Reoptimizing the Vehicle Routing Problem with Urgent Stochastic Customers

Fabian de Vries

December 3, 2021

Abstract

In this paper we propose an alternative online solution to the Vehicle Routing Problem with Urgent Stochastic Customers, a Vehicle Routing Problem with stochastic customers and demands where lowering response times to service requests is the main objective. Here, the bi-objective problem of minimizing tours while also minimizing response times to urgent stochastic customers needs to be solved, taking longer tours as a cost of reducing response times. Our proposition is to recalculate tours whenever stochastic customers request service, using a more efficient (but less optimised) Vehicle Routing Problem with Time Windows. The proposed heuristic (Reoptimization Heuristic) is evaluated through a simulation study. Results show the Reoptimization Heuristic to deliver an increase in travel times without improvement of the response times. However, results indicate that the Reoptimization Heuristic becomes more efficient with a frequency of urgent requests.

1 Introduction

In hospitals, hygiene is a top priority. On a daily basis, a team of janitors are making sure that the Jeroen Bosch Hospital in the Netherlands is always living up to the high hygiene standards the hospital has. However, to do so the hospital splits up the cleaning crew in order to deal with unexpected clean-ups. One team takes care of the a-priori cleaning jobs, while the other team is on permanent standby to make sure any unexpected cleaning jobs can be taken care of as soon as possible. This raises the question: Can all cleaners be deployed such that there is no need for a stand-by crew, while these cleaners can quickly respond to unexpected cleaning jobs, whenever they occur?

This question inspired the development of the Vehicle Routing Problem with Urgent Stochastic Customers [2]. In a Vehicle Routing Problem, the goal is to find the optimal set of tours for a fleet of vehicles to traverse, in order to service a given set of customers. In our situation, the vehicles are our cleaners, the customers are locations that need to be cleaned and the service is the time a cleaner needs to finish cleaning that location. Since we make a distinction between predetermined customers and emergencies, we will refer to them as regular customers and urgent customers, respectively. The urgent customer requests occur with a Poisson distributed probability, and are referred to as urgent stochastic customers.

In this paper, we will take a look at a set of customers where the locations of all customers are known, but if and when an urgent customer will request service is unknown. We propose an alternative heuristic to the Vehicle Routing Problem with Urgent Stochastic Customers, where we recalculate tours of all vehicles whenever an urgent request occurs. We analyse the heuristic through a simulation study, comparing our heuristic to the heuristic proposed by [2].

2 Literature

2.1 Vehicle Routing Problems

Vehicle Routing Problems are optimization problems that aim to find a set of optimal tours for a given fleet of vehicles and a set of locations that need to be serviced. Generally, the tours are deemed optimal if the set of tours minimizes some predetermined cost, for example travel distance.

The Vehicle Routing Problem (VRP), which' first heuristic solution appeared in "The Truck Dispatching Problem" by G.B. Dantzig [4], is a generalization of the Traveling Salesman Problem (TSP). In a TSP, the aim is to find the shortest closed tour for a vehicle (the salesman) between a set of given customers. In a VRP, there is a set of vehicles that have to serve these customers, therefore spreading the workload. Since finding the optimal solution to a VRP is a NP-hard problem, [8], the time needed to find the optimal solution to any VRP becomes polynomial in time. Therefore, a heuristic approach to the problem is required. Probably the most well known heuristic to a VRP is by the hands of Clarke and Wright, called the Savings Algorithm [3].

Additional constraints Vehicle Routing Problems generally come with restrictions. Constraints, as these restrictions are called, put additional strain on the viable solutions for a VRP. In case of the Savings Algorithm, the vehicles were limited by the load they could carry.

In this paper, vehicles are constrained by time windows given for each customers in which that customer needs to be serviced, a constraint first proposed by Linus Schrage in his paper "Formulation and structure of more complex/realistic routing and scheduling problems"[12]. If not, a penalty is imposed on the found solution.

Another variant of the VRP are Dynamic Vehicle Routing Problem with Stochastic Demand (DVRP), of which a comprehensive review is presented in Pillac [10]. In a VRPSD, all or part of the service requests are modeled as a stochastic variable. Generally, DVRPs aim to minimize travel time by combining an a-priori calculated set of tours together with a strategy, if and when an urgent customer requests service ([1], [7], [9]). However, these DVRPs try to minimize the travel distance. In a current study by Bos [2], the Vehicle Routing Problem with Urgent Stochastic Customers is proposed. This VRP is trying to minimize response times to stochastic customers requesting service during the day.

2.2 Definitions

Locations Let us denote the set $R = r_1 \dots r_n$ as the set of n regular customers and the set $U = u_1 \dots u_m$ as the set of m urgent customers. Let d_s be the start depot, d_e be the end depot.

Graph Let $V = R \cup U \cup \{d_s, d_e\}$ be the set of all nodes in the graph and $E = \{(i, j) | i, j \in V, i \neq j\}$. The time to traverse edge $(i, j) \in E$ is denoted as t_{ij}. The VRPs are defined on the undirected graph $G = (V, E)$.

Tours Let $T = t_1 \dots t_k$ be the set of all tours, where k is the number of vehicles in the fleet. A tour is a closed path of a vehicle which starts and ends at a depot and visits at least one location that is not a depot.

Time Every regular customer $r \in R$ is associated with a time window $[a_r, b_r]$ in which a service needs to start. The service time at this customer has length o_r. For every edge $(i, j) \in E$, t_{ij} is the time it takes to travel from i to j over that edge. w_{ij} is the arrival time of vehicle k at customer i. If s be a feasible solution to the vehicle routing problem and S is the set of all feasible solutions, then, given solution s, the latest arrival time at the end depot between all vehicles is t_s.

2.3 VRP algorithms

A mathematical approach is needed in order to find near optimal solutions. In this section, we go over how to define a VRP, as well explaining a few general VRPs in depth.

The Savings Algorithm The Savings Algorithm,[3], works by assigning each customer to their own tour and merging these tours until no further savings can be achieved. Savings are computed for every two customers in the VRP and listed in descending order. Then, in this order these tours are merged, but only if both customers are direct neighbours of the depot, are not already part of the same tour and merging the tours does not violate other given restrictions. The Savings Algorithm was proposed as a fast alternative to the algorithm proposed by Dantzig and Ramsers' Truck Dispatching Problem [4], where trucks are send to deliver gasoline to service stations, but are limited by their carry capacity. The Savings Algorithm is as follows):

1. Create n tours: $d \to r_i \to d$, for each $r_i \in R$, where d is the start and end depot;
2. Compute the savings for merging delivery locations i and j, which are given by: $s_{ij} = t_{id} + t_{jd} - t_{ij}$, $\forall i, j \in R, i \neq j$;
3. Sort all s_{ij} in descending order;
4. Starting at the top of the list of savings, merge the two tours associated with the largest remaining savings, provided that:
    (a) Neither i or j are part of the same tour before merging;
    (b) Neither delivery location is interior to its route (both i and j are still directly connected to the depot in their respective tours;
    (c) The demand G and distance D are not violated by the merged tour;
5. Repeat step 4 until no additional savings can be achieved.

For a visualisation of this algorithm, see figure 2.3

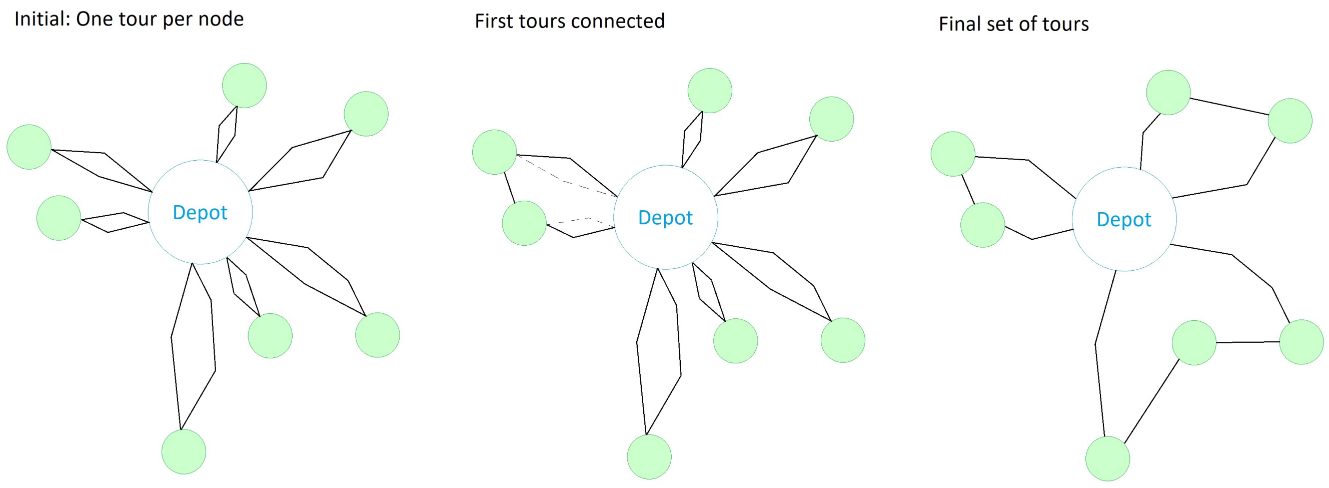

Figure 1: Creating initial tours, merging two tours and the final result of the Savings Algorithm

Sequential insertion heuristic The Vehicle Routing Problem with Time Windows (VRPTW) is the extension of a capacitated vehicle routing problem, where the service at each customer must start within its associated time window and the vehicle must remain at the customer location during service. Time windows can be soft, meaning they can be violated at a cost, or they can be hard. With hard time windows, if a vehicle arrives before the time window, it waits. In this paper, hard time windows are implemented for the VRPs.

As mentioned in Chapter 2.1, VRPs are NP-hard. Even finding a feasible solution to the VRPTW with a fixed fleet size is a NP-complete problem [11]. In 1987, Solomon [13] proposed the Sequential Insertion algorithm. This algorithm assumes partial tours are already found, and new customers must be added to these tours. Then, non-served customers are added to the current set of tours by inserting them at the 'best' position. Once it is not possible to insert a customer into the current trip, the algorithm start with a new trip. For convenience, we will refer to non-served customers as 'candidates'. The algorithm works as follows:

1. Select a vehicle with the largest capacity constraint that has not been selected by this heuristic;
2. List all candidates feasible to service for the vehicle. If none can be found, we go back to Step 1;
3. Calculate the best possible insertion location for each candidate based on the minimum additional distance and time required and list them from in decreasing order;
4. List candidates based on the additional distance and time required, in increasing order;
5. Starting at the top of the list, insert candidates from the list to the tour as long as this does not violate any constraints, similar to the maximum savings concept of the Savings Algorithm;
6. Once no more candidates can be added to a tour, go back to step 1.

When no more candidates with feasible insert solutions can be found, the algorithm is terminated.

2.4 Mixed Integer Linear Programs

In the previous section we took a look at some examples of heuristic VRP algorithms. Although these provide a better understand of how heuristics come to be, they are not used in this paper.

In this section, VRPs based on Mixed Integer Linear Programming (MILP) are introduced. An early example of this is Koskosidis, Powell, and Solomon [14], where an heuristic by Fisher and Jaikumar [5] for a VRPTW is generalized. Here is where we introduce the standard VRP and extend it to a VRPTW, as well as the VRPUSC proposed by Bos [2].

A Basic Vehicle Routing Problem The most basic version of an Mixed Integer Linear Program for a VRP is a VRP that can describe that most basic goal: Given a set of vehicles and customers, find a set of tours that minimize the total cost of the tours. In this case, this cost is equal to the total distance traveled by the fleet.

Let us first take a look at the goal of our VRP, called the Objective Function. Assume that a vehicle travels from node i to node j without visiting any other nodes in between. Then, let c_{ij} be the cost of traveling from node i to node j, which is the sum of the expected service time at node i plus the expected travel time between i and j. The total time a vehicle spends to complete a tour is the sum of all costs c_{ij}, where (i, j) are all edges that are part of this tour.

To filter the costs c_{ij} that are part of the tour of vehicle k, we introduce a decision variable x_{ijk}. As the name suggests, these decision variables decide the dimensions in which the VRP can find a solution and must be determined by our VRP. We define $x_{ijk} = 1$ whenever edge (i, j) is part of the tour of vehicle k, and $x_{ijk} = 0$ otherwise. Its clear to see that $c_{ij}x_{ijk} = c_{ij}$ if (i,j) is part of the tour, and $c_{ij}x_{ijk} = 0$ otherwise. We can sum $c_{ij}x_{ijk}$ over all edges in the graph for a vehicle k to get the total cost of the tour of vehicle k: $\sum_{i \in V} \sum_{j \in V} c_{ij}x_{ijk}$. When we sum the tour costs over all vehicles in the fleet, we get the total cost of a set of tours of our VRP. This sum is our objective function:

$$\min \sum_{k \in K} \sum_{(i,j) \in E} c_{ij}x_{ijk} \tag{1}$$

Now, the constraints of our standard VRP are as follows:

$$\sum_{j \in d_e \cup R} \sum_{k \in K} x_{ijk} = 1 \qquad \forall \quad i \in R \tag{2a}$$

$$\sum_{j \in R} x_{d_s jk} = 1 \qquad \forall \quad k \in K \tag{2b}$$

$$\sum_{i \in d_s \cup R} x_{ijk} - \sum_{i \in d_e \cup R} x_{jik} = 0 \qquad \forall \quad j \in R, k \in K \tag{2c}$$

$$\sum_{i \in R} x_{i,d_e,k} = 1 \qquad \forall \quad k \in K \tag{2d}$$

$$x_{ijk} \in \{0,1\} \qquad \forall \quad (i,j) \in E, k \in K \tag{2e}$$

Here, constraint (2a) ensures every customer has exactly one leaving vehicle. Therefore, every customer is included once in one tour, but not in any other. Constraint (2b) enforces every vehicle to leave the start depot once to visit a customer. Constraint (2c) implies that for every vehicle servicing a customer, that vehicle will depart from that customer. Together, these constraints imply no tour can start or end at a customer. Constraint (2d) makes sure all vehicles will visit the end depot exactly once. With these constraints, every tour needs to begin at the starting depot, visit at least one customer and finish at the end depot. The last constraint defines the decision variable. Due to constraint (2e), our decision variable x_{ijk} is defined as a binary value.

Time Windows In order to extend this VRP to a VRP with Time Windows, the decision variable w_ik is introduced. This decision variable defines the time vehicle k starts service at customer i. w_ik is defined to always be positive, and $w_ik = 0$ if vehicle k does not service costumer i. With this, the VRPTW can be defined by extending the standard VRP with the following additional constraints:

$$x_{ijk}(w_{ik} + o_i + t_{ij} - w_{jk}) \leq 0 \qquad \forall \quad (i,j) \in E, k \in K \tag{3a}$$

$$a_i(\sum_{j \in d_e \cup R} x_{ijk}) \leq w_{ik} \leq b_i(\sum_{j \in d_e \cup R} x_{ijk}) \qquad \forall \quad i \in R, k \in K \tag{3b}$$

$$w_{ik} \geq 0 \qquad \forall \quad i \in V, k \in K \tag{3c}$$

Here, constraint (3a) imposes the start of a service at a customer to be feasible in time by implying that the start time of service at customer j is at least equal to the start of service at the previous customer i, plus the time customer i takes to be serviced and the travel time for i to j. Note that the constraint always holds when (i,j) is not a part of one of the tours. Furthermore, constraint (3b) limits the start of service at customer i to be within the customer's time window $[a_i, b_i]$. Constraint (3c) enforces tours to not service customers for $t < 0$.

3 The Vehicle Routing Problem with Urgent Stochastic Customers

In Bos [2], a VRP called the Vehicle Routing Problem with Urgent Stochastic Customers (VRPUSC) is proposed. The VRPUSC aims to find a set of tours such that expected response times for urgent customers are minimized, while also minimizing the total time needed for completing all tours. With this VRP, we can find a set of tours at the start of a day.

Note that this problem is two-fold. First, we want to minimize expected response times to stochastic urgent requests. Second, we want to minimize travel distances when servicing all regular customers. Since response times are directly related to travel distances, our first goal implies minimal travel distances to urgent customers as well.

Adjusted VRPTW The VRPUSC is based on the VRPTW in (2.4). However, constraint (3a) is substituted for the following constraint:

$$x_{ijk}(w_{ik} + o_i + t_{ij} - w_{jk}) = 0 \qquad \forall \quad (i,j) \in E \cup R, k \in K \tag{4}$$

When interchanging constraint (3a) with constraint (4), we limit w_{ik} further by enforcing an upper bound that is equal to the earliest time a service can start at node i. This way, every customer in the tour must start service as soon as the vehicle arrives at the customer.

3.1 Proximity Scheduling Problem

The VRPTW is extended with an extra set of constraints and an extra objective function. This extention is called the Proximity Scheduling Problem (PSP).

The goal of proximity scheduling is to minimize the expected response time for each urgent customer at any moment during the tour. By assigning a vehicle to every urgent customer at any time in the makespan of the tours, an expected response time for a set of tours can be calculated. By minimizing their sum, an objective function can be obtained:

Objective function Let u be an urgent customer, $(i,j) \in E$ be an edge in the graph, and τ_{iju} the expected response time from (i,j) to u. In this paper, service times are assumed to be 0 and the travel speed of a vehicle is constant and the same for every vehicle in the fleet. Therefore, τ_{iju} can be calculated as the average response time while traversing from customer i to j. By integrating over the distance of each point on (i,j) to u and dividing over the distance of (i,j), τ_{iju} can be determined. Observe that this integration is possible since Poisson arrivals of urgent customers implies that requests for each u arrive uniformly over time [6].

Now, let q_{iju} be a decision variable, that describes time edge (i,j) is assigned to urgent customer u. $\tau_{iju}q_{iju}$ is an approximation of the integral over the response time to u during the time q_{iju}. By summing over all edges for a given u, we have an approximation of total expected response time during for that urgent customer. The lower the sum, the lower the total response time. In order to get the same dimensions as the objective function for the VRPTW, q_{iju} is multiplied by t_s. This way q_{iju} can be interpreted as the unitless fraction of the total time edge (i,j) is assigned to customer u.

Since we want to minimize all response times, all that is left to do is to take the sum of these values for all urgent customers and minimize as such. This gives us the following objective function:

$$\min \frac{1}{t_s} \sum_{u \in U} \sum_{(i,j) \in E} y_{iju} \tau_{iju} q_{iju} \tag{5}$$

Constraints The PSP is constrained as follows:

$$\sum_{k \in K} x_{ijk} \geq y_{iju} \qquad \forall \quad (i,j) \in E, u \in U \tag{6a}$$

$$q_{iju} \leq y_{iju} M_2 \qquad \forall \quad (i,j) \in E, u \in U \tag{6b}$$

$$w_{ik} - (1 - x_{ijk}) M_3 \leq v_{iju} + (1 - y_{iju}) M_3 \qquad \forall \quad (i,j) \in E, k \in K, u \in U \tag{6c}$$

$$v_{iju} + q_{iju} - (1 - x_{ijk}) M_3 \leq w_{jk} + (1 - y_{iju}) M_3 \qquad \forall \quad (i,j) \in E, k \in K, u \in U \tag{6d}$$

$$\sum_{i \in V} \sum_{j \in V} q_{iju} \geq w_{d_e,k} \qquad \forall \quad k, l \tag{6e}$$

$$v_{iju} + q_{iju} \leq v_{cdu} + (1 - f_{ijucd}) M_3 \qquad \forall \quad (i,j),(c,d) \in E, (i,j) \neq (c,d), u \in U \tag{6f}$$

$$v_{cdu} + q_{cdu} \leq v_{iju} + f_{ijucd} M_3 \qquad \forall \quad (i,j),(c,d) \in E, (i,j) \neq (c,d), u \in U \tag{6g}$$

$$y_{iju}, f_{ijucd} \in \{0,1\} \qquad \forall \quad (i,j),(c,d) \in E, (i,j) \neq (c,d), u \in U \tag{6h}$$

$$v_{iju}, q_{iju} \geq 0 \qquad \forall \quad (i,j) \in E, u \in U \tag{6i}$$

Constraints (6a) ensures that an edge $(i,j) \in E$ can only be assigned to urgent customer $u \in U$ when the edge (i,j) is a part of the routing solution, i.e. y_{iju} can be 1, if and only if $x_{ijk} = 1$ for some $k \in K$. Constraint (6b) puts an upper bound to the total time an edge $(i,j) \in E$ can be assigned to an urgent customer and together with (6a) bounds $0 \leq q_{ijl} \leq M_2$ if (i,j) is part of the routing solution and $q_{iju} = 0$ otherwise. Constraints (6c) and (6d) impose that an assignment of $(i,j) \in E$ to $u \in U$ must fall within the interval between arrival times of a vehicle in i and j. Here constraint (6c) gives us a lower bound and constraint (6d) an upper bound. Note that these constraints are always true if either x_{ijk} or y_{iju} or both are 0. Constraint (6e) enforces that the sum of the assignment duration for each urgent customer u has to at least match the latest arrival in the depot and therefore ensuring that there will always be a vehicle coupled to every urgent customer if not all vehicles have finished their tour. Since we want a unique vehicle coupled to an urgent customer at any time, constraints (6f) and (6g) make sure that assignments for one urgent customer don't overlap in time. Constraints (6h) and (6i) determine the scope of the decision variables. The large constant M_2 should be at least t_{ij} and M_3 should at least be t_s.

3.2 VRPUSC

The VRPUSC is a bi-objective problem. We want to minimize response times while also minimizing travel distances of all vehicles. The first goal can be achieved by a PSP, while the latter can by a VRPTW. Since the aim of the VRPUSC is a characterisation of balance between the two objective functions, a scalarization is implemented using a weight factor $\alpha \in [0,1]$.

The VRPUSC can now be described using the constraints for the VRPTW and PSP described above:

$$\min \alpha \sum_{k\in K} \sum_{(i,j)\in E} c_{ij} x_{ijk} + \frac{1-\alpha}{t_s} \sum_{u\in U} \sum_{(i,j)\in E} y_{iju} \tau_{iju} q_{iju} \tag{7a}$$

subject to (7b)

$$\sum_{j\in d_e \cup R} \sum_{k\in K} x_{ijk} = 1 \qquad \forall \quad i \in R \tag{7c}$$

$$\sum_{j\in R} x_{d_s jk} = 1 \qquad \forall \quad k \in K \tag{7d}$$

$$\sum_{i\in d_s \cup R} x_{ijk} - \sum_{i\in d_e \cup R} x_{jik} = 0 \qquad \forall \quad j \in R, k \in K \tag{7e}$$

$$\sum_{i\in R} x_{i,d_e,k} = 1 \qquad \forall \quad k \in K \tag{7f}$$

$$x_{ijk}(w_{ik} + o_i + t_{ij} - w_{jk}) = 0 \qquad \forall \quad (i,j) \in E \cup R, k \in K \tag{7g}$$

$$a_i(\sum_{j\in d_e \cup R} x_{ijk}) \le w_{ik} \le b_i(\sum_{j\in d_e \cup R} x_{ijk}) \qquad \forall \quad i \in R, k \in K \tag{7h}$$

$$\sum_{k\in K} x_{ijk} \ge y_{iju} \qquad \forall \quad (i,j) \in E, u \in U \tag{7i}$$

$$q_{iju} \le y_{iju} M_2 \qquad \forall \quad (i,j) \in E, u \in U \tag{7j}$$

$$w_{ik} - (1 - x_{ijk})M_3 \le v_{iju} + (1 - y_{iju})M_3 \qquad \forall \quad (i,j) \in E, k \in K, u \in U \tag{7k}$$

$$v_{iju} + q_{iju} - (1 - x_{ijk})M_3 \le w_{jk} + (1 - y_{iju})M_3 \qquad \forall \quad (i,j) \in E, k \in K, u \in U \tag{7l}$$

$$\sum_{i\in V} \sum_{j\in V} q_{iju} \ge w_{d_e,k} \qquad \forall \quad k, l \tag{7m}$$

$$v_{iju} + q_{iju} \le v_{cdu} + (1 - f_{ijucd})M_3 \qquad \forall \quad (i,j),(c,d) \in E, (i,j) \ne (c,d), u \in U \tag{7n}$$

$$v_{cdu} + q_{cdu} \le v_{iju} + f_{ijucd} M_3 \qquad \forall \quad (i,j),(c,d) \in E, (i,j) \ne (c,d), u \in U \tag{7o}$$

$$x_{ijk} \in \{0,1\} \qquad \forall \quad (i,j) \in E, k \in K \tag{7p}$$

$$w_{ik} \ge 0 \qquad \forall \quad i \in V, k \in K \tag{7q}$$

$$y_{iju}, f_{ijucd} \in \{0,1\} \qquad \forall \quad (i,j),(c,d) \in E, (i,j) \ne (c,d), u \in U \tag{7r}$$

$$v_{iju}, q_{iju} \ge 0 \qquad \forall \quad (i,j) \in E, u \in U \tag{7s}$$

Bos [2] suggests the optimal weight factor to be $\alpha = 2/3$.

3.3 Reoptimization VRP

During one of the tested heuristics, reoptimization of the tours takes place every time an urgent customer requests service. For this, the Vehicle Routing Problem for Reoptimization is introduced. The VRPRO is an adjusted version of the VRPTW with the same objective as the VRPTW. It aims to find optimal tours for all vehicles, but takes into account the current location of the vehicles, as well as requesting urgent customers.

Redefining the graph for dynamic locations For the VRPRO, the graph must be redefined. Therefore, let $R_w \subseteq R$ be the set of all waiting regular customers, i.e. all customers that have not been serviced when the VRPRO is called. $R_w = R$ at the start of the day, when no regular customers have been serviced yet. Let $U_w \subseteq U$ be the set of all waiting urgent customers. This set consists of the urgent customer requesting service when the VRPRO is called, as well as previously requesting urgent customers that have not been serviced yet.

The VRPRO must take into account the current location of every vehicle. These locations we denote as d_k for $k \in K$, and the set of these locations as $d_K = \underset{k \in K}{\cup} d_k$.

The VRPRO must take into account which vehicle is send to which urgent customer. Therefore, we introduce a ordered set $O_k = \{d_k, o_{k_1}, \ldots, o_{k_n}\}$ for every vehicle $k \in K$, where $\{o_{k_1}, \ldots, u_{o_n}\}$ are the urgent locations coupled to that vehicle. Since every requesting urgent location is coupled to exactly one vehicle, we know that $O_k \cap O_{k^*} = \emptyset$ for $k \neq k^*$. Furthermore, since every waiting urgent customer is coupled to a vehicle, it must hold that $\underset{k \in K}{\cup} O_k = O_w \cup d_K$.

Let $V' = R_w \cup U_w \cup d_K \cup \{d_s, d_e\}$ be the set of all nodes for the VRPRO and $E' = \{(i,j) | i,j \in V', i \neq j\}$ be the edges between these nodes. Then, the VRPRO is defined on the undirected graph $G' = (V', E')$

The VRPRO The VRPRO is an adjusted version of the VRPTW used for the reoptimization heuristic described in Chapter 4.2. Therefore, we only take a look at the adjustments that make the VRPRO.

Since locations of the vehicles may be different than their initial position. Therefore, constraint (2b) is adjusted to be (8b). This way, every vehicle is send from the start depot to their current position. Note that for the edges (d_s, d_k) the distances are set to be zero. This way, vehicles are not penalised by being forced to their current position.

The VRPRO gets a ordered list for every vehicle in the fleet. The VRPRO is required to only find tours where the urgent customers in these lists are serviced by their respective vehicle in the order that is given, before servicing regular customers that have not been serviced yet. This is done by introducing constraint (8c). Here, for each ordered set U_k, all edges between consecutive customers in the set are traversed by vehicle k, ensuring that vehicle visits these customers in order before traversing to other customers that are still requesting service.

These adjustments together with other constraints of the the VRPTW gives us the following MILP of the VRPRO:

$$\sum_{j \in V' \setminus d_s} \sum_{k \in K} x_{ijk} = 1 \qquad \forall \quad i \in V' \setminus \{d_s, d_e\} \tag{8a}$$

$$x_{d_s d_k k} = 1 \qquad \forall \quad k \in K \tag{8b}$$

$$x_{o_i o_j k} = 1 \qquad \forall \quad k \in K, o_i, o_j \in O_k, j = i+1 \tag{8c}$$

$$\sum_{i \in V' \setminus d_e} x_{ijk} - \sum_{i \in V' \setminus d_s} x_{jik} = 0 \qquad \forall \quad j \in V' \setminus \{d_s, d_e\}, k \in K \tag{8d}$$

$$\sum_{i \in V' \setminus \{d_s, d_e\}} x_{i,d_e,k} = 1 \qquad \forall \quad k \in K \tag{8e}$$

$$w_{ik} + s_i + d_{ij} - w_{jk} \leq (1 - x_{ijk}) M_{ij} \qquad \forall \quad (i,j) \in E', k \in K \tag{8f}$$

$$w_{ik} + s_i + d_{ij} - w_{jk} \geq -(1 - x_{ijk}) M_{ij} \qquad \forall \quad (i,j) \in E', k \in K \tag{8g}$$

$$a_i\Big(\sum_{j \in V' \setminus d_s} x_{ijk}\Big) \leq w_{ik} \leq b_i\Big(\sum_{j \in V' \setminus d_s} x_{ijk}\Big) \qquad \forall \quad i \in V' \setminus \{d_s, d_e\}, k \in K \tag{8h}$$

$$E \leq w_{ik} \leq L \qquad \forall \quad i \in \{d_s, d_e\}, k \in K \tag{8i}$$

$$x_{ijk} \in \{0,1\} \qquad \forall \quad (i,j) \in E', k \in K \tag{8j}$$

$$w_{ik} \geq 0 \qquad \forall \quad i \in V', k \in K \tag{8k}$$

4 Heuristics

Now that the VRPs are defined, a heuristic algorithm for dealing with urgent stochastic customers must be defined. Since VRPs are NP-hard, [8], heuristic algorithms or 'heuristics' are proposed as an alternative. These heuristics are designed to solve a problem in a faster and more efficient way than a traditional method by sacrificing optimality for speed. In this chapter, we analyse the heuristic proposed by Bos, Van der Vucht and Boucherie [2], that we refer to as the A-Priori Heuristic, as well as an alternative heuristic called the Reoptimization Heuristic.

4.1 A-Priori Heuristic

The heuristic in Bos, Van der Vucht and Boucherie [2] will be the point of reverence for this study and will form the baseline for the results. In this heuristic, the VRPUSC is used to dispatch vehicles at the start of the day. Whenever an urgent stochastic customer requests service, the closest vehicle to that customer is immediately sent over there for service. When service is completed, the vehicle will continue with his a-priori determined tour. Therefore named the A-Priori Heuristic.

4.2 Reoptimization heuristic

The Reoptimization heuristic is the alternative heuristic proposed in this paper. Similar to the A-Priori Heuristic, vehicles are initially dispatched using the VRPUSC. And similar to the A-Priori Heuristic, once a customer requests service, the closest vehicle is send there for service. However, in contrast to the A-Priori Heuristic, once a vehicle is coupled with an urgent customer, re-optimization of the tours will take place using the VRPRO. Vehicles that are coupled with an urgent customer will remain coupled until service at that customer has been completed.

For re-optimization, the VRPRO is chosen over the VRPUSC. Since the VRPUSC is a fairly slow VRP, the VRPUSC is deemed too slow to be realistically usable during tours. Since the VRPRO is an adjusted version of the VRPTW, which has evolved into a fast algorithm over the years, the VRPRO is used instead.

We will refer to the Reoptimization heuristic as the VRP Reopt.

5 Simulations

Data for the heuristics analysed are generated through simulation. This simulation is programmed as seen in figure 5. The program is designed to create N sets of locations and calculate initial tours using the VRPUSC. Then, for each set of locations, M simulations are run. For every simulation, λ stochastic urgent requests are generated. With every set of urgent requests, both the A-Priori heuristic and the VRP Reopt are simulated and results are saved for analysis. Input settings as used in this paper are described in Chapter 6.

In this study, the simulations are run in Python, using Gurobi as a mathematics optimization solver for the VRPs.

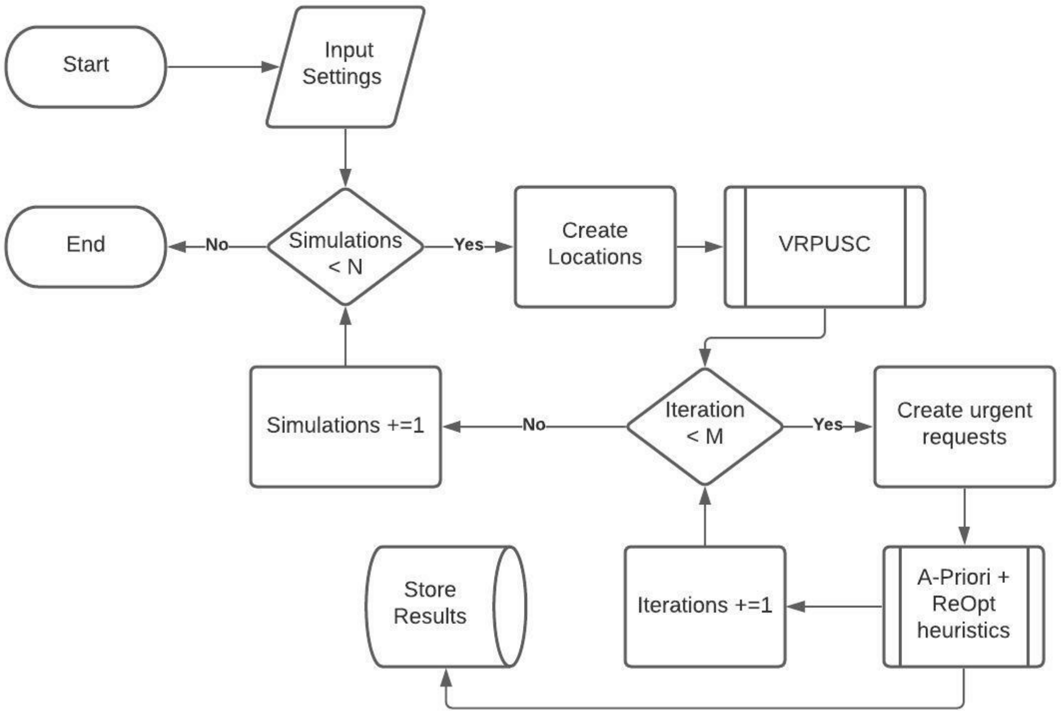

Figure 2: Simulation for generating data

6 Results

With the results from the simulations, comparing the two heuristics is done with statistics. Initially, both the response times and travel times are analysed. Since both heuristics are run on the same set of simulated customers and urgent requests, pooled statistics are then used to analyse whether there are significant differences between the both the response times and the total traveled distances between both heuristics.

Settings For every simulation, a random set of customers is generated in a 10 by 10 grid. This set of customers consists of 7 regular customer locations and 3 urgent customer locations. The coördinates of these locations are drawn from a random number generator using a uniform distributing. The start- and end-depot is located in the middle of the grid, at (5,5). Urgent requests are generated using a Poisson Distribution, with λ being the intensity of requests during one timespan of the tours. In total 20 sets of customers are generated.

For every simulated set of customers, 40 instances of urgent requests are generated and simulated, one each for $\lambda = 1, 2, 3$. The amount of 40 instances was empirically shown to always be sufficient for a 10% relative accuracy in a 95% confidence interval.

6.1 Response Times

The goal of the VRPUSC is to reduce response times to urgent stochastic customers, even if it results in longer tours. Therefore, the main criterium for the VRP Reopt is improving the response times for the VRPUSC and its A-Priori heuristic.

After simulation, the following 95% confidence intervals were found for the total response times of both the A-Priori heuristic and the VRP Reopt, where λ is the amount of urgent requests per simulation.

	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$
A-Priori	2.9 ± 0.8	5.4 ± 1	7.7 ± 1.9
Reopt	2.9 ± 0.8	5.4 ± 1	7.5 ± 1.7

For both $\lambda = 1$ and $\lambda = 2$, response times are the same for both VRPS. Note that this is trivial for $\lambda = 1$, since both VRPs have the same initial set of tours and response to the first urgent request. For $\lambda = 3$, the VRPRO seems to slightly improve on the response times, but this difference is too insignificant to draw any conclusions.

6.2 Travel Distances

The second criterium for both heuristics is the total distance distance traveled by the vehicles. The shorter the tours, the better. Only if longer travel distances result in shorter response times, this can be interpreted as a worth wile cost.

After simulation, the following 95% confidence intervals were found for the total travel distances of both the A-Priory heuristic and the VRP Reopt, where λ is the amount of urgent requests per simulation.

	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$
A-Priori	46.2 ± 7.8	49.1 ± 7.5	51.5 ± 7.3
Reopt	51.6 ± 6.0	55.8 ± 5.6	58.2 ± 6.0

It is clear to see that the VRPUSC results in better travel distances than the VRP Reopt for any given λ. However, the confidence intervals still show overlap. Therefore, these results are not significant enough to prove that the VRPUSC provides better travel distances than the VRP ReOpt.

In order to draw significant conclusions over our results, pooled statistics are used.

6.3 Pooled Statistics

Since we simulate how both VRPs behave for every renerated set of locations, we can analyse their differences. For this, we use Pooled Statistics. With pooled statistics, more data is taken into account to describe one system (in our case the difference between two heuristics). Therefore the sample variance and confidence interval will be smaller, giving us a more accurate result.

With pooled statistics, the estimators are as follows:

$$\mu_1 - \mu_2 = \bar{X} - \bar{Y} \tag{9}$$
$$S^2 = \frac{n_1 - 1}{n_1 + n_2 - 2} S_X^2 + \frac{n_2 - 1}{n_1 + n_2 - 2} S_Y^2, \tag{10}$$

where $\mu_1 - \mu_2$ is the difference of two population means, $\bar{X}$ and $\bar{Y}$ are their sample means, S^2 is the pooled sample variance, S_X^2, S_Y^2 are the sample variance of X and Y, and $n_1 - 1$ and $n_2 - 1$ are their respective degrees of freedom.

Since we are analysing the difference between two random samples, all that is left is to analyse their confidence interval. If this interval does not overlap with zero, we can conclude that the samples cannot be the same. In these results, a positive result means the values for the A-Priori heuristic are higher than those of the VRP Reopt. The pooled results are as follows:

	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$
Response Times	0 (-0.49,0.49)	0.0 (-0.77,0.78)	0.19 (-0.92,1.30)
Travel Distance	-5.42 (-9.70,-1.14)	-6.72 (-10.77,-2.66)	-6.68 (-10.79, -2.58)

Here, it is much clearer to see what we assumed to be true in the previous paragraphs. The difference in response times is too insignificant to be able to speak of a difference between the heuristics, as the confidence interval includes 0.

However, for the travel distance, we can conclude with confidence that the VRP Reopt is not an improvement, but a retrogression to the VRPUSC.

7 Conclusion

In this paper, our aim was to analyse the VRP Reopt, a variation on the VRPUSC, through a simulation study. Our goal was to find whether or not the VRP Reopt was an improvement on the VRPUSC. Two factors were taken into account; the time the VRP took to respond to an emergency, or 'response times', and the total distance traveled by all vehicles after completion of the tours, the 'travel times'.

The most important variable for the VRPs was the response times, since the VRPUSC is developed for the purpose of lowering the response times at the cost of longer travel times. The simulation showed no significant difference between the VRPUSC and the VRP Reopt and therefore we can conclude that the VRP Reopt is not an improvement to the VRPUSC in terms of response times.

When it comes to travel times, the VRP Reopt is doing worse than the VRPUSC. The pooled statistics clearly show that the VRP Reopt does worse in all situations and therefore is not an improvement on the VRPUSC in terms of travel times.

In conclusion, it is clear to conclude that the VRP Reopt is not an improvement over the VRPUSC. It does not show a significant improvement over the VRPUSC, but does result in worse travel times.

8 Discussion and reflection

The VRP Reopt does not show to be an improvement over the VRPUSC. This is not unexpected, as the VRPUSC is designed to find initial tours that minimize response times as much as possible. Since the VRP ReOpt reoptimizes with a simpler VRP, it is not unexpected to see a decrease in effectiveness. However, test results indicate a decreasing effectiveness for the VRPUSC once more urgent customers request service. This also is not unexpected, since the VRPUSC assumes no time loss when calculating optimal tours, while serving a unexpected urgent customer does result in time loss for the vehicle serving this customer. The more customers are serviced, the more out of sync the different tours become.

Since reoptimization does seem to be a promising improvement to the VRPUSC, we would like to give the following recommendations for future research. First, an improvement to the VRP Reopt might be to only reoptimize once every k urgent requests, since reoptimization does seem to have a positive effect on the response times once more requests are put in. Furthermore, if the VRPUSC would become efficient enough, the VRPUSC could be used to reoptimize over a standard VRPTW.

Within the space the simulations are done (7 regular location, 3 stochastic urgent locations), the complexity seems to not be sufficient enough to be able to draw conclusions with any significance. Simulation results may suggest that the VRP Reopt improves with higher complexity, but as stated the low complexity of the space does not allow us to draw any significant conclusions.

References

[1] Mohamed Barkaoui et al. “An adaptive evolutionary approach for real-time vehicle routing and dispatching”. In: *Computers Operations Research* 40 (July 2013), pp. 1766–1776. DOI: 10.1016/j.cor.2013.01.022.

[2] Jasper Bos et al. “The Vehicle Routing Problem with Urgent Stochastic Customers”. unpublished. N.D.

[3] G. Clarke et al. “Scheduling of Vehicles from a Central Depot to a Number of Delivery Points”. In: *Operations Research* 12.4 (1964), pp. 568–581. DOI: 10.1287/opre.12.4.568.

[4] G. B. Dantzig et al. “The Truck Dispatching Problem”. In: *Management Science* 6.1 (Oct. 1959), pp. 80–91. DOI: 10.1287/mnsc.6.1.80.

[5] M.L. Fisher et al. “A generalized assignment heuristic for vehicle routing”. In: *Networks* 11 (1981), pp. 109–124.

[6] Iván García-Magariño et al. “Security in networks of unmanned aerial vehicles for surveillance with an agent-based approach inspired by the principles of blockchain”. In: *Ad Hoc Networks* 86 (Nov. 2018). DOI: 10.1016/j.adhoc.2018.11.010.

[7] Mostepha Khouadjia et al. “Metaheuristics for Dynamic Vehicle Routing”. In: vol. 433. Jan. 2013, pp. 265–289. ISBN: 978-3-642-30664-8. DOI: 10.1007/978-3-642-30665-5_12.

[8] J.K. Lenstra et al. “Complexity of vehicle routing and scheduling problems”. In: *Networks* 11 (Oct. 2006), pp. 221–227. DOI: 10.1002/net.3230110211.

[9] Sandro Lorini et al. “Online vehicle routing and scheduling with dynamic travel times”. In: *Computers Operations Research* 38 (July 2011). DOI: 10.1016/j.cor.2010.10.019.

[10] Victor Pillac et al. “A review of dynamic vehicle routing problems”. In: *European Journal of Operational Research* 225 (Feb. 2013), pp. 1–11. DOI: 10.1016/j.ejor.2012.08.015.

[11] Martin W. P. Savelsbergh. “Local search in routing problems with time windows”. In: *Annals of Operations Research* 4 (1984), pp. 285–305.

[12] Linus Schrage. “Formulation and structure of more complex/realistic routing and scheduling problems”. In: *Networks* 11.2 (1981), pp. 229–232. DOI: 10.1002/net.3230110212.

[13] M.M. Solomon. “Algorithms for the vehicle routing and scheduling problem with time window constraints”. In: *Operations Research* 35 (1987), pp. 254–265.

[14] W.B. Powell Y.A. Koskosidis et al. “An optimization based heuristic for vehicle routing and scheduling with soft time window constraints”. In: *Transportation Science* 26 (1992), pp. 69–85.