A memetic algorithm for solving rich waste collection problems

Lavigne, Carolien; Inghels, Dirk; Dullaert, Wout; Dewil, Reginald

published in
European Journal of Operational Research
2023

DOI (link to publisher)
10.1016/j.ejor.2022.11.023

document version
Publisher's PDF, also known as Version of record

European Journal of Operational Research 308 (2023) 581–604

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, Manufacturing, Transportation and Logistics

A memetic algorithm for solving rich waste collection problems

Carolien Lavigne [a,b,*], Dirk Inghels [c], Wout Dullaert [c], Reginald Dewil [a,d]

[a] KU Leuven Institute for Mobility, Belgium
[b] KU Leuven, Center for Economics and Corporate Sustainability, Faculty of Economics and Business, Campus Brussels, Belgium
[c] Vrije Universiteit Amsterdam, School of Business and Economics, Department of Operations Analytics, the Netherlands
[d] KU Leuven, Group T, Department of Mechanical Engineering, Centre for Industrial Management, Belgium

ARTICLE INFO

Article history:
Received 27 August 2021
Accepted 13 November 2022
Available online 17 November 2022

Keywords:
Routing
Waste collection
Multiple depots
Intermediate facilities
Partial pick-ups

ABSTRACT

Inspired by a real-life case in the Brussels Capital Region, Belgium, this paper provides a Memetic Algorithm with Sequential Split procedure (MASS) for solving a large variety of waste collection problems with multiple depots, a restricted vehicle fleet at each depot, multiple (intermediate) processing facilities, capacity restrictions per processing facility and partial pick-ups. MASS first generates satisfactory initial solutions which are feasible w.r.t. the shift duration and the vehicle capacity using a novel procedure in which (1) a giant tour is split into vehicle routes, (2) intermediate processing facilities are introduced and (3) waste pick-ups can be split further if profitable. Second, MASS improves these solutions through local search. New test instances are created, which are used to evaluate the performance of MASS's components. We show that MASS provides high-quality feasible solutions by comparing MASS with an exact approach on a small example. Furthermore, MASS is tested on existing instances for the multiple-depot vehicle routing problem (MDVRP), (multi-depot) vehicle routing problem with intermediate facilities ((MD)VRPIF) and the multi-depot vehicle routing problem with inter-depot routes (MDVRPI). MASS shows competitive results for the MDVRP, the VRPIF and the MDVRPI. For the MDVRPIF, MASS obtains better results than those currently found in the literature. To assess its practical value, MASS is used to solve a real-life waste collection problem in the Brussels Capital Region in which alternative scenarios for municipal bio-waste collection and treatment are compared.

1. Introduction

Inspired by a real-life case in the Brussels Capital Region, Belgium, we introduce the general Rich Waste Collection Problem (RWCP). In the RWCP, a fleet of collection vehicles needs to pick up waste at waste collection points. The fleet can be stationed at multiple depots and each depot has a limited number of available vehicles. Each vehicle has a limited load capacity and a maximum shift duration. The collected waste can be dropped off at multiple intermediate processing facilities subject to capacity constraints. Additionally, multiple pickup visits at a collection point are allowed.

According to Beliën, De Boeck, & Van Ackere (2014), the most commonly included constraints in operational waste collection problems are the load capacity of the collection vehicles and the maximum shift duration of the work force. Gruler, Fikar, Juan, Hirsch, & Contreras-Bolton (2017), Markov, Varone, & Bierlaire (2016) and Ramos & Oliveira (2011) study waste collection networks with multiple depots. In most cases, the capacity of the vehicles is insufficient to cover an entire shift, which requires multiple visits to the processing facilities during the shift to drop off waste. Hemmelmayr, Doerner, Hartl, & Rath (2013), Markov et al. (2016), Willemse & Joubert (2016), and Gruler et al. (2017) introduce such intermediate processing facilities (IF) in waste collection problems. Moreover, waste processing facilities such as recycling plants, composting facilities, and incinerators have a limited daily capacity, as opposed to landfills, which also necessitates the introduction of processing facility capacity constraints (see e.g., Hemmelmayr et al., 2013).

Waste collection allows for partial pick-ups in which the waste at a collection point can be divided and collected by multiple vehicles. Partial pick-ups for waste collection problems can be modelled similar to split deliveries in Vehicle Routing Problems (SDVRPs). Although allowing for partial pick-ups was shown empirically by Dror & Trudeau (1989,1990) to reduce both travel costs and

* Corresponding author.
E-mail addresses: carolien.lavigne@kuleuven.be (C. Lavigne), dirk.inghels@vu.nl (D. Inghels), wout.dullaert@vu.nl (W. Dullaert), reginald.dewil@kuleuven.be (R. Dewil).

https://doi.org/10.1016/j.ejor.2022.11.023
0377-2217/

the number of required vehicles, partial pick-ups are rarely studied in the waste collection literature (see e.g., Mar-Ortiz, González-Velarde, & Adenso-Díaz, 2013).

The proposed RWCP considers all constraints found in a large-scale real-life case and can be considered as a generalization of several waste collection problems encountered in the literature. The RWCP is able to realistically model complex waste collection and can be used for solving the operational planning of truck routes or for evaluating alternative waste collection networks. Although solution approaches for all of the problem features and constraints in the RWCP have been proposed, these approaches are often disjoint and scattered across the scientific literature. In this paper, we propose a solution approach that simultaneously considers all problem features. Furthermore, we show how the general RWCP can be used to model specific waste collection problems by fixing or specifying certain parameters or variables.

We developed a Memetic Algorithm with Sequential Split procedure (MASS) for solving the RWCP. MASS was constructed by adding novel elements to a combination of promising existing algorithms, developed for VRPs with only a subset of the constraints and features of the RWCP. In particular, elements of algorithms developed for the SDVRP by Boudia, Prins, & Reghioui (2007), Derigs, Li, & Vogel (2010) and Silva, Subramanian, & Ochi (2015) were combined with a genetic algorithm developed by Vidal, Crainic, Gendreau, Lahrichi, & Rei (2012) for the multi-depot VRP.

The main contributions of this study are the following: (1) We provide a new metaheuristic (MASS) for solving instances of a highly general waste collection problem. (2) MASS contains a novel approach for generating sets of satisfactory routes for cases with multiple depots and multiple intermediate facilities, and partial pick-ups. We also adapt the partial pick-up local search moves introduced by Boudia et al. (2007), Derigs et al. (2010) and Silva et al. (2015), and modify common VRP local search moves to allow for intermediate facilities and partial pick-ups. (3) We have shown that MASS can be adapted to the case at hand by omitting or altering certain elements of the algorithm and, thus, offers a single tool for solving a multitude of waste collection problems. The average computation time of MASS is short. According to Vidal et al. (2012), it is therefore suitable to support operational decisions. But it can also be used for tactical decisions by estimating the collection costs under different what-if scenarios such as alternative processing facility and/or depot locations and their respective capacities. (4) Our solution approach has proven to be competitive with the state-of the art methodologies for solving not only waste collection problems, but also more specific vehicle routing problems such as the multi-depot VRP, the (multi-depot) VRP with intermediate facilities, and the multi-depot VRP with inter-depot routes.

This paper is structured as follows. Section 2 provides an overview of the relevant literature. In Section 3, we present the general Rich Waste Collection Problem. We first provide a detailed description of the algorithm in Section 4. Second, we describe how certain parameters and variables can be adjusted to model more specific waste collection problems. In Section 5, we examine the performance of the proposed MASS by using test instances from the literature on related problems and a new set of test instances designed for the proposed RWCP. To evaluate the algorithm's potential of solving real-life RWCPs, we present and solve a case from the Brussels Capital Region in Section 6.

2. Literature review

Waste collection problems have been widely studied in the last decades. A waste collection problem typically deals with generating minimal cost routes for a fleet of vehicles stationed at one or more depots to collect waste at several locations such as households or containers. The waste is dropped off at processing facilities, after which trucks continue their collection round or return to the depot. These problems are therefore often modelled as Vehicle Routing Problems (VRP) with case-specific extensions. Many extensions and constraints have been introduced in the waste collection literature. As the RWCP is inspired by a real-life case, this review focusses on the papers considering the extensions and constraints included in the case study. For an extensive review of the operational waste collection literature and the constraints considered, we refer the interested reader to Beliën et al. (2014).

First, vehicle capacity restrictions and a restriction on the shift duration of a vehicle's crew are often considered (Beliën et al., 2014). For instance, Delgado-Antequera, Laguna, Pacheco, & Caballero (2020) model the waste collection problem as a Capacitated Vehicle Routing Problem (CVRP) and López-Sánchez, Hernández-Díaz, Gortázar, & Hinojosa (2018) consider the vehicle capacity and shift duration constraints for modelling a real-life waste collection problem in the south of Spain.

Second, the waste collection vehicles typically need to be stationed at multiple depots resulting in a VRP with Multiple Depots (MDVRP). While Teixeira, Antunes, & Sousa (2004), Farrokhi-Asl, Makui, Jabbarzadeh, & Barzinpour (2020), Ramos & Oliveira (2011), Markov et al. (2016) and Gruler et al. (2017) study waste collection problems with multiple depots, only Farrokhi-Asl et al. (2020) and Ramos & Oliveira (2011) also impose restrictions on each depot's fleet size.

Third, if the waste quantities that need to be collected are large, a collection vehicle might need to be emptied multiple times at a processing facility along its route. These intermediate stops at (processing) facilities coincide with VRPs with Intermediate Facilities (VRPIF) in the routing literature. Waste collection problems with intermediate facilities were studied by, for instance, Kim, Kim, & Sahoo (2006), Markov et al. (2016), Hemmelmayr et al. (2013), Willemse & Joubert (2016), and Gruler et al. (2017). Beliën et al. (2014) performed a literature review on municipal solid waste collection problems and found that the number of studies including multiple processing facilities greatly exceeds studies with only a single processing facility. Additionally, Hemmelmayr et al. (2013) investigated the impact of processing facility capacity constraints and found that especially biased distributions of processing facility capacities increase collection costs significantly. Vehicle routing problems with both multiple depots and intermediate facilities (MDVRPIF) were studied by Markov et al. (2016), but they leave the assignment of the destination depot flexible. Markov et al. (2016) argue that this flexible destination assignment could be a realistic setting for large rural areas to allow truck drivers to stay at the destination depot (different from their home depot) overnight. When all depots can act as intermediate facilities and vice versa, the problem becomes a multi-depot VRP with inter-depot routes for which Ramos, Gomes, & Barbosa-Póvoa (2020) and Muter, Cordeau, & Laporte (2014) have developed solution approaches. Furthermore, note that reverse logistics VRPs, such as waste collection problems, differ from traditional VRPs for product delivery by the mandatory final facility visit before returning to the depot, i.e., the empty-return constraint.

Fourth, an extension which has only scarcely been applied to waste collection problems is the inclusion of partial pick-ups at the waste collection nodes. Allowing for partial pick-ups is necessary if the waste quantity at a collection point can exceed the vehicle capacity. This can occur either due to the nature of the collection point (see, for instance, the collection of electrical waste in stores presented by Mar-Ortiz et al., 2013) or due to the scale and level of detail of the case study (Lavigne, Beliën, & Dewil, 2021). Partial pick-ups have been studied in the routing literature on Capacitated VRPs with Split Deliveries (CSDVRP). The SDVRP is highly complex and can only be solved to optimality for very specific cases (integer demands, limited number of customers, etc.)

(Archetti & Speranza, 2012). Moreover, Dror & Trudeau (1989,1990) showed empirically that travel costs and the number of required vehicles can be reduced by allowing split deliveries.

Beliën et al. (2014) found that heuristics, as opposed to exact approaches, have been studied extensively for solving operational waste collection problems. The type of heuristic applied, however, varies widely in the literature. Hemmelmayr et al. (2013) use a hybrid algorithm that combines a Variable Neighborhood Search (VNS) heuristic with an exact procedure for inserting the intermediate facilities. First a giant tour of pick-ups is generated after which intermediate facilities are introduced whenever a vehicle's capacity is reached. This approach is based on the SPLIT-procedure introduced by Prins, Lacomme, & Prodhon (2014), which is also used by Vidal et al. (2012) for creating satisfactory routes. Gruler et al. (2017) developed a simheuristic framework combining biased (oriented) randomization, an iterated local search metaheuristic and Monte Carlo simulation for solving a stochastic multi-depot waste collection problem. Markov et al. (2016) employ a multiple neighborhood search heuristic for a multi-depot waste collection problem with multiple intermediate landfills. Delgado-Antequera et al. (2020) present a semi-greedy construction heuristic and local search for constructing collection routes in a Spanish rural region.

Various approaches have been proposed for solving VRPs that consider a selection of the previously mentioned constraints. Montoya-Torres, Franco, Isaza, Jiménez, & Herazo-Padilla (2015) found that hybrid procedures have proven their effectiveness for MDVRPs. In particular, the Hybrid Genetic Algorithm with Adaptive Diversity Control proposed by Vidal et al. (2012) showed impressive results on standard test instances for the MDVRP. For VRPs with intermediate facilities, Crevier, Cordeau, & Laporte (2007) combine a Tabu Search (TS) heuristic with Adaptive Memory to create single-depot routes that start and end at the same depot and assign multiple routes to trucks using a Set Partitioning algorithm. Tarantilis, Zachariadis, & Kiranoudis (2008) also propose a TS heuristic embedded in a VNS framework. Note that the VNS procedure by Hemmelmayr et al. (2013) obtains better results for the same set of test instances. While Ramos et al. (2020) and Muter et al. (2014) developed respectively a matheuristic approach and a branch-and-price algorithm for solving the MDVRPI, only the solution approach of Ramos et al. (2020) was tested on large instances with up to 288 customers.

Vehicle Routing Problems with partial pick-ups have been solved mostly by performing local search within a metaheuristic framework. Boudia et al. (2007) propose a genetic algorithm with population management and diversity control combined with local search moves specifically adapted to partial pick-ups for solving the SDVRP. As stated by Archetti & Speranza (2012), the genetic algorithm proposed by Boudia et al. (2007) outperforms the TS algorithm developed by Archetti, Speranza, & Hertz (2006). Derigs et al. (2010) adapt four standard VRP local search moves for the SDVRP and evaluate several metaheuristic frameworks for solving SDVRPs. Derigs et al. (2010) found that their Attributed-Based Hill Climber (ABHC) method outperformed the algorithms proposed by Boudia et al. (2007). Last, Silva et al. (2015) introduce a multi-start Iterated Local Search (ILS) method for solving the SDVRP and improve the solutions found for an extensive list of test instances.

While the exact solution approach proposed by Lavigne et al. (2021) takes into account the previously mentioned constraints, the authors limit the maximum number of consecutive pick-ups by a vehicle to two and stress that the solution approach is only appropriate for small, aggregated instances. Thus, although the waste collection and routing literature offers solution methods for a large variety of waste collection-specific extensions and constraints, no solution method could be found for realistic cases taking into account all of the elements of the RWCP presented in this paper: multiple depots, a restriction on the fleet residing at each depot, multiple (intermediate) processing facilities, capacity restrictions per processing facility, and partial pick-ups. Considering the solution methods found for VRPs with a selection of these constraints, we present a hybrid approach that combines (1) inserting intermediate facilities in a giant chromosome when the vehicle capacity is reached, (2) a genetic algorithm with (3) a dedicated local search procedure (i.e. a memetic algorithm) which contains (4) VRP moves adapted for cases with partial pick-ups.

3. Problem description

The Rich Waste Collection Problem (RWCP) is a Capacitated Vehicle Routing Problem with a Time Duration constraint, Partial Pick-ups, Multiple and Capacitated Depots and Multiple and Capacitated Intermediate Facilities (MCD-MCIF-CVRP-TD-PP). Let $G=(\mathcal{V}, \mathcal{A})$ be a complete graph with $|\mathcal{V}| = d + f + n$ nodes with $d \in \mathbb{N}_0$ the number of depots, $f \in \mathbb{N}_0$ the number of intermediate processing facilities and $n \in \mathbb{N}_0$ the number of pick-up locations. $\mathcal{V}$ is divided into three sets $\mathcal{V} = \mathcal{V}^{DEP} \cup \mathcal{V}^{FAC} \cup \mathcal{V}^{PU}$. Each $v_i \in \mathcal{V}^{DEP}$, represents a depot with a fleet of Q_i^D collection vehicles with capacity Q^{CV} (tonne). We assume that vehicles must always return to their initial depot. The time available for a truck driver to drive around and collect the waste is limited to the shift duration T^S (hour). All $v_i \in \mathcal{V}^{FAC}$ are intermediate processing facilities, each with a daily capacity of Q_i^F (tonne). The time it takes to unload the waste at facility $v_i \in \mathcal{V}^{FAC}$ equals T_i^{UL} (hour). Furthermore, collection vehicles do not have to be completely full before driving to a processing facility.

Each $v_i \in \mathcal{V}^{PU}$ represents a location where waste quantity q_i should be picked up. The local waste collection scheme, such as door-to-door collection or container bring-systems, data availability, and the size of the region being analysed determine the definition of the waste collection pick-up nodes $v_i \in \mathcal{V}^{PU}$. The waste generated in a pick-up location is allowed to exceed a single collection vehicle's capacity Q^{CV}. Henceforth, we make the distinction between a pick-up location v_i, which represents any type of waste collection node, a pick-up location's waste quantity q_i, which is the total amount of waste that needs to be collected at v_i, and a pick-up $\kappa \in K$, being the combination of a pick-up quantity and a pick-up location. To determine the service time at the pick-up locations, each node $v_i \in \mathcal{V}^{PU}$ has a collection speed τ_i (hour/tonne waste collected). The type of waste collected in each area on a specific day of the week often depends on a calendar determined by the local authorities. The calendar states on which day(s) of the week residents should put out their waste. This includes local preferences and is typically fixed for a longer period of time. The pick-up locations and their corresponding waste quantities on a day must be collected on that same day.

Arcs $a_{ij} \in \mathcal{A}$ represent the direct travel possibility between nodes v_i and v_j and have a corresponding travel time t_{ij} for driving from node v_i to v_j. Throughout this paper, we minimize a cost function $CF = OC \cdot TD + IC \cdot NV$ encompassing both variable and fixed costs, with OC the hourly operational cost, TD the total driving time of all vehicles (in hours), IC the daily fixed cost for employing one vehicle to execute a route and NV the number of vehicles required.

An example of a feasible collection scheme is provided in Fig. 1. Vehicle 1 leaves node v_2 (a depot), visits pick-up locations v_5, v_6, v_7 and v_8, collects their waste and drops off the waste at v_3, a processing facility. It then resumes its route to pick-up location v_{14} where it collects part of the location's waste quantity, after which it goes to pick-up location v_{13} and processing facility v_4. On the same day, vehicle 2 leaves depot v_1 and visits pick-up locations v_{19} and v_{20}. It drops off its waste at processing facility v_4. The vehicle then visits pick-up location v_{14}, picks up the amount of waste that was left after vehicle 1's visit. It then goes to processing facility

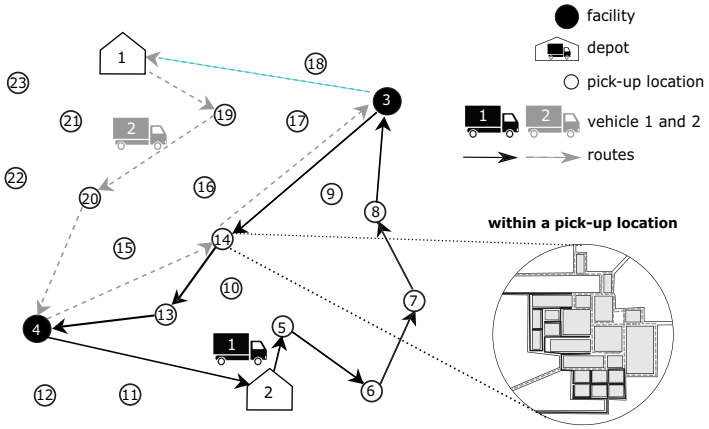


Fig. 1. A waste collection scheme.

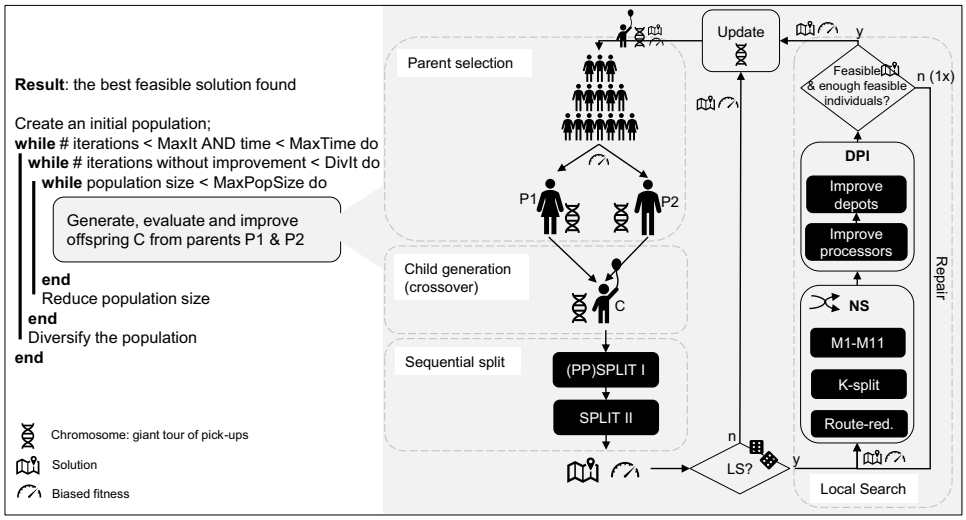


Fig. 2. Outline of the Memetic Algorithm with Sequential Split procedure.

v_3 and returns to its depot. Note that the waste generated in pick-up location v_{14} is collected by two vehicles as partial pick-ups are allowed. In the example in Fig. 1, waste is collected door-to-door. The pick-up location's service time, i.e. how long it takes to collect the waste, is distributed over the two vehicles proportionally to the amount of waste each vehicle collects using the pick-up location's collection speed.

4. The memetic algorithm with sequential split procedure

To solve the RWCP introduced in Section 3, we propose a Memetic Algorithm with Sequential Split procedure (MASS) inspired by Vidal et al. (2012). In particular, we adopt the diversity control, the population management and the split-procedure of the Hybrid Genetic Search with Adaptive Diversity Control (HGSADC) metaheuristic introduced by Vidal et al. (2012). We extend the algorithm by introducing algorithmic components that can additionally handle intermediate facilities, capacity constraints on the depots and processing facilities, and partial pick-ups. Specifically, we introduce a sequential splitting procedure that inserts visits to depots and processing facilities into a giant chromosome of pick-ups and allows pick-ups to be split further. Furthermore, we provide dedicated local search operators that can handle all elements of the RWCP. A general overview of MASS is provided in Fig. 2.

The general structure of the algorithm follows the genetic algorithm as described by Holland (1975). The algorithm starts with creating an initial set of individuals (see Section 4.3). An individual P consists of a giant tour of waste pick-ups $\kappa \in K$ that collect all the waste at the pick-up locations. Each κ represents a waste quantity $q(\kappa) \leq Q^{CV}$ being collected at at pick-up location $pul(\kappa)$. For each individual, the algorithm determines its corresponding solution to the RWCP using the sequential split procedure (see Section 4.2) which splits up the giant tour into vehicle routes, assigns routes to depots and introduces visits to processing facilities to drop off the waste.

The individuals are evaluated and compared to one another based on the level of "fitness" of their solution, which depends on the solution's incurred costs and its feasibility with regards to processor and depot capacities (see Section 4.4). From this initial set of individuals, defined as the parent population, an offspring is generated. Using an order-based crossover procedure, a child C is created from two parents who are selected using a binary tournament procedure following Vidal et al. (2012). A detailed description of the crossover procedure is provided in Appendix A. Child solutions are generated using the sequential split procedure and then improved through local search with a probability of $Prob^{ls}$. The offspring is then added to the parent population. If the parent population exceeds *MaxPopSize*, it is reduced to *MinPopSize* by removing the worst individuals. MASS uses a *MinPopSize* of μ, an offspring size of λ and a *MaxPopSize* of $\mu + \lambda$.

The algorithm proceeds by iteratively creating offspring, adding them to the parent population and selecting good individuals for

(a)

[1]	[2]	[3]	[4]	[1]	[8]	[5]	[6]	[9]	[7]	[7]	[8]
0.2	0.6	0.5	0.4	0.8	0.8	0.3	0.7	0.5	0.4	0.1	0.2

(b)

route 1	D3	[1] 0.2	[2] 0.6	F1	[3] 0.5	[4] 0.4	F4	[1] 0.8	F1	D3
route 2	D1	[8] 0.8	F2	[5] 0.3	[6] 0.7	F3	D1			
route 3	D5	[9] 0.5	[7] 0.4	F3	[7] 0.1	[8] 0.2	F2	D5		

Fig. 3. (a) An individual's chromosome including the amount of waste picked up at a pick-up location; (b) The solution of the individual in (a).

generating the next generation. When the number of iterations exceeds *MaxIt* or if the algorithm exceeds its time limit (*MaxTime*), the algorithm is stopped and the individual with the best feasible solution to the problem is reported.

Three main diversification elements are used. (1) The fitness value of an individual depends on how different it is with respect to the population in addition to the individual's incurred costs and its depot and processor capacity feasibility. (2) If no better individuals were found for *DivIt* iterations, a new population is generated. (3) While managing the population size, identical individuals are removed. For a detailed description of the population management and population diversification process, we refer to Vidal et al. (2012). In the following sections, the modified elements of the HGSADC and the novel approaches introduced are discussed.

4.1. Individual and solution representation

In a memetic algorithm, an individual is represented by a chromosome, i.e. a long sequence of genes in which the location of each gene in the sequence is key for evaluating the fitness of the individual. The genes often represent customers or, for waste collection problems, pick-up locations (see Baker & Ayechew, 2003, Prins, 2004 and Vidal et al., 2012). As partial pick-ups are allowed in the RWCP, a pick-up location can occur multiple times in the chromosome (Boudia et al., 2007). An example of such a chromosome is presented in Fig. 3 a). It contains information on what pick-up location is visited (presented between brackets) and how much waste is picked up at that occurrence (as a % of the vehicle capacity).

To evaluate the individual's fitness, the chromosome is transformed into a solution following the methodology described in Section 4.2. Similar to Vidal et al. (2012), the chromosome is split into vehicle routes to generate the individual's solution and it is determined which vehicles should leave from which depots. MASS additionally inserts processing facilities where each vehicle should drop off waste. However, as opposed to Vidal et al. (2012), a solution is always feasible with regards to the vehicle capacity and the maximum shift duration. Feasibility with respect to the capacities of the depots and the processing facilities is, however, not imposed.

To illustrate the main steps of the sequential split procedure, consider the chromosome presented in Fig. 3 a). One of the splitting operators in Section 4.2 (SPLIT II) splits the chromosome into the three routes shown in (b), once after the second pick-up at pick-up location 1 when a total quantity of 0.8 is collected and once after the only pick-up at pick-up location 6. This operator also assigns the vehicles performing these routes to depots. One vehicle leaves depot 3 (D3), a second vehicle leaves depot 1 (D1), while another vehicle leaves depot 5 (D5). Another splitting operator ((PP)SPLIT I) introduces visits to the processing facilities, denoted by the letter F in (b). For instance, in route 1, a vehicle leaves depot 3, then visits pick-up locations 1 and 2 and drops off the waste at processing facility 1 (F1). Each chromosome defines a unique solution.

4.2. Sequential split procedure

We create a solution by dividing the chromosome of an individual P into vehicle routes specifying the depot the vehicle leaves from, the pick-up locations it visits, the quantities that are collected and where the waste is dropped off at a processing facility. Each route is divided into subroutes which are a sequence of pick-ups, followed by a drop-off. For example, route 2 in Fig. 3 b) contains two subroutes. In the first subroute, the vehicle visits pick-up location 8 and drops off the waste at processing facility 2. In the second subroute, it visits pick-up locations 5 and 6 and continues to processing facility 3.

The sequential split methodology introduced in this study contains two distinct procedures. The first procedure (SPLIT I), inspired by the split procedure of Prins (2004), splits an individual's chromosome into subroutes, taking into account the vehicle capacity. We additionally assign processing facilities to each subroute. Two alternatives of this procedure are presented: a procedure in which the collected quantities of the pick-up locations are kept fixed (simple SPLIT I) and a procedure which allows further division of the pick-ups and their collection quantities (PP (partial pick-ups) SPLIT I). The second procedure (SPLIT II) groups the subroutes into vehicle routes, taking into account the maximum shift duration and assigns a depot to each route. The SPLIT II-procedure thus guarantees feasibility w.r.t. the maximum shift duration constraint, as opposed to the SPLIT-procedure applied in Vidal et al. (2012).

SPLIT I

The simple SPLIT I procedure is explained using the small example depicted in Fig. 4. An individual's chromosome (top table in Fig. 4) is divided into a sequence of subroutes (lower table in Fig. 4). We create an auxiliary graph $HI = (\mathcal{V}, A, C)$ with $\mathcal{V}$ containing $|K| + 1$ nodes from 0 to $|K|$ (Fig. 4 d)), and $|K|$ the number of pick-ups in the chromosome. A is the set of arcs $a_{i,j}$, with $i < j$, if a vehicle can perform pick-ups κ_{i+1} to κ_j before dropping off the waste without exceeding vehicle capacity Q^{CV} and without exceeding the maximum shift duration T^S (if the travel time from and to the depot would be added). We determine which arcs meet these constraints and, thus, can be included in A by applying a procedure, which we explain using the following example.

In Fig. 4 a), arcs feasible with regards to the vehicle capacity $Q^{CV} = 1$ are shown.

In Fig. 4 b), we insert after each potential sequence of pick-ups, the processing facility for which the sum of the travel time from κ_j to this facility, the travel time from this facility to κ_{j+1}, and the unload time at this facility is lowest. To check the shift duration feasibility in Fig. 4 c), we determine the depot for which the sum of the travel time from this facility to the depot and from the depot to κ_{i+1} is the lowest. If the duration of this route, containing only one subroute, is smaller than or equal to T^S, then the arc is added to A. Otherwise, we investigate whether choosing an alternative processing facility, i.e. the processing facility stemming from the processing facility-depot combination which minimizes the route's duration, would render the arc feasible. If choosing the

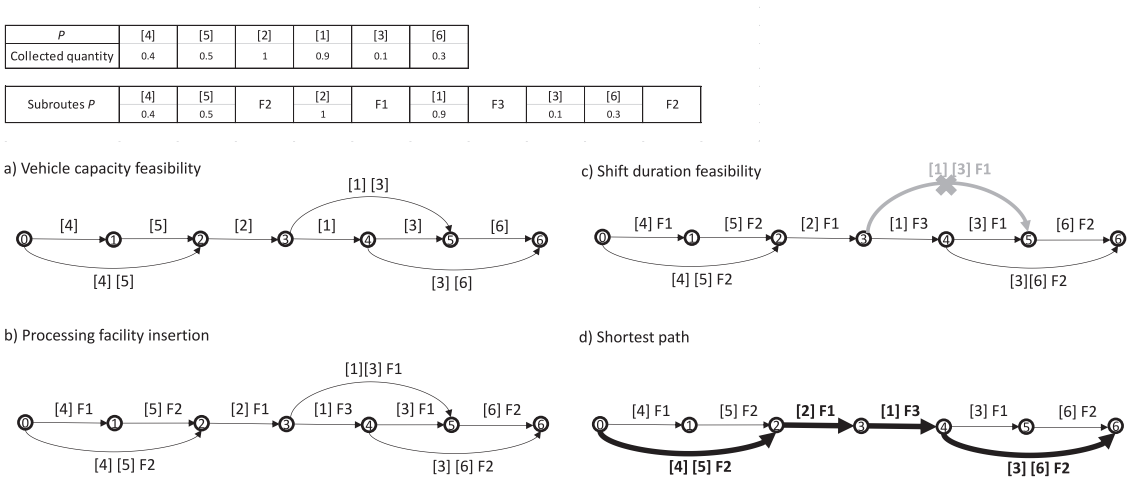


Fig. 4. SPLIT I; the chromosome of individual P and the created subroutes; (a) Arcs of pick-ups feasible w.r.t. vehicle capacity; (b) Insertion of processing facilities; (c) Checking shift duration feasibility and removing infeasible arcs; (d) Shortest path optimization establishing the best subroutes.

alternative processing facility would lead to a feasible route w.r.t. the shift duration, the alternative processing facility is selected. Otherwise, as is the case for arc $\{[1][3]F1\}$ in Fig. 4 c), the arc is not included in A.

The arc-costs $c_{i,j}$ correspond to the duration of the subroutes, but they can be modified to include other cost types such as, for instance, distance-dependent costs. Note that $c_{i,j}$ includes the driving time from the arc's facility to the first pick-up of the next arc to account for this travel time if these arcs (i.e. subroutes) are executed in one route. The resulting graph HI is an acyclic graph for which the shortest path problem from source 0 to sink $|K|$ with arc-cost $c_{i,j}$ can be solved with Dijkstra's algorithm using a minimum priority queue within $O(|\mathcal{V}| + |A| \cdot \log(|\mathcal{V}|))$ time (Cormen, Stein, Rivest, & Leiserson, 2001). The shortest path in Fig. 4 d) (in bold) provides the result of the SPLIT I procedure: a sequence of subroutes, each containing a sequence of pick-ups and the chosen processing facility.

For this procedure to work, the sink of graph HI (node $|K|$, with $|K|$ the last pick-up in the chromosome) must always be reachable from its source (0). To guarantee this, the subroutes with a single pick-up should always be feasible w.r.t. T^S and Q^{CV}. As will be explained in Section 4.3, these single-pick-up subroutes will always be feasible with regards to the shift duration if the processing facility-depot combination, resulting in the shortest route duration, is chosen. The second search for an appropriate processing facility, thus, ensures that all single-pick-up subroutes are feasible and included in A. This guarantees that node $|K|$ can be reached from the graph's source.

PPSPLIT I

The partial pick-up split procedure (PPSPLIT I) is an adaptation of the SPLIT I-procedure and can be used if partial pick-ups are allowed. PPSPLIT I allows us to divide a pick-up in the chromosome of P into two pick-ups. If a pick-up κ_i is divided, the pick-up is duplicated. The duplicate pick-up $\kappa_{i'}$ is inserted after κ_i and part of the collected quantity of κ_i is passed on to $\kappa_{i'}$. The latter depends on the pick-ups preceding κ_i in the chromosome of P. If a vehicle still has some free capacity (Q^{free}) after performing its last pick-up κ_{i-1}, the original κ_i will be assigned a quantity Q^{free} (assuming that Q^{free} does not exceed $q(\kappa_i)$) and pick-up κ_i will be added to the route of the vehicle. The duplicate pick-up will receive the remainder of $q(\kappa_{i'}) = q(\kappa_i) - Q^{free}$.

In the PPSPLIT I-procedure, an auxiliary graph $HPPI = (\mathcal{V}, A, C)$ is created to split the chromosome into subroutes. Fig. 5 shows how $HPPI$ in Fig. 5 d) is generated. The procedure starts in Fig. 5 a) by adding $|K|+1$ nodes to $\mathcal{V}$, with $|K|$ the number of pick-ups in the chromosome. Then, arcs $a_{i,j}$ with $i < j$ are added to A if performing pick-ups κ_{i+1} to κ_j subsequently by one vehicle is feasible w.r.t. the vehicle capacity Q^{CV}.

We then generate additional arcs by allowing pick-ups to be split. Define a primary arc $pa_i = a_{i,i+1}$ as the arc leaving node i and performing a subroute with only a single pick-up. Primary arcs coincide with the pick-ups in P's chromosome. The primary arcs in Fig. 5 a) are $a_{0,1}$, $a_{1,2}$, $a_{2,3}$, $a_{3,4}$, $a_{4,5}$ and $a_{5,6}$. For each primary arc pa_i, we investigate if the pick-up in this arc can be divided in such a way that part of this pick-up can be added to a preceding arc $a_{k,i}$ to create a full truck. First, we assign each arc $a_{i,j}$ a load $L_{a_{i,j}}$ which is the sum of the waste quantities collected during the pick-ups in $a_{i,j}$. Second, for each node i, define AR_i as the set of arcs arriving in node i. Assign each node $i \in \mathcal{V}\backslash\{0, |K|\}$ a $MaxL_i = \max_{a_{k,i} \in AR_i}\{L_{a_{k,i}}\}$ and $MinL_i = \min_{a_{k,i} \in AR_i}\{L_{a_{k,i}}\}$ which are respectively the maximum and minimum vehicle loads of the arcs in AR_i. The resulting L, $MaxL$ and $MinL$ values are added in Fig. 5 b).

We then perform the following procedure on each primary arc, starting at the primary arc leaving the source and ending with the primary arc arriving at the sink of $HPPI$. If $MinL_i < Q^{CV}$ (condition 1), some arcs arriving in node i still have some free capacity and, thus, some waste quantity of pa_i could be added to these arcs. If $MaxL_i + L_{pa_i} \leq Q^{CV}$, then, for all arcs $a_{k,i}$ arriving at node i, an additional arc will already have been added to A containing the pick-ups of $a_{k,i}$ and the pick-up of pa_i. If, however, $MaxL_i + L_{pa_i} > Q^{CV}$ (condition 2), we know that for at least one of the arcs arriving in i, say $a_{k,i}$, no arc was added to A yet, combining the pick-ups of arc $a_{k,i}$ and the (whole) pick-up in pa_i. Thus, if both conditions are met, we can still add a quantity of waste to at least one of the arcs $a_{k,i} \in AR_i$ without creating duplicates and we can proceed with splitting the primary arc.

In Fig. 5 c1) primary arc pa_0 ($a_{0,1}$) cannot be divided as no arcs arrive in node 0. In primary arc pa_1 ($a_{1,2}$), $MaxL_1 + L_{pa_1} < Q^{CV}$. Therefore, the pick-ups in this arc cannot be divided. This illustrates how duplicates are avoided as arc $a_{0,2}$, containing the whole pick-ups in pa_0 and pa_1, already exists. For primary arc pa_2 ($a_{2,3}$) (in bold), the $MaxL_2$ is 0.9, $MinL_2$ is 0.5 and the load $L_{a_{2,3}}$ is 1. Therefore, both conditions are met and we proceed with dividing the pick-up in primary arc pa_2 ($a_{2,3}$).

We define AR_i^{pot} as the subset of AR_i containing arcs with a load $L_{a_{k,i}}$ smaller than Q^{CV}. To these arcs, we can potentially still add a pick-up. In Fig. 5, c1), both arcs arriving in node 2 meet this condition. For each arc $a_{k,i} \in AR_i^{pot}$, a new node i' and a new arc $a_{k,i'}$ is added to the graph $HPPI$ (the dashed dots and arcs in Fig. 5 c1)). The sequence of pick-ups in the new arc $a_{k,i'}$ is a copy of the corre-

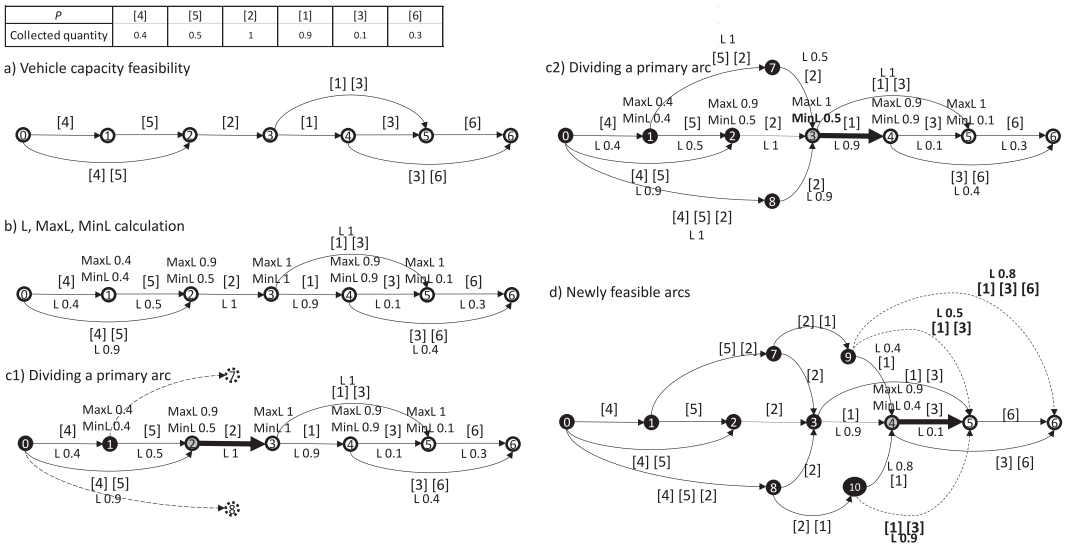


Fig. 5. PPSPLIT I; the chromosome of individual P; (a) Arcs feasible w.r.t. vehicle capacity; (b) Determining the load of each arc and the MaxL and MinL of each node; (c1-c2) Division of a primary arc; (d) Adding new feasible arcs.

sponding arc $a_{k,i}$, followed by part of the pick-up κ_{i+1} in the next primary arc pa_i ([2] with quantity 1 in the example). The waste quantity collected in this added pick-up in $a_{k,i'}$ equals the free vehicle capacity in $a_{k,i}$ (0.5 for $a_{1,2}$ and 0.1 for $a_{0,2}$ in the example). Finally, an arc $a_{i',j}$ is added from each newly added node i' to the end node j of pa_i (or $a_{i,j}$). This arc contains a copy of the pick-up in primary arc $a_{i,j}$ with a reduced waste quantity. The latter is the waste quantity of the pick-up in $a_{i,j}$ minus the waste quantity added to the new arc $a_{k,i'}$. The new nodes and arcs and their corresponding pick-ups and loads are shown in Fig. 5 c2). After each division, the *MaxL* and *MinL* values are updated.

The arcs created during the PPSPLIT I division process could bring forth new feasible arcs, such as the dashed arcs in Fig. 5 d). The two new arcs $a_{9,4}$ and $a_{10,4}$ can be combined with the primary arc $a_{4,5}$. Furthermore, a third arc can be added which combines the new arc $a_{9,5}$ with primary arc $a_{5,6}$. After each division, we search for such potentially new feasible arcs (w.r.t. the vehicle capacity), after which again the *MaxL* and *MinL* values are updated.

It could occur that during the procedure, an arc is created with two visits to the same pick-up location. The procedure then merges these two pick-ups to the first position. The collected waste quantities are summed and assigned to the first occurrence in the pick-up sequence, while the second occurrence is removed. We continue the procedure in the same manner as SPLIT I: Processing facilities are now inserted, the shift duration constraint is checked for each arc and a shortest path optimization is executed to find a set of subroutes which minimizes costs.

SPLIT II

In the SPLIT II-procedure, the subroute sequence, including already assigned processing facilities, is split into a sequence of routes and each route is assigned a depot. Fig. 6 provides a small example of the procedure in which the resulting subroute sequence of the example in Fig. 4 (see also the first table in Fig. 6) is split into the sequence of routes presented in the second table in Fig. 6. We create a second auxiliary graph $HII = (\mathcal{V}, A, C)$ with $\mathcal{V}$ containing a source 0 and a node for each subroute. A contains arcs $a_{i,j}$ with $i < j$ if a vehicle can perform subroutes σ_{i+1} to σ_j without surpassing the maximum shift duration.

In Fig. 6, we show how *HII* (Fig. 6 c)) is created. First, in Fig. 6 a), we add an arc $a_{i,j}$ to A for each subroute in the sequence resulting from (PP)SPLIT I. Second, the depot with the shortest sum of the travel times from the depot to the first pick-up location in the first (and only) subroute and from the last processing facility to the depot is selected for the route associated with arc $a_{i,j}$. The selected depots for each arc are added in Fig. 6 b). Note that these arcs are all feasible w.r.t. the maximum shift duration. This ensures that the sink of *HII* can be reached from its source. Third, in Fig. 6 b), we also add arcs for routes, comprising of sequences of subroutes, for which depots are selected and whose feasibility w.r.t. the shift duration are checked.

An arc's cost $c_{i,j}$ consists of the duration of the route represented in $a_{i,j}$ (duration of the subroutes and the travel time from and to the selected depot), multiplied with the cost per unit of driving time (e.g., cost per hour). Also, the cost of one vehicle (IC) is added to $c_{i,j}$ to account for the use of one vehicle driving this route. This arc-cost can be adjusted if alternative cost-functions are used.

We want to identify the lowest-cost sequence of arcs (i.e. the lowest-cost set of routes) linking the source of *HII* to the sink, which is also feasible w.r.t. the number of vehicles available. We, therefore, apply the Bellman-Ford K-shortest path algorithm on the acyclic graph *HII* with K the total number of available vehicles (i.e. the sum of the depot fleet capacities: $K = \sum_{g=1}^{d} Q_g^D$). Using the Bellman-Ford algorithm, this K-shortest path can be found within $O(|\mathcal{V}^2|)$ time (Cormen et al., 2001). As opposed to Dijkstra's algorithm, Bellman-Ford keeps track of the shortest path between two nodes with up to k arcs ($k \in \{1..|V|\}$). We use this feature by preferably selecting the shortest path with up to K arcs. If such a path does not exist (i.e., the sequence of subroutes does not allow a split into K routes or less while respecting the maximum shift duration, the shortest path with the lowest number of arcs possible ($k > K$) is selected. Therefore, feasibility w.r.t. the number of available vehicles is aimed for, but not guaranteed.

Finally, the shortest path, in bold in Fig. 6 c) represents the routes that will be performed by the vehicles. The second table in Fig. 6 shows the three selected routes. Thus, by sequentially executing the (PP)SPLIT I and SPLIT II procedures on an individual's chromosome, we find a set of routes, satisfactory w.r.t. the costs incurred, and which are feasible with respect to the vehicle capacity and the maximum shift duration.

4.3. Initial population generation

The creation of an initial population entails a simple process which is closely related to the heuristics often applied to SDVRPs

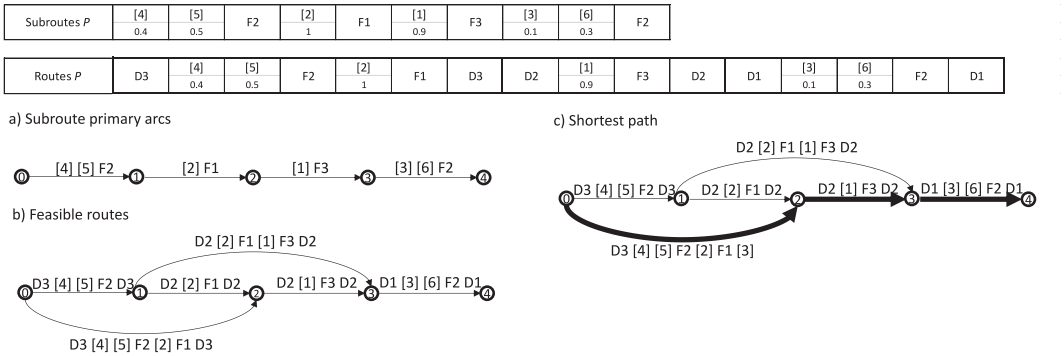


Fig. 6. SPLIT II; the chromosome of individual P and the created subroutes and routes; (a) Primary route arcs feasible w.r.t. vehicle capacity and shift duration; (b) Additional feasible route arcs and insertion of the depots; (c) Shortest path optimization establishing the best routes.

in which the demand (or waste quantity) at a node divided into several full truckloads and one "residual" truckload with a quantity less than the truck's capacity (see e.g., Archetti et al., 2006). An initial population is generated equal to four times the minimum population size. For each individual we create a chromosome for which we can guarantee that a feasible solution exists w.r.t. the vehicle capacity and the maximum shift duration as follows.

First, if the waste that is to be collected at a pick-up location exceeds the maximum vehicle capacity, the waste at the location is decomposed into an integer number of pick-ups of maximum vehicle capacity, taking into account the maximum shift duration, and the residual quantity. Then, we calculate for each pick-up location the largest truckload for which the vehicle capacity and the maximum shift duration constraints are met if only this pick-up is performed in a route. For this route, we select the processing facility and depot combination which results in a route with the shortest duration.

Thus, a solution in which each route contains only one pick-up will be feasible w.r.t. the vehicle capacity and the maximum shift duration. As mentioned in Section 4.2, this ensures that the primary arcs in (PP)SPLIT I are always feasible w.r.t. vehicle capacity and shift duration, which allows the sinks of *HI* and *HPPI* to be reached from their respective sources.

Second, an individual's chromosome is built by generating a permutation of these pick-ups. The chromosomes of half the initial population is generated in a random order, while for the other half, a nearest neighbour heuristic is applied. The latter randomly selects a first pick-up in the permutation and consecutively adds the closest (in terms of travel time) remaining pick-up to the previously appended pick-up. This ensures the creation of both satisfactory good and diverse initial solutions in the initial population.

4.4. Solution evaluation

Individuals are evaluated and compared to one another based on their biased fitness. The biased fitness of an individual is a summary of three distinct elements: (a) its general fitness ($\psi(P)$) (i.e. driving cost using the cost function), (b) its level of feasibility and (c) its diversity compared to other individuals in the population. An individual's general fitness is determined by the operational cost (operational cost per hour multiplied by hours driven) and the fixed vehicle cost (fixed cost per day multiplied by the number of vehicle routes in the solution).

Vidal et al. (2012) introduce a penalty for infeasible solutions with regards to the vehicle capacity and maximum shift duration, which is added to the general fitness of the individual to come up with a fitness level $\varphi(P)$ (lower values are better). The RWCP, additionally includes depot capacity constraints (number of vehicles) and processing facility capacity constraints (amount of waste per day). Although MASS creates solutions which are always feasible w.r.t. the vehicle capacity and the shift duration, penalties are required for the two constraints for which feasibility is not guaranteed: the depot and processing facility capacities. The adjusted function $\varphi(P)$ is presented in Eq. (1) with d the number of depots, $\beta(g)$ the number of vehicles leaving depot g, Q_g^D the maximum number of vehicles allowed to leave depot g, f the number of processing facilities, $\alpha(m)$, the amount of waste dropped off at processing facility m and Q_m^F, the capacity at facility m.

$$\varphi(P) = \psi(P) + \omega^D \sum_{g=1}^{d} max\left(0, \beta(g) - Q_g^D\right) + \omega^F \sum_{m=1}^{f} max\left(0, \alpha(m) - Q_m^F\right) \tag{1}$$

$$\xi(P, P2) = \sum_{i=1}^{n} \left\{ \sum_{j \in H(P,i)} \mathbb{1}(j \in H(P2, i)) + \sum_{j \in H(P2,i)} \mathbb{1}(j \in H(P, i)) \right\} \tag{2}$$

$$\Xi(P) = \frac{1}{n_{close}} \sum_{P_i \in \Pi_{close}} \xi(P, P_i) \tag{3}$$

$$fit(P) = rank(\varphi(P)) \tag{4}$$

$$dc(P) = rank(\Xi(P)) \tag{5}$$

$$BiasedFitness(P) = fit(P) + (1 - PercElite)dc(P) \tag{6}$$

The parameters ω^D and ω^F are initialized by dividing the average solution duration of the initial population by respectively the average number of vehicles surpassing the depot capacities or the average amount of dropped off waste surpassing the processor capacities. These parameters are updated during the iterative process of the metaheuristic. The update process depends on the metaheuristic parameters θ_D^{REF} and θ_F^{REF}, which are the desired proportion of feasible individuals in the population with regards to respectively the depot capacity constraints and the processing facility capacity constraints. If the proportion of feasible individuals in the population is higher than (is lower than) the desired proportion, the penalty for exceeding the respective capacity should be reduced (increased) to allow more (less) infeasible individuals to remain in the population. If the proportion of feasible individual solutions concerning the depot capacity (θ_D) exceeds $\theta_D^{REF} + 0.05$,

ω^D is reduced with 15%. If, however, θ_D is lower than $\theta_D^{REF} - 0.05$, ω^D is increased with 20%.[1] The same procedure is used for ω^F.

For the third evaluation element, diversity, we introduce a diversity function $\xi(P, P2)$ which calculates the similarity between individual P and individual $P2$. In Eq. (2) we determine for each pick-up location whether it is followed by the same pick-up locations in the chromosome of individual P and individual $P2$. In (2), n is the number of pick-up locations, $H(P, i)$ the set of pick-up locations following pick-up location i in the chromosome sequence of individual P and $\mathbb{1}$ the unit function. In Eq. (3), the total diversity contribution of individual P is calculated with n_{close} the metaheuristic parameter deciding on the number of closest individuals to consider and Π_{close} the set of n_{close} closest individuals (largest $\xi(P, P2)$). Functions $fit(P)$ and $dc(P)$ rank the individuals according to their fitness value and diversity contribution. An individual's biased fitness is then calculated in (6) with *PercElite* being a metaheuristic parameter determining the percentage of elite individuals out of all individuals one desires to transfer to the next generation. If the latter parameter is close to 1, the algorithm prefers lower-cost and feasible solutions over more diverse solutions. A *PercElite* closer to 0, however, will create more diverse individuals, which are not necessarily good w.r.t. costs and feasibility.

4.5. Local search moves

The genetic algorithm is complemented with several local search moves which are described below (M1-M15). We included the classic local search moves used in VRPs which are presented in Vidal et al. (2012) and Prins (2004): insertions, swaps, and 2-opts (M1-M7). We add to these non-partial-pick-up (NPP) moves, partial-pick-up (PP) adjustments. Derigs et al. (2010) offer alternatives for insertions (M8-M10) (called relocates) and 2-opt moves (M7). In the latter, the 2-opt move is extended with merging of multiple visits at the same pick-up location within a route. Boudia et al. (2007) present a PP alternative to the swap move (called customer exchange) (M11).

Additionally, the improved K-split procedure (M12) provided by Boudia et al. (2007) and the route-reduction (M13) procedure by Silva et al. (2015), originally developed for the SDVRPS, are adapted for the RWCP. The K-split procedure removes all pick-ups at a specific pick-up location from the individual's chromosome. It then inserts these pick-ups into the chromosome at positions (or insert locations) using a greedy heuristic, inspired by the well-known solution approach for the Knapsack problem. Each insert location receives an insertion price equal to the increased route duration caused by the insertion divided by the waste quantity inserted at the insert location. Insertions are performed at the lowest insertion price, thus favouring pick-ups at insert locations with a high waste quantity and a limited impact on the route's duration. The adjusted procedure is described in Algorithm 2 in Appendix B.

As reducing the number of required vehicles (i.e., the number of routes) decreases costs, a route-reduction procedure is introduced (see, for instance, Nagata, Bräysy, & Dullaert, 2010 and Bräysy, Dullaert, Hasle, Mester, & Gendreau, 2008). In the route-reduction procedure (M13), the solution's shortest route is identified and the pick-ups performed along this route are inserted in other existing routes. To find appropriate locations to insert these pick-ups, a K-split procedure is performed on each pick-up location visited in this route. The pick-ups are not allowed to be put back into this shortest route, thus emptying the route. Lastly, we use two moves to shift solutions towards full feasibility for the processing facility (M14) and depot (M15) capacity constraints.

[1] These percentages were also used for penalizing infeasibilities w.r.t. the vehicle capacity and the maximum shift duration in Vidal et al. (2012).

The moves are adjusted to the use of routes and subroutes in this paper. For instance, not only can pick-ups be inserted between two other subsequent pick-ups, but also between a pick-up and a visit to a processing facility. Furthermore, in order to maintain the solutions' feasibility with regards to vehicle capacity and shift duration, most moves need to be subjected to a feasibility check. The vehicle capacity check is performed on the subroute-level, while feasibility w.r.t the shift duration is checked on the route-level. If the adjusted solution is not feasible in terms of these two constraints, the move is not executed. If, after executing a move, a subroute contains multiple visits to the same pick-up location, these pick-ups are merged. Each move is evaluated with respect to the fitness level $\varphi(P)$ of the new solution of individual P. If the move improves $\varphi(P)$, the move is executed, if not, the original solution is kept.

To describe the local search moves, the following notation is used. Take two pick-ups κ_i and κ_j in individual P's solution. Define $route(\kappa_i)$ and $subroute(\kappa_i)$ as the route $\gamma_r \in \Gamma, r \in \{1..|\Gamma|\}$ and subroute $\sigma_s \in \Sigma, s \in \{1..|\Sigma|\}$ in which pick-up κ_i is executed. Define $Pred(\kappa_i)$ and $Succ(\kappa_i)$ as the part of $route(\kappa_i)$ respectively preceding and succeeding κ_i. $MaxQ(l, \kappa_i)$ denotes the maximum waste quantity that can be collected at pick-up location $l \in \{1..n\}$, if we would insert a pick-up to l after κ_i in $subroute(\kappa_i)$ while respecting Q^{CV} and T^S. $MaxQ(l, \sigma_s)$ denotes the maximum waste quantity that can be collected at pick-up location $l \in \{1..n\}$, if we would insert a pick-up to l at the front of σ_s while respecting Q^{CV} and T^S. The following list details each move. The moves listed below are followed with (Q^{CV}) and/or (T^S) if they need to be checked w.r.t. the vehicle capacity Q^{CV} and/or shift duration T^S constraints.

- M1: Remove κ_i from $subroute(\kappa_i)$, insert it after κ_j. (Q^{CV}, T^S)
- M2: If $subroute(\kappa_i) = subroute(\kappa_{i+1})$, remove $\{\kappa_i;\kappa_{i+1}\}$ from $subroute(\kappa_i)$ and insert $\{\kappa_i;\kappa_{i+1}\}$ after κ_j. (Q^{CV}, T^S)
- M3: If $subroute(\kappa_i) = subroute(\kappa_{i+1})$, remove $\{\kappa_i;\kappa_{i+1}\}$ from $subroute(\kappa_i)$ and insert $\{\kappa_{i+1};\kappa_i\}$ after κ_j. (Q^{CV}, T^S)
- M4: Swap κ_i and κ_j. (Q^{CV}, T^S)
- M5: If $subroute(\kappa_i) = subroute(\kappa_{i+1})$, swap $\{\kappa_i; \kappa_{i+1}\}$ and κ_j. (Q^{CV}, T^S)
- M6: If $subroute(\kappa_i) = subroute(\kappa_{i+1})$, swap $\{\kappa_{i+1}; \kappa_i\}$ and κ_j. (Q^{CV}, T^S)
- M7: If $route(\kappa_i) \neq route(\kappa_j)$ set $route(\kappa_i) = \{Pred(\kappa_i), \kappa_i, Succ(\kappa_j)\}$ and $route(\kappa_j) = \{Pred(\kappa_j), \kappa_j, Succ(\kappa_i)\}$. Each route however keeps its original start and end depot. (Q^{CV}, T^S)
- M8: Insert κ_i after κ_j with the maximum amount of $q(\kappa_i)$ for which $subroute(\kappa_j)$ remains feasible w.r.t. Q^{CV} and T^S. If $q(\kappa_i)$ is transferred entirely to $subroute(\kappa_j)$, then remove κ_i from $subroute(\kappa_i)$, otherwise, reduce $q(\kappa_i)$ in $subroute(\kappa_i)$ with the quantity added in $subroute(\kappa_j)$.
- M9: If $subroute(\kappa_i) = subroute(\kappa_{i+1})$, insert $\{\kappa_i; \kappa_{i+1}\}$ after κ_j, first with the maximum amount of $q(\kappa_i)$, second, with the maximum amount of $q(\kappa_{i+1})$ for which $subroute(\kappa_j)$ remains feasible w.r.t. Q^{CV} and T^S. If $q(\kappa_i)$ or $q(\kappa_{i+1})$ is transferred entirely to $subroute(\kappa_j)$, remove respectively κ_i and κ_{i+1} from $subroute(\kappa_i)$, otherwise, reduce the waste quantities in $subroute(\kappa_i)$ accordingly.
- M10: See M9, but first insert κ_{i+1} after κ_j.
- M11: Assume $q(\kappa_i) > q(\kappa_j)$. Divide κ_i into $\kappa_{i'}$ and $\kappa_{i''}$ with $q(\kappa_{i'}) = q(\kappa_j)$ and $q(\kappa_{i''}) = q(\kappa_i) - q(\kappa_j)$. Replace κ_i in $subroute(\kappa_i)$ with $\kappa_{i''}$. Insert κ_j behind $\kappa_{i''}$ and insert $\kappa_{i'}$ at κ_j's previous position. (T^S)
- M12: Perform the K-Split procedure (see Algorithm 2) on all pick-up locations $l \in \{1..n\}$.
- M13: Select the route $\gamma_r \in \Gamma$ with the shortest duration. Perform the K-split procedure (Algorithm 2) on each $pul(\kappa_i) | \kappa_i \in \gamma_r$.

- M14: Go over all subroutes $\sigma_s \in \Sigma, s \in \{1..|\Sigma|\}$, select the processing facility $m \in \{1..f\}$ with the largest available capacity. Set the processing facility of σ_s to m. (T^S)
- M15: Go over all routes $\gamma_r \in \Gamma, r \in \{1..|\Gamma|\}$, select the depot $g \in \{1..d\}$ with the largest available capacity. Set the depot of γ_r to g. (T^S)

The local search phase is executed in three steps. First, the algorithm executes the neighbourhood search (NS) phase in which three categories of moves are performed in a random order. In the first category, the procedure goes over all pick-ups in the solution. For each pick-up κ_i a move MX (M1-M11) is randomly selected. Within this move, the procedure traverses the other pick-ups κ_j ($i \neq j$). Toth & Vigo (2003) show that restricting the neighbourhood moves to promising elements can speed up the computation time, while still finding good feasible solutions. Specifically, the procedure only traverses pick-ups of the nh closest pick-up locations (shortest travel time) with $h \in [0, 1]$ the granularity threshold which restricts the search to nearby pick-ups. The algorithm checks whether a move with κ_j would improve the solution's fitness value. Once a κ_j resulting in an improvement is found, the move is executed. The procedure then randomly selects a new move MX (M1-M11) and continues in the same manner. If a move does not improve the fitness value, it is taken out of the set of available moves and is labeled "unavailable". Once an improvement is found, all moves are relabeled "available". When all moves (M1-M11) have been made "unavailable", we continue to the next pick-up κ_{i+1}. The second and third category of moves respectively consist of the K-split (M12) and the route-reduction (M13) procedure.

Second, in the depot-processing facility improvement (DPI) phase, the algorithm tries to select alternative processing facilities (M14) and alternative depots (M15) to reduce costs and improve the solution's feasibility w.r.t. the depot and processing facility capacities. Third, if necessary, a repair phase is executed. If the individual's solution is not feasible w.r.t. the depot or processing facility capacities and if the proportion of feasible individuals w.r.t. the respective constraints is lower than the reference proportions θ_D^{REF} and θ_F^{REF}, the algorithm tries to "repair" the solution. While the first NS-DPI sequence is performed with the normal feasibility penalties as described in Section 4.4, the penalties in the repair phase are multiplied by 1000. The NS-DPI sequence is then executed with these augmented penalties.[2]

After performing the local search procedure on the solution of an individual, the chromosome of the individual is updated to reflect the correct sequence of pick-ups. When all individuals have gone through the local search process with probability $Prob^{ls}$, each altered individual's biased fitness is recalculated.

4.6. Flexibility of the memetic algorithm with sequential split procedure

The RWCP is a generalization of many, more specific, waste collection problems. MASS is designed in a way that allows omitting or adapting some of MASS's components so that it may be used for solving these more specific versions of the RWCP. For instance, if partial pick-ups are not allowed, the simple version of SPLIT I should be used instead of PPSPLIT I, and only the local search moves (M1-M7) for full pick-ups should be included. Alternative constraints such as a fleet-wide depot capacity or a total processing capacity can be included by adapting the penalty functions.

[2] Vidal et al. (2012) use a repair rate to decide whether a solution needs to be repaired. MASS, however, determines this endogenously which reduces the number of parameters to calibrate. This also assists MASS in reaching the desired proportion of feasible solutions in the population.

MASS can also be used to solve more specific vehicle routing problems studied outside the waste collection setting, such as the (MD)VRPIF, the MDVRPI and the MDVRP, if small adjustments are made in the SPLIT II procedure and the local search moves. As opposed to instances of the RWCP, instances of these problems do not require a final visit to an intermediate/processing facility before returning to the depot. The last subroute in a vehicle's route must therefore receive a processing facility with a location equal to that of the vehicle's depot. To remove this so-called empty-return constraint, an additional processing facilities set $\mathcal{V}^{FAC'}$ is generated with the same locations as the depots and zero drop-off times. In SPLIT II, MASS chooses a depot resulting in the shortest travel time to the first pick-up and from the last pick-up in the route. The last subroute will receive a "mock"-processing facility with a location equal to that of the selected depot. In a MDVRP, a vehicle can only return to its own depot for a drop-off (waste collection setting) or a replenishment (general VRP setting). Therefore, the location of a vehicle's depot and the locations of the processing facilities it will visit must be the same. To ensure this, the processing facilities in $\mathcal{V}^{FAC}$ will have the same locations as the depots. Additionally, in SPLIT II, the location of the depot and the processing facilities is determined simultaneously by selecting the location resulting in the shortest arc duration. Special care must be given in the local search whenever depots and processing facilities are altered: M7, M14 and M15. An additional check is required to ensure that the location of the last subroute's processing facility remains the same as that of the depot ((MD)VRPI(F)) or that the locations of the processing facilities and the depot in a route are identical (MDVRP).

5. Computational experiments

Experiments were run to test the performance of the proposed MASS. First, the metaheuristic parameters are calibrated on a real-life case. Second, we test how the solution approach performs in three established problems from the literature: the (MD)VRPIF, the MDVRPI and the MDVRP. Third, test instances for the RWCP are created and tested. Fourth, MASS's solution quality is investigated by comparing it with an exact approach for a small case study. Last, a sensitivity analysis is conducted on the building blocks of MASS by systematically leaving out a specific part of the algorithm.

The algorithm was coded in C++. Each instance was run on the Genius High Performance Cluster of the Flemish Super Computer Center (2 Xeon Gold 6140 CPUs@2.3 GHz, Skylake, Processor Base Frequency 2.30 GHz, Max Turbo Frequency 3.70 GHz, 18 cores (36 threads) per node) with 12 threads per instance.

5.1. Parameter calibration

We calibrate MASS based on real-life instances of the case study detailed in Section 6. Four instance types are tested: 1 depot and 1 processing facility (1D1P); 1 depot and multiple processing facilities (1DmP); multiple depots and 1 processing facility (mD1P); and multiple depots and multiple processing facilities (mDmP). Eight heuristic parameters are calibrated: the child population size λ; the maximum population size μ; the parameter *PercElite* deciding on the relative importance of good feasible solutions and the diversity of the solutions; the number of closest individuals to be considered for calculating the diversity contribution (n_{close}), which is calculated by multiplying parameter *PercClose* with μ; the granularity threshold h used in the local search; the desired proportion of feasible individuals w.r.t. the capacity of the depots θ_D^{REF} and the processing facilities θ_F^{REF}; and the local search rate $Prob^{ls}$.

A systematic approach for fine-tuning the heuristic's parameters is advised by Sörensen (2015). We therefore employ the systematic procedure proposed by Coy, Golden, Runger, & Wasil (2000) which

Table 1
Parameter calibration results.

Parameter	1D1P	1DmP	mD1P	mDmP
λ	23	29	24	59
μ	50	61	52	49
PercElite	0.9208	0.4469	0.4500	0.8099
PercClose	0.8000	0.7308	0.4813	0.4764
h	0.4653	0.4735	0.3906	0.2095
θ_F^{REF}	n.a.	0.5797	n.a.	0.3050
θ_D^{REF}	0.7000	0.4000	0.3775	0.4926
$Prob^{ls}$	0.8523	0.9092	0.9224	0.6000

is based on statistical design and gradient descent. A detailed description of how the procedure is applied, accompanied by the intermediate results of the approach can be found in Appendix C. Table 1 presents the final parameter settings for each of the four instance types.

The calibrated child population size, granularity threshold and local search rate are quite similar for the 1D1P, 1DmP, mD1P case, but differ for the mDmP case. These three parameters affect how many iterations can be preformed. Lower values increase the number of iterations, while higher values result in fewer iterations for a fixed amount of computation time. Notice that in mDmP, the impact of a higher child population size is combined with a lower local search rate and granularity threshold. This reduces the number of individuals being improved using local search and shrinks the neighbourhood space for individuals being improved. Thus, a trade-off exists between these three parameters.

The parameters *PercElite* and *PercClose* are generally lower if the number of depots and processing facilities in the instance increases. Lower levels of *PercElite* indicates a need for more diversity in de population for cases with multiple depots or processing facilities. Lower levels of *PercClose* could be driven by a need for speeding up the algorithm as fewer calculations are required when comparing an individual's diversity with a limited number of close individuals.

If only one depot is used, the impact of θ_D^{REF} is rather limited. For cases with multiple depots, however, a θ_D^{REF} around 40% seems appropriate. For the parameter θ_F^{REF}, a lower value should be used if also multiple depots are included. This could be explained by a need for additional flexibility regarding the feasibility of solutions to allow the algorithm to guide the search in the right direction of the search space. The parameter *DivIt*, representing after how many iterations without finding an improved solution the algorithm should reinitialize the population, is adjusted endogenously, similar to Vidal et al. (2012), by setting it to 0.3*MaxIt*. The latter parameter is estimated by recording the average calculation time per iteration and using this value to determine how many iterations the algorithm will be able to run within the allowed timespan.

5.2. Test instances (MD)VRPIF, MDVRPI, MDVRP

To evaluate MASS's ability to achieve high-quality solutions, we test its performance on the VRP with intermediate facilities (VRPIF), the multi-depot VRPIF (MDVRPIF) and the multi-depot VRP with inter-depot routes (MDVRPI) using variations of two sets of test instances: a1-l1 and a2-j2, provided by Crevier et al. (2007). Although Crevier et al. (2007) label these two sets of test instances "multi-depot", they only contain a single depot for the vehicles. The multiple depots are, in these original instances, multiple intermediate facilities which makes them instances of the VRPIF. The sets are solved by Crevier et al. (2007), Tarantilis et al. (2008), Markov et al. (2016) and Hemmelmayr et al. (2013). Crevier et al. (2007) developed a three-phase methodology (CCL algorithm) which first generates a set of routes using a combination of Tabu Search and Adaptive Memory. The routes are then assigned to vehicles using a set partitioning algorithm. Tarantilis et al. (2008) use the first set of instances (a1-l1) to test their algorithm in which a metaheuristic employs Tabu Search within a Variable Neighbourhood Search framework after which a Guided Local Search is performed (H-GLS algorithm). While Markov et al. (2016) introduce a Multiple Neighborhood Search heuristic (MNS algorithm) for the VRPIF, Hemmelmayr et al. (2013) solve the test instances using a hybrid solution method based on Variable Neighborhood Search and Dynamic Programming (VNS algorithm).

A first variation of the instances was created by Markov et al. (2016). The intermediate facilities now also serve as depots, creating test instances with more than one depot for the collection vehicles. Furthermore, a vehicle is allowed to end its route at a depot that differs from the depot it departed from. This is modelled in MASS as a MDVRPIF with a final visit to an intermediate facility before returning to the depot, combined with a zero-travel time between the processing facilities and the depots. Ramos et al. (2020) made a second variation of the instances in which all intermediate facilities can also act as depots and vice versa to test their matheuristic approach (MAT) for the MDVRPI. Note that the "central depot" of the original test instances can now also act as an intermediate facility, which is not the case in the variation by Markov et al. (2016). A second set of test instances, solved by Ramos et al. (2020) and Muter et al. (2014), was omitted from our analysis as these instances contain smaller sets of customers, which is less realistic for waste collection problems. We also compare our MASS with the 32 test instances for the MDVRP reported by Vidal et al. (2012). A description of these test instances is given in Appendix D.

Due to the generic and flexible nature of MASS, alternative metaheuristic parameter settings could be appropriate for different, more specific cases of the RWCP. We do not determine dedicated parameter settings for each of the problem instances and instead apply the settings provided by Vidal et al. (2012). Furthermore, MASS was tailored to solve the instances of the different VRP types, as detailed in Section 4.6. In order to check the robustness of the algorithm, we ran each instance 10 times using different input seeds, thus creating ten different initial populations. We record the lowest-cost feasible solution found after one hour and after running the algorithm for the same amount of time as Hemmelmayr et al. (2013), Markov et al. (2016), Ramos et al. (2020) and Vidal et al. (2012) for respectively the original instance set (a1-l1, a2-j2), variation 1 (MDIFa1-MDIFl1, MDIFa2-MDIFj2), variation 2 (MDIa1-MDIl1, MDIa2-MDIj2) and the MDVRP instances.

Table 2 presents the average percentage difference (over the different instances) between the average results (over the different runs) reported by previous studies and the average results after running MASS. To check the robustness of the algorithm, Table 2 also presents the 95%-confidence intervals of the percentage differences between the result of each instance-run (10 runs per instance) for MASS and the results reported by previous studies applying alternative algorithms. The confidence intervals not containing zero can be considered as significant (at the 5% level).

The results presented in Table 2 show that, for instances with only a single central depot, MASS finds better results than the CCL algorithm and results similar to those of the H-GLS and the MNS algorithms. However, the solution method by Hemmelmayr et al. (2013) generates the best results. MASS achieves impressive results for the instance sets of the MDVRPIF. When MASS is run for the same amount of time as the MNS algorithm, better results are achieved and new best results are found for most instances. Ramos et al. (2020) ran their matheuristic for four or eight hours to solve the MDVRPI test instances. MASS is only run for one hour, but obtains similar results for the first instance set (MDIa1-MDIl1) and only slightly worse results for the second instance set (MDIa2-

Table 2
MASS performance on the test instances.

		Average and 95%-confidence interval of Δ %				
		MASS at T(VNS)	**CCL**	**H-GLS**	**MNS**	**VNS**
VRPIF	a1-l1	+1.26% [1.03,1.48]	-1.12% [-1.37,-0.87]	-0.06% [-0.20,0.09]	-0.67% [-0.85,-0.49]	+0.98% [0.79,1.16]
	a2-j2	+1.67% [1.44,1.90]	-1.08% [-1.40,-0.77]	n.a.	-0.09% [-0.40,0.22]	+1.90% [1.57,2.23]
		MASS at T(MNS)	**MNS**			
MDVRPIF	MDIFa1-MDIFl1	-0.83% [-0.96,-0.71]	-1.76% [-1.87,-1.64]			
	MDIFa2-MDIFj2	-0.95% [-0.97,-0.46]	-1.76% [-1.93,-1.59]			
		MASS at T(MAT)	**MAT**			
MDVRPI	MDIa1-MDIl1	n.a.[a]	+0.06% [-0.12,0.24]			
	MDIa2-MDIj2	n.a.[a]	+0.71% [0.38,1.05]			
		MASS at T(HGSADC)	**HGSADC**			
MDVRP	pr01-pr10	+2.89% [2.47,3.32]	-0.35% [-0.54,-0.16]			
	p01-p23	+4.02% [3.72,4.62]	+0.87% [0.71,1.03]			
	all	+3.67% [3.44,4.12]	+0.5% [0.36,0.64]			

[a] Ramos et al. (2020) ran their matheuristic for four or eight hours, while MASS was only run for one hour. We therefore cannot report results for MASS at T(MAT).

MDIj2). Moreover, it finds new best results for 15 out of the 22 test instances and matches the MAT results of Ramos et al. (2020) for two instances.

Although a direct comparison of computation time is difficult as different processors and a higher number of threads are used, the results indicate that MASS is capable of generating high-quality results. MASS is a competitive algorithm for solving waste collection problems with multiple processing facilities, full (as opposed to partial) pick-ups and one depot. But, as can be expected, the VNS algorithm by Hemmelmayr et al. (2013), which is dedicated to the VRPIF with a single depot, can find slightly better solutions. If, additionally, vehicles can depart from multiple depots, our computational experiments indicate that MASS is the most powerful algorithm available. For instances where depots can also act as intermediate facilities (MDVRPI), MASS achieves similar results as the best algorithm found in the literature with much shorter computation times.

MASS was constructed by adding elements to the HGSADC algorithm for MDVRPs by Vidal et al. (2012) for being able to solve a broad set of waste collection problems. Obviously, using MASS for solving a MDVRP is more computationally intensive than using the HGSADC as several unnecessary calculations are executed. Despite the fact that MASS is, by design, less efficient for such problems, the lowest-cost feasible result obtained after one hour is on average only 0.5% higher than the result obtained by Vidal et al. (2012). In addition, note that MASS performs better on the first ten instances presented in Table 2 than on the last 23 instances. The former set of instances include a service time at the customers, which is also typically the case for waste collection problems. Thus waste collection problems resembling a MDVRP including service times at the pick-up locations could still be solved using MASS. Note that good feasible solutions could probably be found faster using the HGSADC by Vidal et al. (2012).

5.3. Test instances rich waste collection problem

As no test instances could be found containing all elements of the RWCP, we generate 22 new test instances, which are described in Table 3. These new test instances are provided on the website http://www.vrp-rep.org. The instances are based on the six smallest random instances provided by Crevier et al. (2007) and Tarantilis et al. (2008) (a1, b1, d1, e1, g1 and j1) in which the customers to be visited are now regarded as pick-up locations.

First, we generate a set of instances for which partial pick-ups are logical. The vehicle capacity is reduced to the average waste quantity per pick-up location, which ensures that partial pick-ups are necessary at the pick-up locations with the largest waste quantities. Second, the number of available vehicles is adjusted through testing by ensuring that feasible solutions could be found for the instances while remaining sufficiently difficult to solve.

Third, for each of the six instances, four variations are created w.r.t. the number of depots (d) and the number of processing facilities (f): 1 depot - 1 processor (1D1P), 1 depot - multiple processors (1DmP), multiple depots - 1 processor (mD1P) and multiple depots - multiple processors (mDmP). The number and locations of the depots and processing facilities are based on the intermediate facilities of the Crevier et al. (2007) instances. For the mDmP instances, the intermediate facilities are divided evenly over the set of depots and the set of processing facilities. If the number is odd, we choose to assign more intermediate facilities to the processing facilities than the depots as Lavigne et al. (2021) found that increasing the number of processing facilities significantly increases the complexity of the waste collection problem, while this was not the case for increasing the number of depots.

Table 3
Results of the RWCP instances.

Inst.	Based on	Type	n	$Q^d_{tot}/d/f$	T^S	Q^{CV}	MASS (avg)	MASS (best)
rwc1	a1	1D1P	48	13/1/1	700	13	6865.40	6859.97
rwc2	a1	1DmP	48	10/1/2	700	13	5092.42	5054.06
rwc3	a1	mD1P	48	16/2/1	700	13	7910.17	7906.67
rwc4	b1	1D1P	96	14/1/1	1200	14	12227.45	12213.30
rwc5	b1	1DmP	96	12/1/2	1200	14	9853.26	9810.39
rwc6	b1	mD1P	96	16/2/1	1200	14	14876.58	14843.20
rwc7	d1	1D1P	48	15/1/1	600	15	6312.59	6308.66
rwc8	d1	1DmP	48	12/1/3	600	15	4558.69	4544.36
rwc9	d1	mD1P	48	15/3/1	600	15	5911.00	5909.08
rwc10	d1	mDmP	48	14/2/2	600	15	5535.11	5507.70
rwc11	e1	1D1P	96	16/1/1	1300	13	15620.97	15593.90
rwc12	e1	1DmP	96	12/1/3	1300	13	9574.86	9469.27
rwc13	e1	mD1P	96	18/3/1	1300	13	16067.28	16039.50
rwc14	e1	mDmP	96	14/2/2	1300	13	11902.19	11875.70
rwc15	g1	1D1P	72	17/1/1	750	14	9471.04	9468.13
rwc16	g1	1DmP	72	12/1/4	750	14	6472.54	6428.39
rwc17	g1	mD1P	72	16/4/1	750	14	8983.07	8972.53
rwc18	g1	mDmP	72	16/2/3	750	14	6957.30	6933.40
rwc19	j1	1D1P	72	15/1/1	800	12	8790.60	8785.18
rwc20	j1	1DmP	72	11/1/5	800	12	6171.24	6142.56
rwc21	j1	mD1P	72	20/5/1	800	12	9321.75	9312.16
rwc22	j1	mDmP	72	12/3/3	800	12	6290.72	6262.67

Table 4
Sensitivity analysis main component MASS.

	No ls	No pop	No cross	No inf
Δ % objective[5]	3.69%	0.57%	0.55%	0.09%
% feasible solution found	54%	100%	100%	55%

Fourth, we introduce capacity constraints for the depots and processing facilities by limiting the number of vehicles that can leave each depot and by limiting the amount of waste that can be dropped off at each location. The total number of vehicles and the total amount of waste to be collected are distributed evenly over the different locations. An additional 5% of the total waste quantity was added to the total processor capacity, which is distributed evenly over the different processing facilities. Finally, the objective function is altered by introducing a vehicle cost and including a service-time at the pick-up locations and a drop-off time at the processing facilities in the calculation of the operational cost. Each instance is run 10 times with a different seed for one hour with the appropriate calibrated parameters. The last two columns in Table 3 present the average and the best objective value found by MASS.

5.4. Sensitivity analysis

To test the performance of each component of MASS, we perform a sensitivity analysis in which each component is systematically disabled. The 22 new RWCP test instances are solved using each version of MASS (10 times with different seeds).

The four versions of MASS presented in Table 4 remove large parts of the algorithm. The “no ls”-version removes the local search component entirely, including the repair component. In the “no pop”-version, the child population size is put to 1. We take out the main component of the genetic algorithm in “no cross”, by removing the cross-over component. Thus, in this version, individuals are selected based on their biased fitness and are gradually improved through local search. The “no inf”-version eliminates all components regarding infeasibility penalties. The algorithm is therefore not guided towards feasible solutions.

Table 4 presents the impact on the objective value[3] and for how many instances the MASS version was able to find a feasible solution. Removing any component from MASS on average leads to worse results (higher costs). Moreover, the 95%-confidence intervals of the “no ls”, “no pop” and “no cross”-versions are entirely positive for the four subsets (1D1P, 1DmP, mD1P, mDmP). Especially removing the local search component significantly affects the performance of the algorithm. Furthermore, when the local search component or the components associated with penalizing infeasible solutions are removed, MASS is not able to find a feasible solution for almost half the instances. We thus conclude that the four components contribute to finding good feasible solutions.

We also test the performance of the local search moves by systematically removing a (set of) local search move(s). Table 5 shows that removing one local search component has a minor impact on the obtained objective values. The largest impact is caused by leaving out the 2-opt move. Leaving out the 2-opt move increases costs by 0.11% on average. Considering that the local search component significantly improves the performance of MASS, we can conclude that the local search moves are essential. However, as removing one (set of) move(s) does not highly impact the results, not all moves should necessarily be implemented. We provide some guidance on what moves are appropriate by reporting in Table 5 on the relative CPU-time spent per (set of) move(s) and inspecting the 95%-confidence intervals of the entire instance set and the four subsets.

Insert moves, swap moves and the route-reduction move have a relatively low CPU-usage. However, leaving out one such move does not significantly impact the overall results. Only for the instance type mD1P did we find a significant effect in which leaving out the respective move(s) would improve results slightly. Excluding the two-opt move generally has a significant negative effect on results. When we drill down to the subset level however, we see that this negative effect originates from negative effects on the multiple processing facilities subsets. For less complex instances with only a single processing facility, excluding the 2-opt move improves the results. The 2-opt move takes up a large amount of

[3] The objective value excluding the fixed operational cost associated with the service time at the pick-up locations.

Table 5
Sensitivity analysis local search.

	No insert	No insert split	No swap	No swap split	No 2-opt	No K-split	No route-reduction
Δ % objective[5]	-0.03%	-0.01%	-0.02%	-0.06%	0.11%	-0.06%	-0.03%
% CPU time in MASS	0.02%	0.10%	0.11%	7.77%	42.86%	24.28%	2.53%
(+) 95% conf. int.					all,1DmP, mDmP[a]		
(-) 95% conf. int.	mD1P	mD1P	mD1P	mD1P	1D1P, mD1P	mD1P, mDmP	mD1P

[a] 90% confidence interval.

Table 6
Comparison with an exact approach for a simplified case.

	CPLEX gap	MASS gap	Δ%
Average %	3.41%	4.25%	0.88%
Range %	[0.75,8.52]	[1.27,8.79]	[0.22,2.84]

CPU-time due to the high number of additional calculations required. In contrast with the insertion and swap moves, the 2-opt move impacts two entire routes instead of two subroutes. We hypothesize that for less complex cases, sufficient GA iterations can be executed which already ensures convergence to a local optimum. For complex cases however, only a limited number of GA iterations can be performed within the time limit (with or without 2-opt) and it seems that the powerful 2-opt makes more efficient use of the limited computational budget. Thus, although the two-opt move is quite CPU-intensive, it should be included for cases with multiple processing facilities.

The 95%-confidence interval of the "no K-split"-version for all instances includes zero. Upon closer inspection, we find that, for cases with multiple depots, excluding the K-split move would be beneficial. The K-split move has the potential to drastically alter a solution, which could result in the algorithm stepping over local minima of the other moves. Thus, for cases with multiple depots, we advise to carefully investigate the impact of the K-split move or omit this move.

5.5. Solution quality

In Lavigne et al. (2021), a simplified version of the real-life case presented in Section 6 was solved using an exact approach. The instances were aggregated by reducing the number of pick-up locations to allow CPLEX Optimization Studio, Version 12.10 (with the default settings and optimization strategies) to find good solutions within a reasonable computation time. Also, one additional restriction was added: waste collection vehicles were only allowed to collect waste from up to two different pick-up locations before dropping off their waste at a processing facility. To check the quality of the solutions found by MASS, this additional restriction is introduced in MASS. A subset of these aggregated instances are solved using MASS: two collection days for residual waste and bio-waste collection with either one or three depots and one or three processing facilities. A detailed description of these instances can be found in Appendix D.

Table 6 compares the results of MASS to the results obtained by the exact approach. The exact approach applies a branch-and-cut algorithm and, thus, can report a lower bound which is used to calculate the gaps of the two approaches. Table D.5 presents the results of both approaches and reports their respective gaps. MASS performs well as its results are on average only 0.88% larger than the results obtained using the exact approach. Also the gap remains rather low at, on average, 4.25%.

6. Case study: The Brussels Capital Region

Under the latest amendment to the EU waste directive of 2018, all member states are required to collect and treat biological waste (bio-waste) separately by December 2023. Due to this obligation, the Brussels Capital Region (BCR) wants to investigate alternative ways for collecting and treating municipal bio-waste in the near future. Currently, municipal bio-waste is collected in two separate fractions: green (or garden) waste and food (or kitchen) waste. Green waste is only collected separately in specific areas of the BCR. Food waste collection services are offered throughout the region, but participation remains voluntary. Only a small (9%, 2014 data) fraction of the municipal food waste generated in the BCR is collected and treated separately. The lion's share is collected together with residual waste and is incinerated with energy recovery in a local incineration facility. Green waste is composted in a facility located within the BCR's borders (Brussels Compost), while food waste is collected in a transfer station at the border of the BCR and transported to a bio-gas plant 130 km away from the BCR.

Residual, food and green waste are collected door-to-door in bags. The BCR's waste collection services also offer bin-collection for food and residual waste in apartment buildings, offices, schools, etc. Of the former two fractions, 30% is collected in bins and 70% in bags. The BCR's waste collection fleet is stationed at multiple depots. Most fractions (and their collection type, i.e. bins or bags) have one dedicated depot from which the collection vehicles leave. The bag-collection of residual waste, however, is done by vehicles stationed at three different depots.

The scenarios under investigation are based on the treatment scenarios proposed by Bortolotti, Kampelmann, Muynck, Papangelou, & Zeller (2019) and differ with respect to three distinct components. First, the BCR's authorities are interested in the impact of an increased food waste collection rate. What would be the impact on costs and the required fleet size if 30 kt of food waste (scenario 1), instead of the projected 17 kt (scenario 0), would be collected separately from residual waste in 2025?

Second, if the BCR would be able to convince its citizens to separately collect 30 kt food waste, alternative treatment investments within the BCR's border could become viable options. Two investment alternatives are under investigation. Under scenario 1A, the current treatment option is retained. In scenario 1B, three new composting sites are installed, with one located at the old composting facility, while under scenario 1C, a bio-gas facility is installed in the BCR. In the latter two scenarios, the new installations will be able to process both food and green waste.

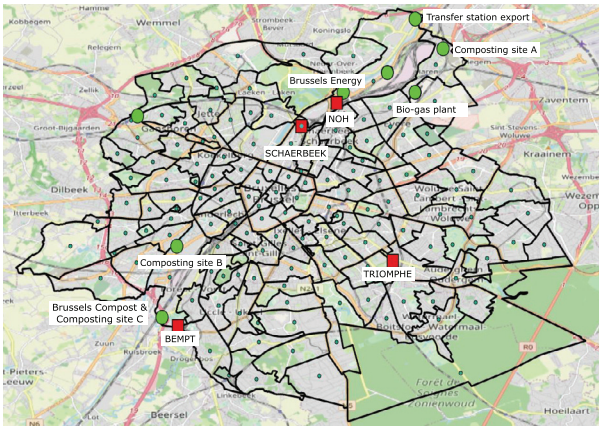
Third, as the new installations are able to process both food and green waste, the two fractions could be collected jointly combined with, if necessary, post-collection separation for managing the food-green waste proportion. Thus, for the two scenarios in which both fractions can be treated together, both a separate and a joint collection is simulated.

This case study can be modelled as a RWCP. The pick-up locations correspond to the 145 districts in the BCR. In Fig. 7, the BCR is divided into districts with their centres of gravity in blue. Also the potential processing facilities (green circles) ($f = 1$ or 3) and the depot locations (red squares) ($d = 1$ or 3) are shown. The waste quantities per pick-up location were derived from data provided by the BCR's waste collection authorities. The collection speed was determined for each district based on the population density, which affects how fast a collection vehicle can drive through the streets, and the distance that needs to be trav-

Table 7
Scenarios and results for the BCR case study.

Scenario		Waste type	Weight (kt)	Processing facility	Collection duration (h)	Truck days	Investment costs[a]	Operational costs[a]	Total costs[a]	Scenario costs[a]
0		residual	328	Brussels Energy	106,129	17,799	271,969	5,837,095	6,109,064	9,423,254
		food	17	Export	40,696	6,255	95,583	2,238,307	2,333,891	
		green	14.5	Brussels Compost	17,067	2,723	41,615	938,685	980,300	
1		residual	315	Brussels Energy	104,489	17,608	269,043	5,746,901	6,015,944	
	1A	food	30	Export	41,162	6,415	98,014	2,263,902	2,361,916	9,358,159
		green	14.5	Brussels Compost	17,067	2,723	41,615	938,685	980,300	
	1B(s)	food	30	3 composting plants	39,498	6,229	95,178	2,172,374	2,267,552	9,276,180
		green	14.5		17,292	2,723	41,615	951,069	992,684	
	1C(s)	food	30	Bio-gas plant	41,433	6,441	98,419	2,278,820	2,377,239	9,450,330
		green	14.5		18,412	2,910	44,471	1,012,677	1,057,148	
	1B(j)	joint	44.5	3 composting plants	40,633	6,808	104,021	2,234,842	2,338,863	8,354,807
	1C(j)	joint	44.5	Bio-gas plant	42,804	6,914	105,640	2,354,222	2,459,862	8,475,806

[a] The cost parameters represent the trade-off between the daily investment cost of a vehicle and the hourly operational cost of collecting waste. This trade-off was provided by the Brussels' waste collection authorities as no real cost parameters could be disclosed.



[a] Map exported from OpenStreetMap contributors (2019).

Fig. 7. Districts, (potential) processing facilities and vehicle depots in the Brussels Capital Region.[a].

elled in each district. The latter was calculated using the GIS-software Arcgis Pro (ESRI, 2019), road maps by OpenStreetMap (OpenStreetMap contributors, 2019), and UrbIS-Adm GIS-data provided by the Brussels Regional Informatics Centre (2019). As actual cost data could not be disclosed, appropriate fictitious investment and operational costs were used that represent a realistic trade-off between the two cost types. This trade-off was provided by the BCR's waste collection authorities.

We minimize the collection costs of each instance, comprising of a scenario (waste quantity, joint or separate collection, processing location(s)), waste type, collection type (bag or bin) and collection day of the week. Note that the costs associated with the treatment of waste are not included. Furthermore, the reported collection costs only comprise the costs associated with collection within the borders of the BCR. The cost of exporting waste is excluded. A system-wide study in which these three elements (collection costs within the BCR, export costs, and treatments costs) are compared, is left for a future study. A representative week in the summer and a representative week in autumn were optimized. The annual results in Table 7 were calculated by taking the average values of the two optimized weeks and multiplying them by 52 weeks.

The results in Table 7 show that if more food waste could be collected separately, the cost reductions attained for residual waste collection outweigh the increased cost for food waste collection. This can be explained by the relatively low quantities for food waste. The collection vehicles are mostly constrained by their shift duration as they arrive at a processing facility with still some free capacity.

Of the three scenarios concerning the processing facilities (export to Ypres, the bio-gas plant, three composting facilities), the option with three new composting sites would result in the lowest waste collection costs. Furthermore, joint collection of food and green waste (scenarios indicated with (j) in Table 7) would result in major cost savings for the BCR's waste collection services as bio-waste collection costs could be reduced by at least 28% ((Total costs 1C(j)-Total costs 1C(s))/Total costs 1C(s) =(2,459,862-2,377,239-1,057,148)/(2,377,239+1,057,148)).

7. Conclusions

Although many studies have been conducted on solving waste collection problems, a general solution approach that can be applied in a wide range of cases is lacking. Inspired by a real-life case, this paper introduces the Rich Waste Collection Problem (RWCP), a general model for waste collection cases with multiple depots, a restriction on the fleet residing at each depot, multiple (intermediate) processing facilities, capacity restrictions per processing facility, and partial pick-ups which allows collection nodes to be visited multiple times.

Moreover, this paper provides a Memetic Algorithm with Sequential Split (MASS) solution procedure for solving RWCPs. The algorithm includes a novel approach for finding satisfactory initial solutions which are guaranteed to be feasible w.r.t. the maximum shift duration of the waste collectors and the capacity of the collection vehicles. These satisfactory solutions are then improved through local search. The local search is based on existing local search operators that were adjusted to accommodate either full or partial pick-ups. Furthermore, we investigated the performance of each move by comparing their impact on the final results and the required computational effort.

We additionally contribute to the municipal solid waste management literature and the vehicle routing literature by creating 22 new test instances for the RWCP. When comparing the performance of MASS with an exact approach for a small instance, we found that MASS is able to generate high-quality feasible solutions. MASS obtains competitive results for VRPIF problem instances from the literature. For MDVRPI problem instances, MASS reaches similar results as the best solution approach found in recent literature, but within much shorter computation times. For MDVRPIF instances, MASS obtains better results than those currently found in the literature. Thus, MASS is not only a useful algorithm for cases in which the waste quantities are allowed to be collected partially, but also for cases in which vehicles are only allowed to visit a pick-up location once (full pick-ups).

Finally, we demonstrate how MASS can be used for providing policy recommendations by solving a real-life case in the Brussels Capital Region (Belgium). Municipal solid waste collection costs could be reduced if residents can be motivated to collect more food waste separately. The installation of three composting plants would result in the lowest bio-waste collection costs, although the cost reductions need to be compared to the processing facility investment costs incurred for this treatment option. Furthermore, a large cost reduction of at least 28% can be achieved for bio-waste collection if food and garden waste would be collected jointly.

Although the RWCP presented in this paper can already be applied to a large variety of cases and the existing literature on the problem provided a point of comparison for the proposed MASS algorithm, the main limitation of this paper is the exclusion of some additional constraints, specific for the waste collection industry. Collection points, depots and processing facilities with time-windows, heterogeneous collection vehicles or time-dependent travel times are left for future research. A second limitation is the lack of solution approaches and test instances for general waste collection problems in the literature that can be used for a performance comparison. Third, this paper considers the waste collection system for a given scenario (predetermined pick-up locations, facilities and depots). However, policy recommendations are typically derived from broader studies in which additional factors need to be considered, for instance, investment costs of depots and processing facilities or the impact of alternative collection calendars. MASS could be used to evaluate such a larger set of scenarios and, thus, offer policy makers very insightful information.

Furthermore, with the rise of cities developing digital twins of their physical infrastructure including IoT (Internet of Things), devices registering near real-time the waste levels of collection bins, several problem extensions are emerging. First, the more fine-grained data collection, both in time and space, allows for more accurate waste generation forecasting and waste collection problems with stochastic demand will become more prevalent. Second, IoT sensors can signal when an anomalous amount of waste is deposited, or conversely not deposited, on a given day which might affect the current routes. This introduces a component of dynamic demand into the operational routing problem.

Acknowledgments

The authors are grateful for the support provided by the Editor in chief and for the helpful comments made by the anonymous reviewers. The authors gratefully acknowledge the financial support of the Brussels Capital Region - Innoviris. The authors would also like to thank Prof. Simon De Jaeger and Prof. Jeroen Belien for their constructive feedback and their support for the development of the case study.

Appendix A. Parent selection and crossover

To create children from an existing population, two parents are selected using a binary tournament based on the individuals' biased fitness. Algorithm 1 describes the order-based crossover procedure used to create a child from the selected parents. In essence, we copy a sequence of pick-ups performed in a sequence of routes from Parent 1. The remaining pick-ups of Parent 1 are filled in from left to right, starting after the last pick-up of the copied sequence of routes, following the order of the pick-up locations in the chromosome of Parent 2. This procedure guarantees that all waste is collected in each pick-up location.

We explain the procedure using the example shown in Fig. A.1. Part a) depicts the chromosomes of two parents $P1$ and $P2$. The chromosomes also display their division into vehicle routes, resulting from the sequential split procedure explained in Section 4.2.

Algorithm 1: Order-based crossover.

Result: chromosome of new child C
Definitions
$K(P)$ = the set of pick-ups κ in P, contains a pick-up location and a pick-up quantity;
$pul(\kappa)$ = the pick-up location of κ;
$move(\kappa)$ = the pick-up κ one gene to the right in the chromosome. If κ is on the last position, move to the front;
$append(\kappa, C)$: appends κ behind the last appended pick-up in C.
Inheritance from $P1$
Select 2 parents $P1$ and $P2$ using a binary tournament;
Determine the start and end κ for each vehicle route in the chromosome of $P1$;
Mark each transition from one route to the next in $P1$ with an increasing integer, starting from 0;
Take R = # routes in $P1$;
Select 2 random integers using a uniform distribution between 0 and R: $r1$ and $r2$ with $r1 < r2$;
Copy the set of κ between $r1$ and $r2$ in $P1$ to the same positions in C;
Inheritance from $P2$
$K^{appended}(P1)$ = the set of κ between $r1$ and $r2$ in $P1$;
$K^{toadd}(P1) = K(P1) \backslash K^{appended}(P1)$;
κ^{P1last} = the last pick-up in $K^{appended}(P1)$;
κ^{currP2} = a randomly selected $\kappa \in K(P2) | pul(\kappa) = pul(\kappa^{P1last})$;
while $K^{toadd}(P1) \neq \varnothing$ **do**
  $move(\kappa^{currentP2})$;
  if $\exists \kappa \in K^{toadd}(P1) | pul(\kappa) = pul(\kappa^{currentP2})$ **then**
    $append(\kappa, C)$;
    $K^{appended} = K^{appended} \cup \kappa$;
    $K^{toadd} = K^{toadd} \backslash \kappa$;
  end
end

The start of each route $\gamma_i \in \Gamma$ receives a value s_i. The first route receives integer 0, the start of the next route receives integer 1, etc. Finally, the end of the chromosome receives a number equal to the total number of vehicle routes ($s_{|\Gamma|} = |\Gamma|$). We randomly select two integers $r1, r2 \in \{0 \dots R\}$ with R the number of routes, and $r1 < r2$. In Fig. A.1 a), $r1 = 1$ and $r2 = 3$. We copy the pick-ups between $r1$ and $r2$ to child C.

We keep track of two sets of pick-ups in $P1$: the pick-ups which were already appended to C, $K^{appended}(P1) = \{[1], [3]\}$, and the pick-ups still to add, $K^{toadd}(P1) = \{[4], [5], [2], [5]\}$. Note that $K^{toadd}(P1)$ contains two pick-ups in pick-up location 5. We fill up the remaining gene positions in C (presented with question marks) starting after [3] (the last pick-up location in $K^{appended}$). Since [3] is visited twice in $P2$, we randomly select one of the pick-ups: the second pick-up. This second pick-up is followed by [2] in $P2$. Since $K^{toadd}(P1)$ still contains [2], this pick-up is appended with the collected quantity of $P1$ to C. The pick-up is added to $K^{appended}(P1) = \{[1], [3], [2]\}$ and removed from $K^{toadd}(P1) = \{[4], [5], [5]\}$. The appended pick-ups are marked with a cross in Fig. A.1 b).

As we have reached the end of $P2$'s chromosome, we return to the front: [1]. $K^{toadd}(P1)$ does not include a visit to pick-up location 1 nor pick-up location 3. We therefore move to [5] in $P2$. $K^{toadd}(P1)$ contains two pick-ups to pick-up location 5. We randomly select one of the two and we update the two sets of pick-ups in $P1$. We move to [4] in $P2$. The corresponding pick-up in $K^{toadd}(P1)$ is appended to C after which we again update the pick-up sets. The result is presented in Fig. A.1 c). Finally, the remaining pick-up [5] in $K^{toadd}(P1)$ is appended to C in Fig. A.1 d).

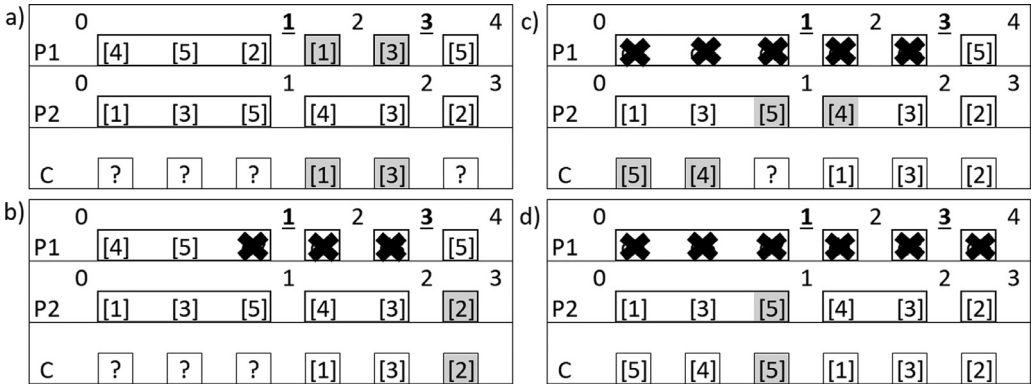


Fig. A.1. Order-based crossover.

Appendix B. K-split procedure

Appendix C. Parameter Calibration

We calibrate MASS heuristic parameters using the systematic experimental design approach proposed by Coy et al. (2000). The approach starts with an initialization phase. First, for each parameter, an initial design center is determined. The initial design center of a parameter is a value that provides satisfactory results and is decided through ad hoc testing. Second, a logical range is determined for each parameter within which the parameter's value is permitted to vary. For the population sizes, the ranges reported by Vidal et al. (2012) were used. For parameters which are proportions, evidently, the parameter values should remain between 0 and 1. Third, we decide on an initial step size. The latter must be small enough to ensure that if it is added to, or subtracted from the initial design center, the resulting value must remain within the previously determined range. However, it must also be suffi-

Algorithm 2: K-split for pick-up location $l \in \{1 \dots n\}$.

Result: new solution of P
Definitions
Define q_l: the generated waste in l.
Define $il_k \in IL, k \in \{1..|IL|\}$ as the set of insert locations where a pick-up κ_{new} to pick-up location l can be inserted. κ_{new} can be inserted after each κ_i in $subroute(\kappa_i)$ or at the front of the pick-up sequence of subroute $\sigma_s \in \Sigma, s \in \{1..|\Sigma|\}$. $|IL| = |K| + |\Sigma|$.
Define IP_k as the insertion price at insert location k.
Define q_{il_k} as the quantity that can be inserted at insert location il_k.
Pick-up location removal
Remove from all subroutes $\sigma \in \Sigma$ all pick-ups $\kappa_i \in K, i \in 1..|K|$ for which $pul(\kappa_i) = l$.
Recalculate for all subroutes and routes the altered duration.
Insertion price calculation
$k = 1$
forall the $\kappa_i \in K$ **do**
  Insert pick-up κ_{new} after κ_i in $subroute(\kappa_i)$ with $q(\kappa_{new}) = \min\{MaxQ(l, \kappa_i), q_l\}$.
  $\Delta(duration)$ = the increase in travel time due to this insertion.
  $IP_k = \Delta(duration)/q(\kappa_{new})$, $q_{il_k} = q(\kappa_{new})$
  $k = k + 1$
  Revert the insertion.
  if κ_i *is the first pick-up in* $subroute(\kappa_i)$ **then**
    Insert pick-up κ_{new} at the front of $subroute(\kappa_i)$ with $q(\kappa_{new}) = \min\{MaxQ(l, \sigma_s), q_l\}$.
    $\Delta(duration)$ = the increase in travel time due to this insertion.
    $IP_k = \Delta(duration)/q(\kappa_{new})$, $q_{il_k} = q(\kappa_{new})$
    $k = k + 1$
    Revert the insertion.
  end
end
Insert pick-up location l
Sort all $il_k \in IL$ in increasing order of IP_k.
$q_l^{res} = q_l$
$k = 1$
while $q_l^{res} > 0$ *AND* $k \leq |IL|$ **do**
  Insert κ_{new} at il_k with $q(\kappa_{new}) = min\{q_{il_k}, q_l^{res}\}$
  If $subroute(\kappa_{new})$ visits l multiple times, merge the pick-ups to the first position.
  $q_l^{res} = q_l^{res} - q(\kappa_{new})$
end
If $q_l^{res} > 0$, create additional routes to collect the residual waste quantity.

Table C.1
Experimental design: initial values.

Parameter	Start design center	min	max	initial step size
λ	70	1	200	65
μ	50	5	200	40
PercElite	0.4000	0	1	0.2000
PercClose	0.5000	0	1	0.3000
h	0.4000	0	1	0.3000
θ_F^{REF}	0.5000	0	1	0.2000
θ_D^{REF}	0.5000	0	1	0.2000
$Prob^{ls}$	0.8000	0	1	0.2000

Table C.2
Fractional factorial design.

	λ	μ	*PercElite*	*PercClose*	h	θ_F^{REF}	θ_D^{REF}	$Prob^{ls}$
1	-1	+1	-1	+1	-1	+1	-1	-1
2	-1	+1	+1	-1	+1	-1	+1	+1
3	-1	-1	+1	+1	-1	+1	+1	+1
4	-1	+1	-1	+1	+1	+1	-1	+1
5	+1	-1	-1	+1	-1	+1	-1	+1
6	-1	-1	-1	-1	+1	-1	-1	-1
7	+1	+1	-1	-1	-1	-1	-1	-1
8	-1	+1	+1	-1	-1	-1	+1	-1
9	-1	+1	+1	+1	+1	-1	-1	-1
10	+1	-1	-1	-1	-1	+1	+1	-1
11	+1	+1	+1	+1	-1	+1	+1	-1
12	-1	-1	-1	+1	+1	-1	+1	+1
13	-1	+1	+1	+1	-1	-1	-1	+1
14	+1	+1	+1	-1	+1	+1	-1	-1
15	+1	+1	-1	-1	+1	-1	-1	+1
16	+1	+1	-1	+1	-1	-1	+1	+1
17	-1	-1	+1	-1	-1	+1	-1	-1
18	+1	-1	-1	+1	+1	+1	-1	-1
19	+1	+1	+1	+1	+1	+1	+1	+1
20	+1	+1	-1	+1	+1	-1	+1	-1
21	+1	-1	+1	-1	+1	-1	+1	-1
22	+1	-1	+1	+1	-1	-1	-1	-1
23	-1	+1	-1	-1	-1	+1	+1	+1
24	-1	-1	-1	+1	-1	-1	+1	-1
25	+1	-1	+1	-1	-1	-1	+1	+1
26	-1	+1	-1	-1	+1	+1	+1	-1
27	+1	-1	-1	-1	+1	+1	+1	+1
28	-1	-1	+1	+1	+1	+1	+1	-1
29	-1	-1	-1	-1	-1	-1	-1	+1
30	+1	-1	+1	+1	+1	-1	-1	+1
31	+1	+1	+1	-1	-1	+1	-1	+1
32	-1	-1	+1	-1	+1	+1	-1	+1

ciently large to test the impact of a significant difference in a parameter's value. The initial design centers, initial step sizes and the ranges of all parameters investigated are presented in Table C.1. As Vidal et al. (2012) found that the probability of an individual being subjected to the local search procedure should be near to 1, we selected a higher design center for $Prob^{ls}$ than for the other parameters.

The second phase consists of an iterative procedure that searches for good parameter settings within the range through gradient descent. A two-level factorial design (-1, +1) is set up for the eight parameters. The "-1"("+1") represents a value in which the step size is deducted from (added to) the design center. A fractional factorial design is used to reduce the number of experiments to be run and thus the computational effort. The parameter θ_F^{REF} is irrelevant for the instances with only one processing facility (1D1P and mD1P). Therefore, we use a 2^{7-2} fractional factorial design for the 1D1P and the mD1P case and a 2^{8-3} fractional factorial design for the other two cases. The fractional factorial design can be found in Table C.2. The resulting 32 instances are run 5 times with different input seeds.

The 32 average objective values obtained from the respective five runs are used to perform a linear regression analysis. The parameter with the largest significant coefficient (in absolute values) (parameter j with coefficient b_j) is selected to perform a gradient descent. For all significant parameters i, the step sizes and the design centers are adjusted. The new step size is obtained by multiplying the old step size with b_i/b_j with b_i and b_j the estimated coefficients for respectively parameter i and j. The design center of parameter i is adjusted by moving half a step down the path of the steepest descent. This is done by subtracting or adding (depending on the sign of the estimated coefficient) half the new step size from or to the current design center. Coy et al. (2000) propose small steps of 1/4 to avoid stepping over good local minima. However, as MASS contains many parameters to be calibrated, a step of 1/2 was used to reduce computational efforts.

The new design centers and step sizes are used to adjust the experimental design. The 32 instances are run again, followed by a linear regression analysis and the appropriate adjustment of design centers and step sizes. If none of the parameters have a significant effect on the objective value, the procedure presented by Coy et al. (2000) stops the iterative process. However, we extend this procedure by looking for additional local improvements. Specifically, if no significant effects are found, all step sizes are reduced by 50%. If then, still no significant results can be found,

Table C.3
Parameter calibration 1D1P.

Step	λ		μ		*PercElite*		*PercClose*		h		θ_D^{REF}		$Prob^{ls}$	
	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS
0	70	65	50	40	0.400	0.200	0.500	0.300	0.400	0.300	0.500	0.200	0.800	0.200
1	38	37	65	29	0.465	0.152	0.500	0.300	0.465	0.131	0.500	0.200	0.800	0.200
2	19	18	76	22	0.490	0.133	0.500	0.300	0.490	0.049	0.500	0.200	0.835	0.070
3	23	8	76	22	0.490	0.133	0.500	0.300	0.490	0.049	0.500	0.200	0.852	0.035
4	21	5	76	22	0.490	0.133	0.500	0.300	0.490	0.049	0.500	0.200	0.852	0.035
5	21	5	76	22	0.490	0.133	0.500	0.300	0.490	0.049	0.500	0.200	0.870	0.035
6	21	5	76	22	0.490	0.133	0.500	0.300	0.490	0.049	0.500	0.200	0.870	0.035
7	21	5	76	22	0.490	0.133	0.650	0.300	0.490	0.049	0.500	0.200	0.870	0.035
8	19	4	76	22	0.490	0.133	0.650	0.300	0.490	0.049	0.500	0.200	0.870	0.035
9	19	4	67	17	0.490	0.113	0.650	0.300	0.490	0.049	0.600	0.200	0.870	0.035
10	19	4	58	17	0.490	0.113	0.650	0.300	0.490	0.049	0.600	0.200	0.870	0.035
11	21	4	50	16	0.490	0.113	0.650	0.300	0.490	0.049	0.600	0.200	0.870	0.035
12	21	4	42	16	0.490	0.113	0.650	0.300	0.490	0.049	0.600	0.200	0.870	0.035
13	***21***	***2***	***42***	***8***	***0.490***	***0.057***	***0.650***	***0.150***	***0.490***	***0.025***	***0.600***	***0.100***	***0.870***	***0.017***
14	21	2	42	8	0.490	0.057	0.650	0.150	0.490	0.025	0.550	0.100	0.870	0.017
15	21	2	46	6	0.490	0.057	0.650	0.150	0.490	0.025	0.550	0.100	0.878	0.017
16	21	2	46	6	0.490	0.057	0.650	0.150	0.490	0.025	0.550	0.100	0.870	0.017
17	21	2	42	6	0.490	0.057	0.650	0.150	0.490	0.025	0.550	0.100	0.870	0.017
18	*21*	*1*	*42*	*3*	*0.490*	*0.028*	*0.650*	*0.075*	*0.490*	*0.012*	*0.550*	*0.050*	*0.870*	*0.009*

Table C.4
Parameter calibration 1DmP.

Step	λ		μ		*PercElite*		*PercClose*		h		θ_F^{REF}		θ_D^{REF}		$Prob^{ls}$	
	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS
0	70	65	50	40	0.400	0.200	0.500	0.300	0.400	0.300	0.500	0.200	0.500	0.200	0.800	0.200
1	70	65	64	29	0.540	0.200	0.500	0.300	0.540	0.281	0.500	0.200	0.500	0.200	0.900	0.100
2	38	37	64	29	0.540	0.200	0.500	0.300	0.540	0.281	0.500	0.200	0.500	0.200	0.950	0.050
3	57	37	64	29	0.495	0.200	0.500	0.300	0.495	0.091	0.500	0.200	0.500	0.200	0.950	0.050
4	38	37	64	29	0.480	0.200	0.500	0.300	0.480	0.030	0.540	0.080	0.500	0.200	0.950	0.050
5	38	37	64	29	0.480	0.134	0.592	0.185	0.480	0.030	0.540	0.080	0.500	0.200	0.925	0.050
6	57	37	61	7	0.480	0.134	0.592	0.185	0.480	0.030	0.540	0.080	0.500	0.200	0.925	0.050
7	38	37	59	3	0.480	0.134	0.592	0.185	0.480	0.030	0.540	0.080	0.500	0.200	0.925	0.050
8	57	37	59	3	0.480	0.134	0.592	0.185	0.480	0.030	0.540	0.080	0.500	0.200	0.917	0.016
9	38	37	59	3	0.476	0.134	0.592	0.185	0.476	0.009	0.540	0.080	0.500	0.200	0.917	0.016
10	57	37	59	3	0.476	0.134	0.592	0.185	0.476	0.009	0.540	0.080	0.500	0.200	0.917	0.016
11	*57*[a]	*19*	*59*	*1*	*0.476*	*0.067*	*0.592*	*0.092*	*0.476*	*0.004*	*0.540*	*0.040*	*0.500*	*0.100*	*0.917*	*0.008*
12	47	19	59	1	0.476	0.067	0.592	0.092	0.476	0.004	0.540	0.040	0.500	0.100	0.917	0.008
13	47	19	59	1	0.476	0.040	0.639	0.092	0.476	0.004	0.540	0.040	0.500	0.100	0.917	0.008
14	**47**	**19**	**59**	**1**	**0.478**	**0.040**	**0.639**	**0.092**	**0.478**	**0.004**	**0.540**	**0.040**	**0.500**	**0.100**	**0.917**	**0.008**
15	47	19	59	1	0.476	0.040	0.639	0.092	0.476	0.003	0.540	0.040	0.450	0.100	0.917	0.008
16	38	19	59	1	0.476	0.040	0.639	0.092	0.476	0.003	0.540	0.040	0.450	0.100	0.917	0.008
17	38	19	59	1	0.476	0.019	0.639	0.092	0.476	0.003	0.540	0.040	0.400	0.100	0.917	0.008
18	38	19	59	1	0.476	0.019	0.685	0.092	0.476	0.003	0.540	0.040	0.400	0.100	0.917	0.008
19	*38*	*9*	*59*	*1*	*0.476*	*0.010*	*0.685*	*0.046*	*0.476*	*0.002*	*0.540*	*0.020*	*0.400*	*0.050*	*0.917*	*0.004*

[a] An oscillation occurs. The only significant parameter is λ. Its DC oscillates between 38 and 57 while its step size remains unchanged. We break the oscillation by reducing the step sizes.

the iterative process is stopped. Furthermore, if, during the procedure, a parameter's "-1" value or "+1" value would surpass respectively the minimum or maximum value of its range, Coy et al. (2000) propose to keep the value of this parameter constant for the remainder of the iterative process. Instead, we propose to further fine-tune the values for this parameters by reducing its step size to the maximum value for which the range's limits are not exceeded.

The design centers (DC) and the step sizes (SS) of each iteration are presented in Table C.3, Table C.4, Table C.5, Table C.6 for respectively the 1D1P, the 1DmP, the mD1P and the mDmP case. Significant results are underlined and iterations in which the step sizes are halved are put in italics.

After 14 iterations, no significant effects could be found for the mD1P case. Throughout these iterations, no significant effect was found for the parameter *PercClose*. In iteration 15, we introduce an additional test with a larger step size for *PercClose*. As no significant effects could be found for *PercClose* in this additional test, the iterative process is stopped. For the instance type mDmP, no significant results were found after 20 iterations. The step sizes have become relatively small and also no significant effects were found for the parameter $Prob^{ls}$. We suspect that the search for good parameter settings has ended up in a local minimum. We therefore extend the procedure proposed by Coy et al. (2000) and widen the search space again by increasing the step size of $Prob^{ls}$ to its initial value. Indeed, in the next iteration, significant effects were found, thus leading the search out of the local minimum.

In the third and final phase, parameter settings are selected for each instance type. We select the parameter settings that resulted in the lowest average objective value (over the five runs). The iteration in which the parameter settings with the lowest average objective value was found, is indicated in bold in Tables C.3, C.4, C.5, C.6. The final parameter settings can be found in Table 1 in Section 5.

During the iterative parameter calibration process, significant effects were found for all parameters. We can, thus, conclude that all building blocks of MASS represented by these parameters play a crucial role in finding high-quality feasible solutions for RWCPs.

Table C.5
Parameter calibration mD1P.

Step	λ		μ		*PercElite*		*PercClose*		h		θ_D^{REF}		$Prob^{ls}$	
	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS
0	70	65	50	40	0.400	0.200	0.500	0.300	0.400	0.300	0.500	0.200	0.800	0.200
1	39	38	50	40	0.250	0.200	0.500	0.300	0.250	0.250	0.427	0.145	0.800	0.200
2	39	38	50	40	0.375	0.200	0.500	0.300	0.375	0.250	0.427	0.145	0.800	0.200
3	39	19	50	20	0.375	0.100	0.500	0.150	0.375	0.125	0.427	0.073	0.800	0.100
4	31	16	50	20	0.375	0.100	0.500	0.150	0.375	0.125	0.427	0.073	0.850	0.100
5	31	16	50	20	0.375	0.100	0.500	0.150	0.375	0.125	0.399	0.057	0.900	0.100
6	*31*	*8*	*50*	*10*	*0.375*	*0.050*	*0.500*	*0.075*	*0.375*	*0.063*	*0.399*	*0.029*	*0.900*	*0.050*
7	29	5	53	6	0.375	0.050	0.500	0.075	0.375	0.063	0.385	0.029	0.900	0.050
8	27	3	53	6	0.375	0.050	0.500	0.075	0.375	0.063	0.385	0.029	0.900	0.050
9	*27*	*2*	*53*	*3*	*0.375*	*0.025*	*0.500*	*0.038*	*0.375*	*0.031*	*0.385*	*0.014*	*0.900*	*0.025*
10	26	2	53	3	0.375	0.025	0.500	0.038	0.375	0.031	0.385	0.014	0.900	0.025
11	25	2	53	3	0.375	0.025	0.500	0.038	0.375	0.031	0.385	0.014	0.900	0.025
12	25	1	53	3	0.375	0.025	0.500	0.038	0.375	0.031	0.385	0.014	0.900	0.025
13	24	1	53	3	0.375	0.025	0.500	0.038	0.375	0.031	0.385	0.014	0.911	0.022
14	***24***	***1***	***53***	***2***	***0.375***	***0.013***	***0.500***	***0.019***	***0.375***	***0.016***	***0.385***	***0.007***	***0.911***	***0.011***
15	24	1	53	3	0.375	0.025	0.500	*0.300*	0.375	0.031	0.385	0.014	0.911	0.022

Table C.6
Parameter calibration mDmP.

Step	λ		μ		*PercElite*		*PercClose*		h		θ_F^{REF}		θ_D^{REF}		$Prob^{ls}$	
	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS	DC	SS
0	70	65	50	40	0.4000	0.2000	0.5000	0.3000	0.4000	0.3000	0.5000	0.2000	0.5000	0.2000	0.8000	0.2000
1	38	37	61	22	0.400	0.200	0.5000	0.3000	0.4000	0.3000	0.500	0.200	0.5000	0.2000	0.8000	0.2000
2	54	33	61	22	0.400	0.190	0.5000	0.3000	0.4000	0.3000	0.4000	0.2000	0.5000	0.2000	0.8000	0.2000
3	41	26	61	22	0.4000	0.1902	0.5000	0.3000	0.4000	0.3000	0.400	0.200	0.5000	0.2000	0.8000	0.2000
4	41	26	61	22	0.400	0.190	0.5000	0.3000	0.4000	0.3000	0.3000	0.2000	0.5000	0.2000	0.8000	0.2000
5	41	26	61	22	0.400	0.190	0.5000	0.3000	0.4000	0.3000	0.3000	0.2000	0.5000	0.2000	0.8000	0.2000
6	41	26	61	22	0.4000	0.1902	0.5000	0.3000	0.4000	0.3000	0.3000	0.2000	0.5000	0.2000	0.8000	0.2000
7	*41*	*13*	*61*	*11*	*0.400*	*0.095*	*0.500*	*0.150*	*0.400*	*0.150*	*0.300*	*0.100*	*0.500*	*0.100*	*0.800*	*0.100*
8	41	13	56	10	0.4000	0.0951	0.5000	0.1500	0.4000	0.1500	0.2500	0.1000	0.5000	0.1000	0.8000	0.1000
9	48	13	56	10	0.400	0.095	0.5000	0.1500	0.400	0.150	0.250	0.100	0.5000	0.1000	0.8000	0.1000
10	48	13	52	7	0.3250	0.0543	0.5000	0.1500	0.325	0.150	0.2844	0.0687	0.5000	0.1000	0.8000	0.1000
11	48	13	50	4	0.2500	0.0543	0.5000	0.1500	0.2500	0.1500	0.2844	0.0687	0.5000	0.1000	0.8000	0.1000
12	*48*	*6*	*50*	*2*	*0.250*	*0.027*	*0.500*	*0.075*	*0.250*	*0.075*	*0.284*	*0.034*	*0.500*	*0.050*	*0.800*	*0.050*
13	48	6	50	2	0.2125	0.0272	0.5000	0.0750	0.2125	0.0750	0.284	0.034	0.5000	0.0500	0.8000	0.0500
14	51	6	51	1	0.2125	0.0272	0.5000	0.0750	0.2125	0.0750	0.2956	0.0224	0.5000	0.0500	0.8000	0.0500
15	54	6	51	1	0.2125	0.0272	0.500	0.075	0.2125	0.0750	0.296	0.022	0.5000	0.0500	0.8000	0.0500
16	57	6	50	1	0.2125	0.0272	0.476	0.047	0.2125	0.0750	0.3028	0.0145	0.5000	0.0500	0.8000	0.0500
17	57	6	50	1	0.2125	0.0272	0.4528	0.0472	0.2125	0.0750	0.3028	0.0145	0.5000	0.0500	0.8000	0.0500
18	*57*	*3*	*50*	*1*	*0.213*	*0.014*	*0.453*	*0.024*	*0.213*	*0.038*	*0.303*	*0.007*	*0.500*	*0.025*	*0.800*	*0.025*
19	57	3	50	1	0.2125	0.0136	0.4528	0.0236	0.213	0.038	0.3028	0.0072	0.4875	0.0250	0.8000	0.0250
20	57	3	50	1	0.1938	0.0136	0.4528	0.0236	0.1938	0.0375	0.3028	0.0072	0.4875	0.0250	0.8000	0.0250
21	57	3	50	1	0.1938	0.0136	0.4528	0.0236	0.1938	0.0375	0.3028	0.0072	0.4875	0.0250	0.800	0.200[a]
22	57	3	50	1	0.1938	0.0136	0.4528	0.0236	0.1938	0.0375	0.303	0.007	0.4875	0.0250	0.700	0.200
23	58	1	50	1	0.1938	0.0136	0.4528	0.0236	0.1938	0.0375	0.3006	0.0044	0.4875	0.0250	0.800	0.200
24	58	1	50	1	0.1938	0.0136	0.4528	0.0236	0.194	0.038	0.3006	0.0044	0.4875	0.0250	0.700	0.200
25	58	1	50	1	0.2014	0.0136	0.4528	0.0236	0.2014	0.0153	0.3006	0.0044	0.4875	0.0250	0.800	0.200
26	58	1	50	1	0.2014	0.0136	0.4528	0.0236	0.201	0.015	0.3006	0.0044	0.488	0.025	0.700	0.200
27	59	0[b]	49	0[b]	0.2041	0.0136	0.4528	0.0236	0.2041	0.0054	0.3006	0.0044	0.4824	0.0103	0.800	0.200
28	59	0	49	0	0.204	0.014	0.4528	0.0236	0.2041	0.0054	0.3006	0.0044	0.4824	0.0103	0.700	0.200
29	**59**	**0**	**49**	**0**	**0.204**	**0.005**	**0.453**	**0.024**	**0.204**	**0.005**	**0.301**	**0.004**	**0.482**	**0.010**	**0.800**	**0.200**
30	59	0	49	0	0.2041	0.0045	0.4528	0.0236	0.204	0.005	0.301	0.004	0.4824	0.0103	0.700	0.200
31	59	0	49	0	0.2052	0.0045	0.4528	0.0236	0.2052	0.0021	0.2992	0.0028	0.4824	0.0103	0.800	0.200
32	59	0	49	0	0.2052	0.0045	0.4528	0.0236	0.2052	0.0021	0.2992	0.0028	0.4824	0.0103	0.700	0.200
33	59	0	49	0	0.2052	0.0045	0.4528	0.0236	0.2052	0.0021	0.2992	0.0028	0.4824	0.0103	0.8000	0.100[c]
34	*59*	*0*	*49*	*0*	*0.205*	*0.002*	*0.453*	*0.012*	*0.205*	*0.001*	*0.299*	*0.001*	*0.482*	*0.005*	*0.800*	*0.050*

[a] Additional search for significant effect for $Prob^{ls}$: increase the step size to its initial level.
[b] Step sizes < 1: fix values to the factor (+1/-1) with the best results.
[c] $Prob^{ls}$ oscillates. Take half the step size.

Appendix D. Details computational experiments

Tables D.1, D.2, D.3 and D.4 provide a detailed description of the test instances used for comparing the performance of MASS with respectively VRPIF instances, MDVRPIF instances, MDVRPI instances and MDVRP instances. The number of customers n, the number of vehicles available Q_{tot}^d, the number depots d, and the number of intermediate facilities f of the instance sets are presented. We report on the lowest-cost feasible solution found after one hour and after running the algorithm for the same amount of time as Hemmelmayr et al. (2013), Markov et al. (2016) and Vidal et al. (2012) for respectively the original Crevier et al. (2007) instances (a1-l1, a2-j2), variation 1 (MDVRPIF) of the Crevier et al. (2007) instances (MDIFa1-MDIFl1, MDIFa2-MDIFj2), variation 2 (MDVRPI) of the Crevier et al. (2007) instances (MDIa1-MDIl1, MDIa2-MDIj2) and the MDVRP instances. Furthermore, if a new best feasible so-

Table D.1
Results for the VRPIF test instances.

Inst.	n	Q^d_{tot} /d/f	CCL	CCL T(s)	H-GLS	H-GLS T(s)	MNS	MNS T(s)	VNS	VNS T(s)	MASS (3600s) (Δ%)	New best (MASS)	MASS at T(VNS)[a] (Δ%)	T(MASS)[b] at result VNS (s)
a1	48	6/1/2	1211.28	275	1189.70	203	1202.89	21	1180.57	85	1182.78(0.19%)	1179.79	1187.18(0.56%)	856
b1	96	4/1/2	1232.67	550	1225.08	468	1231.33	191	1217.07	383	1232.25(1.25%)		1233.01(1.31%)	
c1	192	5/1/2	1893.01	2173	1898.92	2053	1910.21	712	1867.96	1224	1911.06(2.31%)		1939.30(3.82%)	
d1	48	5/1/3	1076.31	513	1064.29	352	1071.19	19	1059.43	94	1061.71(0.22%)	1056.04	1062.89(0.33%)	124
e1	96	5/1/3	1311.60	811	1309.12	517	1333.99	157	1309.12	373	1309.12(0.00%)		1310.40(0.10%)	512
f1	192	4/1/3	1601.54	2485	1585.83	2329	1597.78	1149	1573.05	1536	1601.14(1.79%)		1604.00(1.97%)	
g1	72	5/1/4	1202.00	3313	1190.21	347	1202.28	73	1183.32	203	1182.97(-0.03%)	1181.04	1183.39(0.01%)	120
h1	144	4/1/4	1598.51	1924	1577.54	664	1571.26	532	1548.61	877	1558.59(0.64%)		1564.26(1.01%)	
i1	216	4/1/4	1976.11	3061	1956.17	2550	1956.97	1224	1923.52	2015	1974.03(2.63%)		1977.80(2.82%)	
j1	72	4/1/5	1161.77	3534	1128.86	331	1139.20	66	1115.78	167	1115.78(0.00%)		1116.33(0.05%)	63
k1	144	4/1/5	1618.45	3877	1591.74	724	1598.25	555	1577.96	874	1588.81(0.69%)		1593.61(0.99%)	
l1	216	4/1/5	1917.08	6256	1904.39	3083	1903.15	1436	1869.70	2129	1908.24(2.06%)		1909.51(2.13%)	
Average Δ% CCL				-1.12% [-1.37,-0.87]										
Average Δ% H-GLS				+0.06% [-0.20,0.09]										
Average Δ% MNS				-0.67% [-0.85,-0.49]										
Average Δ% VNS				+0.98% [0.79,1.16]										
Average Δ% (T(VNS))[a]				+1.26% [1.03,1.48]										
a2	48	4/1/4	1005.16	383			998.90	38	997.94	74	997.94(0.00%)		998.32(0.04%)	53
b2	96	4/1/4	1333.20	883			1343.87	218	1291.19	385	1308.20(1.32%)		1315.35(1.87%)	
c2	144	4/1/4	1792.46	3701			1756.83	432	1715.84	901	1733.18(1.01%)	1715.60	1738.38(1.31%)	1449
d2	192	3/1/4	1898.21	2432			1884.91	1031	1860.92	1808	1897.80(1.98%)		1908.16(2.54%)	
e2	240	3/1/4	1995.75	4427			1979.30	1621	1922.81	2959	1976.80(2.81%)		1979.40(2.94%)	
f2	288	3/1/4	2312.15	9733			2291.38	2451	2233.43	4274	2360.71(5.70%)		n.a.	
g2	72	4/1/6	1185.93	1771			1167.65	78	1153.17	223	1152.64(-0.05%)	1152.02	1158.83(0.49%)	1177
h2	144	4/1/6	1611.75	9647			1601.21	506	1575.28	940	1602.39(1.72%)		1604.72(1.87%)	
i2	216	3/1/6	1998.20	19345			1958.01	1402	1922.24	2515	1959.73(1.95%)		1966.08(2.28%)	
j2	288	3/1/6	2325.18	15411			2291.22	3057	2250.21	4403	2307.81(2.56%)		n.a.	
Average Δ% CCL				-1.08% [-1.40,-0.77]										
Average Δ% MNS				-0.09% [-0.40,0.22]										
Average Δ% VNS				+1.90% [1.57,2.23]										
Average Δ% (T(VNS))[a]				+1.67% [1.44,1.90]										

[a] The average result reached after running MASS for the same amount of time as the reference study. If no feasible solution was found at that time, the result was excluded for calculating this value.
[b] The average run time to reach the result of VNS of runs that reached the result using the VNS method.

Table D.2
Results for the MDVRPIF test instances.

Inst.	n	Q^d_{tot} /d/f	MNS	MNS T(s)	MASS (3600s)(Δ%)	New best (MASS)	MASS at T(MNS)[a] (Δ%)	T(MASS)[b] at result MNS (s)
MDIFa1	48	6/3/2	1106.46	21	1094.85(-1.05%)		1095.42(-1.00%)	4
MDIFb1	96	4/3/2	1218.30	132	1200.10(-1.49%)	1200.10	1207.63(-0.88%)	86
MDIFc1	192	5/3/2	1885.82	764	1825.03(-3.22%)	1822.54	1859.20(-1.41%)	545
MDIFd1	48	5/4/3	1023.26	27	1013.09(-0.99%)		1013.56(-0.95%)	7
MDIFe1	96	5/4/3	1294.99	147	1277.48(-1.35%)	1275.80	1285.72(-0.72%)	109
MDIFf1	192	4/4/3	1568.29	946	1533.34(-2.23%)	1526.80	1556.10(-0.78%)	688
MDIFg1	72	5/5/4	1138.56	65	1124.66(-1.22%)	1124.66	1126.60(-1.05%)	23
MDIFh1	144	4/5/4	1542.88	448	1517.93(-1.62%)	1512.32	1532.12(-0.70%)	269
MDIFi1	216	4/5/4	1936.75	1443	1891.97(-2.31%)	1883.24	1907.49(-1.51%)	853
MDIFj1	72	4/6/5	1080.02	69	1061.47(-1.72%)	1061.04	1076.58(-0.32%)	50
MDIFk1	144	4/6/5	1542.00	520	1509.23(-2.13%)	1501.97	1530.33(-0.76%)	383
MDIFl1	216	4/6/5	1874.47	1249	1841.93(-1.74%)	1830.38	1875.54(0.06%)	1356
Average Δ% MNS			-1.76% [-1.87,-1.64]					
Average Δ% (T(MNS))[a]			-0.83% [-0.96,-0.71]					
MDIFa2	48	4/5/4	887.58	45	884.56(-2.91%)	884.56	885.43(-2.82%)	8
MDIFb2	96	4/5/4	1256.27	185	1245.09(-2.27%)	1243.06	1246.90(-2.13%)	45
MDIFc2	144	4/5/4	1691.70	421	1672.46(-2.48%)	1665.96	1696.39(-1.09%)	250
MDIFd2	192	3/5/4	1860.77	834	1831.76(-2.08%)	1820.53	1874.12(0.18%)	968
MDIFe2	240	3/5/4	1913.66	2017	1923.77(-1.44%)	1911.72	1937.55(-0.74%)	1564
MDIFf2	288	3/5/4	2249.43	2472	2269.43(-0.23%)	2238.55	2294.91(0.89%)	2781
MDIFg2	72	4/7/6	1070.38	119	1066.58(-1.78%)	1064.34	1068.55(-1.60%)	21
MDIFh2	144	4/7/6	1550.94	369	1542.51(-1.56%)	1534.11	1572.82(0.37%)	481
MDIFi2	216	3/7/6	1903.29	946	1888.93(-1.91%)	1873.99	1929.51(0.20%)	1085
MDIFj2	288	3/7/6	2239.79	2913	2250.70(-0.89%)	2228.58	2261.47(-0.42%)	2328
Average Δ% MNS			-1.76% [-1.93,-1.59]					
Average Δ% (T(MNS))[a]			-0.95% [-0.97,-0.46]					

[a] The average result reached after running MASS for the same amount of time as the reference study. If no feasible solution was found at that time, the result was excluded for calculating this value.
[b] The average run time to reach the result of MNS of runs that reached the result using the MNS method.

Table D.3
Results for the MDVRPI test instances.

Inst.	n	$Q^d_{tot}/d/f$	MAT	MAT T(s)	MASS (3600s)(Δ%)	New best (MASS)	MASS at T(MAT)[a] (Δ%)	T(MASS)[b] at result MAT (s)
MDIa1	48	6/3/3	1102.58	14400	1102.95(0.03%)		n.a.	5
MDIb1	96	4/3/3	1203.98	14400	1203.35(-0.05%)	1203.35	n.a.	296
MDIc1	192	5/3/3	1825.43	14400	1832.21(0.37%)	1823.53	n.a.	1990
MDId1	48	5/4/4	1020.15	17030	1017.32(-0.28%)	1017.32	n.a.	8
MDIe1	96	5/4/4	1293.03	20394	1288.35(-0.36%)	1285.78	n.a.	128
MDIf1	192	4/4/4	1556.69	14400	1563.18(0.42%)		n.a.	
MDIg1	72	5/5/5	1151.67	28801	1138.83(-1.11%)	1134.84	n.a.	21
MDIh1	144	4/5/5	1520.99	14400	1525.75(0.31%)	1517.29	n.a.	1596
MDIi1	216	4/5/5	1908.26	14400	1920.68(0.65%)	1901.45	n.a.	1382
MDIj1	72	4/6/6	1079.27	28801	1073.35(-0.55%)	1064.37	n.a.	144
MDIk1	144	4/6/6	1553.06	28800	1534.90(-1.17%)	1529.78	n.a.	301
MDIl1	216	4/6/6	1831.47	14400	1876.26(2.45%)		n.a.	
Average Δ% MAT			+0.06% [-0.12,0.24]					
MDIa2	48	4/5/5	900.84	14571	900.85(0.00%)		n.a.	140
MDIb2	96	4/5/5	1266.38	28801	1254.75(-0.92%)	1252.24	n.a.	533
MDIc2	144	4/5/5	1683.29	14400	1690.33(0.42%)	1681.62	n.a.	1282
MDId2	192	3/5/5	1849.97	14400	1871.43(1.16%)	1841.75	n.a.	1952
MDIe2	240	3/5/5	1915.69	14400	1967.13(2.69%)		n.a.	
MDIf2	288	3/5/5	2259.00	28800	2319.40(2.67%)		n.a.	
MDIg2	72	4/7/7	1112.35	28801	1085.94(-2.37%)	1081.20	n.a.	30
MDIh2	144	4/7/7	1539.39	14400	1566.85(1.78%)		n.a.	
MDIi2	216	3/7/7	1927.95	28800	1948.95(1.09%)	1923.27	n.a.	3017
MDIj2	288	3/7/7	2272.53	28800	2286.46(0.61%)	2263.43	n.a.	3164
Average Δ% MAT			+0.71% [0.38,1.05]					

[a] Ramos et al. (2020) ran their matheuristic for four or eight hours, while MASS was only run for one hour. We therefore cannot report results for MASS at T(MAT).
[b] The average run time to reach the result of MAT of runs that reached the result using the MAT method.

Table D.4
Results for the MDVRP instances.

Inst.	n	Q^d_{tot}/d	HGS ADC	HGS ADC T(s)	MASS (3600s) (Δ%)	New best	MASS at T(HGSADC)[a] (Δ%)	T(MASS)[b] at result HGSADC (s)
pr01	249	6/5	861.32	10	861.32 (0.00%)		862.14 (0.09%)	8
pr02	80	5/2	1307.34	46	1297.93 (-0.72%)	1296.3	1313.92 (0.50%)	62
pr03	80	5/2	1803.8	115	1806.82 (0.17%)		1869.08 (3.62%)	
pr04	80	5/2	2059.36	313	2036.10 (-1.13%)	2029.6	2116.96 (2.80%)	673
pr05	160	5/4	2340.29	574	2339.75 (-0.02%)		2413.29 (3.12%)	2705
pr06	160	5/4	2681.93	600	2706.51 (0.92%)		2856.64 (6.51%)	
pr07	160	5/4	1089.56	20	1075.12 (-1.33%)	1075.1	1104.50 (1.37%)	31
pr08	240	5/6	1665.05	123	1664.43 (-0.04%)	1661.9	1723.15 (3.49%)	669
pr09	240	5/6	2134.17	366	2151.82 (0.83%)		2255.69 (5.69%)	
pr10	240	5/6	2886.59	600	2824.72 (-2.14%)	2816.1	2936.25 (1.72%)	1345
Average Δ% (3600s):					-0.35% [-0.54,-0.16]			
Average Δ% (T(HGSADC))[a]:					+2.89% [2.47,3.32]			
p01	360	5/9	576.87	14	576.87 (0.00%)		576.87 (0.00%)	5
p02	360	5/9	473.53	13	473.53 (0.00%)		480.78 (1.53%)	740
p03	360	5/9	641.19	26	645.01 (0.60%)		650.15 (1.40%)	
p04	48	2/4	1001.23	116	1005.21 (0.40%)		1006.74 (0.55%)	
p05	96	4/4	750.03	64	754.14 (0.55%)		759.49 (1.26%)	
p06	144	6/4	876.5	68	880.79 (0.49%)		883.96 (0.85%)	
p07	192	8/4	884.43	93	889.86 (0.61%)		895.50 (1.25%)	
p08	240	10/4	4397.42	600	4425.78 (0.64%)		4551.93 (3.51%)	
p09	288	12/4	3868.59	570	3887.46 (0.49%)		4002.49 (3.46%)	
p10	72	3/6	3636.08	589	3667.88 (0.87%)		3742.58 (2.93%)	
p11	144	6/6	3548.25	428	3574.80 (0.75%)		3719.59 (4.83%)	3092
p12	216	9/6	1318.95	31	1320.79 (0.14%)		1357.90 (2.95%)	107
p13	288	12/6	1318.95	34	1318.95 (0.00%)		1319.54 (0.04%)	28
p14	50	4/4	1360.12	33	1391.39 (2.30%)		1419.49 (4.37%)	697
p15	50	2/4	2505.42	115	2523.04 (0.70%)		2639.44 (5.35%)	470
p16	75	3/2	2572.23	118	2577.15 (0.19%)		2665.20 (3.61%)	366
p17	100	8/2	2709.09	128	2748.46 (1.45%)		2919.64 (7.77%)	
p18	100	5/2	3702.85	271	3737.45 (0.93%)		3962.22 (7.00%)	2577
p19	100	6/3	3827.06	252	3842.16 (0.39%)		4143.82 (8.28%)	977
p20	100	4/4	4058.07	262	4146.36 (2.18%)		4438.96 (9.39%)	
p21	249	14/2	5476.41	600	5522.02 (0.83%)		5870.26 (7.19%)	
p22	249	12/3	5702.16	600	5733.15 (0.54%)		6321.49 (10.86%)	
p23	249	8/4	6078.75	600	6380.68 (4.97%)		n.a.[c]	
Average Δ% (3600s):					+0.87% [0.71,1.03]			
Average Δ% (T(HGSADC))[a]:					+4.02% [3.72,4.62]			
Average Δ% (3600s) (all):					+0.5% [0.36,0.64]			
Average Δ% (T(HGSADC))[a] (all):					+3.67% [3.44,4.12]			

[a] The average result reached after running MASS for the same amount of time as the reference study. If no feasible solution was found at that time, the result was excluded for calculating this value.
[b] The average run time to reach the result of HGSADC of runs that reached the result of HGSADC.
[c] No feasible solution found within the time span.

Table D.5
Comparison with an exact approach on a simplified case study.

Waste type	Day	d	f	CPLEX	Gap% CPLEX	MASS	Gap%[a] MASS	Δ%
Bio-waste	1	1	1	12213.60	1.24%	12279.70	1.77%	0.54%
Bio-waste	2	1	1	15113.00	7.05%	15176.70	7.44%	0.42%
Bio-waste	1	1	3	11710.80	7.55%	11736.70	7.76%	0.22%
Bio-waste	2	1	3	14473.60	8.52%	14516.50	8.79%	0.30%
Bio-waste	1	3	1	11437.20	2.27%	11465.00	2.51%	0.24%
Bio-waste	2	3	1	14012.20	0.75%	14085.00	1.27%	0.52%
Bio-waste	1	3	3	11384.90	1.60%	11421.50	1.91%	0.32%
Bio-waste	2	3	3	14043.00	6.82%	14074.20	7.02%	0.22%
Residual waste	1	1	1	19352.50	1.68%	19440.70	2.12%	0.46%
Residual waste	2	1	1	22546.20	3.13%	22847.00	4.40%	1.33%
Residual waste	1	1	3	18007.10	1.49%	18518.40	4.21%	2.84%
Residual waste	2	1	3	21450.60	2.89%	21842.40	4.63%	1.83%
Residual waste	1	3	1	18525.50	2.93%	18691.40	3.79%	0.90%
Residual waste	2	3	1	21584.80	2.71%	21910.30	4.15%	1.51%
Residual waste	1	3	3	17741.30	0.84%	17979.10	2.15%	1.34%
Residual waste	2	3	3	21133.00	3.15%	21352.70	4.15%	1.04%
Average gap% CPLEX:		3.41%						
Average gap% MASS:		4.25%						
Average Δ%:		0.88%						

[a] Using the lower bound reported by CPLEX.

lution is found for an instance, the improved objective value is reported. Finally, if the objective value obtained by the reference study is reached within an hour, we report how long it took MASS to reach this result. Each table also provides a summary that reports on the average percentage difference (over the different instances) between the average results reported by previous studies and the average results after running MASS. To check the robustness of the algorithm, 95%-confidence intervals were constructed of the percentage differences between the result of each instance-run (10 runs per instance) using MASS and the results reported by previous studies applying alternative algorithms.

Table D.5 compares the performance of MASS with an exact approach using CPLEX Optimization Studio. The routes for collecting residual waste and bio-waste in the Brussels Capital Region on two weekdays are optimized. Instances differ with respect to the number of depots and the number of processing facilities. Table D.5 presents the percentage gap reported by CPLEX and the percentage gap of MASS results using the same lower bound as reported by CPLEX.

References

Archetti, C., & Speranza, M. G. (2012). Vehicle routing problems with split deliveries. *International Transactions in Operational Research, 19*(1–2), 3–22. https://doi.org/10.1111/j.1475-3995.2011.00811.x. http://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2011.00811.x

Archetti, C., Speranza, M. G., & Hertz, A. (2006). A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science, 40*(1), 64–73. https://doi.org/10.1287/trsc.1040.0103. https://pubsonline-informs-org.kuleuven.ezproxy.kuleuven.be/doi/10.1287/trsc.1040.0103

Baker, B. M., & Ayechew, M. A. (2003). A genetic algorithm for the vehicle routing problem. *Computers & Operations Research, 30*(5), 787–800. https://doi.org/10.1016/S0305-0548(02)00051-5. https://www.sciencedirect.com/science/article/pii/S0305054802000515

Beliën, J., De Boeck, L., & Van Ackere, J. (2014). Municipal solid waste collection and management problems: A literature review. *Transportation Science, 48*(1), 78–102. https://doi.org/10.1287/trsc.1120.0448. http://pubsonline.informs.org/doi/abs/10.1287/trsc.1120.0448

Bortolotti, A., Kampelmann, S., Muynck, S. D., Papangelou, A., & Zeller, V. (2019). Conditions and concepts for interdisciplinary urban metabolism research - the case of an inter-project collaboration on biowaste in brussels. *ULB Institutional Repository 2013/292639*. Universite Libre de Bruxelles.

Boudia, M., Prins, C., & Reghioui, M. (2007). An effective memetic algorithm with population management for the split delivery vehicle routing problem. In T. Bartz-Beielstein, M. J. Blesa Aguilera, C. Blum, B. Naujoks, A. Roli, G. Rudolph, & M. Sampels (Eds.), *Hybrid metaheuristics*. In *Lecture Notes in Computer Science* (pp. 16–30). Springer Berlin Heidelberg.

Brussels Regional Informatics Centre (2019). UrbIS-Adm: main administrative divisions of the territory and various themed data. Retrieved from https://bric.brussels/en/our-solutions/urbis-solutions/urbis-data/urbis-adm.

Bräysy, O., Dullaert, W., Hasle, G., Mester, D., & Gendreau, M. (2008). An effective multirestart deterministic annealing metaheuristic for the fleet size and mix vehicle-routing problem with time windows. *Transportation Science, 42*(3), 371–386. Publisher: INFORMS. http://www.jstor.org/stable/25769409

Cormen, T. H., Stein, C., Rivest, R. L., & Leiserson, C. E. (2001). *Introduction to algorithms*. McGraw-Hill Higher Education.

Coy, S. P., Golden, B. L., Runger, G. C., & Wasil, E. A. (2000). Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics, 7*, 77–97.

Crevier, B., Cordeau, J., & Laporte, G. (2007). The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research, 176*, 756–773.

Delgado-Antequera, L., Laguna, M., Pacheco, J., & Caballero, R. (2020). A bi-objective solution approach to a real-world waste collection problem. *Journal of the Operational Research Society, 71*(2), 183–194.

Derigs, U., Li, B., & Vogel, U. (2010). Local search-based metaheuristics for the split delivery vehicle routing problem. *The Journal of the Operational Research Society, 61*(9), 1356–1364. http://www.jstor.org/stable/40802255

Dror, M., & Trudeau, P. (1989). Savings by split delivery routing. *Transportation Science, 23*(2), 141–145. http://www.jstor.org/stable/25768367

Dror, M., & Trudeau, P. (1990). Split delivery routing. *Naval Research Logistics*. https://iaorifors.com/paper/3368

ESRI (2019). Arcgis pro,. Redlands, CA: Environmental Systems Research Institute.

Farrokhi-Asl, H., Makui, A., Jabbarzadeh, A., & Barzinpour, F. (2020). Solving a multi-objective sustainable waste collection problem considering a new collection network. *Operational Research, 20*(4), 1977–2015. https://doi.org/10.1007/s12351-018-0415-0.

Gruler, A., Fikar, C., Juan, A., Hirsch, P., & Contreras-Bolton, C. (2017). Supporting multi-depot and stochastic waste collection management in clustered urban areas via simulation–optimization. *Journal of Simulation, 11*, 11–19.

Hemmelmayr, V., Doerner, K., Hartl, R., & Rath, S. (2013). A heuristic solution method for node routing based solid waste collection problems. *Journal of Heuristics, 19*, 129–156. https://doi.org/10.1007/s10732-011-9188-9.

Holland, J. (1975). *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor.

Kim, B.-I., Kim, S., & Sahoo, S. (2006). Waste collection vehicle routing problem with time windows. *Computers & Operations Research, 33*(12), 3624–3642. https://doi.org/10.1016/j.cor.2005.02.045. http://www.sciencedirect.com/science/article/pii/S0305054805001322

Lavigne, C., Beliën, J., & Dewil, R. (2021). An exact routing optimization model for bio-waste collection in the brussels capital region. *Expert Systems with Applications, 183*, 115392. https://doi.org/10.1016/j.eswa.2021.115392. https://www.sciencedirect.com/science/article/pii/S0957417421008150

López-Sánchez, A., Hernández-Díaz, A., Gortázar, F., & Hinojosa, M. (2018). A multiobjective GRASP–VND algorithm to solve the waste collection problem. *International Transactions in Operational Research, 25*(2), 545–567. https://doi.org/10.1111/itor.12452. https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12452

Mar-Ortiz, J., González-Velarde, J. L., & Adenso-Díaz, B. (2013). Designing routes for WEEE collection: The vehicle routing problem with split loads and date windows. *Journal of Heuristics, 19*(2), 103–127. https://doi.org/10.1007/s10732-011-9159-1.

Markov, I., Varone, S., & Bierlaire, M. (2016). Integrating a heterogeneous fixed fleet and a flexible assignment of destination depots in the waste collection VRP with intermediate facilities. *Transportation Research Part B, 84*, 256–273.

Montoya-Torres, J. R., Franco, J. L., Isaza, S. N., Jiménez, H. F., & Herazo-Padilla, N. (2015). A literature review on the vehicle routing problem with

multiple depots. *Computers & Industrial Engineering, 79*, 115–129. https://doi.org/10.1016/j.cie.2014.10.029. https://www.sciencedirect.com/science/article/pii/S036083521400360X

Muter, I., Cordeau, J.-F., & Laporte, G. (2014). A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. *Transportation Science, 48*(3), 425–441. http://www.jstor.org/stable/43666695

Nagata, Y., Bräysy, O., & Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research, 37*(4), 724–737. https://doi.org/10.1016/j.cor.2009.06.022. https://www.sciencedirect.com/science/article/pii/S0305054809001762

OpenStreetMap contributors (2019). Planet dump [data file from 04.02.2019]. Retrieved from https://planet.openstreetmap.org

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research, 31*(12), 1985–2002. https://doi.org/10.1016/S0305-0548(03)00158-8. http://www.sciencedirect.com/science/article/pii/S0305054803001588

Prins, C., Lacomme, P., & Prodhon, C. (2014). Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C: Emerging Technologies, 40*, 179–200. https://doi.org/10.1016/j.trc.2014.01.011. http://www.sciencedirect.com/science/article/pii/S0968090X14000230

Ramos, T. R. P., Gomes, M. I., & Barbosa-Póvoa, A. P. (2020). A new matheuristic approach for the multi-depot vehicle routing problem with inter-depot routes. *OR Spectrum, 42*(1), 75–110. https://doi.org/10.1007/s00291-019-00568-7.

Ramos, T. R. P., & Oliveira, R. C. (2011). Delimitation of service areas in reverse logistics networks with multiple depots. *Journal of the Operational Research Society, 62*(7), 1198–1210. https://doi.org/10.1057/jors.2010.83.

Silva, M. M., Subramanian, A., & Ochi, L. S. (2015). An iterated local search heuristic for the split delivery vehicle routing problem. *Computers & Operations Research, 53*, 234–249. https://doi.org/10.1016/j.cor.2014.08.005. http://www.sciencedirect.com/science/article/pii/S0305054814002159

Sörensen, K. (2015). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research, 22*(1), 3–18.

Tarantilis, C., Zachariadis, E., & Kiranoudis, C. (2008). A hybrid guided local search for the vehicle- routing problem with intermediate replenishment facilities. *INFORMS Journal on Computing, 20*(1), 154–168.

Teixeira, J., Antunes, A. P., & Sousa, J. P. (2004). Recyclable waste collection planning – a case study. *European Journal of Operational Research, 158*(3), 543–554.

Toth, P., & Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing, 15*(4), 333–346. https://search-proquest-com.kuleuven.ezproxy.kuleuven.be/scholarly-journals/granular-tabu-search-application-vehicle-routing/docview/200521672/se-2?accountid=17215

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research, 60*(3), 611–624. https://doi.org/10.1287/opre.1120.1048. https://pubsonline-informs-org.kuleuven.ezproxy.kuleuven.be/doi/10.1287/opre.1120.1048

Willemse, E. J., & Joubert, J. W. (2016). Constructive heuristics for the mixed capacity arc routing problem under time restrictions with intermediate facilities. *Computers & Operations Research, 68*, 30–62. https://doi.org/10.1016/j.cor.2015.10.010. https://www.sciencedirect.com/science/article/pii/S0305054815002415