ERASMUS UNIVERSITY ROTTERDAM
ERASMUS SCHOOL OF ECONOMICS
BACHELOR THESIS ECONOMETRIE EN OPERATIONELE RESEARCH

SEPARATING CONSTRAINTS IN THE CAPACITATED VEHICLE ROUTING PROBLEM

The capacitated vehicle routing problem (CVRP) is a classic optimization problem. In the CVRP a set of customers needs to be satisfied in their demand by vehicles with limited capacity. Augerat et al. (1998) describe the cutting plane algorithm to obtain lower bounds for the CVRP. This algorithm iteratively solves linear relaxations and adds violated capacity constraints. Our research is mainly concerned with investigating heuristic methods to identify violated capacity constraints. We implement the methods described in Augerat et al. (1998) and also design our own advanced tabu and simulated annealing methods. Finally we investigate a variant of the CVRP which has emission zones where only electric vehicles are allowed to come. We develop constraint sets and identification heuristics to obtain lower bounds for this variant of the CVRP.

HARM DE WITH
453504hw

Supervisor:
NAUT BULTEN
Second assessor:
DR. SHADI SHARIF AZADEH

July 6, 2019

Contents

1 Introduction

The Vehicle Routing Problem (VRP) is one of the more well-known optimization problems in logistics. In this problem, a set of customers with a certain demand needs to be supplied by a fleet of vehicles. The problem consists of planning routes for the vehicles such that the demand of each customer is satisfied while minimizing the cost of these routes.

The VRP is a very relevant problem for companies where transport costs make up for a great part of the total operating costs. However, the assumptions behind the VRP are often not a very realistic representation of the transport process in these companies. Therefore, in practice, some constraints need to be added to the VRP. For example, some companies have to supply their customers within certain time windows. Another example is that in practice vehicles have limited capacity. The VRP which also takes into account the capacity of the vehicles is called the Capacitated Vehicle Routing Problem (CVRP), this report is mainly concerned with studying this problem. We also study an extension of the CVRP in which there are some customers which can only be accessed by electrical vehicles, which have limited range.

The CVRP is a well-studied problem, for which numerous solution methods exist, both exact and heuristic. One of these methods is the cutting plane method, as described by Augerat et al. (1998). This approach makes use of linear relaxations of an exact formulation of the problem, and iteratively adds constraints until no violated constraints can be found. If the solution to the linear relaxation is integer and feasible, we have found the optimal solution to the CVRP. However, in most cases, the solution will not be integer. This means we only have a lower bound to the exact solution of the CVRP, which can be used in the branch-and-cut algorithm to obtain an exact solution.

The main goal of this report is to research which ways to find the violated constraints (separating the constraints) are the most effective. More effective separating procedures yield better cuts and thus better lower bounds. To separate the constraints we use several heuristic procedures, such as the greedy randomized algorithm and the tabu search. These procedures are from Augerat et al. (1998). We investigate some alternative versions of the greedy randomized algorithm. Furthermore, we also design our own simulated annealing algorithm. At the start, the simulated annealing algorithm has a high probability of accepting a worse solution. As we iterate, this probability decreases, such that a local (and hopefully global) optimum will be found. Finally we extend the tabu search by incorporating long term memory structures and elite solutions. We investigate how well our methods perform compared to the methods of Augerat et al. (1998).

For the CVRP with emission zones we find a mathematical formulation, and then design a cutting plane algorithm using the same principle as the cutting plane algorithm of Augerat et al. (1998). Finally we investigate the effect of adding some new cuts to this CVRP variant.

In this report we first do a literature study in Section 2. Then, in Section 3, we introduce the mathematical formulations for the CVRP and for the CVRP with emission zones. Next, in Section 4, we briefly explain what data we use. After that, we clarify what how the cutting plane algorithm works, and introduce heuristic methods to find violated constraints in Section 5. Then, in Section 6, we present the results of the described methods. Finally, we give a conclusion in Section 7.

2 Literature review

The VRP was first studied by Dantzig and Ramser (1959). Since then, numerous variants of the VRP have been studied. Also, many solution techniques have been developed to solve these variants of the VRP, both exact and heuristic. For recent surveys, see Caceres-Cruz et al. (2015)

and Braekers et al. (2016).

Laporte et al. (1985) introduced the branch and cut approach for the CVRP, and Augerat et al. (1995) were the first to develop a complete branch and cut approach. Augerat et al. (1998) further studied ways to separate the constraints, which this report is mainly concerned with. Furthermore, Lysgaard et al. (2004) proposed some new procedures to find violated constraints, which solved three benchmark instances to optimality for the first time.

Another class of fairly successful exact approaches to solve the CVRP consists of the branch-and-bound algorithms. One of the earliest of these approaches is by Christofides et al. (1981). Christofides et al. (1981) used a branch-and-bound algorithm, with lower bounds calculated by using Lagrangian relaxation. This procedure solved CVRP instances with up to 25 customers.

Fukasawa et al. (2006) combined the branch-and-cut approach with Lagrangian relaxation. They used Lagrangian relaxation over q-routes, these are routes which visit some customers with total capacity not exceeding the capacity of the vehicles, where customers can be visited more than once. This branch-and-cut-and-price approach solved numerous benchmark instances for the first time, such that all instances with up to 135 nodes were solved to optimality. Fukasawa et al. (2006) showed that combining solution methods may yield more performance improvement than extending the branch-and-cut methods.

Pecin et al. (2017) improved on the branch-and-cut-and-price approach of Fukasawa et al. (2006), and solved all the benchmark instances for exact algorithms. The improved branch-and-cut-and-price algorithm even solved an instance with 360 customers.

Lately, due to increased relevance, the green vehicle routing problem has become popular. This is a variant of the CVRP where the vehicle fleet consists of electrical vehicles with limited range, this problem also takes into account the possibility to recharge the trucks at certain reload stations. Koç and Karaoglan (2016) design a branch-and-cut algorithm for this problem. However, as the CVRP variant we consider simultaneously uses normal and electric vehicles these problems are different.

Glover et al. (1993) give an overview of possible attributes in tabu search methods. For example, they describe possible long term memory structures, which can be used to get a more advanced tabu search procedure than Augerat et al. (1998), which only uses short term memory.

3 CVRP formulations

3.1 Classic CVRP

The CVRP is defined on an undirected and complete graph (V, E). We define $V = \{0, 1, ..., n\}$, where 0 is the depot, and the other nodes are customers. We represent the set of customers by V_0. Further, we denote the demand of customer i by q_i, where the demand is strictly positive. Note that we deviate from Augerat et al. (1998), where d_i is used to denote demand. We do this because later we use d_{ij} to denote distance. Let c_{ij} denote the non-negative cost of traversing edge (i, j). We denote the capacity of each vehicle by C. Finally, let k be the number of vehicles which need to be used.

Let decision variable x_{ij} denote the number of times edge (i, j) is traversed in the solution for all edges $(i, j) \in E$. Define the coboundary of S, $\delta(S)$, as follows: $\delta(S) = \{(i, j) \in E : i \in S, j \in V \setminus S\}$. So the coboundary corresponds to all the edges with one node in S, and the other node outside of S. Let $(S : S')$ be the set of edges with one node in S, and the other node in S', so $(S : S') = \{(i, j) \in E : i \in S, j \in S'\}$. Define $\gamma(S)$ to be the set of edges for which both adjacent nodes lie in S ($\gamma(S) = \{(i, j) \in E : i, j \in S\}$). Furthermore, let $q(S) = \sum_{i \in S} q_i$, be the total demand in set S, and let Q_T be the total demand of all the customers. If F is a set of edges, we denote the sum of the x values of these edges by $x(F) = \sum_{(i,j) \in F} x_{ij}$. Now we can

formulate the CVRP as the following integer problem:

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in E} c_{ij}x_{ij} && && (1)\\
\text{s.t.} \quad & x(\delta(\{0\})) = 2k && && (2)\\
& x(\delta(\{i\})) = 2 && \forall i \in V_0 && (3)\\
& x(\delta(S)) \geq 2r(S) && \forall S \subseteq V_0, S \neq \emptyset && (4)\\
& 0 \leq x_{ij} \leq 1 && \forall (i,j) \in \gamma(V_0) && (5)\\
& 0 \leq x_{ij} \leq 2 && \forall (i,j) \in \delta(\{0\}) && (6)\\
& x_{ij} \in \mathbb{N} && \forall (i,j) \in E && (7)
\end{aligned}$$

Constraint (2) makes sure that k vehicles leave and return back to the depot. Constraints (3) demand that every customer node is visited exactly once.

In constraints (4), for each subset S of the customer set $r(S)$ vehicles are needed, where $r(S)$ is the solution to the bin packing problem (BPP). The solution to the BPP is needed because it determines how many vehicles are needed to satisfy the demand in S, as all goods have to be packed in the vehicles, and these vehicles need to both enter and leave S. However, instead of using the solution of $r(S)$, it is also possible to use any lower bound for $r(S)$. Augerat et al. (1998) use the straightforward lower bound $\lceil q(S)/C \rceil$. Augerat et al. (1995) suggest that solving the BPP hardly yields any improvements compared to using the lower bound $\lceil q(S)/C \rceil$. They do not back up this claim with results, so we investigate the effect of using the exact solution. See for example De Carvalho (1999) for an exact algorithm. It is also possible to use stronger lower bounds for the BPP to obtain better bounds in reasonable time. Martello and Toth (1990) describe some lower bounds. The best performing lower bound can be calculated in $O(n^3)$ makes use of a dominance criterion. The dominance criterion roughly means that when there are two sets of customers, the first set dominates the second if the optimal solution will be better when the first set is assigned to a bin, then when the second set is assigned to a bin.

Constraints (5) - (7) assure that every edge which is not incident to the depot is used once or not at all. We can use this constraint because an edge not adjacent to the depot can not be used twice or more as every node need to be visited by exactly one vehicle. The edges which are incident to the depot can have value 2, this happens on trips where the vehicle only serves one customer, such that the same edge is used twice.

3.2 CVRP with emission zones

Now we introduce an extension to the CVRP. There are some customers in emission zones where only electric vehicles are allowed to come. These electric vehicles have different capacity, different costs, and also have a limited range that they can drive. In contrast to the CVRP in Augerat et al. (1998), we now do not have a certain number of vehicles that we must use. However, we do have an upper limit on both normal and electrical vehicles used, as well as on total vehicles used.

Let k_N and k_E be the maximum number of respectively normal and electric vehicles. Denote the capacities of the normal and electrical trucks by respectively C_N and C_E. Let c_{ij}^N and c_{ij}^E be the costs of traversing edge (i,j) for respectively normal and electric vehicles. R_E is the range that an electric truck can drive. We denote the distance on edge (i,j) by d_{ij}. The set of customers that can only be served by electric vehicles is represented by V_E. The other customers, that can be served by any truck, is $V_A (= V_0 \setminus V_E)$. The new decision variables x_{ij}^N and x_{ij}^E indicate whether edge (i,j) is used by an normal or electric vehicle. We denote whether

an edge is used at all by $x_{ij} = x^N_{ij} + x^E_{ij}$. Finally, y_i is an decision variable which indicates whether node i is served by an normal vehicle (y_i has value 1), or by an electric vehicle (y_i has value 0). We can now describe a integer problem to model the CVRP with emission zones.

$$\begin{aligned}
\min \quad & \sum\nolimits_{(i,j)\in E} c^E_{ij}x^E_{ij} + c^N_{ij}x^N_{ij} && && (8)\\
\text{s.t.} \quad & x(\delta(\{0\})) \leq 2k && && (9)\\
& x^N(\delta(\{0\})) \leq 2k_N && && (10)\\
& x^E(\delta(\{0\})) \leq 2k_E && && (11)\\
& x^N(\delta(\{i\})) = 2y_i && \forall i \in V_0 && (12)\\
& x^E(\delta(\{i\})) = 2(1-y_i) && \forall i \in V_0 && (13)\\
& y_i = 0 && \forall i \in V_E && (14)\\
& x^N(\delta(S)) \geq 2\frac{\sum_{i\in S} q_i y_i}{C_N} && \forall S \subseteq V_0, S \neq \emptyset && (15)\\
& x^E(\delta(S)) \geq 2\frac{\sum_{i\in S} q_i (1-y_i)}{C_E} && \forall S \subseteq V_0, S \neq \emptyset && (16)\\
& x^E(\delta(S)) \geq 2\frac{\sum_{(i,j)\in \gamma(S)\cup\delta(S)} d_{ij}x^E_{ij}}{R_E} && \forall S \subseteq V_0, S \neq \emptyset && (17)\\
& 0 \leq x^N_{ij}, x^E_{ij} \leq 1 && \forall (i,j) \in \gamma(V_0) && (18)\\
& 0 \leq x^N_{ij}, x^E_{ij} \leq 2 && \forall (i,j) \in \delta(\{0\}) && (19)\\
& x^N_{ij}, x^E_{ij} \in \mathbb{N} && \forall (i,j) \in E && (20)\\
& y_i \in \mathbb{B} && \forall i \in V_A && (21)
\end{aligned}$$

Constraints (9) - (11) impose a maximum on the number of vehicles used. Constraints (12) and (13) make sure that each customer i is either visited by a electric truck or by a normal truck, such that the type of the truck does not switch halfway through the route. Constraints (14) make sure that each customer in an emission zone is visited by an electric truck. Constraints (15) and (16) are the capacity constraints for the normal and electric vehicles. In the situation with more types of vehicles, for a given subset S, not all nodes in S have to be served by the type of vehicle considered. This means we can not just use $r(S)$ in the right hand side of the equation. Instead, the total demand served by the considered vehicle is determined in the summations by using the y variables. Then this demand is divided by the capacity of the vehicles considered. We can not easily round this number up, because there are decision variables in this part. Constraints (17) are used to model the range constraint on the electric vehicles. The number of vehicles needed in S satisfy the range constraint, is the total distance covered in S and leaving S by the vehicles by the vehicles leaving S, divided by the range of the electric vehicle. Therefore we sum over all the edges in S, and the edges leaving S. Constraints (18)-(21) are straightforward and need no further explanation.

Constraints (15)-(17) have another advantage. If the integrality of y is relaxed, constraints (15)-(17) will partly prevent mixing of vehicle types on a single route. For example, if there are two "normal" edges (which are not adjacent) on an electric route, S can be chosen such that the left hand side of the equations (15)-(17) is zero, and this yields a violation, as the right hand sides are not zero.

4 Data

4.1 Classic CVRP

For the classic CVRP we use the same data sets as Augerat et al. (1998). The first library with CVRP we use consists of 11 difficult instances and is denoted by LITLIB. Furthermore, we use the ALIB library. This libary consists of instances with customers uniformly distributed on a 100 by 100 plane. Finally we use the library BLIB. The BLIB library consists of instances with clustered customers, to get an approximation of real life problems.

4.2 CVRP with emission zones

To test our methods on the CVRP with emission zones, we use the data sets and upper bounds from de Feijter et al. (2018) to evaluate the quality of the lower bounds obtained. These are small instances with around twenty customers per instance. Cost per kilometer is 4 for normal vehicles, and 5 for electric vehicles. The capacity of the normal vehicles is 104, and the electrical vehicles have a capacity of 78. The range of the electrical vehicles is 100 kilometers. Finally, 8 normal vehicles and 5 electrical vehicles can be used. The maximum total trucks is also 8. See table 1 for more details.

Table 1: Data instances CVRP with emission zones

Data instance	Customers	Emission zone customers	Upper bound
Monday	21	10	1607.48
Tuesday	21	10	1530.77
Wednesday	20	10	1356.27
Thursday	20	10	1879.60
Friday	19	10	1353.95
Saturday	20	10	1564.76
Sunday	20	10	1678.64

5 Cutting plane algorithm

The cutting plane algorithm is used to find a lower bound for the CVRP, we describe this algorithm in this section. First we solve a relaxation of the described mathematical program. We omit the capacity and integrality constraints (4) and (7), to get a Linear Program (LP). If the solution is integer, we can easily check if it is feasible by verifying the induced routes, which yields a violated capacity constraint if the solution is not feasible. If it is not integer, we attempt to identify a violated capacity constraint, and add this constraint to the relaxation. We repeat this process until we have found a feasible solution, or when we cannot find a violated capacity constraint. In the last case we only have a lower bound for the optimal solution.

5.1 Identifying violated constraints in the CVRP

We try to find the most violated constraint. Thus we compute $min\{x(\delta(S)) - 2r(S) : S \subseteq V_0, S \neq \emptyset\}$. To find the violated capacity constraints, we propose a couple of heuristic methods. Also note that we use the lower bound $\lceil q(S)/C \rceil$ in these methods, as in most methods this lower bound can not be replaced by another the solution to $r(S)$ without changing the structures of the algorithms. Finally, if more than $min\{125, 2n\}$ cuts are identified in one iteration, we only select the $min\{125, 2n\}$ best of the identified cuts. If this happens we also discard the cuts for which the violation is less than 0.01.

Simple identification techniques

Let $G(x)$ be the graph induced by a solution x to the LP. Edge (i, j) is in $G(x)$ if $x_{ij} > 0$. We refer to $G(x)$ as the support graph. Then, if there is an edge (i, j) with $x_{ij} = 1$, nodes i and j are shrunk to a super node p. p then has demand $q_i + q_j$, and $x_{vp} = x_{vi} + x_{vj}$ for all v in V. This process is repeated until there are no edges with weight 1 left. Note that every super node still has degree 2. So if a node occurs with demand greater than the capacity of a vehicle, we have identified a violated capacity constraint. It turns out that using the reduced $G(x)$ does not cause us to miss a violated constraint if there are any.

Another simple identification technique is to consider the connected components of $G(x)$ and $G(x) \setminus \{0\}$. Note that if S is a connected component of $G(x)$ with the depot in S, we remove the depot from S.

Finally, whenever we find a subset S of V_0 for which the capacity constraint is violated, we apply the following procedure: if $pC \geq q(S) \geq (p - 0.33)C$, for a $p \in \mathbb{N}$, we check all sets $S \cup \{v\}$ for all v which are adjacent to at least one node in S. If $\lceil q(S)/C \rceil = p$ and $\lceil q(S \cup \{v\})/C \rceil = p + 1$, then $x(\delta(S \cup \{v\})) \geq 2p + 2$ implies that $x(\delta(S)) - x((S : \{v\})) \geq 2p$. As $x((S : \{v\})) \geq 0$ this equation dominates the capacity constraint of S, so then we only add the capacity constraint of $S \cup \{v\}$. Also, every time we check whether a set S violates the capacity constraint, we also check it for the complement set $V_0 \setminus S$.

Constructive heuristic

In the first method described in Augerat et al. (1998), restrictions (4) are relaxed to $x(\delta(S)) \geq 2q(S)/C$. The advantage of this procedure is that the set S with the highest violation of the constraint can be found in polynomial time by calculating a minimum capacity cut. This minimum cut is in an extended graph $H(x)$ which is a modification of support graph $G(x)$. The extended graph has an extra node $n + 1$ which is connected to all nodes in V_0. Each edge is assigned a capacity as follows: the capacity of edges (i, j) with $i, j \in V_0$ is x_{ij}. For edges adjacent to the depot the capacity is $max\{0, x_{0j} - 2q_j/C\}$. The edges adjacent to the new node $n+1$ have capacity $max\{0, 2q_j/C - x_{0j}\}$. If a set is found which violates the relaxed restriction, it obviously also violates the stronger restriction with $2\lceil q(S)/C \rceil$ or $2r(S)$ on the right hand side.

Greedy randomized algorithm

In this procedure, in each iteration, a node v (other than the depot) is added to a set S, such that $x(\delta(S \cup \{v\}))$ is minimized. This process is repeated until S contains all the customers. As described in Augerat et al. (1998) there are several possible methods to build an initial set S, such as starting with a random subset of V_0 or a single node. Augerat et al. (1998) do not explain how the initial sets are exactly created. We build the initial sets by adding each node to the initial set S with a certain probability. This process is applied ten times the number of customers in the problem instance. We investigate the greedy algorithm for several selecting probabilities.

Alternative greedy

In stead of minimizing $x(\delta(S \cup \{v\}))$ in the greedy randomized algorithm, it is also possible to maximize the violated inequality. That is, we maximize $2\lceil (q(S) + q_v)/C \rceil - x(\delta(S \cup \{v\}))$.

Greedy with exact BPP bounds

We use the greedy randomized algorithm to investigate the effect of using the exact BPP bounds in the capacity constraints. The lower bound $\lceil q(S)/C \rceil$ is still used when identifying the constraints, because the solving the BPP is computationally expensive. However, when a LP solution violates the relaxed constraint $x(\delta(S)) \geq 2\lceil q(S)/C \rceil$, it will also violate the constraint $x(\delta(S)) \geq 2r(S)$. Therefore, when a violation of a relaxed constraint is identified, we compute the solution to the bin packing problem $r(S)$, and add the potentially stronger restriction to the model. The BPP can be modeled by the mathematical formulation described in De Carvalho (1999).

$$\begin{aligned}
\min \quad & \sum_{i=1}^{k} y_i && && (22)\\
\text{s.t.} \quad & \sum_{j \in S} q_j z_{ij} \leq C y_i && \forall i = 1, ..., k && (23)\\
& \sum_{i=1}^{k} z_{ij} = 1 && \forall j \in S && (24)\\
& y_i \in \mathbb{B} && \forall i = 1, ..., k && (25)\\
& z_{ij} \in \mathbb{B} && \forall i = 1, ..., k, \forall j \in S && (26)
\end{aligned}$$

y_i indicates whether 'bin' (vehicle) i is selected. Furthermore, z_{ij} represents whether 'item' (customer) j is assigned to bin i. The objective function minimizes the number of bins used. Constraints (23) demand that the contents of the bin do not exceed its capacity, and constraints (24) make sure that each item is assigned to exactly one bin.

Greedy with relaxed bounds on x_{ij}

Augerat et al. (1998) at some point are not clear whether they also relax constraints (5) and (6). However, our results show that the non-negativity of the x-variables is necessary. We relax the upper bound on the x-variables in the greedy randomized algorithm, and check how the results change. In fact, in the appendix we show in theorem 1 that the upper bounds for the x-variables automatically follow if constraints (3) and (4) are satisfied.

Tabu search

In the tabu search we distinguish between different classes of capacity constraints, namely for $p = 1, ..., k-1$ we analyze the sets $S \subseteq V_0$ for which $(p - llimit)C \leq q(S) \leq (p + ulimit)C$. These sets S need around p vehicles to satisfy the demand of the customers.

If S is a subset of V_0 we define a simple neighbourhood $N^+(S)$ for adding nodes to S: a node in $V_0 \setminus S$ belongs to $N^+(S)$ if it is adjacent in $G(x)$ to a node in S. In the same spirit, $N^-(S)$ denotes the set of nodes in S which are adjacent to a node in $V_0 \setminus S$. This last neighbourhood can be used for removing nodes from S. Now we denote the set of candidate nodes to be added as $C^+(S)$. We want to stay in the collection of sets for which around p vehicles are needed after adding the new node, that means that we need to consider $C^+(S) = \{v \in N^+(S) : q(S) + q_v \leq C(p + ulimit)\}$. The same reasoning is used when defining the canidate nodes for removal by $C^-(S) = \{v \in N^-(S) : q(S) - q_v \geq C(p - llimit)\}$.

The tabu search starts with the set $S = \{i\}$, for a certain customer i. Initially, p is set to 1. The algorithm is run for a randomly selected half of the nodes of $G(x)$. Then, in the expansion phase of the algorithm, the set S is expanded by adding nodes until $C^+(S)$ is empty. Instead of adding $v \in N^+(S)$ such that $x((S : \{v\}))$ is maximal (which is equivalent to $x(\delta(S \cup \{v\}))$ minimal), a node is randomly selected to construct more diverse solutions. However, v is chosen

such that $x((S : \{v\}))$ is close to the maximum value, to also maintain some quality. The parameter *per* determines how high this quality standard is.

After the expansion phase, the interchange phase of the algorithm is called. In the interchange phase, nodes are removed and added for *tope* iterations. When a node is added or removed from S, it becomes a tabu move, and thus it will be forbidden to reverse this move. This tabu lasts for a couple of iterations, which number is determined by the parameter *tll*. When there are no possible moves, the interchange phase is terminated early. After the interchange phase p is increased by one, and the algorithm goes back to the expansion phase if $p \leq k-1$. If $p = k$, the algorithm terminates. An overview of the algorithm can be found in algorithm 1. The described procedure is applied *ntimes*.

Algorithm 1: Tabu algorithm

Set $p = 1$.
Expansion phase
E.1: compute $C^+(S)$. If $C^+(S)$ is empty, go to I.0.
E.2: compute $M = max\{x((S : \{v\})) : v \in C^+(S)\}$ and randomly select a node $v \in C^+(S)$ such that $M - per \leq x((S : \{v\})) \leq M$.
E.3: add v to S, check equation (4), and go to E.1.
Interchange phase
I.0: set $iter = 1$.
I.1: compute $C^-(S)$ and $C^+(S)$, and remove the tabu moves. If $C^-(S) \cup C^+(S)$ is empty, go to I.4.
I.2: let v be the node which maximizes

$$\{\{x((S : \{v\})), v \in C^+(S)\}, \{x((V_0 \setminus S : \{v\})), v \in C^-(S)\}\}.$$

I.3: depending on whether $v \in C^+(S)$ or $v \in C^-(S)$, add v to or remove v from S, and check equation (4). Increment *iter* by one. If $iter > tope$, go to I.4, else, go to I.1.
I.4: do $p = p + 1$. If $p \leq k - 1$, go to E.1, otherwise, stop.

Advanced tabu search

Now we propose a more advanced version of the tabu search. This tabu search algorithm makes use of long term memory. We penalize moves which occur relatively often, to force the tabu search to find more diverse solutions. So, when adding a node v to S, instead of evaluating the maximal $x((S : \{v\}))$, we evaluate $a(S, v) = x((S : \{v\})) - \beta^+ \cdot af_v$. Here af_v is the percentage which adding v makes up of the total executed add-moves, multiplied by the number of nodes in $G(x)$, for scaling purposes. β^+ is a positive parameter determining the weight of the penalty. When removing a node v from S, we evaluate $r(S, v) = x((V_0 \setminus S : \{v\})) - \beta^- \cdot rf_v$. Here rf_v denotes the frequency of removing v relative to all other executed remove-moves, also multiplied by the number of nodes in $G(x)$. The positive parameter β^- denotes the weight of the penalty. By using the penalty function, which provides diversity in the solutions, the need to randomly choose nodes in phase E.2 is gone. So in E.2 we can just choose the node with the best score.

Another extra feature is that the advanced tabu search keeps track of a set with elite solutions. The elite solutions are the globally best solutions (minimal values of $x(\delta(S)) - 2\lceil q(S)/C \rceil$). This set of elite solutions has a maximum of n_{elite} elements. Whenever a move yields a solution which is better than one of the elite solutions, we ignore both the short and long term tabu measures, and select this move anyway.

Note that it would be useless to also have the *ntimes* parameter for the advanced tabu algorithm, as there is no randomness involved. Instead we employ higher values of *tope* to take

optimal advantage of the long term memory structure. The updated algorithm can be found in algorithm 2.

Algorithm 2: Advanced tabu algorithm

Set $p = 1$.
Expansion phase
E.1: compute $C^+(S)$. If $C^+(S)$ is empty, go to I.0.
E.2: select v that maximizes $\{a(S, v) : v \in C^+(S)\}$.
E.3: add v to S, check equation (4), and go to E.1.
Interchange phase
I.0: set $iter = 1$.
I.1: compute $C^-(S)$ and $C^+(S)$.
I.2: let v be the node which minimizes

$$\{\{x(\delta(S\cup\{v\}))-2\lceil q(S\cup\{v\})/C\rceil, v \in C^+(S)\}, \{x(\delta(S\backslash\{v\}))-2\lceil q(S\backslash\{v\})/C\rceil, v \in C^-(S)\}\}.$$

I.3: depending on whether $v \in C^+(S)$ or $v \in C^-(S)$, add v to or remove v from S, check equation (4) and go to I.6 if it is better than one of the elite solutions.
I.4: remove the tabu moves from $C^+(S)$ and $C^-(S)$ and compute v which maximizes

$$\{\{a(S, v), v \in C^+(S)\}, \{r(S, v), v \in C^-(S)\}\}.$$

I.5: depending on whether $v \in C^+(S)$ or $v \in C^-(S)$, add v to or remove v from S, and check equation (4).
I.6: increment *iter* by one. If $iter > tope$, go to I.7, else, go to I.1.
I.7: do $p = p + 1$. If $p \leq k - 1$, go to E.1, otherwise, stop.

Simulated annealing

Finally, we introduce a simulated annealing method. The method is similar to the tabu search of Augerat et al. (1998), with the difference that the interchange moves are not guided by a tabu list, but by a simulated annealing principle. The quality of a new solution is evaluated by the function $f(S) = x(\delta(S)) - 2\lceil q(S)/C\rceil$, where a lower value indicates higher quality. If a solution is of higher quality, then it is always accepted. However, if a solution is of lower quality, it is selected with probability $e^{-(f(S')-f(S))/T}$. This probability depends on T, which is the 'temperature'. The temperature starts high, and will gradually decrease as the iteration counter increases. Lower temperature means a lower probability of selecting a worse solution, thus being flexible with worse solutions at the start, to explore the search space, but to also get to a high quality solution at the end. We calculate the temperature by using the formula $T = tope/(10 \cdot iter)$. This procedure is also described in algorithm 3.

Algorithm 3: Simulated annealing algorithm

Set $p = 1$.
Expansion phase: identical to the expansion phase in algorithm 1.
Interchange phase:
I.0: set $iter = 1$.
I.1: set $T = tope/10iter$.
I.2: pick a random node v from $C^+(S) \cup C^-(S)$.
I.3: let S' be $S \cup \{v\}$ if $v \in C^+(S)$ and $S \setminus \{v\}$ if $v \in C^-(S)$, and check equation (4).
I.4: always select S' as the new solution if $f(S') < f(S)$, and if $f(S') \geq f(S)$ select S' with probability $e^{-(f(S')-f(S))/T}$. Increment $iter$ by one. If $iter > tope$ go to I.5, else, go to I.1.
I.5: do $p = p + 1$. If $p \leq k - 1$, go to E.1, otherwise, stop.

5.2 Identifying violated constraints in the CVRP with emission zones

To compute a lower bound, we use the same cutting plane algorithm as for the standard CVRP. The only difference is that there is more than one constraint family. We check in every iteration for every constraint family if there is a violated constraint. The algorithm only stops if no violated constraint can be found in all of the constraint families. We relax constraints (15)-(17) and the integrality constraints (20)-(21). We use a tabu search heuristic to identify violations of constraints (15)-(17), and add these to the model. Also we investigate the effect of enforcing the integer restriction on the y variables.

Extra cuts

Because constraints (15)-(17) are rather weak as they can not be rounded up, we introduce additional restriction sets to obtain better lower bounds. The first extra restriction set strengthens the electric capacity constraints (restrictions (16)) by introducing

$$x^E(\delta(S)) \geq 2\lceil q(S)/C_E \rceil \tag{27}$$

for all sets S which are contained in V_E. This restriction is possible as $y_i = 0, \forall i \in V_E$. So there are no decision variables in the right hand side, and thus it can be rounded up. We refer to these constraints as the improved electric capacity constraints.

The second set of restrictions we introduce is very similar to the capacity constraints in the classic CVRP:

$$x(\delta(S)) \geq 2\lceil q(S)/max\{C_N, C_E\} \rceil \tag{28}$$

for all subsets S of V_0. This constraint omits the decision variables in the right hand side by using the fact that every customer has to be served by a vehicle, normal or electric. And then, on the right hand side the minimal number of vehicles is calculated by using the minimal number of vehicles needed if all customers in S are served by the vehicle with the biggest capacity. This way we omit the decision variable in the right hand side such that we can round up. If C_N is close to C_E this set of constraints is relatively strong. These constraints will be referred to as the total capacity constraints

These two new constraints are very useful for sets with demand just exceeding an integer multiple of C_N or C_E, as these sets will see a strong increase on the right hand side compared to the old capacity constraints.

Updated simple identification techniques

Note that we cannot use most of the simple identification techniques. The shrinking procedure for example can possibly miss violated range constraints. We still use the connect component identification technique, however we do use it on the graph $G(x^N)$ for the normal capacity constraints. We use support graph $G(x^E)$ for the electric capacity constraints, range constraints, and the improved electric constraints. Finally, we use $G(x)$ for the total capacity constraint. We also refrain from checking $V_0 \setminus S$ every time a violated capacity constraint is found, as computing complements is not as easy in this problem. We also refrain from checking and $S \cup \{v\}$ for all v adjacent in the relevant support graph, because we do not round up, so we cannot use the same criteria for "promising sets" as in the classical CVRP.

Tabu search with emission zones

The tabu search procedure we use is very similar to the tabu search for the classical CVRP. However, instead of analyzing the sets $S \subseteq V_0$ with $(p - llimit)C \leq q(S) \leq (p + ulimit)C$ for an integer p, we analyze sets $p - llimit \leq |S| \leq p$. We do this because in constraints (15)-(17) the sets with demand around C_N or C_E are not necessarily more promising, as we do not round up the right hand side. So now we allow all possible S, and to not search a space too large at a time, we restrict the sets S based on cardinality. However, we let p range from 1 to n, to not exclude any possible subsets S.The sets $C^+(S)$ and $C^-(S)$ are changed accordingly.

6 Results

Now we present the results of the described methods. To obtain these results, we used an Intel Core i7-5500U and 8 GB RAM. The programming language we used is Java, and we solved the linear programs using CPLEX. We use the DirectedGraph and DirectedGraphEdge classes code of Bouman (2018) for a undirected graph class.

6.1 Comparison with Augerat et al. (1998)

We do an extensive comparison for our results of three of the algorithms in Augerat et al. (1998): the constructive heuristic, the greedy randomized heuristic and the tabu1 heuristic. The last of which is the tabu heuristic with certain parameter settings.

Constructive heuristics

The results of our implementation and the results of the implementation of Augerat et al. (1998) can be found in table 2 and 3. Note that we updated the gaps of Augerat et al. (1998), as some of the upper bounds they used are outdated by now.

Table 2: LITLIB results for the constructive heuristic

Data set	Lower bound	Gap	Cuts	Iterations	Separating time	CPLEX time
M-n101-k10	818.5	0.18	848	83	1.19	10.01
E-n30-k3	508.5	4.78	43	12	0.01	0.00
E-n33-k4	832.5	0.30	213	34	0.04	0.03
E-n51-k5	507.769	2.54	123	23	0.06	0.06
E-n101-k8	791.357	3.14	692	51	1.61	4.93
E-n76-k10	773.672	6.79	660	39	0.43	0.71
E-n76-k8	702.547	4.42	734	61	0.57	0.81
E-n76-k7	659.510	3.30	707	49	0.70	0.77
F-n135-k7	1152.35	0.83	2007	159	4.36	78.28
F-n45-k4	720.5	0.48	139	21	0.04	0.02
F-n72-k4	232.5	1.90	113	20	0.05	0.03

Table 3: LITLIB results for the constructive heuristic in Augerat et al. (1998)

Data set	Lower bound	Gap	Cuts	Iterations	Separating time	CPLEX time
M-n101-k10	818.167	0.22	542	85	13.06	46.85
E-n30-k3	508.5	4.78	68	11	0.84	0.50
E-n33-k4	831.667	0.40	171	26	2.14	1.48
E-n51-k5	510.792	1.96	100	16	1.33	1.38
E-n101-k8	789.816	3.33	421	56	11.23	26.41
E-n76-k10	773.206	6.84	383	50	8.08	22.45
E-n76-k8	703.009	4.35	647	77	14.12	32.08
E-n76-k7	658.641	3.43	461	57	11.30	18.46
F-n135-k7	1152.66	0.80	1547	255	64.84	434.65
F-n45-k4	723.333	0.09	129	27	1.90	1.84
F-n72-k4	232.5	1.90	105	10	0.90	1.76

First, we observe that the average gaps are quite close to each other. The average gap for our lower bounds is 2.60, while for Augerat et al. (1998) it is 2.55. However, one could expect the results to be exactly the same, as there is no randomness involved in the constructive heuristic. This is true, however, the randomness does occur when shrinking the support graph $G(x)$. Augerat et al. (1998) does not specify a certain rule for selecting the edges to shrink. So the edges are arbitrarily selected, leading to different support graphs, and other identified constraints (see the appendix for an example). The constructive heuristic may be extra sensitive to this, as not many cuts are generated in this method. Simpler instances also are more sensitive to this, as one missed important constraint may have more impact on the final lower bound.

The number of cuts and iterations seems to be reasonable close. Due to the arbitrariness of the shrinking of the support graph it is not surprising to see some differences, as different identified constraints in the early phases may lead to different paths in exploring the cuts.

Finally we note that our implementation is almost fifteen times as fast the implementation in Augerat et al. (1998), which is not surprising considering Augerat et al. (1998) was written in 1998.

Tabu1

Now we present the results of the tabu1 heuristic. The tabu1 works in two phases. In the first iterations the tabu search is run with settings: $ntimes = 1, tope = 10, tll = 5, ulimit = 0.3, llimit = 0.1, per = 0$. If the increase of the lower bound in the last three iterations is less than 0.5%, *ntimes* changes to 3, *per* to 0.2 and *tope* to 20. The results four our implementation and the implementation of Augerat et al. (1998) can be found in tables 4 and 5. For both

implementations the average gap is 2.02. Our implementation performs slightly better on five instances, and slightly worse on one instance. So the results for the tabu1 heuristic seem to agree more than the constructive heuristic. This may be due to that tabu1 explores more cuts, and thus is less likely to miss one important cut. Now we see slightly more differences in the number of cuts and iterations, compared to the constructive heuristic. This is not surprising, as the tabu search itself also employs randomness.

Table 4: LITLIB results for the tabu 1 heuristic

Data set	Lower bound	Gap	Cuts	Iterations	Separating time	CPLEX time
M-n101-k10	819.5	0.06	1027	58	7.71	6.82
E-n30-k3	508.5	4.78	71	10	0.02	0.01
E-n33-k4	833.5	0.18	214	19	0.19	0.03
E-n51-k5	514.524	1.24	419	27	1.05	0.11
E-n101-k8	796.349	2.53	908	48	16.84	4.85
E-n76-k10	789.415	4.89	982	45	23.49	1.45
E-n76-k8	711.205	3.24	780	40	11.53	0.81
E-n76-k7	661.278	3.04	610	39	7.70	0.47
F-n135-k7	1158.25	0.32	1710	96	29.55	46.18
F-n45-k4	724	0.00	213	20	0.10	0.02
F-n72-k4	232.5	1.90	119	13	0.06	0.02

Table 5: LITLIB results for the tabu 1 heuristic in Augerat et al. (1998)

Data set	Lower bound	Gap	Cuts	Iterations	Separating time	CPLEX time
M-n101-k10	819.333	0.09	1014	25	14.69	34.46
E-n30-k3	508.5	4.78	70	16	0.15	0.58
E-n33-k4	833.5	0.18	272	24	1.38	2.15
E-n51-k5	514.524	1.24	544	21	5.89	3.10
E-n101-k8	796.284	2.53	1597	30	89.70	50.75
E-n76-k10	789.344	4.90	2442	47	136.51	89.25
E-n76-k8	711.141	3.25	2063	48	98.14	45.35
E-n76-k7	661.299	3.04	1568	47	71.40	25.37
F-n135-k7	1158.02	0.34	2506	33	60.42	232.27
F-n45-k4	724	0.00	156	20	0.76	1.55
F-n72-k4	232.5	1.90	171	13	0.47	2.26

Greedy randomized heuristic

Finally we compare the greedy randomized heuristic implementations. First we investigate which selecting probability is the most successful. Due to the randomness of the greedy algorithm, we run the greedy algorithm 50 times for each selecting probability, each time with different seeds. The results can be found in table 6.

Table 6: LITLIB results greedy heuristic with different selecting probabilities

Selecting probability	Mean	Standard deviation mean
5%	2.159	0.003
10%	2.111	0.002
15%	2.131	0.005
20%	2.176	0.005

It turns out that the selecting probability 10% yields the best results. In table 7 we present the results for one of the runs of the greedy algorithm. The average gap in this run is 2.13,

which is above the average of the 50 runs. The gap in the implementation of Augerat et al. (1998) is 2.09, which is slightly below average of the 50 runs. However, the difference is not significant, as the standard deviation of one run of the greedy heuristic is 0.014.

We also see that Augerat et al. (1998) often has more cuts. This can easily be explained due to the fact that we do not know how the initial sets are generated in the implementation of Augerat et al. (1998).

Finally we note that our implementation of the greedy randomized algorithm seems to be inefficient, as the total separating time is 264.71, while the separating time of Augerat et al. (1998) is only 232.07. Which is significantly slower, considering the increase in computing speed in the last twenty years.

Table 7: LITLIB results for the greedy heuristic

Data set	Lower bound	Gap	Cuts	Iterations	Separating time	CPLEX time
M-n101-k10	819.5	0.06	1313	50	17.94	6.70
E-n30-k3	508.5	4.78	47	10	0.02	0.01
E-n33-k4	833.5	0.18	241	20	0.35	0.03
E-n51-k5	514.027	1.34	334	20	1.69	0.06
E-n101-k8	795.381	2.65	731	26	28.63	2.31
E-n76-k10	786.605	5.23	1222	45	38.11	1.74
E-n76-k8	708.524	3.60	803	31	27.78	1.00
E-n76-k7	659.840	3.25	624	29	13.00	0.45
F-n135-k7	1157.705	0.37	1920	78	136.71	49.67
F-n45-k4	723.5	0.07	200	24	0.37	0.04
F-n72-k4	232.5	1.90	126	11	0.11	0.02

Table 8: LITLIB results for the greedy heuristic in Augerat et al. (1998)

Data set	Lower bound	Gap	Cuts	Iterations	Separating time	CPLEX time
M-n101-k10	819.333	0.08	1470	40	20.37	46.84
E-n30-k3	508.5	4.78	86	21	0.73	0.80
E-n33-k4	833.5	0.18	311	21	1.41	1.44
E-n51-k5	514.524	1.24	408	22	3.72	2.82
E-n101-k8	795.866	2.59	1381	38	27.37	31.45
E-n76-k10	787.873	5.08	1526	47	28.59	48.95
E-n76-k8	709.64	3.45	1256	26	10.87	24.67
E-n76-k7	660.675	3.13	952	41	19.17	17.55
F-n135-k7	1156.41	0.48	4310	102	114.75	489.96
F-n45-k4	723.667	0.05	182	27	1.58	2.06
F-n72-k4	232.5	1.90	183	20	3.51	2.61

6.2 Variants on the greedy randomized heuristic

In table 9 the results for some variants of the greedy algorithm are displayed. First we observe that relaxing the bounds on x, or using BPP bounds for the capacity constraints does not really have a great effect. Note that the greedy algorithm with exact BPP bounds has a significant increase in separating time, this due to the fact that we counted solving the bin packing problem as separating time and not CPLEX time.

Alternative greedy does perform well. It has a lower average gap and finds more cuts. However, it still may be due to the randomness in the greedy methods. Due to the success for greedy we also try maximizing the constraint violation instead of minimizing $x(\delta(S))$ in the tabu search heuristic. With the same settings as in tabu1, this yields a average gap of 2.04,

which is worse than the 2.02 of tabu1. So in the tabu search it is not a profitable deviation to maximize the violation of the capacity constraints. This can be explained by the fact that the tabu heuristic already searches for sets which need around p vehicles. This has as a consequence that it is not necessary to include the number of vehicles needed in the move evaluation.

Table 9: Summary results for variants of the greedy randomized heuristic

Greedy variant	Avg. gap	Max gap	Best	Cuts	Iterations	Sep. time	CPLEX time
Standard	2.13	5.23	4	687.36	31.27	24.06	5.64
Relaxed bounds on x	2.10	5.29	6	692.27	34.64	19.38	6.06
Exact BPP bounds	2.12	5.16	4	715.18	34.27	73.88	5.88
Alternative	2.08	5.05	4	871.45	36.00	33.23	10.73

6.3 Summary results

Now we present the summarized results of some additional methods. The results can be found in table 10. First we describe the parameter settings of some of the methods. We introduce two additional versions of the tabu heuristic. Tabu2 is the same as tabu1 in the first iterations, and in the second phase (when the objective value has not improved at leas 0.5% in the last three iterations) it still uses the same settings as tabu1 in phase one. Tabu3 is the same as tabu1, but when no violated constraint is found, the tabu search is run again with the following parameters changed: $ntimes = 3$, $per = 0.4$, $tope = 30$, $tll = 15$, $ulimit = 0.45$ and $llimit = 0.25$. If that still does not yield a violated constraint, the tabu search is run again with $ntimes$ and per changed to 6 and 0.6.

The advanced tabu heuristic has similar settings as tabu3. In the first iteration the settings are exactly the same, and the long term memory parameters are both 0. In the second phase, the settings are still the same as in tabu3, with the only difference that $tope = 60$ instead of $tope = 20$. This is to compensate for the fact that tabu3 has $ntimes = 3$ in this phase and that advanced tabu does not have a $ntimes$ parameter. This phase also makes use of the long term memory by setting β^+ and β^- to 0.1. If no violations are found, we run the advanced tabu again with the same parameters as in tabu3, with some exceptions: again $tope$ is higher, and is set to 500, tll is set to 5 (instead of 15) as long tabu lists are not necessary due to the long term memory. The long term parameters are also increased to 0.2. If this still does not yield a violated capacity constraint, the advanced tabu search is run again with $tope = 2000$, $\beta^+ = 1, \beta^- = 1$ and $tll = 5$. Finally we note that in every phase we set n_{elite} to 10.

Simulated annealing (SA) only uses the first two phases, like tabu1. It also has the same parameter settings as tabu1, except $tope$ is 30 in the first phase and 60 in the second phase.

Table 10: Summary results for the LITLIB instances

Method	Avg. gap	Max gap	Best	Cuts	Iterations	Sep. time	CPLEX time
Constructive	2.60	6.79	2	570.82	50.18	0.82	8.70
Greedy	2.13	5.23	4	687.36	31.27	24.06	5.64
Alt. greedy	2.08	5.05	4	871.45	36.00	33.23	10.73
Tabu1	2.02	4.89	7	641.18	37.73	8.93	5.52
Tabu2	2.08	5.00	3	694.91	33.55	2.25	4.89
Tabu3	2.01	4.89	9	728.18	39.27	11.57	6.43
Advanced tabu	2.01	4.89	10	796.27	36.27	19.90	6.48
Simulated annealing	2.04	4.94	6	638.36	34.09	16.72	3.72

In table 10 the summary results are shown for the most relevant methods. We observe that constructive yields lower bounds of relatively low quality: the average gap is much higher than

all the other methods. In Augerat et al. (1998) the performance of the greedy randomized heuristic is close to the performance of tabu2. In our implementations, greedy seems to perform a bit worse, which can be caused by the randomness, or by the quality of the initial sets. The tabu heuristics seem to be performing the best. Just as in Augerat et al. (1998), tabu3 performs slightly better than tabu1. Our simulated annealing algorithm does not succeed in performing better than the tabu search heuristics. It performs slightly worse, but it clearly does perform better than the greedy heuristics.

The long term memory and elite solution structures of the advanced tabu algorithm do not seem to improve the simple tabu search. Now it is an interesting question to ask why our proposed did not succeed in improving the gaps of tabu3. Of course, a reasonable explanation could be that the search methods are not advanced enough. However this does not always hold. For example, instance E-n30-k3 has a gap of 4.78% in all the considered constraint identifiers. It turns out that the support graph of the corresponding solution can be shrunk to a graph with only 6 customers. With only 6 customers there are only $2^6 - 1$ possible capacity constraints sets. These sets can be easily checked by brute force, and it turns out that there are no violated capacity constraints. So it could also be a reasonable possibility that the lower bounds achieved by tabu3 and advanced tabu can hardly be improved by brute forcing all capacity constraints.

6.4 Comparison heuristics

To compare the methods with each other directly we try every identification heuristic for one iteration after the last iteration of the cutting plane algorithm for every heuristic. The results can be found in tables 11 and 12. Entry (I, J) means that heuristic I is applied first, and then heuristic J. When heuristic J is applied we mean that we use the parameter settings of the last phase of the algorithm.

Table 11: Average number of cuts

	Constr.	Greedy	Alt. greedy	Tabu1	Tabu2	Tabu3	Adv. tabu	SA
Constr.	-	209.64	423.73	2065.64	264.82	3466.64	1419.64	1114.27
Greedy	2.45	-	42.64	997.09	111.18	2663.09	817.00	120.09
Alt. greedy	1.00	14.82	-	207.00	25.91	438.09	148.82	83.00
Tabu1	0.09	0.64	1.18	-	0.00	7.45	3.45	6.82
Tabu2	2.64	5.82	27.36	250.27	-	590.91	186.64	21.45
Tabu3	0.00	0.82	0.00	0.00	0.00	-	0.45	1.36
Adv. tabu	0.00	0.00	0.00	0.09	0.00	0.36	-	0.45
SA	0.00	0.27	7.00	68.91	0.82	210.18	42.91	-

First the number of cuts in our results is much higher than in Augerat et al. (1998). While in Augerat et al. (1998) the average number of cuts did not once reach 100, in our implementation the average number of cuts easily gets to 1000. Note that in the other overviews only the cuts were counted that were added to the model, which means that no more than 125 cuts per iteration were added, which explains why these numbers are not so high as well. We tried several different ways of counting the cuts, such as only counting the cuts found by the identification algorithm itself, and not by the neighbour and complement checks of the simple identification techniques. None of these succeeded in getting approximately the same number of cuts as Augerat et al. (1998). We suspect that this difference is just due to the way of counting, as the quality of the lower bounds obtained do agree with the results in Augerat et al. (1998).

Table 12: Average slack of cuts

	Constructive	Greedy	Alt. greedy	Tabu1	Tabu2	Tabu3	Adv. tabu	SA
Constructive	-	0.82	0.75	0.85	0.80	0.86	0.84	0.83
Greedy	0.24	-	0.26	0.44	0.27	0.48	0.47	0.37
Alt. greedy	0.13	0.17	-	0.27	0.21	0.27	0.24	0.24
Tabu1	0.01	0.02	0.03	-	0.00	0.05	0.04	0.02
Tabu2	0.17	0.33	0.41	0.46	-	0.46	0.46	0.30
Tabu3	0.00	0.02	0.00	0.00	0.00	-	0.02	0.00
Adv. tabu	0.00	0.00	0.00	0.00	0.00	0.00	-	0.00
SA	0.00	0.01	0.04	0.10	0.02	0.12	0.10	-

We observe that the advanced tabu algorithm performs poorly compared to tabu3. This can be explained by the fact that the advanced tabu algorithm in phase 4 focuses on diversity by having high values for β^+ and β^-, which has the effect that it does not find a lot of cuts, but it does find cuts which are hard to find.

Also in our implementation there is quite a difference between the tabu1 and the tabu3 results, in contrast to Augerat et al. (1998). Tabu3 finds around twice to thrice as many cuts as tabu1, while in Augerat et al. (1998) they find exactly the same number of cuts. The slack of the cuts in tabu1 and tabu3 does follow the pattern of Augerat et al. (1998), where tabu3 is slightly better.

Finally it is interesting to note that simulated annealing and greedy seem to be similar. They both do not find a high number of cuts when applied after each other, but they do find a relatively high number of cuts on the tabu searches. In fact, simulated annealing finds the most cuts on both tabu3 and advanced tabu.

6.5 ALIB and BLIB

In table 13 and 14 we present the results of most heuristics for the ALIB and BLIB instances. Interestingly, alternative greedy performs worse than greedy for both libraries. The rest of the results are consistent with the results of the LITLIB instances. Tabu3 and advanced tabu are the best performing methods and tabu1 is slightly worse. Simulated annealing is almost as good as the aforementioned tabu heuristics, and better than the greedy methods. Finally the constructive heuristic is easily outperformed by all other methods. The results in Augerat et al. (1998) have greater gaps, however this is probably due to the use of upper bounds which are not optimal. The relative performance of our implementations are similar to the relative performances in Augerat et al. (1998).

Table 13: Summary results for the ALIB instances

Method	Avg. gap	Max gap	Best	Cuts	Iterations	Sep. time	CPLEX time
Constructive	3.54	5.68	0	486.27	47.96	0.25	0.55
Greedy	2.41	4.61	9	584.15	31.58	6.67	0.47
Alt. greedy	2.44	4.69	6	644.73	34.62	8.23	0.50
Tabu1	2.32	4.35	17	590.81	34.92	4.14	0.41
Tabu2	2.44	4.51	8	626.88	31.65	0.84	0.38
Tabu3	2.32	4.34	23	588.35	36.46	6.12	0.44
Advanced tabu	2.32	4.35	21	690.35	34.15	12.17	0.59
Simulated annealing	2.36	4.37	10	572.27	34.15	9.30	0.45

Table 14: Summary results for the BLIB instances

Method	Avg. gap	Max gap	Best	Cuts	Iterations	Sep. time	CPLEX time
Constructive	1.24	4.72	1	369.87	49.78	0.17	0.57
Greedy	0.73	2.51	12	546.43	37.09	3.35	0.46
Alternative greedy	0.74	2.67	12	631.91	41.61	6.02	0.88
Tabu1	0.72	2.50	17	535.61	39.83	2.27	0.49
Tabu2	0.75	2.71	10	565.96	38.13	0.55	0.48
Tabu3	0.71	2.50	22	561.56	38.83	2.55	0.48
Advanced tabu	0.71	2.50	21	607.52	38.22	3.97	0.60
Simulated annealing	0.72	2.59	12	506.09	38.35	4.61	0.53

6.6 Results emission zones

In table 15 the results of computing the lower bounds are shown when we do not employ the extra cuts. We use the same tabu search settings as tabu1, except $llimit = 2$ in the first phase and $llimit = 5$ in the second phase. The lower bounds are much worse than our test results in the classical CVRP. This is to be expected as the problem is more complex and the cuts are weaker.

Table 15: Results for emission zones with no extra cuts

Data set	Lower bound	Gap	Cuts	Iterations	Separating time	CPLEX time
Day1	1633.41	15.93	60	12	0.03	0.03
Day2	1564.68	14.24	59	11	0.05	0.02
Day3	1570.80	14.27	47	9	0.03	0.01
Day4	1735.28	24.85	60	12	0.05	0.02
Day5	1323.74	15.27	41	8	0.04	0.01
Day6	1623.73	18.27	55	11	0.03	0.02
Day7	1606.55	17.94	48	11	0.08	0.02

Now we present the summary results for adding the extra constraint families to the CVRP with emission zones in table 16. The total capacity cuts (TCC) and improved electric cuts (IEC) both decrease the gap by almost 3 percent points. Surprisingly adding both cuts (TCC + IEC) does nearly nothing, compared to only adding one of the cuts. We expected these constraint families to complement well, as IEC is useful for sets with only customers in emission zones, and TTC is useful for sets with a high number of normal vehicles. Finally, adding the integer restriction on the y-variables (on top of the two extra constraint families) decreases the gap by almost 7 percent point, and thus almost halves the gap. This confirms that integer valued y-variables are very important, as the solution now consists of separate clusters of normal and electrical routes. However, not surprisingly, the CPLEX time increases.

Table 16: Summary results for emission zones

Method	Avg. gap	Max gap	Best	Cuts	Iterations	Sep. time	CPLEX time
No extra cuts	17.25	24.85	0	52.86	10.57	0.04	0.02
TCC	14.47	17.99	0	85	13.14	0.07	0.03
IEC	14.59	17.56	0	58	12	0.04	0.03
TTC + IEC	14.10	18.24	0	80	11.86	0.01	0.01
TCC + IEC + int y	7.71	11.11	7	141.29	24.57	0.03	3.36

7 Conclusion

In this report we investigated a couple of heuristics from Augerat et al. (1998) to identify violated capacity constraints in the CVRP. Then, we introduced our own methods and compared their performances to the methods of Augerat et al. (1998).

We investigated the effect of using the exact solution of the BPP in the capacity constraints, this does not lead to an improvement in the obtained lower bounds. Then, we investigated the effect of maximizing the constraint violation in the greedy an tabu heuristics, which also seems to have no effect.

The simulated annealing heuristic performs decently, outperforming the greedy and constructive heuristics in most cases. However, it is not able to beat the best tabu versions from Augerat et al. (1998). Finally, we introduced an advanced tabu algorithm, which performs as good as the best tabu heuristic form Augerat et al. (1998), but is not able to outperform it. We noted that our inability to improve the identification heuristics may have to do with the limitations of the cutting plane algorithm, rather than the quality of identification heuristics.

Finally we developed a cutting plane algorithm for the CVRP with emissions. We investigated the effect of using several cut families, the best average gap while relaxing all integer restrictions is 14.10, which is a large gap compared to the average 2.01 obtained by the tabu3 heuristic for the LITLIB instances. It would be interesting for further research to investigate better cut families for the CVRP with emission zones to improve the lower bounds.

References

P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a branch-and-cut code for the capacitated vehicle routing problem. *Technical Report 949-M, Université Joseph Fourier*, 1995.

P. Augerat, J.-M. Belenguer, E. Benavent, A. Corbéran, and D. Naddef. Separating capacity constraints in the cvrp using tabu search. *European Journal of Operational Research*, 106 (2-3):546–557, 1998.

P. Bouman, 2018. URL `obtained at 14 january 2019 from https://github.com/pcbouman-eur/JavaCplexExample/tree /master/src/basic`.

K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313, 2016.

J. Caceres-Cruz, P. Arias, D. Guimarans, D. Riera, and A. A. Juan. Rich vehicle routing problem: Survey. *ACM Computing Surveys (CSUR)*, 47(2):32, 2015.

N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1): 255–282, 1981.

G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1): 80–91, 1959.

J. V. De Carvalho. Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, 86:629–659, 1999.

M. de Feijter, H. de With, W. de Voogd, and B. Spee. Case 1 basiswerkcollege feb22009 groep 25. 2018.

R. Fukasawa, H. Longo, J. Lysgaard, M. P. 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.

F. Glover, M. Laguna, E. Taillard, and D. de Werra. *Tabu search*. Springer, 1993.

Ç. Koç and I. Karaoglan. The green vehicle routing problem: A heuristic based exact solution approach. *Applied Soft Computing*, 39:154–164, 2016.

G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33(5):1050–1073, 1985.

J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

S. Martello and P. Toth. Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics*, 28(1):59–70, 1990.

D. Pecin, A. Pessoa, M. Poggi, and E. Uchoa. Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100, 2017.

Appendix

Proof of redundance upper bounds on x_{ij}

Theorem 1: constraints (5) and (6) can be replaced by $x_{ij} \geq 0, \forall (i,j) \in E$ in the CVRP formulation.

Proof: First suppose that $x_{ij} > 2$ holds for some edge. Let i be a node other than the depot without loss of generality. By restrictions (3) we have $x(\delta(\{i\})) = 2$. Now, as $x_{ij} > 2$, it holds that $x(\delta(\{i\})) - x_{ij} < 0$. However, $x_{i'j'}$ is non-negative for all (i', j') in E, and $(i,j) \in \delta(\{i\})$, thus $x(\delta(\{i\})) - x_{ij} \geq 0$. This is a contradiction, so $x_{ij} \leq 2$ for all $(i,j) \in E$.

Now we have proved that equations (6) hold, but equations (5) still can be violated. Consider an edge $(i,j) \in \gamma(V_0)$. Now we investigate the capacity constraint for the set $\{i,j\}$, which is $x(\delta(\{i,j\})) \geq 2\lceil q(\{i,j\})/C \rceil \geq 2$. This last inequality follows from the fact that the demand in i and j is nonzero. The left hand side can be rewritten as: $x(\delta(\{i\})) + x(\delta(\{j\})) - 2x((\{i\} : \{j\})) = 2 + 2 - 2x_{ij} = 2(2 - x_{ij})$. This yields the following equation: $2(2 - x_{ij}) \geq 2$, which is equivalent to $x_{ij} \leq 1$. □

Example non-determinism shrinking procedure

Consider the following example CVRP: there are 6 customers. Let the capacities of the trucks be 100. The demands are $q_1 = 90, q_2 = 20, q_3 = 10, q_4 = 10, q_5 = 20, q_6 = 20$, such that the problem has a feasible solution with $k = 2$ trucks. Now consider a certain solution of the relaxed LP as in figure 1.

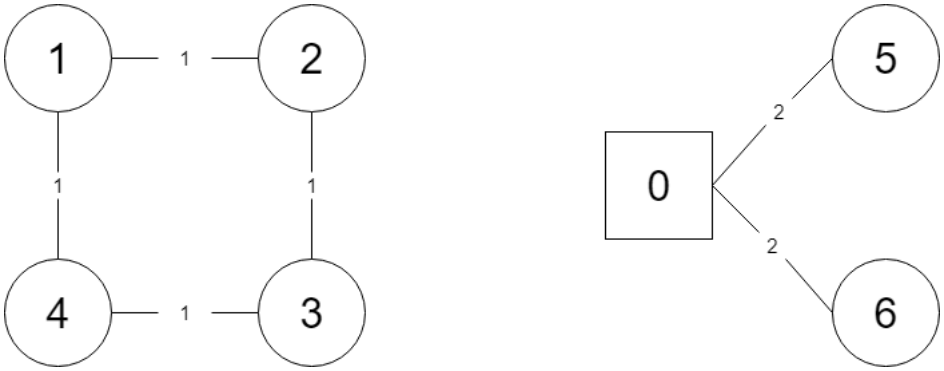

Figure 1: Example LP solution

We consider two possible steps in the shrinking procedure. In the first possibility we shrink both nodes 1 and 2 and 3 and 4 to one node, as can be seen in figure 2. As $q_1 + q_2 = 110 > C$, the supernode $\{1,2\}$ violates the capacity constraint, and consequently is identified in the shrinking procedure. However, none of the resulting supernodes in the other shrinking procedure (see figure 3) yields a violated constraint.

Therefore, we showed that applying the shrinking procedures to different edges may result in other identified constraints. This will possibly result in different LP solutions, and thus a different lower bound.

Figure 2: First possible shrinking move

Figure 3: Second possible shrinking move

Code summary

Code CVRP cutting plane algorithm

AdvancedConstraintIdentifier: an abstract class which is extended by all the different identifier heuristics.
GreedyRandomized: the class for the greedy randomized heuristic.
GreedyRandomizedAltScore: the class for the alternative greedy randomized algorithm.
Tabu, TabuAdvanced, TabuAlternative: the classes for the different tabu heuristics.
Constructive: the class for the constructive heuristic.
SimulatedAnnealing: the class for the simulated annealing heuristic.
BinPackingSolver: the class that solves the bin packing problem.
ComparingData: the class that contains the summary data for each comparison of two heuristics.
Cut: the class which contains data of a certain identified violated capacity constraint.
CVRPData: the class with general data of the CVRP instance.
Edge, EdgeData, Node, Graph: the classes used for the graph.
LBCalculator: the class used to calculate a lower bound for a certain CVRP instance for a certain identification heuristic.
LPModel: the linear program model for a certain CVRP instance.
LPSolution: the class which contains data for a certain solution of the linear program.
Main: the main class which calculates everything.
MainGreedyTest: the main class for the test with different selecting probabilities for the greedy randomized heuristic.

SolvingData: the class which contains summary data about the solving procedure for a certain heuristic for a certain library.

Code for the CVRP with electrical vehicles

AdvancedConstraintIdentifier: an abstract class which is extended by all the different identifier heuristics.
Cut: the class which contains data of a certain identified violated capacity constraint.
CutData: the abstract class for relevant data for the left hand side or right hand side of the capacity/range equations.
CutDataDemand: the CutData class of constraints for which there needs to be kept track of the demand of a customer set.
CutDataDemandElectric: the CutData class for the improved electricity constraitns.
CutFamily: the abstract class which computes the values of left and right hand sides of the constraints.
CVRPData: the class which contains data for a certain instance of the CVRP with electric vehicles.
Edge, EdgeData, Node, Graph: the classes used for the graph.
ElectricCapacityCutFamily, ImprovedElectricCapacityCutFamily, NormalCapacityCutFamily, RangeCutFamily, TotalCapacityCutFamily: the CutFamily classes for all the different constraint sets.
LBCalculator: the class used to calculate a lower bound for a certain CVRP instance for a certain identification heuristic.
LPModel: the linear program model for a certain CVRP instance.
LPSolution: the class which contains data for a certain solution of the linear program.
Main: the main class which calculates everything.
SolvingData: the class which contains summary data about the solving procedure for a certain heuristic for a certain library.