables 4 and 5 this sometimes did result in a (better) feasible solution, in only 9 of the 33 cases, but mostly not. The total number of instances we have found a solution for is now 56 out of 100. We did not test for all the instances where no or no good solution was found whether altering the parameters would improve the performance of the heuristic. We choose to investigate this for rather small instances, up to 32 customers, since we expect that for larger instances we would need to widen the search area through altering the parameters much more.

Of the first 85 2L-CVRP instances depicted in Tables 4 and 5 a solution was found for 51 instances. The $\%gap$ between the best found objective values is on average -1.28 and 22 found solutions are better. This implies that our Tabu Search performs slightly better, if it is able to find a solution. However, the Tabu Search of Gendreau et al. (2008) had extra restrictions to take into consideration. If we leave out the solutions where these restrictions are violated, leaving 36 solutions, the average $\%gap$ becomes 0.65, and only 9 found solution are better. This disproves the idea that our Tabu Search is slightly better than the Tabu Search of Gendreau et al. (2008), and leads to think that our Tabu Search performs on average the same or even slightly worse.

Gendreau et al. (2008) also give results of the Tabu Search without the restrictions as in Tables 4 and 5 and where the Euclidean distances between the vertices were not rounded for the costs used in the objective function. Unfortunately, only a concise representation of the results is provided, making exact comparison with our results difficult. These results for the first 20 CVRP instances can be found in Table 6. The results were divided in two sets: solutions for the first class of every instance, which is in practice equal to pure CVRP, and solutions for the other classes of every instance. The instances of the first class are pure CVRP, because every client demands only one item, which is always of unit length and height. Since all items always would fit in one vehicle, independent of the size of the instance, the loading constraint is never binding. This means only the weight constraint is binding for these instances and this is pure CVRP.

In Table 6 the results are grouped in results for CVRP found with the Tabu Search of Gendreau et al. (2008) (first cluster), results for CVRP found with the Tabu Search of this paper (second cluster), results for 2L-CVRP found with the Tabu Search of Gendreau et al. (2008) (third cluster), and results for 2L-CVRP found with the Tabu Search of this paper (last cluster). The best found objective values for the first class instances (z_1) are depicted for both Tabu Searches. The corresponding time it took to find this solution (CPU_{opt}) and the total computation time (CPU), both in seconds, can be found as well. For instances of class 2 to 5 an average of the found optimal objective values is given ($\bar{z}_{2\text{-}5}$) for both Tabu Searches, also together with the computation times in seconds. Here the computation times are also averages of the computation times of classes 2 to 5 for every CVRP instance. Since the Tabu Search of this thesis is not able to find solutions for all the instances, the classes that the Tabu Search was able to find a solution for are given under *solved* for every instance. Averages of all values are given in the last line of Table 6, where the average of *solved* is the average number of 2L-CVRP instances for which a solution was found. For the average of $\bar{z}_{2\text{-}5}$ of our Tabu Search the weighted average is taken with respect to the number of instances that are solved. Two asterisks indicate the instances for which no solutions are found.

What is clearly visible from Table 6, is that the Tabu Search of Gendreau et al. (2008) never exceeds their maximum computation time of one hour, meaning that it completes all predefined number of iterations for the Tabu Search. Our Tabu Search on the contrary reaches the maximum computation time (2 or 4 hours) more often, and thus almost never completes the Tabu Search for the predefined number of iterations. From this table we can also derive that including loading constraints leads to worse solutions, as the best found objective values for CVRP are smaller on average than those for 2L-CVRP.

7 Conclusions & Summary

We have evaluated a heuristic method, proposed by Gendreau et al. (2008), for solving an extended version of the classical vehicle routing problem in which two-dimensional loading constraints are introduced. This problem combines two classical optimization problems, CVRP and 2BPP. A Tabu Search heuristic was developed for solving the resulting problem: 2L-CVRP. The research of Gendreau et al. (2008) covers two versions of 2L-CVRP, with unrestricted loading and with sequential loading. In this thesis this Tabu Search heuristic was implemented and tested for the unrestricted version of the problem, meaning that with packing a vehicle, the order in which customers are visited is not considered. The heuristic was applied to a large set of 100 instances, which are also used by Gendreau et al. (2008).

Some aspects of the heuristic were differently implemented with this thesis. The procedure of packing a vehicle was exactly implemented as was described by Gendreau et al. (2008). In the Tabu Search itself, however, some alterations were made. For determining a new solution, a different insertion procedure was applied for deleting a customer from a route and adding it to another route. Gendreau et al. (2008) make use of the so-called *Generalized Insertion Procedure* (GENI), which is heuristic for solving the Traveling Salesman Problem (TSP) that considers a subset of the possible 4-opt modifications of both the routes in question. We opted for solving a TSP differently for both routes, mostly to exact optimum. Since most of the time the routes in consideration do not visit a large amount of clients, this was a manageable option. For routes visiting less than 10 clients we solved TSP to exact optimality, for larger routes we used a Genetic Algorithm, another heuristic. By doing so, more individual routes of vehicles were sure to be distance optimal.

One large element of the Tabu Search heuristic developed by Gendreau et al. (2008) was not implemented. A Tabu Search heuristic usually has a diversification phase and a intensification phase in order to escape poor searching areas or more extensively search a promising solution area. The diversification was implemented by penalizing moves with respect to the frequency of the move before. The intensification phase, however, was omitted. This was due to the complexity of the algorithm, it required a mixed integer problem to be solved, and the limited time this thesis had to be completed in. The computational results were probably substantially affected by choosing not to implement the intensification phase.

After implementation, the heuristic was tested and the results were compared to the results of Gendreau et al. (2008). In comparison, the performance of our heuristic proved itself poor. In contrast to the Tabu Search of Gendreau et al. (2008), in a lot of the instances, 53%, not even a feasible solution was found initially, and when a feasible solution was found, it often was worse than the solution found with the Tabu Search of Gendreau et al. (2008). This may probably be attributed mostly to the computation time. Our heuristic needed approximately 200 times more time for completion of the Tabu Search. The computation times may be that much larger, because of the choice of program. Another explanation could be that the heuristic was not efficiently enough implemented. The limit of the computation time was set to two hours, and for most of the instances the Tabu Search did not complete all iterations. Hence it is possible that the heuristic did not find a feasible or optimal solution, because it ran over too few iterations or because the Tabu Search is too much restricted by its parameters.

We therefore chose to widen the search area by altering the parameters the Tabu Search is dependent on for some of the instances where no or no good solution was found. We increased the maximum value for λ, leading to more extensively check for load-feasibility and we squared the value for θ, declaring moves tabu for more following iterations. We also doubled the maximum number of iterations and the maximum computation time in order to let the Tabu Search run over more iterations, in the hope that then good feasible solutions are found. We tested whether this would improve our performance for 33 instances where no or no good solution was found, and with this alteration did now found solutions in 9 of these cases. Hence it means that we can improve the performance of our heuristic somewhat, but this also yields much more computation time.

Concluding we can say that our heuristic does not measure up to the Tabu Search of Gendreau et al. (2008). It does find better solutions sometimes, but overall it performs worse, especially regarding the computation times. Also our heuristic does not always find a solution, contrary to the Tabu Search of Gendreau et al. (2008).

8 Further Research

We would like to touch on some notes for further research. Obviously, looking at the performance of our Tabu Search, our implementation of the heuristic leaves a lot to be desired. The computation times were much too high for finding solutions within a reasonable amount of time. The modules that this heuristic consists of can probably be implemented more efficiently and should be fine-tuned. For example, the adjustment of the packing positions in the procedure TP_{2UL} is divided into 5 different possible situations, which were all individually implemented. This might be done more efficiently by determining covering adjustment rules that would work for all situations. Also a more efficient program to implement the heuristic in should be considered.

Secondly, an intensification phase should be implemented, because then a promising solution that might be infeasible, can be more extensively adapted, trying to make it feasible. As was discussed, our current heuristic lacks such a phase. Gendreau et al. (2008) propose an intensification phase, but different intensification phases could be considered. Also different diversification methods could be considered than the one proposed by Gendreau et al. (2008), which used penalizing frequently made moves in order to diversify the search. An example of a different diversification could be perhaps to allocate more than one customer in one move, if the Tabu Search is stuck in a local minimum.

Also some more robustness testing on the parameters the Tabu Search is dependent on should be done, for determining its sensitivity to these parameters. Different rules on adapting α, β, and λ may be considered as well as different values for θ, γ, and δ. Also more different halting criteria for the Tabu Search could be investigated, altering the maximum number of iterations and maximum CPU time. We did slightly test for some alterations, but this does not give a good overall impression of the dependence of the Tabu Search on these parameters.

Furthermore, the different modules the Tabu Search consists of could be further investigated. Different insertion procedures for reallocating a client may be examined for example. Gendreau et al. (2008) made use of the GENI heuristic, we used a combination of exactly solving TSP and a Genetic Algorithm, but many more options exist for solving TSP. Also checking for load-feasibility may be done differently, for example use a different initial sequence of items that is packed, or swap items in LH_{2UL} more cleverly, instead of randomly.

Lastly, the other algorithm for determining an initial solution, IHE_{2UL}, should be examined as well. With IHG_{2UL} we found feasible solutions only for 21% of the instances. Perhaps better initial solutions would be found with IHE_{2UL}.

References

Bortfeldt, A. (2012). A hybrid algorithm for the capacitated vehicle routing proble with three-dimensional loading constraints. *Computers & Operations Research*, 39(2):2248–2257.

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

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

Duhamel, C., Lacomme, P., Quilliot, A., and Toussaint, H. (2011). A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle routing problem. *Computers & Operations Research*, 38(3):617–640.

Fuellerer, G., Doerner, K. F., Hartl, R. F., and Iori, M. (2009). Ant colony optimization for the two-dimensional loading vehicle routing problem. *Computers & Operations Research*, 36(13):655–673.

Fuellerer, G., Doerner, K. F., Hartl, R. F., and Iori, M. (2010). Metaheuristics for vehicle routing problems with three-dimensional loading constraints. *European Journal of Operational Research*, 201(3):751–759.

Gendreau, M., Hertz, A., and Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1093.

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

Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2008). A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. *Networks*, 51(1):4–18.

Iori, M., Salazar-González, J.-J., and Vigo, D. (2007). An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science*, 41(2):253–264.

Lacomme, P., Toussaint, H., and Duhamel, C. (2013). A GRASP×ELS for the vehicle routing problem with basic three-dimensional loading constraints. *Engineering Applications of Artificial Intelligence*, 28(8):1795–1810.

Leung, S. C., Zheng, J., and Zhang, D. (2010). Simulated annealing for the vehicle routing problem with two-dimensional loading constraints. *Flexible Services and Manufacturing Journal*, 22(1-2):61–82.

Leung, S. C., Zhou, X., Zhang, D., and Zheng, J. (2011). Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem. *Computers & Operations Research*, 38(1):205–215.

Lodi, A., Martello, S., and Vigo, D. (1999). Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 11(4):345–357.

Ma, H.-w., Zhu, W., and Xu, S. (2011). Research on the algorithm for 3L-CVRP with considering the utilization rate of vehicles. *Intelligent computing and information science*, 134:621–629.

Tarantilis, C., Zachariadis, E., and Kiranoudis, C. (2009). A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem. *Intelligent Transportation Systems*, 10(2):255–271.

Wei, L., Zhang, Z., Zhang, D., and Lim, A. (2015). A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, 243(3):798–814.

Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T. (2009). A guided tabu search for the vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, 195(3):729–743.

Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T. (2013). Integrated distribution and loading planning via a compact metaheuristic algorithm. *European Journal of Operational Research*, 228(1):56–71.