An Adaptive Large Neighbourhood Search for Real-Time Waste Collection Inventory Routing with Autonomous Vessels

Tomas Gast[1 2], *Bilge* Atasoy[2], *Fabio* Duarte[1], *Daniela* Rus[1]

[1] *Department of Urban Sciences and Planning, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*
[2] *Transport Engineering and Logistics, Department of Maritime and Transport Technology, Delft University of Technology, Delft, 2628 CD, The Netherlands*

1 Introduction and Problem Definition

The city of Amsterdam has recently installed environmental zones to reduce air pollution. The zones prevent diesel engines from entering, which caused a surge of heavy electric garbage trucks in Amsterdam [1]. Unfortunately, these heavy vehicles induce subsidence of the city's 'grachtengordels'. To prevent this, the city of Amsterdam is exploring other methods of transportation [2]. In [3], Daub presents a solution for the household waste collection, using the autonomous vessels developed by the roboat project to pick-up containers situated at 283 predefined locations [1]. Resembling the current pick-up schedule, Amsterdam was divided into sections, with a weekly pick-up day assigned to each section. In addition to household waste, companies in the hospitality sector located near the canals also generate waste. This waste however, is often collected by third party companies [4]. Two consequences of this system are: (i) no data is gathered and (ii) the third party companies continue using the simpler heavier garbage truck. In this paper, we consider the waste generated by hotels along the canals in order to address this.

We formulate a real-time waste collection inventory routing problem and propose an efficient Adaptive large Neighbourhood Search (ALNS) algorithm for the solution. In this problem, the vessels have a capacity of three full containers, totalling a maximum weight of one ton. A single depot exists, where the vessels are emptied and recharged. When a vessel arrives to pick-up a container, it is swapped with an empty one. In this system, each container measures the amount of waste it contains and the time since it was last served. Both parameters are then combined into a score, i.e., the score increases with the waste level and with the time since last visit. This score is minimised by the ALNS algorithm to optimise the waste collection problem. In Figure (1), a single vessel's solution for an arbitrary network is illustrated. In this example, the vessel recently picked up container B, which is now on-board. Container B's weight and time since service are therefore very low. Since this container should not be revisited any time soon, the score equals zero. Containers C and D both have a low score due to the large amount of waste and the long time since being serviced respectively, and are thus included in the vessel's route. Container A however, is less attractive to visit, since its score is relatively high, and is thus excluded from the route.

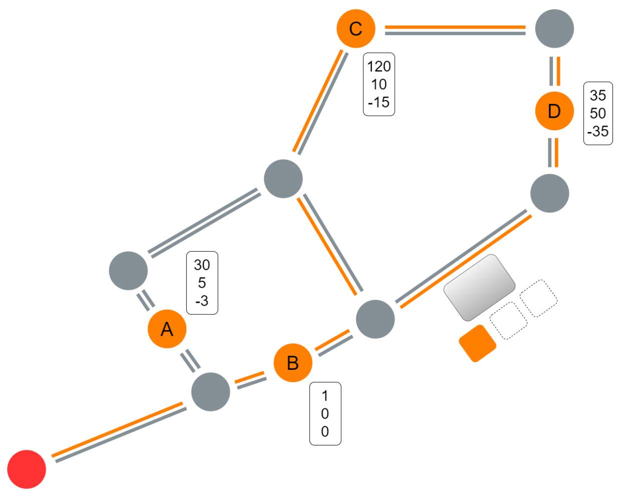

Figure 1: Illustration for a single vessel. The orange arcs represent the remaining route, the orange nodes represent containers. Each container is indicated by its weight, time since last service and the associated score (top - bottom). The grey arcs connect the grey nodes in both directions. The rectangular vessel carries one full and two empty containers. The red node is the depot.

The waste collection problem has existed since 1974, when Beltrami and Bodin aimed to improve the efficiency of the New York City Department of Sanitation [5]. The efficiency of such departments was found to be highly important, as up to 80% of waste disposal costs occur during the collection phase [6], [7]. For this specific type of optimisation problem, heuristics have been heavily used due to the computational demand of real world problem sizes.

[1]The roboat project is a collaboration between the Amsterdam Institute for Advanced Metropolitan Solutions and the Massachusetts Institute of Technology. In developing a fleet of autonomous floating vessels for the city of Amsterdam, it investigates the potential of self-driving technology to change our cities. http://www.roboat.org

Today, the rise of the Internet of Things (IoT) and cheap electronics, allow the design of smart low cost and low power power trash bins, that measure their weight, filling volume and temperature [8], [9].

The ALNS was introduced for a pick-up and Delivery Problem with Time Windows (PDPTW) by Ropke and Pisinger [10]. It was found to be advantageous to use multiple competing subheuristic operators. Since then, the ALNS has been used for different problems and with more complex operators.

In this paper, an efficient ALNS algorithm is proposed, which includes a roulette wheel selection mechanism that chooses a single operator at each iteration based on the operators past performance, accepts new solutions based on simulated annealing , uses A* as the route planning algorithm and allows the infeasible regions to be searched by having soft capacity constraints with penalty terms in the objective function. This ALNS algorithm is then compared to static problems to select the best performing parameters. We evaluate the performance with a case study of 86 hotels facing the canals in Amsterdam. Besides, the concept of the collection system making use of smart containers and small autonomous vessels is another novelty.

2 Methodology

The methodology consists of two types of processes: (i) real-time simulation of the environment each time step and (ii) (re-)optimising a static waste collection problem with the updated inputs generated by the environment. With these two processes, the dynamic nature of the problem is ensured. In Table (1), we provide the used notation. We first present how the real-time simulation is handled and then present the static waste collection problem which is used to re-optimise the system at time steps where new information is received.

2.1 Real-time simulation

First, at each time step, the progress of all vessels is simulated. If t exceeds the opening hour μ_t and lower than the closing hour σ, all vessels move forward at a constant speed. Their progress is saved as a fraction of the current edge length, and once this fraction exceeds 1, the current edge is deleted and the remaining progress is carried over to the next edge. If the deleted edge contains a pick-up location, the collected waste is registered and removed from container's waste level, $w_{i,t}$, the container $\eta_{i,t}$ is updated to reflect the last visit time, the vessel's remaining capacity $\gamma_{k,t}$ is adjusted and δ is subtracted from the progress of the next edge.

Secondly, for each time step a vessel spends at the depot, its battery level increases by κ_k. Note that each vessel is at the depot for at least Δ hours, before departing again.

Thirdly, at each time step, each container generates waste therefore the weight $w_{i,t}$ is updated. The amount generated over all time steps in a day is equal to the expected amount of waste generation for each hotel.

Finally, at each time step, and for each vessel, the remaining length of each route and the score of each container $\lambda_{i,t}$, and thus the total score $f(s)_t$, as explained by equation (2) and (1), is recalculated.

2.2 The mathematical formulation for the static waste collection problem

The waste collection problem is formulated as a MILP. The objective function (1) is built up of three parts: costs, score and distance travelled. The first cost, C_k, represents the cost of using a vessel and variable z_k keeps track of the utilisation of each vessel k. Second cost term is related to the score of each container and the third term is the transportation cost. Additionally, ς_k and ψ_k are added as penalties for violating the capacity constraint of the vessel and the battery capacity respectively. This allows the ALNS to escape local optima by considering infeasible solutions during the course of the algorithm. When the constraint is not violated, the cost equals 0.

$$f(n)_t = \sum_{k\in K} C_k z_k + \sum_{k\in K}\sum_{i\in N}(\varsigma_k + \lambda_{i,t})y_{i,k} + \\ \sum_{k\in K}\sum_{j\in N}\sum_{i\in N}\sum_{p\in P}(\psi_k + \pi_{i,j,t})x_{i,j,k,p} \quad (1)$$

Secondly, the score $\lambda_{i,t}$ of each container is calculated at each $t \in T$ by equation (2) which basically says that the score is 0 when we are below the time and waste level limits and at its maximum magnitude, -40, when we are above those limits. When it between, the score depends on the sensitivity to the waste level, ρ and sensitivity to the time since last visit, ϵ.

$$\lambda_{i,t} = \begin{cases} 0 & \text{if } \eta_{i,t} < \xi \\ 0 & \text{if } \omega_{i,t} < \varepsilon \\ -40 & \text{if } \eta_{i,t} \geq \zeta \\ -40 & \text{if } \omega \geq \vartheta \\ -\rho\omega_{i,t} - \epsilon\eta_{i,t}, & \text{otherwise} \end{cases} \quad (2)$$

The objective function is subject to several constraints including basic vehicle routing constraints with single depot, capacity of the vessels, battery capacity and time.

2.3 Adaptive large neighbourhood search

At each iteration of the ALNS, a single operator is chosen through a roulette wheel selection mechanism and a neighbouring solution s' is proposed. This solution is evaluated based on a simulated annealing approach. A solution with a better score, $f(s')_t < f(s)_t$ is always accepted, while the probability of accepting a worse solution is given by:

$$P(s = s') = P\Big((f(s)_t - f(s')_t) \quad rnd() < T_0 c^n\Big) \quad (3)$$

In this simplified form of simulated annealing [11], the initial temperature T_0 decreases by a cooling factor c

Table 1: Notations

Sets			
K	Set of vessels $k \in K$	T	set of time steps $t \in T$
N	Set of nodes $i, j \in N$	P	Set of route sections $p \in P$
Parameters			
$\pi_{i,j}$		Travel distance of arc(i,j).	
μ_t, σ		Lower and upper bounds for the time window for all vessels at time t.	
δ		Service duration of all containers.	
Δ		Depot time of all vessels.	
ε, ϑ		Lower and upper bounds for the pick-up waste level for all containers.	
h_i		Binary indicating if node i is a pick-up location.	
ξ, ζ		lower and upper time limit between pick-ups for all containers.	
$d_{k,t}$		Maximum travel distance of vessel k at time t, based on its battery level.	
κ_k		Charging speed of vessel k at the depot.	
ρ		Waste scoring factor of all containers.	
ϵ		Time scoring factor of all containers.	
C_k		Cost of using vessel k.	
ς_k		Cost of surpassing the capacity of vessel k.	
ψ_k		Cost of surpassing the battery capacity $d_{k,t}$ of vessel k.	
$\omega_{i,t}$		Waste level of container i at time t.	
$\eta_{i,t}$		Time since container i was last served at time t.	
$\lambda_{i,t}$		The score associated with container i at time t.	
$S_{k,t}$		Starting time of vessel k at time t.	
R_i		Number of rooms linked to container i.	
s		Current solution.	
n		Iteration step in the ALNS.	
Binary Decision Variables			
$x_{i,j,k,p}$		if vessel k traverses edge $(i - j)$ as the p-th section of its route, 0 o/w.	
$y_{i,k}$		1 if vessel k picks up the waste on point i, 0 otherwise (binary).	
$z_{k,t}$		1 if vessel k is used at time t, 0 otherwise (binary).	

at every iteration n. After the evaluation of the solution, the performance of the chosen operator is registered as $f(s)_t - f(s')_t$. The probability of choosing this operator during the next iteration is then based on its past performance over the last 20 executions with respect to the other operators. A predetermined probability order is awarded to the operators $P_{op,rnk}$ and to the operator category $P_{cat,rnk}$ based on their ranking. The probability of choosing a specific operator is thus given by their product: $P_{op,rnk}P_{cat,rnk}$. To obtain a more predictable CPU time, the stopping criterion is set by a maximum number of iterations. Note that, in this problem, not all vessels in K have to be used, and not all customers in h_i have to be visited, so the number of customers or vessels removed does not necessarily equal the number inserted.

There are various operators defined within the ALNS algorithm. The following are the change operators:

- *Swap Operator.* Randomly removes a container from a random vessel and swaps it with another randomly chosen vessel and container.
- *Reorder.* Randomly chooses a container of a random vessel and swaps the visiting order.
- *Donate Operator.* Randomly chooses a container in a random vessel and donates it to another vessel. The receiving vessel and visit order are chosen based on the minimum euclidean distance.

The destroy operators are given as:

- *Random customer removal.* Removes a random container from a random vessel.
- *Worst customer removal.* Chooses a random vessel and finds the worst performing container pick-up and removes it for the vessel's route.
- *Remove random vessel.* Removes the entire route of randomly chosen vessel.

Finally, the following repair operators are used:

- *Random customer insertion.* Picks a random container not in s and places it at random location on a random vessel's route.
- *Closest customer insertion.* Picks a random container not in s and places it on the closest path of an existing route.
- *Greedy vessel insertion.* Finds a vessel with no route in s and builds a route until it reaches maximum capacity or until no containers exist with non-zero waste, i.e.

$\lambda_{i,t} > 0$. The containers and their order are chosen by consecutively selecting the closest option in terms of euclidean distance.

2.4 A* path planning

The A* algorithm was first presented in [12] and can be viewed as a heuristic extension of Dijkstra's algorithm [13].

Consider the simple network in Figure (2), where a vessel travels from the depot to container A to C, and back to the depot. Between A and C the route: depot - A-1-3-4-6-C - depot is used. When container B is added by the *random customer insertion* operator between A and C, the A* algorithm will create a temporary route: A-1-2-B-5-6-C. This temporary route is then inserted in the old route, creating: depot - A-1-2-B-5-6-C - depot.

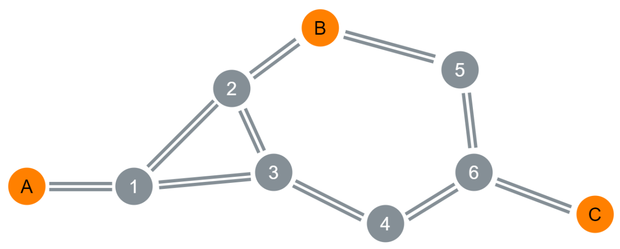

Figure 2: A detailed view of a graph network, showing how container node A,B and C are connected. The depot and the network connecting A and C to the depot is not shown.

3 Numerical Experiments

We performed numerical experiments for the Amsterdam case presented in the introduction section.

3.1 Parameter selection for the ALNS

In this section, the simulated annealing initial temperature T_0, the simulated annealing cooling factor c, the number of iterations n, the predetermined operator probability distribution $P_{op,rnk}$ and operator category probability distribution $P_{cat,rnk}$ are investigated. The other parameters: κ_k, $d_{k,0}$, υ, C_k, ς_k and ψ_k are set to 14.8 km/h, 40 km, 3.6 km/h, 5, 40 and 50 respectively.

In the dynamic version of the system, numerous initial situations exist, that can all require different strategies. Here, the aim is to choose T_0, c, $P_{op,rnk}$ and $P_{cat,rnk}$ in such a way that the ALNS performs well in both cases.

First, looking at the simulated annealing, the score of each container $\lambda_{i,t}$ is uniformly set to -15. Then, for different values of T_0 and c, the ALNS is run starting with an initially empty solution and twelve available vessels, and with an initially mediocre solution (named "full solution"). For the latter, a solution with the same number of pick-ups as the best found solution was chosen, to contrast the operator selection of the empty initial solution. The ALNS was run 500 times and obtained the best average solution for T_0 and c equal to 5 and 0.997 respectively. To simulate an ALNS that can only hill climb, the initial temperature T_0 was set to 10e-5 and the cooling factor to 1. For the empty and full initial solution, the average score equalled -471 ± 6.1 and 507 ± 0.2 respectively.

Next, since the initial solutions have shown to perform similarly, the influence of $P_{op,rnk}$ and $P_{cat,rnk}$ is investigated using only the empty initial solution. Since ALNS has three categories, each containing three operators, the probability is given by the following equation:

$$P_{cat,rnk} = \frac{\chi_{cat}^{(3-rnk)}}{\sum_{rnk=1}^{rnk=3} \chi_{cat}^{(3-rnk)}} \tag{4}$$

The value of $\chi_{(-)}$ is ranged between 0.7 and 2.95. The lowest score was achieved for $\chi_{op} = 2.5$ and $\chi_{cat} = 2.05$.

3.2 Benchmark comparison

Using the best performing parameters found in section (3.1), we compare the ALNS to an exact approach (Gurobi solver) under the static setting in two sets of experiments: (i) small size problem where the optimal solution is available and (ii) full size problem where optimal solution is not available and the best found solution is used instead. We present the mean values for ALNS with a 99% confidence interval. The ALNS is repeated 500 times for 850 iterations across all the experiments.

The small size problem consists of 19 nodes and 8 pick-up locations, sampled from the full size problem. The pick-up locations are given a different score, ranging from -1 to -12. The performance is measured in score, total distance and number of vessels used as displayed Table (2). It is shown that the ALNS approaches the optimal solution by 1.6% on average with a CPU time of 3.5s.

Table 2: Results for the small size problem

ALNS	Mean	Low	High
Score	-31.4 ± 0.06	-31.85	-30.15
Distance (km)	5.5 ± 0.10	5.15	7.85
Pick-ups	6.1 ± 0.08	6	8
Time (s)	3.5 ± 0.05	2.97	5.25

Optimal	Value
Score	-31.9123
Distance (km)	5.1234
Pick-ups	6
Time (s)	5×10^4

No optimal result for the static full size problem (where there are 402 nodes in total 86 of which are container locations) was available, the lowest found objective function, i.e., score, still provides insights into the performance of the algorithm. The same setup is used as in (3.1). The results are displayed in Table (3). From these results, it can be seen that the algorithm approaches the best found score for the full size problem by 4.2% on average.

Table 3: Results for the full size problem with ALNS

ALNS	Mean	Low	High
Score	-378 ± 1.1	-393.86	-344.42
Distance (km)	97 ± 0.66	86.14	122.27
Pick-ups	35.7 ± 0.08	33	36
Time (s)	6.0 ± 0.02	5.55	6.63

Best found	Value
Score	-393.86
Distance (km))	86.14
Pick-ups	33

3.3 Dynamic experiments

The performance of the ALNS is measured over $t = \{0, ..., 40320\}$, with t representing one minute. Here, the performance is indicated by the overflowing frequency when $w_{i,t} > \vartheta$. The values for μ_0, σ_t, δ, ε, ϑ, ξ and ζ are 7 am, 20 pm, 5 min, 50 kg, 300 kg, 6 h and 120 h respectively. The maximum number of vessels is 20 and the re-optimisation frequency is 12 min.

The value of ρ and ϵ represent the planner's sensitivity to the waste level $w_{i,t}$ and time since last visit $\eta_{i,t}$ while their ratio prioritises one's scoring over the other. In Figure (3), ρ and ϵ are varied.

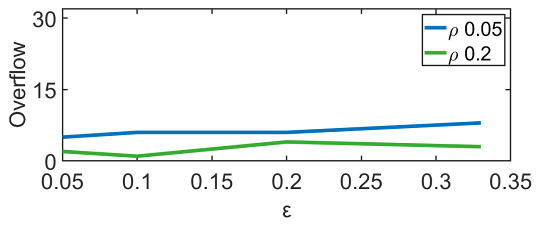

Figure 3: The Number of overflowing containers for different values of ρ and ϵ, using the ALNS

The ALNS is able to achieve a lower overflow and a higher efficiency compared to a greedy method that works based on a threshold value, without compromising on the late pickups. Choosing these values means that over a one month period, we can expect approximate 1728 pickups, 6 overflowing containers, 302 late pickups and an efficiency of 32.6 kg per km travelled.

4 Conclusions and Future Directions

This paper presents a novel approach to the waste collection problem for the canals in Amsterdam with autonomous vessels and smart containers. An efficient ALNS algorithm is proposed for the solution of the problem and thanks to re-optimisation, it is able to respond to the environment's waste generation in a dynamic fashion, while avoiding overflowing of containers and improving the routing efficiency. The computational time is small enough to allow for a time step of one-minute.

Ongoing work investigates different waste generating scenarios and multiple re-optimisation strategies. Future work could look into a heterogeneous set of containers with different sizes, to reduce the number of late picks. Besides, smarter, more complex operators can be investigated to enable the ALNS converge to a better solution. Furthermore, stochastic version of this framework is a promising future direction.

References

[1] D.N. Staat, ed., *Milieuzones in Nederland* (Amsterdam, The Netherlands, 2019), `https://www.milieuzones.nl/uitleg-milieuzones`

[2] A.I. for Advanced Metropolitan Solutions, ed., *What we do* (Amsterdam, The Netherlands, 2019), `https://www.ams-institute.org/about-ams/who-we-are/`

[3] V.B. Daub, *Routing and Scheduling of Autonomous Boats for Amsterdam's Waste Collection*, Technische Universität Darmstadt **1** (2019)

[4] R. Koops, ed., *Amsterdam begint de afvalcrisis te voelen* (Het Parool, Amsterdam, The Netherlands, 2019)

[5] E.J. Beltrami, L.D. Bodin, *Networks and vehicle routing for municipal waste collection*, Networks **4**, 65 (1974)

[6] U. Sonesson, *Modelling of waste collection - a general approach to calculate fuel consumption and time*, Waste Management & Research **18**, 115 (2000)

[7] V.N. Bhat, *A Model for the Optimal Allocation of Trucks for Solid Waste Management*, Waste Management & Research **14**, 87 (1996)

[8] Á. Lozano, J. Caridad, J.F. De Paz, G. Villarrubia Gonzalez, J. Bajo, *Smart waste collection system with low consumption LoRaWAN nodes and route optimization*, Sensors **18**, 1465 (2018)

[9] S.K.T. Utomo, T. Hamada, N. Koshizuka, *Low-Energy Smart Trash Bin Architecture for Dynamic Waste Collection System*, in *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems* (Association for Computing Machinery, New York, NY, USA, 2018), ICFNDS '18, ISBN 9781450364287

[10] S. Ropke, D. Pisinger, *An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows*, Transportation Science **40**, 455 (2006)

[11] E. Aarts, J. Korst, *Simulated annealing and boltzmann machines* (1988)

[12] P.E. Hart, N.J. Nilsson, B. Raphael, *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, IEEE Transactions on Systems Science and Cybernetics **4**, 100 (1968)

[13] E.W. Dijkstra et al., *A note on two problems in connexion with graphs*, Numerische mathematik **1**, 269 (1959)