Solving a real-life roll-on-roll-off waste collection problem with column generation

Solving a real-life roll-on–roll-off waste collection problem with column generation

Joël Raucq[1], Kenneth Sörensen[2], and Dirk Cattrysse[3]

[1]Université catholique de Louvain, CORE, B-1348 Louvain-la-Neuve, Belgium, joel@raucq.be
[2]University of Antwerp Operations Research Group – ANT/OR, Prinsstraat 13, 2000 Antwerp, Belgium, kenneth.sorensen@uantwerpen.be
[3]Katholieke Universiteit Leuven, Centre for Industrial Management, Celestijnenlaan 300a, 3001 Heverlee, Belgium, dirk.cattrysse@cib.kuleuven.be

(December 2018)

Abstract

In this paper, we develop an intelligent solution to a complex, real-life vehicle routing problem in the waste management sector. Waste management companies deliver various types of empty waste containers to industrial customers, to be filled with different types of waste, after which the containers are picked up again. The full containers are transported to a so-called *waste management depot* or *waste handling depot*, where they are emptied, after which they are reused. Empty containers are stocked in various *stock depots*. Time windows in which containers may be picked up and delivered at the customers and opening hours of the different depots have to be taken into account additionally. Very importantly, there are two types of waste collection trucks, that can respectively carry one or two containers.

The resulting vehicle routing problem belongs to a class of so-called *roll-on–roll-off* problems, characterized by unit demand. Additionally, the problem discussed in this paper has several characteristics (multiple waste types, multiple container types, multiple depots, a heterogeneous fleet, pick-up and delivery, time window constraints, and other) that make solving it a difficult task.

To solve this problem, we develop a novel column generation scheme that incorporates a heuristic approach to generate new columns. The master problem is solved by linear relaxation of a set partitioning problem. New routes (columns) are generated by a constructive heuristic loosely based on Solomon's insertion heuristic for the vehicle routing problem with time windows. The construction heurisic operates on a reformulated version of the roll-on–roll-off problem, that is a generalized vehicle routing problem with time windows. The algorithm — called WMPopt — is

submitted to an extensive sensitivity analysis to determine its response to different parameter settings. After this, we test it on four real-life problem instances and compare its results to those obtained by a commercial solver. We show that WMPopt achieves much better solutions than the commercial solver in similar computing times.

Key words: Roll-on–roll-off, vehicle routing, waste management, column generation, constructive heuristic

1 Introduction

For the disposal of industrial waste, companies often have a contract with a waste management company that picks up the waste in standardized containers. To this end, waste containers of different types are delivered by the waste management company and picked up when they are full. Transport is usually done using specialized trucks, that have specific handling equipment to pick up and drop off the containers. The transport of the empty and full containers between the customers' sites and the depots gives rise to a so-called *roll-on–roll-off* vehicle routing problem with some specific characteristics.

In this paper, we describe a specific variant of this problem faced by several waste management companies. These companies have two types of trucks that are able to transport the containers, with a capacity of one or two containers respectively. The motivation for using two-container trucks is that they are much cheaper to operate than two one-container trucks, even though some flexibility is lost. Furthermore, only a limited number of each truck type is usually available. Empty containers are stored at so-called *stock depots*, whereas full containers are emptied in *waste management depots*. Some waste management depots may also serve as stock depots (i.e., they may store empty containers), but this is not required.

The waste management companies receives four types of service requests:

- *Exchange* orders (EXC): An empty container is picked up from a stock depot and brought to the customer's site. The container can also be picked up from a waste handling depot that maintains a stock of empty containers, or from another customer, in which case it is first emptied at a waste depot. A full container is picked up at the customer immediately after the empty container has been delivered. The full container is brought to a waste handling depot to be emptied.

- *Round trip* orders (ROU): A full container is picked up at the customer first, emptied at a waste handling depot, after which the same container is brought back to the customer. The truck makes the return trip to the waste handling depot without intermediate stops.
- *New customers* (NEW): An empty container is picked up from a stock depot and dropped at the customer's site. No container is picked up.
- *End-of-Contract customers* (END): A full container is picked up at the customer's site and brought to a waste management depot. No empty container is brought back.

The waste management companies handle different types of waste, and each waste type can be deposited only in a subset of the available waste management depots. Stock depots usually store every type of container. For each container type, there is also a list of waste types that it may or may not contain, although situations arise in which each container can hold every type of waste.

Companies may call for any of the service requests, indicating the day they want to receive the service. The planning is done on a daily basis, based on service requests made up to the previous day. Each customer may also specify a time window for their service request, i.e., an earliest and latest time at which the truck should arrive. The truck type (one or two containers) with which a service request is carried out, is irrelevant for all service requests (except for the special case in which a customer demands two containers at the same time, see further). Also, each truck can carry each container type and the order in which containers are loaded or unloaded is unimportant.

The objective of this problem is to minimize the total active time. This includes driving time between customers and depots, service and waiting time at the customers and handling time at the depots. All of these times are assumed to be known and deterministic and may depend on the truck type. The vehicle routing problem described here is a pick-up and delivery vehicle routing problem with time windows and unit demands, in which each truck has a capacity of either one or two. Such problems are known in the literature as *roll-on–roll-off* problems.

The decisions that have to be made in our problem are the following:

- The truck type (one or two containers) with which to execute each route.
- The order in which to visit the customers on a given route.

- The position in the route to visit stock depots and waste management depots.
- The actions to perform at stock and waste management depots (how many containers to pick up or drop off).

The difference between our problem and most other pick-up-and-delivery vehicle routing problems is the fact that the depot to pick up or drop off a container is not given. In other words, either the pick-up or the drop-off location becomes a decision variable. For example, a container picked up from a customer can be delivered to any waste depot. Which waste depot to visit is a decision that is an integral part of this roll-on–roll-off problem.

An example of a solution to a waste collection vehicle routing problem, that shows the complexity of the problem, is shown in figure 1. Notice that we assume for reasons of simplicity that there is only one type of container and one type of waste in this example. This solution consists of two routes, one performed by a two-container truck (dashed line) and the other by a one-container truck (solid line). The time windows are not shown. Note that all trucks start and end from the same location in this figure, something which is not required. We also assume that the truck needs to return to the depot empty. The numbers on the arcs between trucks stops show the order in which the stops are executed.

2 Literature review

Several types of waste collection vehicle routing problems can be distinguished. *Urban garbage collection problems*, are usually modeled as a *capacitated arc routing problem* (Golden and Wong, 1981; Hertz et al., 2002). The objective of such problems is to determine the order in which to drive through a set of streets (i.e., visit the arcs of a graph, rather than the nodes), possibly in both directions, minimizing the number of vehicles used and the total distance traveled. Each street has a certain amount of waste to be picked up and the capacity of the vehicles should not be exceeded. Ghiani et al. (2005) use a two-stage cluster-first route-second method to solve a waste collection problem in Italy. An ant-colony based algorithm for an urban waste collection problem can be found in Bautista et al. (2007).

Whereas urban waste collection problems are best modeled as arc routing problems, this is not the case for industrial waste collection problems. In these problems, waste is picked up at specific locations and not in streets and hence these problems are best tackled as

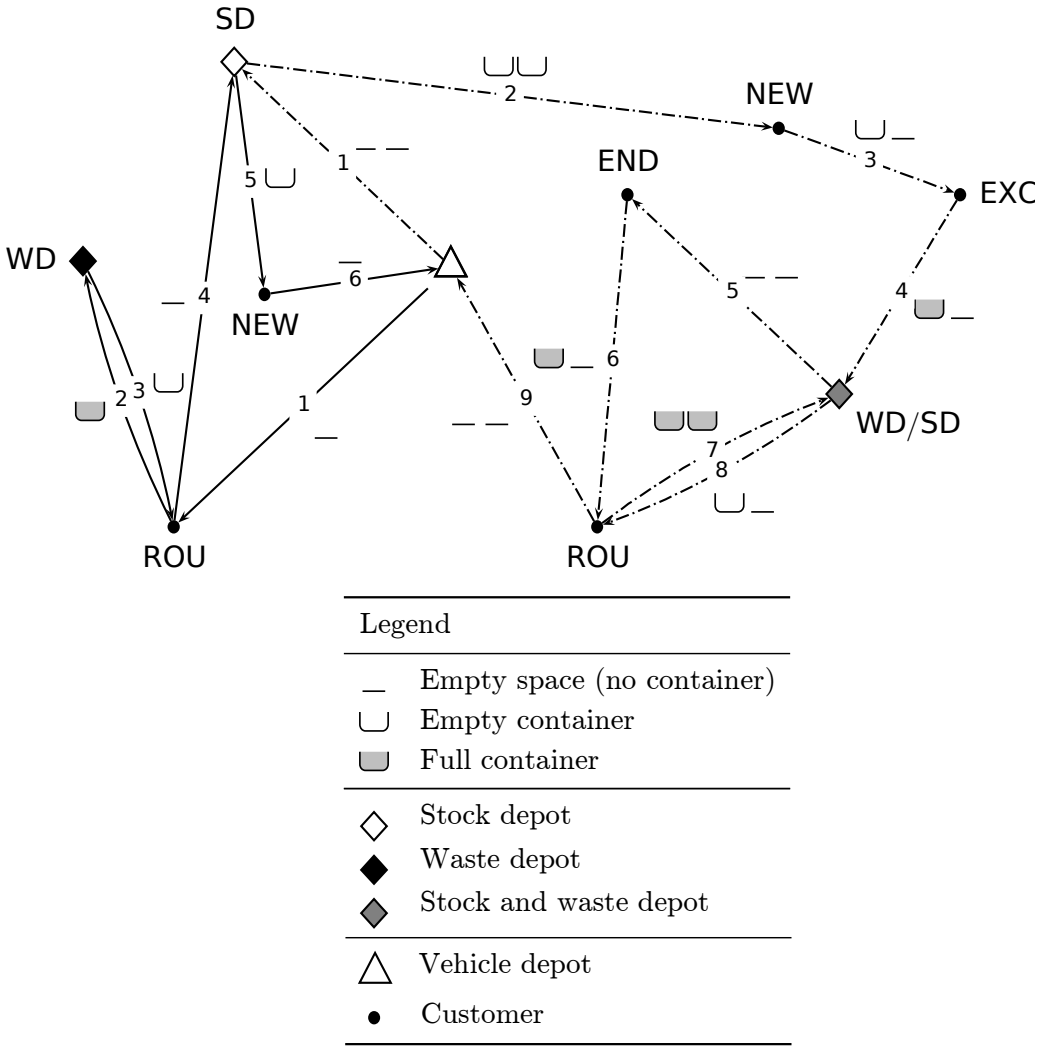

Figure 1: An example of a solution with two trucks of different types

node routing problems. Waste collection vehicle routing problems belong to the general class of *pick-up and delivery* vehicle routing problems (Parragh et al., 2008; Berbeglia et al., 2007). In several papers, a periodic vehicle routing model is used, in which a schedule for several days is created. One of the earliest uses of the PVRP for waste collection problems is due to Beltrami and Bodin (1974). In some papers, the specific waste collection variant is made difficult by the necessity to schedule intermediate waste disposal facilities, where the waste can be dropped off once the vehicle is full. This problem, the *periodic vehicle routing problem with intermediate facilities* is surveyed in Guastaroba et al. (2016) and addressed by Angelelli and Speranza (2002a), who develop a tabu search procedure for it. This algorithm is applied in Angelelli and Speranza (2002b) to estimate and compare the operating costs of different waste-collection systems. A similar model can be found in Kim et al. (2006), who add time windows for both the stops and the waste disposal facilities to the formulation.

The problem discussed in this paper is an example of a so-called *roll-on–roll-off* vehicle routing problem (RRVRP) (Cristallo, 1994; Cattrysse et al., 1996; Golden et al., 2002; Wy and Kim, 2013; Derigs et al., 2013). Roll-on–roll-off vehicle routing problems are special cases of pick-up and delivery problems (Nanry and Wesley Barnes, 2000; Mitrović-Minić et al., 2004) but differ from them in that the product (waste, in this case) is not picked up as such, but in containers of unit capacity. These containers are delivered to waste management depots where they are emptied, after which they may be reused. In the models of De Meulemeester et al. (1997), Bodin et al. (2000), and Baldacci et al. (2006), the vehicles can only move one container at a time. De Meulemeester et al. (1997) present a problem with multiple disposal facilities and a single inventory/depot location. They present two heuristics and an exact algorithm. Bodin et al. (2000) present a problem with a single disposal facility and a single depot/inventory location. They develop a mathematical model for this problem, two lower bounds and four heuristic algorithms. An extension of this problem can be found in Baldacci et al. (2006), who introduce the *multiple disposal facilities and multiple inventory locations roll-on–roll-off vehicle routing problem*. They formulate the problem as a time-constrained vehicle routing problem on a multigraph and develop an exact algorithm for it. This problem is very similar to our problem, with the exception that our problem has two different types of vehicles, which can take either one or two containers. The number of containers on a single vehicle is extended to two in le Blanc et al. (2006). Hauge et al. (2014), on the other hand, model a similar problem where vehicles can take up to eight containers, two of which can be full, and solve it using a hybrid column generation approach. Contrary to our method, the authors use a tabu search algorithm to generate columns. Their model

also differs from ours in the specificities of the actions performed on the containers. Another related problem is modelled in Wy et al. (2013): a roll-on roll-off problem with multiple disposal facilities, multiple container storage yards, different service types of customer demands, time windows for customer demands and facilities, various types and sizes of containers, and the lunch break of tractor drivers. They solve this problem with a large neighborhood search based algorithm. Aringhieri et al. (2017) consider a real-life case study of a waste collection RRVRP in which two objectives are considered, the number of vehicles as well as the total route duration. Finally, Elbek and Wøhlk (2016) develop a multiperiod model of the RRVRP.

Recently, authors have started to look at waste collection routing problems (Ramos et al., 2018), where sensors in the various bins transmit the fill level. Such "smart" installations may allow a waste collection company to pro-actively determine which sites to visit before containers fill up completely and potentially offers more opportunities for further optimization.

3 Problem formulation and solution overview

3.1 Mathematical formulation

In this roll-on–roll-off problem we are given a set of n service requests, and a set of m depots. We will from now on assume without loss of generality that each customer places exactly one service request and, consistent with the routing literature, use the term "customer" to indicate a service request in a specific location. Customers and depots therefore jointly form $n + m$ nodes on a graph, from now on referred to as *locations*. The customers require one of four possible operations (end-of-contract, new contract, exchange or round-trip). The service time at customer or depot i is denoted s_i. At the depot, the service time may depend on the depot and the operation performed there. The start of service at customer i, denoted b_i, must be between an earliest start e_i and a latest start l_i. The interval $[e_i, l_i]$ is called the *time window* of customer i. Although the start of service at depot i is also referred to as b_i, there is no time window for the depot. The travel time between location i and location j is denoted t_{ij}. If a truck arrives early (before e_i) at customer i, it will wait until e_i.

In this paper, we reformulate our waste collection problem as a generalized vehicle routing problem with time windows (GVRPTW). In the generalized version of routing problems a set of node *clusters* are given, which are mutually exclusive, i.e., no node appears

in more than one cluster. The aim is to find the best solution that visits exactly one node in each cluster. Ghiani and Improta (2000) solve the GVRP by reformulating it as a capacitated arc routing problem. In this paper, we use a route generation heuristic loosely based on the famous I1 insertion heuristic due to Solomon (1987).

In the real-life roll-on–roll-off problem, routes usually consist of a relatively small number of customers, mostly not more than ten. This motivates the use of a column generation solution method, that decomposes the waste collection problem into a master problem and a subproblem.

The master problem is a set covering problem given by the following formulation:

$$\min \sum_{j=1}^{q} c_j x_j \tag{1}$$

$$\text{s.t.} \sum_{j=1}^{q} a_{ij} x_j \geq 1 \qquad \forall i \in \{1, \ldots, n\} \tag{2}$$

$$\sum_{j=1}^{q} b_{kj} x_j \leq V_k \quad \forall k \in \{1, 2\} \tag{3}$$

$$x_j \in \{0, 1\} \qquad \forall j \in \{1, \ldots, q\} \tag{4}$$

In this formulation, each column j of the matrix $\{a_{ij}\}$ represents a route. The number of routes is equal to q and c_j is the cost of route j, equal to the total time spent in this route, and a_{ij} is equal to 1 if customer i is included in route j and 0 otherwise. The decision variables x_j are equal to 1 if route j is selected and 0 otherwise. The objective function (1) represents the total cost of all selected routes. Constraints (2) ensure that each customer is included in a route at least once. V_k is the number of trucks of type k available, b_{kj} is 1 if route j is executed with a truck of type k. Constraints (3) ensure that all routes can be executed with an available truck. Finally, constraints (4) ensure that the decision variables are binary. To obtain the final feasible solution at the end of the algorithm, equation 2 is replaced by $\sum_j a_{ij} x_j = 1 \quad \forall i$, which transforms the master problem into a set *partitioning* problem and enforces each customer to be visited in a single route. It is important to note that this step is necessary to ensure that the final solution is feasible. If a customer is planned twice, it does not suffice to just delete the most expensive copy of this customer from the route (unlike in the standard VRPTW). The reason is that there may be intermediate operations planned before or after this customer that make the route infeasible once this customer is removed (e.g., a container

is picked up there, and dropped at another customer).

3.2 Column generation method overview

Even though recent alternative aggregated/disaggregated formulations (Bruck and Iori, 2017; Bianchessi and Irnich, 2019) have also been successful, LP-based column generation approaches remains an important approach to solve various types of problems in production planning and scheduling (Cattrysse et al., 1993, 1990), generalized assignment (Savelsbergh, 1997; Cattrysse et al., 1994), cutting stock (Cintra et al., 2007), and also in routing (Desrochers et al., 1992; Bramel and Simchi-Levi, 1997; Fukasawa et al., 2006; Choi and Tcha, 2007). The decomposition results in a master problem that is a set covering or set partitioning problem. The subproblem contains all the characteristics of the problem and is solved using dynamic programming, metaheuristics a.o. For routing problems the LP-bound generated with Dantzig-Wolfe decomposition is very tight and frequently an integer solution is obtained. The integer solution obtained by solving the master problem is not necessarily an optimal one but the LP-bound of the master problem provides a performance measure for the heuristic.

At each iteration of the column generation heuristic, the dual variables obtained by solving the linear relaxation of the restricted master problem are used to determine which customers will be used to form a new route. Forming a new route is done by the subproblem pricing heuristic, that schedules as many customers as possible with an insertion heuristic. In section 5 we explain in detail how the linear relaxation of the restricted master problem is solved and how customer lists are generated to be supplied to the subproblem heuristic. First, we discuss how this heuristic finds good routes to add to the master problem. Algorithm 1 describes the iterative procedure.

At each iteration, information from the optimal solution of the linear relaxation of the restricted master problem is used to generate a candidate list of service requests or customers. Of this list, as many customers as possible should be scheduled in an efficient route, including the picking up and dropping off of containers at stock and waste management depots respectively. This is done by applying a heuristic route construction procedure loosely based on the I1 insertion heuristic of Solomon (1987).

Note that the resulting route should have two (contradicting) characteristics: (1) it should contain as many customers as possible and (2) it should be as short as possible, i.e., the total time required to execute all operations in the route (including the waiting time at a customer when a vehicle arrives early) should be minimal. A balance between

Algorithm 1: Column generation for the roll-on–roll-off vehicle routing problem

```
Initialize: generate "virtual" routes, one for each customer;
Generate a mono-customer route per vehicle for each customer (Phase I);
repeat
    (Phases II+III);
    Solve the linear relaxation of the restricted master problem;
    Determine the candidate list of customers to plan;
    Solve the subproblem, generating routes using customers in this candidate
     list;
    Insert the corresponding columns in the restricted master problem;
until stopping condition;
Solve the restricted master problem to (integer) optimality;
Optimize the individual routes (Phase IV);
```

these characteristics is found by applying a constructive heuristic that attempts to add as many customers as possible, i.e., until no more customers can be feasibly added or until no more customers are left to schedule. The constructive heuristic attempts to schedule each customer in such a way that the resulting route will be of minimal length.

Efficiently scheduling the intermediate depots adds a new layer of complexity to the problem. We therefore develop a route construction procedure that operates on a reformulated version of the roll-on–roll-off problem, effectively removing the depots from the formulation. The resulting problem is a generalized vehicle routing problem with time windows, and the transformation is explained below.

3.3 Modeling as a generalized vehicle routing problem with time windows

Modeling the roll-on–roll-off problem as a *generalized vehicle routing problem with time windows* (GVRPTW) is done in three steps:

1. Determining the node clusters for each customer.
2. Adding the inter-cluster arcs.
3. Determining the inter-cluster travel times.

3.3.1 Determining node clusters for each customer

In the new formulation, each customer is modeled as a *node cluster*, that contains different nodes. The nodes are used to model the fact that different trucks (one or two container capacity) may arrive at a customer in different *states*, i.e., empty, or carrying empty or full containers of different types and containing different waste types. In our new formulation, the costs and travel times of the edges between nodes in different clusters now include the costs and times of intermediate operations (dropping off or picking up empty or full containers at stock or waste depots), effectively removing the depots from the formulation.

For each customer in the original formulation, we define a node cluster in the new formulation. The types of nodes included in a node cluster depend on the order type. Figure 2 shows the different node clusters for the different order types for a problem with one type of waste and one type of container. A node in a node cluster should be interpreted as the combination of (1) a customer and (2) the state the truck that services this customer is in upon *arrival*.

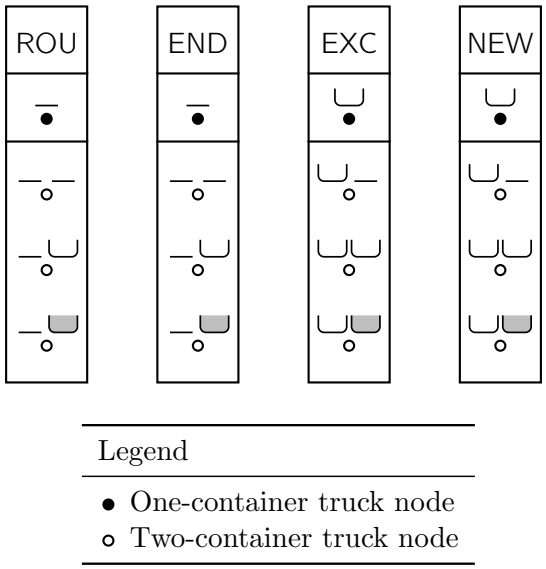

Figure 2: Node clusters in the GVRPTW formulation

Trucks arriving at an end-of-contract customer or a round trip customer, need to have at least one empty position. Therefore, all vehicle states that have at least one empty position are allowed. Trucks arriving at a new customer or an exchange customer need to carry an empty container. Allowable states are therefore those that have at least one

empty container. The node clusters for round-trip and end-of-contract customers are therefore the same, as are the node clusters for new customers and exchange customers.

Some customers demand two containers to be picked up and/or delivered at the same time. For these customers, only one configuration is possible, and hence their node clusters contain only one node, such as can be seen in figure 3.

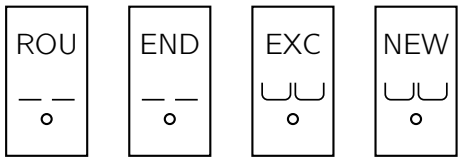

Figure 3: Node clusters for customers that require pickup and/or delivery of two containers at the same moment

When the problem involves multiple waste and/or container types, the number of nodes in a node cluster is increased. Figure 4 shows examples of node clusters when two types of waste and two types of containers are present. The exchange customer on the right demands a container of type 2. Note that forbidden combinations of waste type and container type can be taken into account by simply not adding a node in the node clusters. In the example in figure 4, containers of type 2 are not allowed to contain waste of type 2.

3.3.2 Adding the inter-cluster arcs

When the nodes of the GVRPTW formulation graph have been determined, the edges are added to this graph. In principle, an edge is added for each pair of nodes in different node clusters, although some edges represent an impossible operation, as we will see, and therefore do not need to be added. For each edge, the cheapest possible route between the two nodes it connects, including the visit to a waste or stock depot, can be calculated. Because a node inside of a cluster unambiguously defines the vehicle state at the visit to this customer, the actions that need to be undertaken to arrive at the customer in this state, are easy to define.

To illustrate this, we give in figure 5 an example of the different edges that are drawn between a new customer and an end-of-contract customer (again assuming one type of waste and one type of container). In table 1, these edges are explained, showing the intermediate operations that are required to traverse this edge in the graph. For

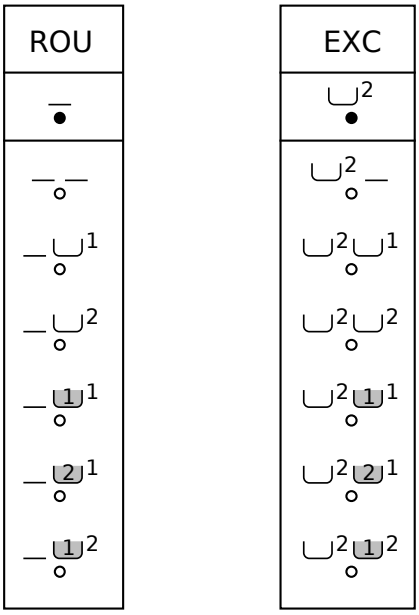

Figure 4: Example node clusters for two types of waste and two types of containers

simplicity, we assume again that only one type of container and one type of waste are present.

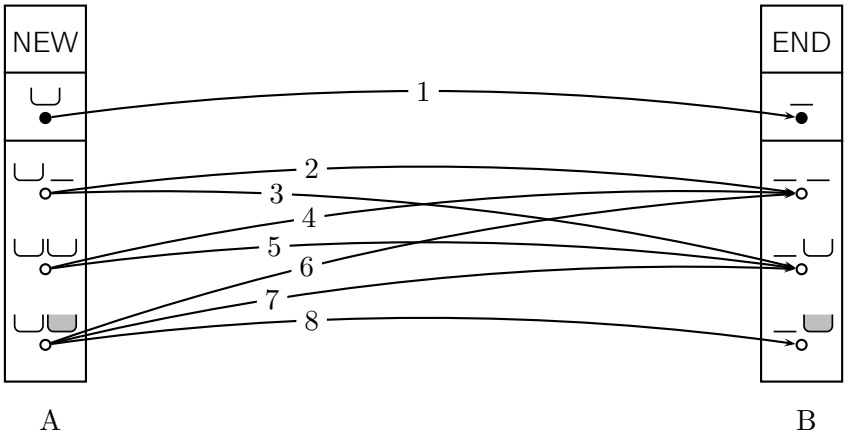

Figure 5: Edges between a new customer A and an end-of-contract customer B

305 It is clear that a one-container truck cannot transform into a two-container truck. Therefore, there are no edges between nodes corresponding to different vehicle types. In practice, this means that the graphs corresponding to each vehicle type are completely disjoint, since they do not share any edges or any vertices. The subproblem heuristic

Table 1: Edges between a new customer A and an end-of-contract customer B, including intermediate operations

#	Vehicle state At A	After A	At B	Intermediate operations
1	⊔	_	_	-
2	_ ⊔	_ _	_ _	-
3	_ ⊔	_ _	_ ⊔	Pick up empty container
4	⊔ ⊔	_ ⊔	_ _	Drop off empty container
5	⊔ ⊔	_ ⊔	_ ⊔	-
6	⊔ ▭	_ ▭	_ _	Dispose waste, drop off empty container
7	⊔ ▭	_ ▭	_ ⊔	Dispose waste
8	⊔ ▭	_ ▭	_ ▭	-

therefore always attempts to find routes for both vehicle types separately (see section 4).

Note that even between the nodes representing a two-container truck, two potential edges are not included: those that would require the truck to pick up a full container between two customers. Clearly, full containers are only picked up at customer locations, and including one of these edges would therefore imply that a customer was visited between A and B. Similarly, edges that are impossible due to time constraints are not added. To perform this check, we calculate the duration of the travel from A to B, taking into account the intermediate actions that need to be performed to arrive and depart in the correct configuration, and check whether this travel from A to B is feasible while respecting both customer's time windows.

3.3.3 Determining edge costs

The cost of an edge between a node in node cluster A and a node in node cluster B should be equal to the minimum cost of driving from customer A to customer B, including the time required for performing the intermediate operation(s). For example: to traverse edge 6 in figure 5 (i.e., to start from the new customer carrying a full container and arrive at the end-of-contract customer with two empty positions), the two-container truck needs to drop the full container at a waste management depot. This should be done in the waste depot for which the cost of travelling is minimized. Moreover, the cost of the edge in the new graph should be equal to sum of travel costs between customer

A and the waste depot and between the waste depot and customer B and may include the service time at the waste depot.

Some of the edges in the GVRPTW graph may represent rather complex operations. For example, an edge from the "no container + full container" node of a round-trip customer C to the "empty container + empty container" node of an exchange customer D will represent the following operations: traveling from C to the waste management depot and unloading the container, traveling back to C to drop the empty container, traveling to a storage depot to pick up two empty containers and finally, driving from the storage depot to customer D. In each of the possible cases, however, it is easy to determine the cheapest possible way in which the complex sequence of operations can be accomplished.

In our method, we calculate the minimum cost of an edge by an exhaustive enumeration of all feasible intermediate combinations. Feasibility of an edge cost requires that all necessary operations be performed, potentially taking into account precedence constraints. For example, a one-container truck that picks up a full container at customer A and needs to arrive with an empty container at customer B needs to go to a waste management depot *first* to drop off the full container and can only then go to a stock depot to pick up an empty container (except if B requires the same container type as A, in which case the truck does not need to go to a stock depot, but can use the emptied container from A). In all cases, the number of combinations of intermediate operations is small, as there are only a limited number of stock and waste management depots.

One of the most important advantages of the column generation approach is that we only need to generate edge costs if they are required. In other words, edge costs are only calculated when the algorithm generates a route, and only between customers in that route. The reason this is important is that the new formulation is defined on a graph that has a number of vertices that is a multiple of the number of customers in the original formulation (because each customer corresponds in the new graph to a customer cluster that consists of a vertex per truck configuration in which the truck can arrive at this customer). When a large number of container and waste type combinations are possible, the number of vertices can grow quite large (e.g., 70 nodes per customer). Of course, the increase in number of edges is a quadratic function of this. Generating the entire graph, including all edge costs, is therefore intractable for all but the smallest of problems.

In the GVRPTW formulation, each node cluster has a time window, which is equal

to the time window of the customer it corresponds to. Alternatively, we may say that each node in a node cluster has a time window equal to the customer this node cluster corresponds to.

4 Subproblem route construction heuristic

Because of the expected size of the GVRPTW graph, we opt for a simple constructive heuristic, that is a modified version of the I1 insertion heuristic by Solomon for the VRPTW. Our procedure builds a route by inserting one customer at a time into the route that is currently being built. It chooses these customers from a candidate list that is created based on information from the optimal solution to the linear relaxation of the restricted master problem (see section 5). Contrary to the original I1 heuristic (that starts another route when no more customers can be inserted), our procedure stops when no more customers can be inserted into the route. Another difference is that our procedure uses only travel time, instead of an (arbitrarily weighted) combination of travel time and travel distance.

Given the fact that the number of customers is small and not all customers need to be added to the route, our procedure attempts to use each node as a seed. After this, the method iteratively adds one customer at a time until no more customers can be added. Finally, the best solution found using this procedure is returned to be added as a column to the restricted master problem.

Customers are added one by one to the route as follows. First, the best possible insertion position is calculated for each unplanned customer. Given a partial route $(i_0, i_1, i_2, \ldots, i_m)$, the best possible feasible insertion position for a customer u is determined as

$$\kappa^*(u) = \min_p [\kappa(i_{p-1}, u, i_p)], \quad p = 1, \ldots, m \tag{5}$$

where i_{p-1} and i_p do not belong to the same node cluster, and

$$\kappa(i, u, j) = \alpha\kappa_1(i, u, j) + (1 - \alpha)\kappa_2(i, u, j), \quad \alpha \in [0, 1] \tag{6}$$

$$\kappa_1(i, u, j) = t_{iu} + t_{uj} - \mu t_{ij}, \quad \mu \geq 0; \tag{7}$$

$$\kappa_2(i, u, j) = b_j^u - b_j \tag{8}$$

where b_j^u is the new time for service to start at customer j, given that customer u is inserted between customers i and j. When the best possible insertion location for each customer has been determined, the customer u^* to insert is established:

$$u^* = \arg\max[\lambda t_{0u} - \kappa^*(u)], \quad u \text{ unrouted and feasible.}$$

The procedure has three parameters: μ and α and λ.

Algorithm 2: Insertion heuristic (based on Solomon I1)

```
Input: A candidate list of customers C, a seed node
Output: One route for each combination of vehicle type and starting depot,
        each containing a subset of the customers in C
for each vehicle type/starting depot combination do
    for each customer i do
        Start a new route using seed customer i ;
        for each customer s ∈ C do
            Add s to the route;
            repeat
                Calculate cheapest insertion location for each unrouted customer;
                Calculate cheapest customer u* to insert;
                Insert u* in its cheapest insertion location;
            until no more customers can be inserted;
    Determine the cheapest route for the combination vehicle type/starting
     depot;
```

5 Solution of the linear relaxation of the restricted master problem and generation of new columns

At each iteration of the main loop, i.e., after one or more new columns have been added, the linear relaxation of the restricted master problem is solved to optimality using the dual simplex method implemented in the GLPK callable library. Dual prices are then calculated for all customers and added to the travel time from this customer to the seed customer, i.e., the first customer added to the route (see further). The obtained value is an approximation of the reduced cost generally used in column generation methods and is used to create a candidate list of "interesting" customers that are then reported to the subproblem. The subproblem heuristically attempts to create an efficient route incorporating as many customers as possible. The costs of these routes, the vehicle type

used, and the list of customers they contain, are reported back to the master problem and inserted as new columns. In this section, we explain this column generation scheme in detail.

5.1 Initialization of the master problem

An initial (infeasible) solution is created by adding a *virtual* route for each customer. These routes become the first n columns of the restricted master problem (n is the number of customers). The columns corresponding to these virtual routes contain a single 1 in the row of the corresponding customer, and are given an arbitrarily high cost. They do not correspond to any vehicle type, so in these columns $b_{kj} = 0 \quad \forall j \in \{1, \ldots, n\}, k \in \{1, 2\}$. Because of this, the fleet size constraints do not apply to these columns, ensuring that the linear programming relaxation of the restricted master problem (which we will refer to as LP) has at least one feasible solution, albeit one that cannot be implemented because no vehicle types are assigned to the routes.

The reason for adding these virtual routes is the following. When virtual routes appear in a solution of the non-relaxed problem, this means that the customers corresponding to these routes are yet unplanned (if the customer would appear in another route, the virtual column could not have been included in the optimal solution). This knowledge is used to determine later on which customers to select for route building. Since the first n columns are virtual, the value of $\sigma_x = \sum_{j=1}^{n} x_j$ is an indication of the number of customers that are still unplanned. Note that this value may be fractional, since the x_j-values result from a linear relaxation of the master problem. This value is used to determine later whether we should explicitly attempt to plan unplanned customers.

5.2 Generation of columns

In a first phase, we generate a second set of initializing columns, one per customer–vehicle type combination. The generation of "real" columns happens in two phases (II and III), that are executed sequentially and iterated a limited number of times. After each column is added, the linear relaxation of the restricted master problem is re-optimized, starting from the previous solution to save time.

Phase I generation of mono-vehicle columns In phase I, a column is generated for each customer–vehicle type combination, i.e., we generate routes with one service, for each type of vehicle. Usually no implementable solution can be found after this phase because of vehicle restrictions. Note that — because of the virtual columns — the master problem always has a solution that is technically feasible. Therefore we use the term "implementable" to mean a solution that does not retain any virtual columns.

Phase II generation of multi-vehicle columns In phase II, we attempt to generate candidate lists of customers that can be put together in a route. These lists are reported to the subproblem, that attempts to put as many of them in a route as possible. Generating these candidate lists is done by first selecting a *seed* customer and then adding customers that appear well-suited for inclusion in the same route as the seed customer. When many customers are unplanned in the current solution, we attempt to plan them explicitly. The value of $\sigma_x = \sum_{j=1}^{n} x_j$ is used as proxy of the actual number of unplanned customers.

If $\sigma_x > \sigma_{\max}$ ("high") many customers are unplanned and we use an unplanned customer as the seed. If σ_x is "medium", the customer with the highest dual value is chosen as the seed customer. This is the case if $2 \leq \sigma_x \leq \sigma_{\max}$ during the first iteration of this phase, and if $0 < \sigma_x \leq \sigma_{\max}$ during all other iterations.

When the seed customer has been determined, a fixed number of customers are determined, based on their desirability to be in the same route as the seed customer. The desirability of a customers is calculated as the sum of two factors: (1) the dual price of the row corresponding to this customer, which gives a general indication of how "difficult" this customer is to plan; and (2) its distance to the seed customer. The resulting value is an approximation of the reduced cost generally used in column generation algorithms. However, if the vehicle is of type 1 and the order is an exchange order or an end order, we take the distance to the waste depot into account because we know that the truck will have to return immediately to the waste depot after this visit and cannot visit another customer first.

The number of customers to add to the seed customer varies throughout the algorithm in order to examine both long and short routes. At each iteration, we generate two large and two small routes, for a one-container and a two-container vehicle respectively. As the iterations progress, the length of the large routes generated is reduced and that of the short routes increased slowly. In general, we found that this strategy yields a good

mix of different column lengths. The length of the small, respectively the large route are given by $\lfloor j/P \rfloor + M$ and $\bar{M} - \lfloor j/P \rfloor$ where M is the minimum route length and $\bar{M}$ the maximum route length. j is the current iteration and P is a parameter that determines how fast the size of the large route shrinks and how fast the small route grows.

In this phase, multi-vehicle columns are generated until a certain maximum number of columns $c_{\max}$ is reached or until σ_x is "low" ($\sigma_x < 2$ during the first iteration of this phase, $\sigma_x = 0$ during all other iterations). The algorithm then moves to phase III.

Phase III: generation of merged columns In this phase, we attempt to merge routes in order to attempt to find better columns than the ones already generated. In this phase, if a route has a cost that is lower than a certain threshold, the route becomes a candidate for merger. Additionally, all unplanned orders are also candidates for merger into existing routes.

The algorithm then attempts to merge all candidate routes into all existing routes, provided that both routes are not run by a truck of capacity 2. The reason for this restriction is that we found that a successful merger of two such routes is extremely rare, as high-quality routes generally have quite a large number of customers in them, which makes them difficult to combine and stay within the time windows of the customers and the depots. The routes are merged by combining the two candidate lists of customers into one and using this list in the route construction heuristic. If a feasible merged route is found, it is added as a new column.

If not all customers are planned after phase 3, phases 2 and 3 are repeated until a maximum number of iterations (usually 4) has been executed. When this is the case, the algorithm moves to phase IV.

Phase IV: route improvement heuristic To further improve the quality of a given route, a route improvement heuristic is implemented. This heuristic does not change the order in which to visit the customers, but determines the optimal visits of depots given a certain sequence of customer visits. The procedure uses the GVRPTW formulation and solves a shortest path problem with time windows on this graph, from starting depot to ending depot, thereby visiting one node in each node cluster. This shortest path problem with time windows is also solved using the GLPK callable library.

Unfortunately, the shortest path with time windows is NP hard. Therefore, the computing time for this procedure is quite large. We therefore only perform this operation once at the end of the algorithm.

6 Experimental results

6.1 Experimental setup and real-life test cases

The column generation algorithm was implemented in C++ using the open source GNU Linear Programming KIT (GLPK) callable library version 4.33 (`http://www.gnu.org/software/glpk/glpk.html`). This prototype version of the code was called WMPopt. Since no data sets are available for testing or comparison (even to closely related problems), we are forced to compare WMPopt to the performance of the commercial solver it was designed to replace. To this end, several real-life instances, obtained from a Belgian waste management company, were used in the testing and calibration phases. These data sets are available from the authors upon request. Table 2 describes these different cases.

Table 2: Description of the 4 real-life cases

Case no.	Customer requests	Trucks	
		Capacity 1	Capacity 2
1	86	11	8
2	101	13	9
3	69	9	7
4	92	10	9

6.2 Sensitivity analysis

Prior to solving the real-life case, WMPopt was subjected to a thorough sensitivity analysis. The main goals of this phase were to calibrate the parameters and to judge the robustness of the performance of WMPopt with respect to these parameters. In this analysis, we varied the values of several parameters in a systematic way and measured the performance of WMPopt in terms of (1) the objective function value (total distance traveled), (2) the number of trucks of both types used, (3) the number of unplanned customers and (4) the computing time.

The parameters that were tested where the maximum number of columns to generate $c_{\max}$, the threshold value of σ_x for using unplanned customers as the seed for new columns $\sigma_{\max}$, and P, the value that determines the speed of growth or shrinkage of the generated routes. For each of these parameters, we determined several values that we considered reasonable and ran a series of 64 experiments. The results are shown in tables 3 to 5. Each row shows the value of the parameter, and the averages of the cost, number of unplanned customers, number of trucks used, and CPU time in seconds for the experiments corresponding to this parameter value.

Table 3: Influence of the value of P on the performance of WMPopt

P	Avg. Cost	Avg. Unpl.	Avg. Trucks	CPU (s)
6	542374	0,75	19	320
7	546978	0,75	19	293
8	554084	0,44	19	261
9	541563	1,75	19	391

Table 4: Influence of the value of $\sigma_{\max}$ on the performance of WMPopt

$\sigma_{\max}$	Avg. Cost	Avg. Unpl.	Avg. Trucks	CPU (s)
5	542687	1,25	19	326
6	550402	0,75	19	224
7	555696	0	19	229
8	551246	0,6	19	278
9	559096	0	19	276
10	560230	0,25	19	290
11	553439	0,75	19	277

Table 5: Influence of the value of $c_{\max}$ on the performance of WMPopt

$c_{\max}$	Avg. Cost	Avg. Unpl.	Avg. Trucks	CPU (s)
30	559139	0,25	19	270
40	557611	0	19	207
50	551007	0,625	19	286
60	557611	0	19	227
70	557611	0	19	229

The experiments clearly show that the performance of WMPopt is very robust with respect to the different parameter values. The different performance measures only deviate slightly when the parameters are changed. Based on this result, we can confidently set

the parameters to the values that yield lowest cost, i.e., $P = 9$, $\sigma_{\max} = 5$, $c_{\max} = 50$.

6.3 Performance benchmark

To benchmark the performance of WMPopt, we compare it to the performance of a commercial vehicle routing solver, that was adapted as much as possible to the specific problem formulation. Unfortunately, the solver cannot cope with some of the more complex constraints present in the real problem (which was one of the main motivations for the commissioning of this work). The solution of the commercial solver therefore needs to be "repaired" to restore its feasibility. An example of such a problem with the solver is that it cannot be forced to perform a round-trip operation in which no intermediate operations are scheduled (i.e., in which the truck immediately goes to the nearest waste depot with the full container and immediately returns to the customer with an empty container). If such a situation occurs, the truck needs to be manually forced to perform these operations before continuing with the tour. The repair procedure consists of trivial manipulations to the final solution and were performed exactly like a planner would apply them. Contrary to WMPopt, the commercial solver makes intensive uses of randomness to search for better solutions. We therefore perform three runs and report the average cost of the commercial solver's solution *after* the repair.

Table 6 shows the results of the experiments. It is clear that the performance of WMPopt far exceeds that of the commercial solver, finding a solution with on average 12.1% lower costs with the same number of trucks. Additionally, computing times are considerably lower in most cases, even without taking the extra time needed for the repair into account. Most importantly however, the commercial solver is unable to model all the complex constraints and therefore solves a relaxed version of the problem. Limits on its modeling capacity render the commercial solver unusable in practice. As mentioned, the final step in the procedure to solve the problem using the commercial solver is a trivial but time-consuming and error-prone repair phase. This makes it extremely difficult to interact with the solver software through its graphical user interface to make manual changes to the solution and re-optimize the solution (partially), a common practice in almost all planning departments. Finally, as mentioned, the commercial solver uses randomness intensively and results are therefore prone to variation. Although solutions resulting from different runs of the commercial software have very similar objective function values, the company producing the software found the varying solutions difficult to explain to its customers and therefore very undesirable.

Table 6: Comparison of WMPopt versus the commercial solver

Case	WMPopt			Commercial solver			
no.	CPU (s)	Cost	Trucks	CPU (s)[a]	Cost[b]	Trucks	ΔCost (%)[c]
1	118	547493	19	194	611099	19	11.6
2	263	660848	22	202	771874	22	16.8
3	104	448755	16	212	485432	16	8.2
4	129	573347	19	148	641588	19	11.9

[a] Average CPU time in seconds of three runs of the commercial solver, not including the repair phase
[b] Average cost for three runs of the commercial solver, after the repair phase
[c] Decrease in cost achieved by WMPopt with respect to the average of the cost of the three solutions found by the commercial solver after repair

7 Extensions

The GVRPTW formulation is very flexible and may be extended in several ways to allow for more complex problems to be modeled. Currently, we are implementing several of these extensions. Some of the extensions are rather trivial to formulate (such as time windows at the depots or multiple time windows at the customers). Two deserve some further investigation: washing depots and free customers.

7.1 Washing depots

In some situations, containers have to be washed before they are returned to a customer. This is done in so-called *washing depots*. These may be located at the site of a waste depot or a stock depot, but this is not required.

Washing depots can be easily added to the problem formulation. The shortest distance between two customer nodes, now may include the visit to a washing depot if the customer receiving the container requires it to be cleaned. No adaptation of the node clusters is necessary.

7.2 Free customers

In this paper, we have assumed that a customer that had a full container and was not an end-of-contract customer was either a round-trip customer or an exchange customer. I.e., either the container of this customer was delivered at the same time the full container was picked up, or the full container was emptied at a waste depot and then returned to the customer.

In practice, some customers may be indifferent as to whether they are serviced as a round-trip or an exchange customer. Such customers are called *free* customers. In this case, the model should be able to decide the cheapest way to service such a customer. Free customers can be easily added to our formulation. A truck arriving at such a customer must have either an empty container (so the customer can be serviced as an exchange customer) or an empty spot (so the customer can be serviced as a round-trip customer). A customer node for a free customer therefore looks as in figure 6.

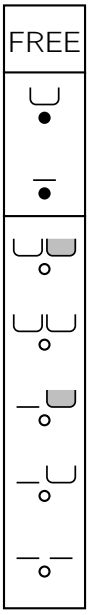

Figure 6: Free customer node cluster

8 Conclusions and future research

In this paper we have described a solution to a real-life roll-on–roll-off waste collection vehicle routing problem that arises when a waste management company handles industrial waste in standardized containers. The main difficulty of the routing problem treated in this paper is the fact that not only visits to customers, but also visits to stock depots and waste management depots need to be scheduled. Moreover, not all trucks necessarily have the same capacity: some can take only one container whereas others can take two. Also, the company can collect different types of waste in different types of containers.

To solve this problem efficiently we have developed a column generation method that combines a set partitioning master problem, solved using the GLPK open source linear programming solver, and a route generation subproblem, solved heuristically using a variant of Solomon's I1 insertion heuristic. To allow the insertion heuristic to efficiently deal with intermediate visits to depots, the roll-on–roll-off problem was reformulated as a generalized vehicle routing problem with time windows.

The method has been tested on four real-life case studies and compared to a commercial solver that has been adapted as well as possible to the specific problem at hand.

Our method, called WMPopt, was able to find dramatically better solutions than the commercial software. For this and other reasons, WMPopt was judged by the company producing the vehicle routing software to be far better suited at solving the problem than the commercial solver. Currently, the method is therefore being integrated into the commercial solver code, something which will take several more man-months to be accomplished.

Several topics for future research arise naturally. First, several of the extensions mentioned in the previous section remain to be implemented. These extensions will make the method more general and will allow us to implement it on more roll-on–roll-off routing problems. Second, the route improvement heuristic that we developed takes too much time. A more efficient heuristic, that can be implemented immediately after the route generation phase, during the column generation phase, would be a valuable addition to our method. Finally, to obtain truly excellent solutions, the column generation method needs to be followed by a local-search based method. The commercial solver in which our method will be integrated, contains a large arsenal of such local search operators. An interesting study would be to investigate the best strategy with which to follow up on the column generation method.

References

E. Angelelli and M.G. Speranza. The periodic vehicle routing problem with intermediate facilities. *European Journal of Operational Research*, 137:233–247, 2002a.

E. Angelelli and M.G. Speranza. The application of a vehicle routing model to a waste-collection problem: two case studies. *Journal of the Operational Research Society*, 53: 944–952, 2002b.

R. Aringhieri, M. Bruglieri, F. Malucelli, and M. Nonato. A special vehicle routing problem arising in the optimization of waste disposal: A real case. *Transportation Science*, 52(2):277–299, 2017.

R. Baldacci, L. Bodin, and A. Mingozzi. The multiple disposal facilities and multiple inventory locations rollon-rolloff vehicle routing problem. *Computers and Operations Research*, 33(9):2667–2702, 2006.

J. Bautista, E. Fernández, and J. Pereirac. Solving an urban waste collection problem using ants heuristics. *Computers & Operations Research*, In Press, 2007.

B. Beltrami and L.D. Bodin. Networks and vehicle routing for municipal waste collection. *Networks*, 4:65–94, 1974.

G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1–31, 2007.

N. Bianchessi and S. Irnich. Branch-and-cut for the split delivery vehicle routing problem with time windows. *Transportation Science*, 2019.

L. Bodin, A. Mingozzi, R. Baldacci, and M. Ball. The rollon-rolloff vehicle routing problem. *Transportation Science*, 34:271–288, 2000.

J. Bramel and D. Simchi-Levi. On the effectiveness of set covering formulations for the vehicle routing problem with time windows. *Operations Research*, 45:295–301, 1997.

B.P. Bruck and M. Iori. Non-elementary formulations for single vehicle routing problems with pickups and deliveries. *Operations Research*, 65(6):1597–1614, 2017.

D. Cattrysse, J. Maes, and L. Van Wassenhove. Set partitioning and column generation heuristics for capacitated dynamic lotsizing. *European Journal of Operational Research*, 46(1):38–47, 1990.

D. Cattrysse, M. Salomon, R. Kuik, and L. Van Wassenhove. Heuristics for the discrete lotsizing and scheduling problem with setup times. *Management Science*, 39(4):477–486, 1993.

D. Cattrysse, M. Salomon, and L. Van Wassenhove. A set partitioning heuristic for the generalized assignment problem. *European Journal of Operational Research*, 72: 167–174, 1994.

D. Cattrysse, K. Geeroms, A. Proost, and C. Van der Heyde. Container transport : a case study. *Logistics Information Management*, 9(6):15–23, 1996.

E. Choi and D.W. Tcha. A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers and Operations Research*, 34(7):2080–2095, 2007.

G.F. Cintra, F.K. Miyazawa, Y. Wakabayashi, and E.C. Xavier. Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. *European Journal of Operational Research*, 191(6):61–85, 2007.

G. Cristallo. *Optimisation de Tournées de Véhicules de Transport Container*. Mémoire de license en sciences economique et sociales, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgium, 1994.

L. De Meulemeester, G. Laporte, F.V. Louveaux, and F. Semet. Optimal sequencing of skip collections and deliveries. *Journal of the Operational Research Society*, 48:57–64, 1997.

U. Derigs, M. Pullmann, and U. Vogel. A short note on applying a simple ls/lns-based metaheuristic to the rollon–rolloff vehicle routing problem. *Computers & Operations Research*, 40(3):867–872, 2013.

M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992.

M. Elbek and S. Wøhlk. A variable neighborhood search for the multi-period collection of recyclable materials. *European Journal of Operational Research*, 249(2):540–550, 2016.

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

G. Ghiani and G. Improta. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research*, 122:11–17, 2000.

G. Ghiani, F. Guerriero, G. Improta, and R. Musmanno. Waste collection in Southern Italy: solution of a real-life arc routing problem. *International Transactions in Operational Research*, 12(2):135–144, 2005.

B.L. Golden and R.T. Wong. Capacitated arc routing problems. *Networks*, 11:305–315, 1981.

B.L. Golden, A.A. Assad, and E.A. Wasil. Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries. In P. Toth and D. Vigo, editors, *The vehicle routing problem*, pages 245–286. SIAM, Philadelphia, PA, 2002.

Gianfranco Guastaroba, Maria Grazia Speranza, and Daniele Vigo. Intermediate facilities in freight transportation planning: a survey. *Transportation Science*, 50(3): 763–789, 2016.

Kristian Hauge, Jesper Larsen, Richard Martin Lusby, and Emil Krapper. A hybrid column generation approach for an industrial waste collection routing problem. *Computers & Industrial Engineering*, 71:10–20, 2014.

A. Hertz, G. Laporte, and M. Mittaz. A tabu search heuristic for the capacitated arc routing problem. *Operations Research*, 48(1):129–135, 2002.

B.I. Kim, S. Kim, and S. Sahoo. Waste collection vehicle routing problem with time windows. *Computers and Operations Research*, 33(12):3624–3642, 2006.

I. le Blanc, M. van Krieken, H. Krikke, and H. Fleuren. Vehicle routing concepts in the closed-loop container network of arn—a case study. *OR Spectrum*, 28(1):53–71, 2006.

S. Mitrović-Minić, R. Krishnamurti, and G. Laporte. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B*, 38(8):669–685, 2004.

W.P. Nanry and J. Wesley Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B*, 34(2):107–121, 2000.

S.N. Parragh, K.F. Doerner, and R.F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

Tânia Rodrigues Pereira Ramos, Carolina Soares de Morais, and Ana Paula Barbosa-Póvoa. The smart waste collection routing problem: Alternative operational management approaches. *Expert Systems with Applications*, 2018.

M. Savelsbergh. A Branch-and-Price Algorithm for the Generalized Assignment Problem. *Operations Research*, 45:831–841, 1997.

M.M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):254–265, 1987.

J. Wy and B.-I. Kim. A hybrid metaheuristic approach for the rollon–rolloff vehicle routing problem. *Computers & Operations Research*, 40(8):1947–1952, 2013.

J. Wy, B.-I. Kim, and S. Kim. The rollon–rolloff waste collection vehicle routing problem with time windows. *European Journal of Operational Research*, 224(3):466–476, 2013.

9 Acknowledgements

The work of J. Raucq has been funded by the Institute for Scientific Research and Innovation in Brussels (IRSIB).

The authors are grateful to Prof. L. Wolsey for many helpful discussions.