Utrecht University

Department of Information and Computing Sciences

Scheduling Electric Buses with Stochastic Driving Times

Master's Thesis

ICA-6197140

Author:
P. de Bruin

Supervisors:
Dr. ir. J.M. van den Akker
Dr. J.A. Hoogeveen

July 2022

Abstract

In this thesis we look at the Stochastic Electric Vehicle Scheduling Problem. In this problem we are given a set of trips, and we need to schedule vehicles such that each trip is driven. We apply this problem to electric buses, where we want to minimize the operating cost taking the battery life into account. Here, we want to make our schedules more robust against delays. These delays could be caused by, for example, various traffic conditions, or passenger loads, as these factors have an effect on the driving time. Thus, in order to make our schedules more robust against these delays, we use stochastic driving times instead of deterministic driving times. Not only the driving time itself could be a source of delays, but also the energy consumption. Bus drivers have different driving styles, which effect the energy consumption, and thus the time needed for charging. Thus, we also consider the energy consumption to be stochastic.

To achieve this, we use a combination of simulated annealing and simulation. Here, we use simulation to calculate the cost of a solution. This, however, comes with a performance penalty. Thus, we try different methods of determining how many simulations we need, such that we still make a correct choice about which solution is better. The techniques we consider here are: *Optimal Computation Budget Allocation*, *Indifference Zones*, and a method we developed ourselves, which uses t-tests.

We show that the use of some of these methods can increase the runtime performance while performing similar in terms of their final score. Furthermore, with our use of stochastic driving times, we see in increase in the punctuality of the buses, and they also arrive a bit earlier at the start of their trip. However, we also see a slight increase in operating cost, as we need slightly more buses compared to when we use deterministic driving times.

Acknowledgements

First and foremost, I would like to thank my supervisors Marjan van den Akker and Han Hoogeveen for their guidance and support during my thesis. They gave excellent and frequent feedback on the project, which helped me a lot in completing this thesis. I always enjoyed our meetings and learned a lot in the process.

I would also like to thank Marcel van Kooten Niekerk, our contact person at Qbuzz, who provided both the data I needed and insight into their operations, which helped me build and verify the models in this thesis project.

Finally, I am also thankful of my friends and family for their support and encouragement.

Contents

1 Introduction **1**
- 1.1 Problem Description . . . 1
- 1.2 Solution Approach . . . 2
- 1.3 Outline . . . 3

2 Literature Overview **4**

3 Goal **6**
- 3.1 Research Questions . . . 6
- 3.2 Methodology . . . 6
  - 3.2.1 Local Search . . . 6
  - 3.2.2 Simulation . . . 7
  - 3.2.3 Modeling Stochastic Driving Times . . . 7

4 The Hybrid Algorithm **8**
- 4.1 ILP . . . 8
- 4.2 Simulated Annealing . . . 9
- 4.3 Neighbourhoods . . . 11

5 Robustness **12**
- 5.1 Simulation . . . 12
- 5.2 Simulating Driving Times . . . 13
- 5.3 Simulating Energy Consumption . . . 13

6 Required Number of Simulations **15**
- 6.1 Optimal Computation Budget Allocation . . . 15
- 6.2 Indifference Zones . . . 16
- 6.3 T-Test . . . 17
- 6.4 Integration in Simulated Annealing . . . 19

7 Driving Time Analysis **20**
- 7.1 Dataset . . . 20
- 7.2 Variables . . . 20
  - 7.2.1 Time of Day . . . 20
  - 7.2.2 Weather . . . 22
  - 7.2.3 Passengers . . . 23
- 7.3 Distributions . . . 25

8 Experiments and Results **27**
- 8.1 General Setup . . . 27
- 8.2 Required Number of Simulations . . . 27

8.2.1 Performance . . . 28
8.2.2 Number of simulations . . . 28
8.2.3 Hillclimb . . . 30
8.2.4 Best solution comparison . . . 31
8.2.5 Recombination . . . 31
8.3 Stochastic E-VSP . . . 31
8.3.1 Recombination . . . 33
8.3.2 Lateness . . . 34
8.3.3 Depth of Discharge . . . 35

9 Conclusion **39**
9.1 Summary . . . 39
9.2 Conclusion . . . 39
9.2.1 Required Number of Simulations . . . 39
9.2.2 Stochastic E-VSP . . . 40
9.2.3 Recombination . . . 40
9.3 Future Research . . . 40

Bibliography **I**

Chapter 1

Introduction

In an effort to make the world more sustainable, electrification in the public transport sector is becoming more and more important. More specifically, everywhere in the Netherlands diesel buses are replaced for their electric counterparts. This, however, is not as easy as it might sound, as it introduces more constraints when scheduling these buses. In their current status, electric buses are constrained in their range, and thus they need to be recharged during the day. This is not a problem for diesel buses, as they could generally drive the whole day on a single tank.

Introducing range restrictions on the buses leads to various scheduling problems. For example, these electric buses need to be recharged, or their batteries need to be swapped during the day. This means that we can look into optimizing the locations of such 'service points'. Scheduling electric buses becomes more difficult compared to scheduling diesel buses, as we now not only need to determine the routes for the buses, but also when to recharge the vehicle.

There already exist several solutions to this problem. However, most of these solutions assume deterministic driving times. However, this is not a realistic assumption, as traffic conditions and passenger loads vary from day to day, which will cause delays. In this thesis, we will look at scheduling these electric buses, where we investigate this extension of using stochastic driving times in order to make our schedules more robust against these delays. Not only the driving times are a source for delays for these buses, because the driving behaviour of a bus driver could mean that he uses more electricity, which causes longer charging times. Thus, we will also take this behaviour into consideration. A more exact description of this problem will be given in the next section. After that, we will discuss several solution approaches to this problem, and we will also give an outline of the rest of this thesis.

1.1 Problem Description

In the Vehicle Scheduling Problem (VSP), we are given a set of trips. These trips consist of a departure and arrival location, a starting time, and a driving time. Thus, these trips could, for example, represent a timetable for buses in a city. The goal is then to schedule vehicles such that every trip is driven. For this, we want to minimize the cost of using these vehicles. Such costs typically consist of a fixed cost of using the vehicle, a cost per kilometer driven, and a cost per *block*. A *block* is a set of trips driven after each other without going back to the garage. A cost for these blocks is included to penalize situations where a bus only drives a single trip before going back to the garage. In this case, we consider only one depot location. All the vehicles must start and end at this location, as the vehicles will be parked here overnight. We also assume that all vehicles are identical.

As mentioned before, we could also apply this problem to electric vehicles, giving us the E-

VSP problem. However, since these electric vehicles have less range than their non-electric counterparts, we need to think about charging strategies for these vehicles. These strategies could come in various forms, such as battery swapping technologies, or charging at specified recharging stations. We will focus on the latter strategy, as the use of this is more widespread and battery swapping technologies do not seem to exist yet. We also do not expect battery swapping technologies to be used soon. When recharging the battery, we take the battery life into consideration, as certain strategies significantly degrade the battery life resulting in more maintenance costs. We do this by looking at the effect of charge cycles on the battery life. For this, we calculate the cost of charging a battery and minimize this cost. Furthermore, we will only be looking into charging the batteries at specified locations. For this, we will not consider a capacity constraint on these recharging locations. However, we will be taking the non-linearity of the charging time into account.

Usually, when solving these VSP (or E-VSP) problems, we assume deterministic driving times. However, in practice this assumption is often violated, as buses drive in various traffic conditions in which delays may occur. Furthermore, weather conditions or other factors can introduce more passengers leading to higher dwell times, which causes higher driving times. These delays can cause various issues, such as delay propagation in our network or inconveniences for the end-user, as it results in longer waiting times, delayed arrivals, and possibly missed transfers.

We do not expect that the charge required depends much on the driving time, as the distance driven remains the same, but it will affect the time left for charging. This could result in further delays or the risk of running out of charge. To cope with this risk, we ensure that a bus leaving a charging location has enough energy to reach the next charging location. Note that this could result in the bus not departing on time.

Due to different driving styles of the bus drivers, our energy consumption does differ between these drivers. This could also result in longer charger times and buses not departing on time, resulting in potential delays. To take this into account, we need to make the energy consumption stochastic as well.

Taking all these potential delays into account allows us to create a schedule that is more robust against this behaviour. However, this means that our driving times become stochastic. The goal of this thesis is to find a good way to create such a robust schedule for electric buses, taking these stochastic driving times into account. Thus, in this case we want to make our schedule robust against delays, meaning that delays will not cause massive issues.

To measure the robustness, we look at the starting times of a trip, because a delayed vehicle would start its next trip late when there is not enough slack between the trips. We also apply some non-linearity to this measure such that being 10 minutes late is penalized significantly more compared to being just 1 minute late.

All in all, this means that we schedule electric buses on a given set of trips, where we minimize operational costs, the robustness penalty, and a cost for the battery lifetime. These operational costs consist of a fixed cost for using a vehicle, a cost per kilometer driven, and a cost per driven block. The robustness penalty is as explained before, where we penalize buses that start their trip later than planned. For the battery lifetime, we use the cost of a battery and calculate the effect of charging on the lifetime of the battery. We use this to calculate the cost of charging, that we minimize in order to maximize the lifetime of the battery.

1.2 Solution Approach

A more classical way to deal with stochastic driving times, is to include some slack based on the distribution of the driving time. This slack time could be based on a factor multiplied

by the mean of the distribution, or a certain percentile of the distribution. With this, all the stochastic driving times could be converted back into deterministic ones. This results, however, in an approximation where the different trips do not affect each other. Also, the variance of the distribution is only somewhat accounted for, as we will not encounter the more extreme delays that realistically still could occur. Thus, this approximation does not necessarily represent reality very well.

A better way to deal with stochastic driving times is to work with the expected start- and end-times of a trip given the trips that are driven before it. However, it is not easy to compute these values. To solve this, we could try to estimate these values by assuming normal distributions [16]. Another way to estimate these values is by using simulations in a local search algorithm [25]. In this thesis we will focus on the latter approach.

To solve the E-VSP problem, we use the simulated annealing approach used by ten Bosch et al. [24]. This method is based on a column-generation approach by van Kooten Niekerk et al. [26]. For this, each column represents the trip of a single vehicle. However instead of solving a pricing problem to find new columns, we use simulated annealing to find a schedule and use the trips in this schedule as columns. We recombine these trips into a final solution using an ILP-solver.

Lastly, we will also need to determine what distributions to use for the driving times. For this, we worked closely with Qbuzz, a bus company from The Netherlands, who provided data and insights of their operations. With their information we determined various sources for delays in the driving times. Furthermore, we also determined that we would need a simulation where the driving times are dependent on each other. The idea here is that people taking the bus in the morning, will also take the bus back in the afternoon. Thus, if we have higher passenger demands in the morning, we will also see these passenger demands in the afternoon, likely resulting in higher driving times both in the morning and the afternoon.

1.3 Outline

In Chapter 2 we will first discuss the relevant literature for this problem. Then, in Chapter 3 we will set out our research question and further goals for this thesis. Furthermore, we will also discuss the approach we will take to reach these goals. In Chapters 4 and 5 we will go into the details of our model, where we discuss our local search approach and its extension with simulation into account. Furthermore, we will discuss various options of increasing the runtime performance of our simulation in Chapter 6. We will also analyze historic driving times in Chapter 7. Lastly we will run some experiments to test our model in Chapter 8, which we will discuss in Chapter 9.

Chapter 2

Literature Overview

In recent years, the E-VSP problem is studied more and more. A recent study of Perumal et al. [18] divides the research into the different challenges related to the planning and scheduling of electric vehicles and how different studies overcome those challenges. When it comes to the scheduling of electric vehicles, we can think of this as scheduling vehicles with resource constraints, as the vehicles are limited in their range. In 1983, Raff [20] showed that the VSP problem with any resource constraint is NP-hard. In 2007, Wang and Shen [27] added fuelling time constraints to the VSP problem, where they specifically focus on electric vehicles. For the recharging of these electric vehicles, multiple different technologies can be considered [5, 13]. For example, Chao and Xiaohong [3] developed a genetic algorithm to solve the E-VSP problem with battery swapping stations. Other approaches mainly focus on recharging stations, which could either provide slow- or fast-charging capabilities. Wen et al. [28] present a large neighbourhood search heuristic for solving E-VSP with recharging stations, where they assume the charging time to be linear. They also allowed for partial recharging. However, this assumption of linear charging times is not realistic and, as shown by Olsen and Kliewer [14], may lead to infeasible routes, or too large charging gaps. Van Kooten Niekerk et al. [26] incorporated such non-linear charging times and proposed a column-generation approach to solve E-VSP.

Ten Bosch et al. [24] built on this approach, where they use simulated annealing to generate new columns. They do this by generating multiple schedules using simulated annealing. Then, they take all the vehicle tasks from these schedules and use the restricted master problem for column generation to recombine these tasks into a single schedule. This approach is very successful compared to column generation, where it is both faster and has a smaller integrality gap to the LP-relaxation.

The use of some sort of local search to generate columns is used in similar problems. Subramanian et al. [22] use it to solve different variants of the vehicle routing problem, where they show competitive results for the results of different vehicle routing problem instances. Pirkwieser and Raidl [19] enhance a variable neighbourhood search in a similar way for the vehicle routing problem with time windows. Thus, they use a variable neighbourhood search to generate columns. This approach was built upon by Parragh and Schmid [15], who use it in combination with a large neighbourhood search for the dial-a-ride problem. They were able to find new best solutions for different instances, while requiring less runtime.

Not much research has been done when it comes to E-VSP with stochastic driving times. Tang et al. [23] propose a branch-and-price framework for solving E-VSP under both static and dynamic traffic conditions. They do this by using a so-called buffer-distance, which makes sure that the bus does not run out of charge while in traffic. Furthermore, while they propose a model to avoid running out of charge due to the traffic conditions, they still use the average travel time

for cost and delay calculations. So, while they look at stochastic driving conditions, they still solve it deterministically. When it comes to a charging strategy, their model ensures that buses always leave the depot or a charging station with a fully charged battery. Furthermore, they use a constant charging time, in order to reduce the number of variables in the model. Bie et al. [2] use a Non-dominated Sorting Genetic Algorithm with the elitist strategy (NGSA-II) to solve E-VSP for stochastic driving times. For their recharging strategy, they set a range in which to keep the battery's state of charge. They will then recharge a bus when it is idle, that is when it is currently waiting for its next trip to start. For recharging, they make use of linear charging times.

There is also some research done on incorporating stochasticity into similar optimization problems. Van den Akker et al. [25] adapt a local search algorithm for the Job Shop Scheduling problem to include simulations of stochastic processing times. This approach worked well compared to more classical methods of adding slack. However, these simulations are quite slow to run. Passage et al. [16] look at the Stochastic Parallel Machine Scheduling problem and approximate the expected makespan within the local search framework. They do this by assuming normal distributions. Then, using the results of Clark [6], you can compute the expected maximum of two of these normally distributed random variables. Computing these estimations is pretty quick, especially when compared to using simulation. Furthermore, this approach also yielded better results than using simulation, unless we do a lot of simulations each iteration of the local search, which slows down the algorithm.

Another way of improving the runtime of simulations in a local search algorithm, is by trying to minimize the number of simulations needed while still doing enough simulations to make correct decisions. In a local search framework, we constantly compare different solutions to each other in order to select the best one. Note that to do this, we do not need to accurately calculate the score of each solution as long as we can create a correct ordering of solutions. This is called *Ordinal Optimization* [8]. One way to do this is by employing so-called *Optimal Computing Budget Allocation* (OCBA). Yang et al. [30] did this for the Stochastic Job Shop Scheduling problem. This works by not spreading all the simulations evenly over all solutions, but some solutions get more simulations based on the variance of earlier results. Another technique for *Ordinal Optimization* is by using *Indifference Zones* [1, 7, 12]. This is similar to OCBA, but where OCBA maximizes the probability of a correct selection, Indifference Zones guarantee a minimum for the probability of a correct selection [12].

Chapter 3

Goal

3.1 Research Questions

As mentioned in Chapter 1, we will be looking at the E-VSP problem with a focus on robustness. For this, we need to create a model that incorporates stochastic driving times. Thus, in this thesis we will focus on the question of creating a model to solve the E-VSP problem with stochastic driving times by using simulation in a local search algorithm. Furthermore, we look into combining the results of this local search algorithm with an ILP.

We expect the simulations to form a performance bottleneck in the local search, and hence we need to explore techniques for optimizing this step, as this will likely lead to better results more quickly. Thus, we will look into Ordinal Optimization techniques to find out what works well for this problem. Furthermore, for our local search algorithm, we also want to answer whether recombination of different solutions found by our local search algorithm leads to further significant improvements.

We will not only create a model that solves this problem, but we also need to find out how to accurately model these stochastic driving times. Thus, we need to analyse historic driving times in order to find representative distributions. This analysis should also give an insight into different conditions that should be modelled in the simulation, such as weather conditions or dependencies on, for example, the driving times in the morning.

3.2 Methodology

3.2.1 Local Search

To answer our research questions, we will first need to develop a local search algorithm that solves the E-VSP problem with deterministic driving times. For this, we will create a simulated annealing algorithm based on the work of ten Bosch et al. [24]. Here, we will use two neighbourhoods that are also used by ten Bosch et al. Namely, a *2-opt swap* and a *trips move* neighbourhood. These neighbourhoods will be further explained in Section 4.3.

As simulated annealing is not guaranteed to find optimal or near-optimal solutions, we will need to run the algorithm multiple times for different seeds. We could also try slightly different input parameters for each run. With this, we will get multiple solutions and thus a lot of possible routes a vehicle can drive. We will also combine these routes into a single schedule using an ILP to see how much this can improve our overall solution.

3.2.2 Simulation

To calculate the cost of a certain route a vehicle drives, we will use discrete-event simulation. Here, we sample the duration of a trip or deadhead from distributions based on historic driving times. We use these durations to schedule the event that marks the end of a trip or deadhead. In this simulation we keep track of the delayed starting times and the state of charge of the vehicle. Such a simulation will be run multiple times after which we use the average of the costs as the cost of the vehicle task.

When comparing two solutions, we need to be careful that they are compared fairly, as the randomness of the driving times can cause a bad solution to be lucky (i.e., it has a low cost in the simulation), and consequently not being rejected. To make the comparison more fair, we can apply a technique called *Common Random Numbers* (CRN) [11]. The idea of this technique is to use the same realizations of driving times for both schedules. Further techniques that could be used are, for example, *cut-off sampling* [25], although we would need to experiment with this as it might interfere with the convergence of the simulated annealing [16]. In our model we will not look at such techniques, thus we only employ CRN.

Using simulations in a local search algorithm is computationally relatively expensive. Thus, we want to minimize the number of simulations needed while still using enough simulations to make sure we select the better vehicle schedule. To reduce the number of simulations needed, we will need to further explore Ordinal Optimization techniques to find out what works best in this situation. Techniques to consider here are OCBA [30] or Indifference Zones [1, 7, 12]. However, we will also test our own method, which uses t-tests to determine whether we need to simulate more.

3.2.3 Modeling Stochastic Driving Times

For our simulation model, we need to know the distributions of the driving times of each trip. We will analyse historic driving times in order to find these distributions. For this, we will also consider wether these driving times are dependent on other external factors, such as weather conditions or a delayed start of the trip. These dependencies could give us correlations between subsequent driving times, but also between driving times over a whole day. Thus, with these correlations we will be able to create more realistic simulations, which in turn could result in more robust schedules. Lastly, we can also use these historic driving times to test the robustness of a schedule in a real-world setting.

Chapter 4

The Hybrid Algorithm

4.1 ILP

To solve the E-VSP problem, we will use an ILP formulation as the basis. This ILP can be used when using deterministic driving times. In order to easily extend this ILP for stochastic driving times, we will already include the notion of (expected) lateness.

Let $\bar{T}$ denote the set of trips that need to be driven, and let V be the set of all possible tasks. Here, a task is a set of trips that can be driven by a single vehicle. When working with deterministic driving times, this means that the vehicle can drive these trips without running out of charge and can start every trip on time. In the stochastic case however, we drop the requirement of starting a trip on time as this cannot be guaranteed. Thus, in this case, starting a trip late will be penalized. In these tasks, we also account for the recharging itself, meaning that we both account the recharging that is needed and the time it takes to recharge. To calculate the recharging time, we take the same approach as van Kooten Niekerk et al. [26]. Thus, we assume the charging time to consist of two linear parts, where charging from 0% to 80% takes the same time as charging from 80% to 100%.

For a task $v \in V$, we define C_v to be the operating cost of this task. This operating cost consists of the cost of a vehicle, a cost per driven kilometer, a cost per driven minute (this includes the time waiting before the next trip can be driven), and a cost per driven *block*. Remember from Section 1.1 that a *block* is a set of trips that are driven after each other without going back to the garage. Lastly, we define L_v to be the lateness of the task and D_v to be the charging cost of the task. We will further explain these two terms later.

Let x_v be the decision variable that task $v \in V$ is included in the schedule, and let α and β be some constant. We use the parameter r_{vt} to indicate if trip $t \in \bar{T}$ is included in v. Then our objective is to

$$\textbf{minimize} \sum_{v \in V} x_v(C_v + \alpha L_v + \beta D_v). \tag{4.1}$$

Which is subject to the constraints:

$$\sum_{v \in V} r_{vt} x_v = 1 \qquad \forall t \in \bar{T}, \tag{4.2}$$

$$x_v \in \{0, 1\} \qquad \forall v \in V. \tag{4.3}$$

Here, Equation (4.2) ensures that we drive every trip in the final schedule and Equation (4.3) sets the domain of our decision variables.

Lateness When using deterministic driving times, we require that a vehicle can drive its tasks without starting a trip late. This means that $L_v = 0$ for all $v \in V$. However, in the stochastic case, we do not know when a trip ends, thus we need to penalize tasks where trips are likely to start late. For this, we define two cost variables C_{l_1} and C_{l_2}, with $C_{l_1} \leq C_{l_2}$. For a trip $t \in \bar{T}$ with a planned starting time of p_t and an actual starting time of a_t, we define the lateness cost of this trip as

$$l_t = \begin{cases} C_{l_1}(a_t - p_t) & \text{if } a_t - p_t < M, \\ C_{l_1}M + C_{l_2}(a_t - p_t - M) & \text{otherwise.} \end{cases} \tag{4.4}$$

Note that we do not allow trips to start earlier than their planned starting time, thus $a_t \geq p_t$. Furthermore, M is constant such that being less than M minutes late is less costly than being more than M minutes late, as then the C_{l_2} cost factor will be used. In our research we will use $M = 3$. Then L_v is defined as the sum of the expected values of l_t for the trips t that are driven in task v.

Charging Cost In order to reduce the maintenance costs of the electric buses, it is important to take the battery longevity into account. For this, we penalize tasks with a high *Depth-of-Discharge* (DoD), which percentage value is used to indicate how much a battery is discharged. To account for this, we use the cost factors found by van Kooten Niekerk et al. [26]. Let C_{battery} denote the cost of buying a new battery, then they define the cost of a charge cycle as

$$c(x) = \frac{e^{2.519x}}{4825.4} C_{\text{battery}}, \tag{4.5}$$

where $x \in [0, 1]$ denotes the DoD. To calculate the cost of charging the vehicle, they use the formula

$$c(s, f) = c(s) - c(f) = \frac{e^{2.519s} - e^{2.519f}}{4825.4} C_{\text{battery}}, \tag{4.6}$$

with $s, f \in [0, 1]$ and $s \geq f$. Here, s denotes the DoD when we start charging and f denotes the DoD when we are done with charging. Then we define D_v as the sum of these charging costs of each charging session along the route. For this we assume that we need to fully charge the bus at the end of its route. The cost of this is also included in D_v. Note that for this charging cost we only consider the cost related to battery deprecation. It does not include the cost of the electricity itself.

Charging strategy To have a valid task, we need to make sure that buses do not run out of charge along their route. For this, we need to define a charging strategy. For this we will charge the vehicle whenever possible for as long as possible. Thus, if a bus needs to wait for its trip to start at a place at which it can also charge, it will charge until either the battery is full or it needs to start the trip. Note that this charging strategy also minimizes the DoD over the whole trip. When working with stochastic driving times, we need to extend this strategy a bit. A trip might run longer than expected, resulting in not enough time to charge the battery in order to reach the next charging location. Thus, when charging, we charge for as long as possible, but we also make sure that we can reach the next charging location without running out of charge. This may result in running late for the next few trips, but that is better than being stranded somewhere along the route.

4.2 Simulated Annealing

To find tasks that can be used in the ILP described in the previous section, we will make use of a local search algorithm in the form of simulated annealing. For this we start with an initial solution s_{initial} where every vehicle drives just one trip. Using the neighbourhoods that will be

described in Section 4.3, we generate new solutions that are accepted with a certain probability. Given the cost of the current solution c, the cost of a potential new solution c_{new}, and the temperature T, we could define this acceptance probability as

$$\mathbb{P}(c, c_{\text{new}}, T) = \begin{cases} 1 & \text{if } c > c_{\text{new}}, \\ e^{\frac{c - c_{\text{new}}}{T}} & \text{otherwise.} \end{cases} \tag{4.7}$$

Furthermore, every Q iterations the temperature is decreased by a factor α. We continue iterating until we reach the stopping condition.

In our algorithm, we will start with this simulated annealing strategy and after a certain amount of time switch over to a more greedy hillclimb algorithm. A hillclimb algorithm accepts a new solution if and only if it is better than our current solution. In our case, we switch over to this hillclimb algorithm when T reaches 1. The acceptance probability function that represents this behaviour is

$$\mathbb{P}(c, c_{\text{new}}, T) = \begin{cases} 1 & \text{if } c > c_{\text{new}}, \\ 0 & \text{if } T < 1 \text{ and } c \leq c_{\text{new}}, \\ e^{\frac{c - c_{\text{new}}}{T}} & \text{otherwise.} \end{cases} \tag{4.8}$$

Then, after a set number of iterations the best solution gets returned. The pseudocode of this algorithm is provided in Algorithm 1. In here, we assume that GenerateNeighbour(s) returns feasible neighbours of a solution s, and that Random$(0, 1)$ returns a random value x that is distributed uniformly on the interval $[0, 1]$. Furthermore, the function Cost(s) calculates the cost as defined in Equation (4.1) for a solution s. This can be done deterministically or by using simulation, in which case the mean cost is returned.

Algorithm 1 Simulated annealing algorithm we use for finding good schedules.

```
Require: An initial solution s_initial
  procedure SimulatedAnnealing
    s ← s_initial
    s_best ← s
    while stop condition is not met do
      ▷ Generate new solution ◁
      s_new ← GenerateNeighbour(s)
      if Q iterations since last temperature reduction then
        ▷ Reduce the temperature ◁
        T ← T · (1 − α)
      ▷ Accept new solution with the acceptance probability from Equation (4.8) ◁
      if ℙ(Cost(s), Cost(s_new), T) ≥ Random(0, 1) then
        s ← s_new
        ▷ Also check if we found a new best solution ◁
        if Cost(s) < Cost(s_best) then
          s_best ← s
    return s_best
```

To generate multiple different schedules, we execute this simulated annealing procedure multiple times with different input parameters. This is also called *multistart simulated annealing*. Examples of these input parameters are the initial temperature, the cooling rate, or the seed for the random number generator. However, we will only be changing the seed between different simulated annealing runs. Varying these parameters ensures that we find different solutions, of which the best solution could be selected. However, we will also make use of the *simulated annealing with recombination* technique used by ten Bosch et al. [24]. This technique combines all

the tasks from the different solutions in the set V, such that we can use the ILP from Section 4.1 to find a solution.

4.3 Neighbourhoods

For our local search we use two neighbours which are selected with equal probability. These are the *2-opt swap* and *move range* neighbour. When generating a new neighbour, we need to make sure that it results in a feasible solution. If this is not the case, the neighbour is rejected, and we generate a new one until it results in a feasible schedule. A solution is feasible if all its vehicles correspond to a valid task as described in Section 4.1. This means that none of the vehicles run out of charge, and, in case of deterministic driving times, every trip starts on time. Furthermore, we also need to make sure that all trips are served. However, since our neighbours do not delete trips from the solution, we will always serve all the trips as long as we start with a solution that serves all trips.

2-Opt swap The *2-opt swap* neighbour selects two random vehicles v_1 and v_2 from the solution. Furthermore, it selects a random time t. It will then swap the trips that occur after t in v_1 with the trips that occur after t in v_2. This is also illustrated in Figure 4.1.

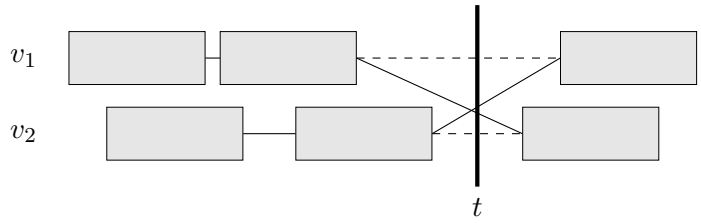

Figure 4.1: Example of the *2-opt swap* neighbour.

Move range Our second neighbour, the *move range* neighbour, also selects two random vehicles v_1 and v_2 from the solution. Then, it selects a range of trips R from v_1 and tries to move these trips to v_2. This is illustrated in Figure 4.2. Note the set R can contain all the trips in v_1, which results in v_1 and v_2 being merged into one vehicle.

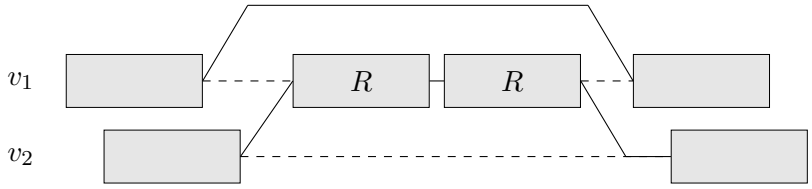

Figure 4.2: Example of the *move range* neighbour.

Chapter 5

Robustness

In order to create schedules that are robust against different traffic conditions and passenger loads, we introduce stochastic variables to the problem. This means that we need to find the expected total cost. However, this is not easy to compute, thus as is mentioned in Section 1.2, we estimate this by using simulation. We simulate a vehicle schedule multiple times and then take the average score of these runs.

To test the robustness of a schedule in various traffic conditions and passenger loads, we make use of stochastic driving times. Because on a day which is more busy, a bus accumulates more dwell time (that is the time it is stationary at a stop) resulting in a higher total driving time. Furthermore, different traffic conditions also result in different driving times. As different bus drivers have different driving styles, the energy consumption between bus drivers is slightly different. Thus, we also try to account for this by making the energy consumption also stochastic. This will be further explained in Section 5.3.

5.1 Simulation

To simulate a schedule, we make use of discrete-event simulation. Each vehicle (or task as it is defined in Section 4.1) consists of subtasks. These are the trips, deadheads, and the trips from and to the garage that need to be driven in order to complete the task. We call trips from and to the garage pull-outs and pull-ins respectively. A subtask has a specified driving time, which might be stochastic. Furthermore, trips and pull-outs also have a planned starting time and are not allowed to start earlier than this time. The discrete-event simulation consists of two events: the start of a subtask and the end of a subtask. It starts by scheduling the start of the first subtask. The 'start subtask' event will schedule the end of the subtask making use of the stochastic driving time. Then, the 'end subtask' event will either schedule an event for starting the next subtask or ending the simulation. This is also illustrated in the event diagram in Figure 5.1. Note that in our model the driving times of each of the buses do not directly influence each other, thus for our implementation we can simulate each bus individually.

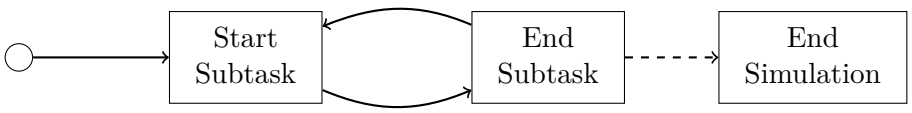

Figure 5.1: Event diagram to simulate a single vehicle.

To integrate this into our simulated annealing algorithm described in Section 4.2, we need to replace the cost function that is used. Instead of calculating a deterministic score, it will simulate the given solution a few times and return the average of these results. However, when

comparing two solutions, we need to make sure that they are compared fairly. Specifically, the randomness of the driving times and energy consumption can cause a worse solution to be more 'lucky' and outperform the better solution. In order to make comparisons more fair, we employ a technique called *Common Random Numbers* (CRN) [11]. With this technique, we make sure that both solutions get the same realizations of driving times, thus solutions cannot gain an advantage by drawing shorter driving times. However, this technique is not applicable to the energy consumption, which we will explain further in Section 5.3.

5.2 Simulating Driving Times

To simulate the driving times, we first need to find appropriate distributions for these driving times. We will go into further detail about this in Chapter 7. As we also explained in Section 1.2, we want to create a simulation where the driving times are also dependent on each other. This way we can create scenarios where, for example, longer driving times in the morning also lead to longer driving times in the evening. This allows us to create days with higher passenger demands that could lead to higher driving times over the whole day. To accomplish this, we generate *instances* of simulated driving times. These contain the simulated driving times of all the trips in the timetable. Thus, within an instance, we simulate a whole day based on expected passenger loads. We explain the distributions we use for this in Section 7.3.

We create different instance types for multiple scenarios. First of all, we have a scenario for 'normal' days. These are days with an average passenger load and mostly average driving times. But, as we want our schedules to be more robust against delays, we will also create scenarios for busier days, where we have more passengers and thus more above average driving times. By simulating on a mix of these scenarios, we make our schedules more robust against these busy days, while maintaining a good schedule under more normal loads.

Within our simulation, we randomly select one of these instances and simulate the whole schedule with the selected instance. Before running the simulations, we can also decide how many of each of the instance types we want to simulate per run. This will be set beforehand, such that we always run the simulation with the same distribution of instance types. This way we also ensure that each instance type is accounted for.

5.3 Simulating Energy Consumption

In our simulation, we account for different bus drivers having different driving styles, and thus different energy consumption figures. However, since we do not create a crew schedule, we need to estimate this, as we do not know who is driving when. In our simulation we create three scenarios. We have scenarios for drivers with below or above average energy consumption and a scenario for a driver with average fuel consumption. Then, before the vehicle pulls out of the garage, we select one of these scenarios randomly. Thus, we simulate one of the available driving styles. In our simulation we assume that these driving styles do not have an influence on the driving time. This might not be a completely realistic assumption, but we do not expect the driving style to have a big effect on the driving time.

As drivers need breaks, there are possibilities along a route where bus drivers can be swapped. However, the places where this can happen are not known in our simulation. Therefore, we allow these driver swaps at the start of every trip. However, to make sure that drivers are not swapped too frequently, a driver needs to drive the bus for at least 2 hours before he is allowed to be swapped. After these 2 hours we will try to swap the drivers as soon as possible, but since we only employ three different driving styles, this will not always lead to a change in energy consumption.

Note that this approach does not allow for CRN to be used on these stochastic energy consumptions, because some buses will drive a different route. This could be solved by using the same driver scenario over the whole solution, but this could lead to unrealistically large energy usage, and thus we will probably overestimate the amount of charging that is required. Another approach could be to select a driver scenario per task, however this could lead to an excessive number of driver swaps. Furthermore, because the vehicles between solutions drive different routes, we do not have the same deadheads in each solution. Thus, the driven distance is different between solutions. For these reasons we will not use CRN for these driver scenarios in our model.

Chapter 6

Required Number of Simulations

The use of simulations to calculate the cost of a solution is a computationally expensive task; especially compared to calculating the cost with deterministic driving times. Thus, in order to increase the runtime performance of our local search method, we want to minimize the number of simulations we perform. However, we still need to make sure that we perform enough simulations in order to make 'correct' decisions. In other words, we want to minimize the number of simulations while still making sure that we perform enough simulations to make sure we select the better solution.

The simplest way to compare two solutions, is by performing an equal number of simulations on both solutions and comparing the averages of those results. In order to do this we need to find out how many simulations are generally needed such that we generally select the better solution. Note that the simulated annealing framework sometimes accepts worse solutions, thus we do not have to be perfect in deciding which solution is better. However, this decision still has to be 'good enough', because otherwise the simulated annealing will have difficulties converging to an optimal solution. The number of simulations we need to compare two solutions, generally depends on the difference in quality of the solutions we compare. Sometimes we need few simulations, while other times you need many simulations. With a constant number of simulations, this means that we need to find a balance between simulating too much, sacrificing computation time, and simulating too little, potentially sacrificing the quality of the final solution.

In order to improve the performance of our algorithm, we will try different techniques to determine how many simulations are needed while comparing the solutions. In Section 6.1 we will explain Optimal Computation Budget Allocation and in Section 6.2 we explain Indifference Zoning. Lastly, in Section 6.3, we explain a technique that uses paired t-tests to determine how many simulations are needed.

6.1 Optimal Computation Budget Allocation

The general idea of Optimal Computation Budget Allocation (OCBA) is to divide a budget of simulations between the solutions we compare such that we maximize the probability of a correct selection. For this we implemented the algorithm described by Chen et al. [4]. For this algorithm we define three parameters: n_0, Δ, and N. These are the initial number of simulations, the budget increase per step, and the maximum budget respectively. The algorithm first simulates each solution n_0 times. It will then repeatedly distribute Δ simulations over the solutions until we hit our computation budget N.

In our simulated annealing, we are comparing two solutions with each other. Let N_1 be the number of simulations we used for the first solution and N_2 the number of simulations we used

for the second solution. This means that after the initial simulations, we have $n_0 = N_1 = N_2$. Furthermore, s_1^2 and s_2^2 denote the sample variances of the corresponding solutions. Without loss of generality, we assume that the first solution is currently better than the second one. Using the results of Chen et al. [4], we will allocate the additional Δ simulations using the rule:

$$\frac{N_1}{N_2} = \frac{s_1}{s_2}. \tag{6.1}$$

Note that we have already done some simulations, thus we need to update the previous values of N_1 and N_2 such that we better approximate the ratio $\frac{s_1}{s_2}$.

A potential drawback of this allocation framework is that it does not allow us to implement CRN. For CRN we would need both solutions to get the same number of simulations, which this framework does not guarantee.

6.2 Indifference Zones

Indifference Zones (IZ) work differently compared to OCBA in that they guarantee a minimum probability of correct selection. This comes at the cost of not having a maximum number of simulations per iteration, as we cannot guarantee that the required minimum probability of correct selection is reached within those simulations. Of course, we could still set a maximum, but then we are not always certain about the probability of correct selection. For this we define an IZ width δ^* and a confidence value α. If two solutions are within δ^* units of each other, the decision maker considers them to be the same, or "indifferent". Then IZ procedures guarantee the following [9]:

$$\mathbb{P}(\text{CS}) = \mathbb{P}(\text{Solution 1 is observed as best} \mid \mu_1 + \delta^* \leq \mu_2) \geq 1 - \alpha. \tag{6.2}$$

Here, $\mathbb{P}(\text{CS})$ denotes the probability of correct selection, and μ_1 and μ_2 denote the mean scores of solutions 1 and 2 respectively.

One such procedure that minimizes the probability of correct selection is Rinott's two-stage procedure [21]. It does this by first simulating each solution n_0 times. Then, it calculates the sample variances of each solution s_i^2. Lastly we can calculate the number of simulations we need for a solution i as follows:

$$N_i = \max\left\{n_0, \left\lceil\left(\frac{hs_i}{\delta^*}\right)^2\right\rceil\right\}. \tag{6.3}$$

In here, h denotes the solution to Rinott's double integral

$$\int_0^\infty \left[\int_0^\infty \Phi\left(\frac{h}{\sqrt{(n_0-1)(\frac{1}{x}+\frac{1}{y})}}\right) f(x)\,\mathrm{d}x\right]^{k-1} f(y)\,\mathrm{d}y = 1-\alpha, \tag{6.4}$$

where k denotes the number of solutions we compare, and f denotes the probability density function of the χ^2 distribution with $n_0 - 1$ degrees of freedom. On modern computers this integral can be solved quickly, however tables for h are also available in [29]. Lastly, each solution gets simulated an additional $N_i - n_0$ number of times, after which the solution with the lowest mean is selected.

This procedure has later been adapted by Yoon and Bekker [31] into the following procedure:

1. Simulate each of the k solutions n_0 times and calculate the sample means $\bar{x}_i$ and sample variances s_i^2. Furthermore, let I be the set containing all the solutions, let $N_i = n_0$, and let $b = \text{argmin}_i\, \bar{x}_i$.

2. Delete solution i ($i \neq b$) from I if

$$N_i \geq \left\lceil \left(\frac{h_1 s_i}{\delta_i} \right)^2 \right\rceil \text{ and } N_b \geq \left\lceil \left(\frac{h_1 s_b}{\delta_i} \right)^2 \right\rceil, \tag{6.5}$$

and delete solution b from I if

$$N_b \geq \left\lceil \left(\frac{h_1 s_b}{\delta_i} \right)^2 \right\rceil \text{ for all } i \neq b. \tag{6.6}$$

Here, $\delta_i = \max\{\delta^*, \bar{x}_i - \bar{x}_b\}$, and h_1 is the solution to the integral

$$\int_0^\infty \left[\int_0^\infty \Phi \left(\frac{h_1}{\sqrt{(N_i - 1)\frac{1}{x} + (N_b - 1)\frac{1}{y}}} \right) f_{N_i - 1}(x)\, \mathrm{d}x \right] f_{N_b - 1}(y)\, \mathrm{d}y = 1 - \frac{\alpha}{k - 1}, \tag{6.7}$$

where $f_n(x)$ denotes the probability density function of the χ^2 distribution with n degrees of freedom.

3. If $|I| = 0$, stop and return solution b as the best solution.

4. Else give each solution in I one additional simulation. Update the sample mean and variance of each solution. Furthermore, for each $i \in I$, set $N_i \leftarrow N_i + 1$. We also restore I such that it includes all solutions, and update $b = \text{argmin}_i\, \bar{x}_i$ and go back to Step 2.

Yoon and Bekker [31] show that Equation (6.2) holds for this procedure, while it needs significantly fewer simulations compared to Rinott's procedure. Note that in this algorithm the value of h_1 can be pre-computed (as can the value h in Rinott's double integral).

We implemented this procedure into our simulated annealing, and made a few changes in order to improve the performance. In early experiments we found the calculations in Step 2 to be quite expensive. Thus, in Step 4, instead of giving each solution in I one additional simulation, we give each solution Δ additional simulations. While this could lead to simulating more than is necessarily needed, overall it improved the performance quite a bit. We further improved the runtime performance by introducing a maximum number of simulations to be run per iteration. This does break the guarantee in Equation (6.2), but we do not see a significant score difference in our early experiments. Here, the score is the outcome of our objective function. We show this in Figure 6.1, where we see that increasing the maximum number of simulations from 100 to 1000 results in similar scores, while being much slower. This is likely due to the fact that simulated annealing does not always pick the better solution. Thus, as long as the scores are still close to each other it does not seem to matter that we might not be completely sure which solution is better. We will further test this in Section 8.2.3.

Lastly, the procedure does not allow for CRN to be implemented, since there is no guarantee that each solution gets an equal number of simulations. However, by also simulating the solutions that are not in I in Step 4, each solution would get an equal number of simulations. Since we are only comparing two solutions (thus $k = 2$), we expect that this does not impact our runtime performance too much, while we get the benefits of having CRN.

6.3 T-Test

Both the OCBA and IZ procedures are designed for comparisons of k solutions. However, in our simulated annealing, we will only be comparing 2 solutions ($k = 2$). Thus, we developed a simpler third method for determining how many simulations are needed. This method works by

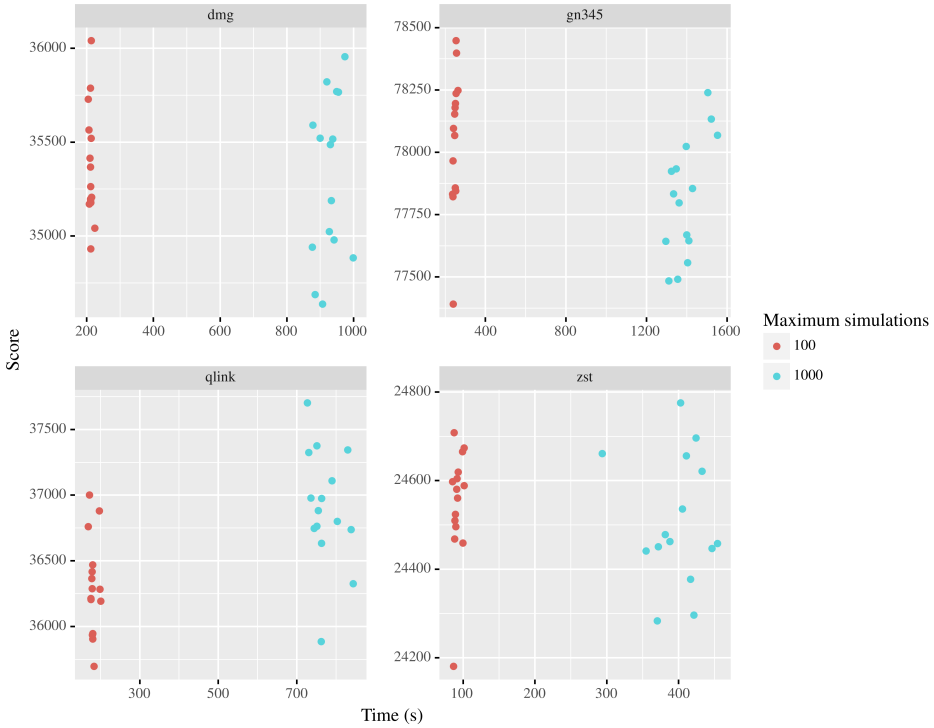

Figure 6.1: Comparison between the score and runtime of different simulated annealing runs for a different maximum number of simulations in the IZ procedure. The other parameters in this experiment are $n_0 = 10$, $\Delta = 10$, and $\alpha = 0.1$. The used datasets are explained in Section 8.1.

performing a paired samples t-test to determine if two solutions are the same. Since we make use of CRN, two solutions with the same random numbers become dependent, justifying the use of the paired samples t-test.

For this procedure, we select four parameters: the initial number of simulations n_0, the number of additional simulations Δ, the maximum number of simulations N, and a confidence parameter α. The procedure is as follows:

1. Simulate each solution n_0 times.
2. Let $\bar{x}$ and s^2 be the sample mean and sample variance of the paired difference in score between each simulation respectively. Then calculate

$$t = \frac{\bar{x}}{\sqrt{\frac{s^2}{n}}}, \tag{6.8}$$

   where n is the number of simulations we ran for a single solution.
3. Use the calculated value of t in a two-sided t-test with $n-1$ degrees of freedom to determine the p-value. Thus,

$$p = 2F_{n-1}(-|t|), \tag{6.9}$$

   where F_{n-1} is the cumulative distribution of the Student's t-distribution with $n-1$ degrees of freedom. Then if $p < \alpha$ or $2n \geq N$, stop and return the solution with the lowest mean as the best solution.
4. Otherwise, simulate each solution an additional Δ number of times and return to Step 2.

A possible way to further improve the runtime performance of this procedure, is to consider that the simulated annealing algorithm does not always select the solution with the better score. Let p_{accept} be a random value that is uniformly distributed on the interval $[0, 1]$, that is used to denote the acceptance probability for the current iteration. Then our simulated annealing algorithm, Algorithm 1, would accept a new solution if $c - c_{\text{new}} \geq T \ln(p_{\text{accept}})$ and $T \geq 1$. Consider two solutions whose scores are not significantly different. Then, if their difference is significantly higher than $T \ln(p_{\text{accept}})$, we can stop simulating further as we know that the simulated annealing algorithm will accept the neighbour. Thus, we also created an extended version of our t-test procedure. If this extended procedure does not find a significant difference between the two solutions, it also checks if the difference between the two solutions is significantly greater than $T \ln(p_{\text{accept}})$. The updated procedure for this is as follows:

1. Simulate each solution n_0 times.
2. Let $\bar{x}$ and s^2 be the sample mean and sample variance of the paired difference in score between each simulation respectively. Then calculate
$$t_1 = \frac{\bar{x}}{\sqrt{\frac{s^2}{n}}}, \tag{6.10}$$
where n is the number of simulations we ran for a single solution.
3. Use the calculated value of t_1 in a two-sided t-test with $n - 1$ degrees of freedom to determine the p-value. Thus,
$$p_1 = 2F_{n-1}(-|t|), \tag{6.11}$$
where F_{n-1} is the cumulative distribution of the Student's t-distribution with $n-1$ degrees of freedom. Then if $p_1 < \alpha$ or $2n \geq N$, stop and return the solution with the lowest mean as the best solution.
4. If $T \leq 1$, go to Step 5. Otherwise, we perform another t-test to see if the difference between the two solutions is significantly greater than $T \ln(p_{\text{accept}})$. For this, calculate
$$t_2 = \frac{\bar{x} - T \ln(p_{\text{accept}})}{\sqrt{\frac{s^2}{n}}}. \tag{6.12}$$
Then,
$$p_2 = F_{n-1}(-t_2). \tag{6.13}$$
If $p_2 < \alpha$, we stop and return the solution with the lowest mean, which is the new solution. Otherwise, we continue to the next step.
5. Simulate each solution an additional Δ number of times and return to Step 2.

6.4 Integration in Simulated Annealing

In our simulated annealing, we have at most two solution comparisons per iteration. Once to evaluate the new neighbour, and, when this new neighbour is accepted, we check if this new neighbour is better than our current best solution. For the first comparison it is important to do enough simulations in order to make a 'correct' decision. Otherwise, the simulated annealing might have a hard time converging to a good optimum. However, when comparing for the best solution, it might be enough to do just a few simulations. Note that we should always resimulate the best solution, otherwise a bad solution might get lucky getting selected as the best solution and not being replaced with a better, but unlucky solution afterward. For this comparison we will test if it makes a difference in solution quality if we use the same method for calculating the number of simulations to perform as our first comparison, or if we can use fewer simulations in this comparison.

Chapter 7

Driving Time Analysis

7.1 Dataset

In order to find a good distribution for the driving times, we analysed historic driving times. This data is mainly from the region of Dordrecht, The Netherlands, and was provided by Qbuzz, the bus company that serves this region. We looked at driving times in this region from the year 2019. This data contains the information about the delay at the start of a trip, the planned driving time of the trip, the actual driving time of the trip, the dwell time during the trip, and the length of the trip. Furthermore, it contains 27 different routes with an average length of 11.8km and on average 22 stops. In total this dataset contains 77 937 trips.

To analyse these driving times, we will first remove some outliers from the dataset. For this, we require the dwell time to be bigger than or equal to 0 and not bigger than the total driving time, as values outside this range are simply not possible. Furthermore, we look at the average speed of the bus. This has to be between 0 and 80 kilometers per hour. Lastly, we also filter trips based on their delay at the start of the trip. We observed some trips to start exactly 1 hour before or after their planned time, suggesting an error in linking the bus with the exact trip they drove. Thus, we filter out trips based on the z-score of their delay at the start. The z-score is the number of standard deviations by which this value is above or below the mean of the observed values. Thus, in this case, it is the number of standard deviations a certain delay at the start is above or below the average delay at the start. For this we will filter out trips where the z-score of the delay at the start is bigger than 2.5. After filtering, our dataset contains 76 485 different trips.

7.2 Variables

To create distributions for the driving times, we first investigate different sources for variation in the driving times. For this, we look into the time of day, the weather conditions, and also the effect of the dwell time.

7.2.1 Time of Day

One source of variation in the driving times is the time of day. Traffic conditions vary over the day, where mornings and afternoons are usually more busy due to people commuting to work or back home. For the same reasons, we also expect there to be more passengers, thereby increasing the dwell time and thus the total driving time. These variations are already accounted for in the bus schedule, as illustrated in Figure 7.1, where we observe higher planned driving times in the morning and late afternoon.

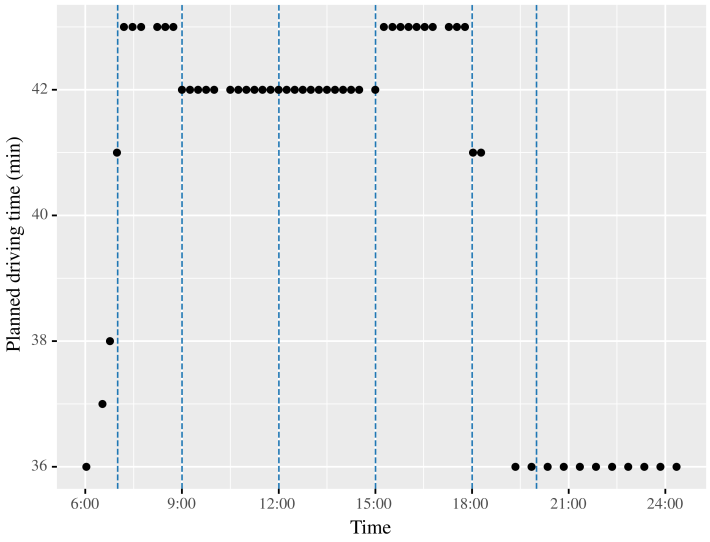

Figure 7.1: Example of planned driving times over a single day. The blue lines indicate the time periods defined in Table 7.1.

Looking at the full data, we extract the average driving time as a percentage of the planned driving time. This is plotted in Figure 7.2, where we grouped each trip by the hour it departs in. In this figure, we do not see big differences in these percentages over the whole day. However, we will still create different distributions for different periods of the day. We base this division on the work of Patnaik et al. [17] and the planned driving times. For our simulation we will use the time periods defined in Table 7.1. These time periods are also indicated by the blue lines in Figures 7.1 and 7.2. These time periods largely correspond to the time periods used by Qbuzz for, for example, their deadhead driving time calculations. The main difference here is that we make use of more time periods.

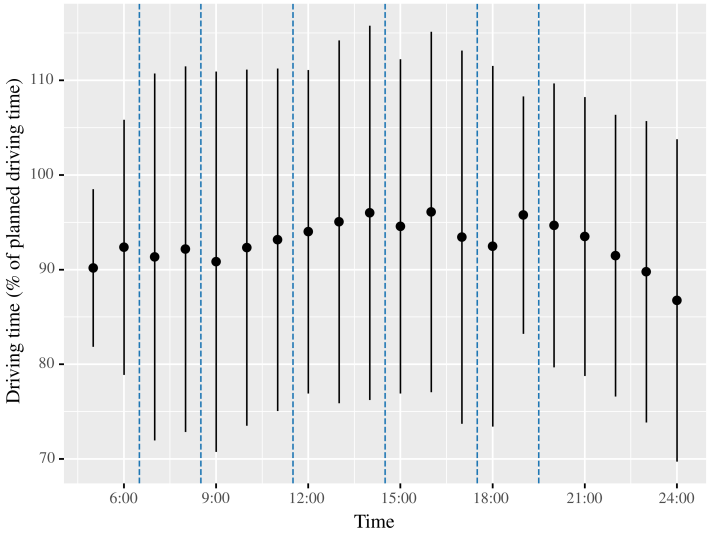

Figure 7.2: Average driving time plus/minus two times its standard deviation as a percentage of the planned driving over a whole day. The blue lines indicate the time periods defined in Table 7.1.

Table 7.1: Time periods used

Time Period	Description
Early Morning	4:00 till 6:59
Morning Peak	7:00 till 8:59
Late Morning	9:00 till 11:59
Early Afternoon	12:00 till 14:59
Afternoon Peak	15:00 till 17:59
Evening	18:00 till 19:59
Late night	20:00 and later

7.2.2 Weather

Another variable we investigated is the effect of the weather on the driving times. For this we expected the driving times to be higher on days with bad weather. The reasoning behind this is that we expect more people to be taking either public transport or go by car, thus increasing driving times due to traffic conditions and higher passenger loads.

To test this hypothesis we used the hourly weather data of 2019 made publicly available by the KNMI [10]. For this, we used the readings from the weather station in Rotterdam. We use information about the duration of rainfall (DR) and the total amount of rainfall (RH) during the timeblock of an hour. For every trip, we calculate the duration and total amount of rainfall during the day the trip took place, the morning of the day the trip took place, and the hour in which the trip departed. For this, we define rain during the morning to be any rain that falls between 6:00 and 9:00, while rain during the day is defined as any rain that falls between 6:00 and 20:00. We performed correlation tests on these variables and the driving time, using Pearson's correlation coefficient. These coefficients are shown in Figure 7.3. These tests indicate no relationship between the driving time and various variables indicating rainfall.

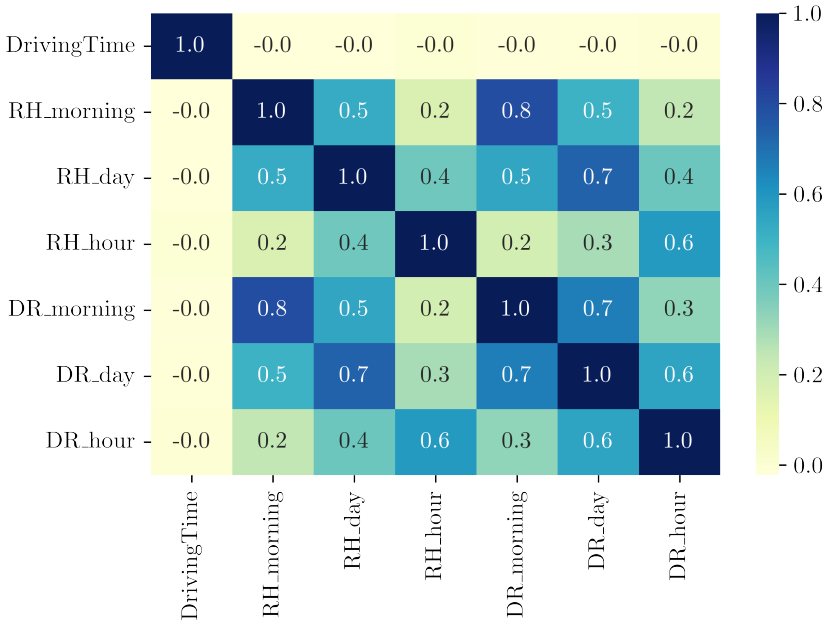

Figure 7.3: Pearson's correlation coefficients between various rainfall parameters and the driving time.

To see why this is the case, we looked at the driving times under various rain conditions. For

this we looked at the average rain intensity in millimeters per hour. We do this both for the whole day (excluding the night) and within a certain hour. We use the rain intensity as this would be the most accurate classifier within the available data. Another factor that could be taken into consideration is, for example, the size of the rain droplets. However, we do not have data for that and, furthermore, this is usually not reported in the weather reports, so we do not expect this to be a major factor when people decide how they travel.

We classify an average rain intensity of 3mm/h or less to be light rain, and higher values are classified as rain. The driving times under these conditions are shown in Figure 7.4. In here there are not always significant differences between rain or no rain. We also note that the amount of rain does not predict the driving time very well, as a lot of rain could lead to lower driving times compared to a bit of rain.

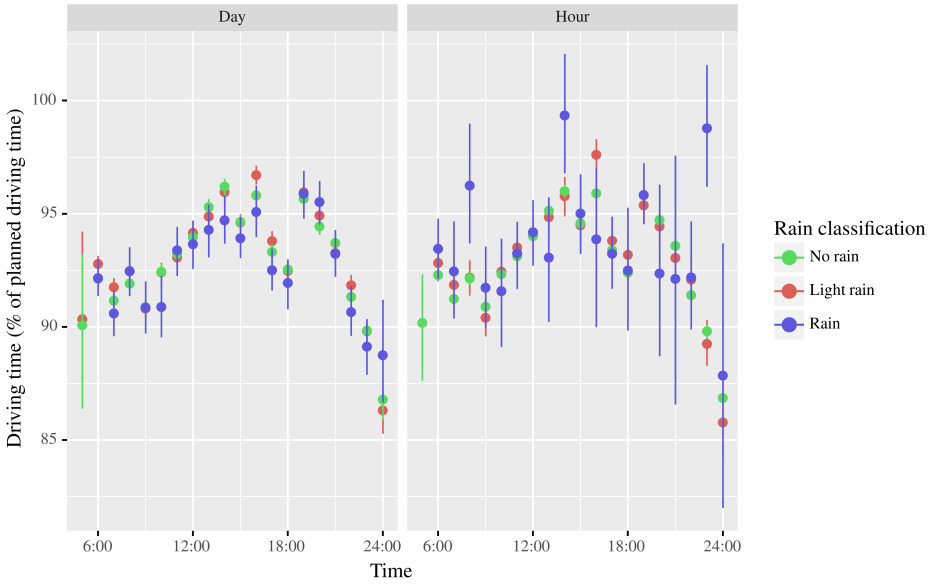

Figure 7.4: Mean driving times with their 95% confidence interval under different rain conditions during the day. Here light rain has an average rain intensity of 3mm/h or less, while it is more than 3mm/h for rain.

From this we conclude that we cannot use these weather patterns in our simulations, because it remains unclear how they influence the driving times. We saw that in some scenarios there do not seem to be significant differences, while in others heavier rain did not necessarily lead to higher driving times. This could be due to passenger behaviour, where for some weather conditions people go by bus rather than by bike, while for other weather conditions people just stay at home. We could not verify this behaviour as we do not have access to passenger data for this route. Thus, we do not include these weather patterns in our simulations, as we can not draw conclusions from our current data.

7.2.3 Passengers

The last variable we looked at is the effect of passenger numbers on the driving times. While we do not have exact passenger data, we do have information about the dwell times, which give an indication of how busy a trip is, because more people moving in or out of the bus leads to longer dwell times. Furthermore, calculating Pearson's correlation coefficient between the driving time and the dwell time, we found a coefficient of 0.6907, showing a correlation between the driving time and the dwell time.

To get a better understanding of how the driving times are influenced and by how much, we

group the dwell times into three categories. For each line variant we calculate the 70th and 90th percentiles of the dwell time and use these to categorize the dwell time of a specific trip. Then we create three groups with driving times. Group 1 contains driving times, where the dwell time is below the 70th percentile of that trip. Group 2 contains driving times, where the dwell time is above the 70th percentile of that trip and below the 90th percentile of that trip. Lastly, group 3 contains the remaining driving times. Grouping on these categories gives us insight into the mean and standard deviation of these driving times. These are shown in Table 7.2.

Table 7.2: Mean and standard deviation of the driving time (as percentage of the planned driving time) grouped by the dwell time category.

	Driving time (% of planned driving time)		
	Mean	Standard deviation	#Trips
Group 1	91.77	9.23	53 519
Group 2	95.72	7.54	15 285
Group 3	99.45	8.62	7 681

From this table we can already see some differences between the driving times with the different dwell times. To confirm that these differences are also significant, we performed Welch's unequal variances t-test. Our null-hypothesis in these tests is: "The means of the driving times from the two tested dwell time categories are equal." For these tests we will use $\alpha = 0.005$, which is lower than the usual 0.05, because we perform multiple t-tests. The p-values for these tests are shown in Table 7.3. Note that some of these values are 0.0, meaning that they are too small to be represented by a 64-bit floating point number. All these p-values are lower than our chosen α, thus these means of the driving times in these categories are significantly different. Note these p-values seem exceptionally low, which is due to the number of trips in each category.

Table 7.3: p-Values of Welch's unequal variances t-test we performed on the driving times in the different dwell time categories.

	Group 1	Group 2	Group 3
Group 1	–	0.0	0.0
Group 2	0.0	–	$7.69 \cdot 10^{-219}$
Group 3	0.0	$7.69 \cdot 10^{-219}$	–

For our implementation it is important to know if there are any patterns in which these higher dwell times happen. For example, are there certain days on which most of the trips encounter higher dwell times? We mainly looked at patterns over a whole day, as our simulation model simulates driving times for a whole day. From Figure 7.5 we can see that these higher dwell times can occur at pretty much any time of the day, except for very early in the morning or late at night, when there are not many bus trips anyway. This is especially visible in the bigger confidence intervals past 19:00. Furthermore, the increase in driving time looks pretty consistent between the categories.

Grouping the trips of each day together, the average dwell time percentile seems to be very consistent with a mean of 0.5049 and a standard deviation of 0.0517. Relating this with the driving time, we find Pearson's correlation coefficient to be 0.3540, indicating that there is some positive correlation. This could be due to some cancellation happening when averaging dwell times and driving times over a whole day. For example, a line might be extremely busy in one direction, but not busy at all in the other direction. Thus, in the average driving times of this line, we might not see the full extend as to how busy it was. We would need to further

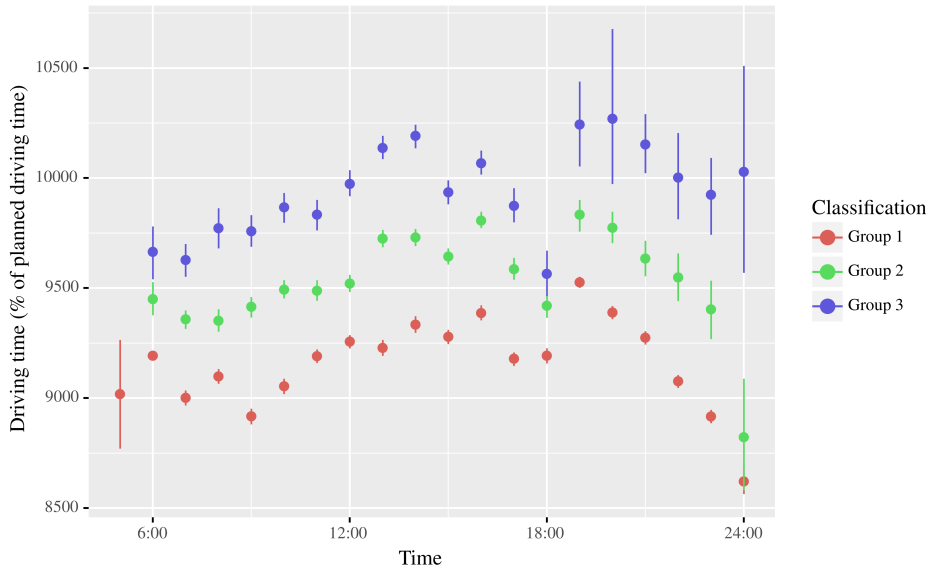

Figure 7.5: Mean driving times with their 95% confidence interval for different dwell time categories.

investigate the data to verify this behaviour. Furthermore, our simulation model then also needs to incorporate such information when simulating driving times. Thus, we generate driving time instances as described in Section 5.2. The distributions used for how often a busy day occurs and the driving times on that day will be further explained in the next section.

7.3 Distributions

For our final implementation we fit different distributions on the driving times for trips departing in the time periods defined in Table 7.1. Based on Section 7.2.3, we will only use driving times with a dwell time that is less than the 70th percentile of the dwell time for that trip. This is to create a baseline distribution, that is not influenced by the more busy days. Then, in our simulation model, we set a probability to generate driving times for a busy day, in which case all simulated driving times are multiplied by a set factor. We base these factors on the results shown in Table 7.2. Thus, with a 20% probability we will generate driving times that are 5% higher and with a 10% probability we will generate driving times that are 10% higher.

We fitted normal distributions for the driving times in each period. These fits are shown in Figure 7.6. For some time periods, we used a single normal distribution to fit the data to, but for others we used a combination of two normal distributions to create a better fit. Thus, these distributions are a mixture of the distributions $N(\mu_1, \sigma_1^2)$ and $N(\mu_2, \sigma_2^2)$ with the weights p and $1 - p$ respectively. The parameters we use for these distributions are given in Table 7.4.

In our simulation, we only use these distributions to generate the driving times of trips, which means that deadheads and trips to and from the garage still use deterministic driving times. This is partly because we do not have data for these driving times, but these driving times vary less in general, since they are not influenced by passenger loads. These deterministic driving times still vary over the day to account for different traffic conditions; we vary these according to specified time periods given in our input data.

All in all, we use the distributions shown in Figure 7.6 for generating driving times of trips, and use deterministic driving times for trips from and to the garage and for deadheads. Furthermore, we simulate busy days by multiplying the generated driving time with a certain factor. Here we

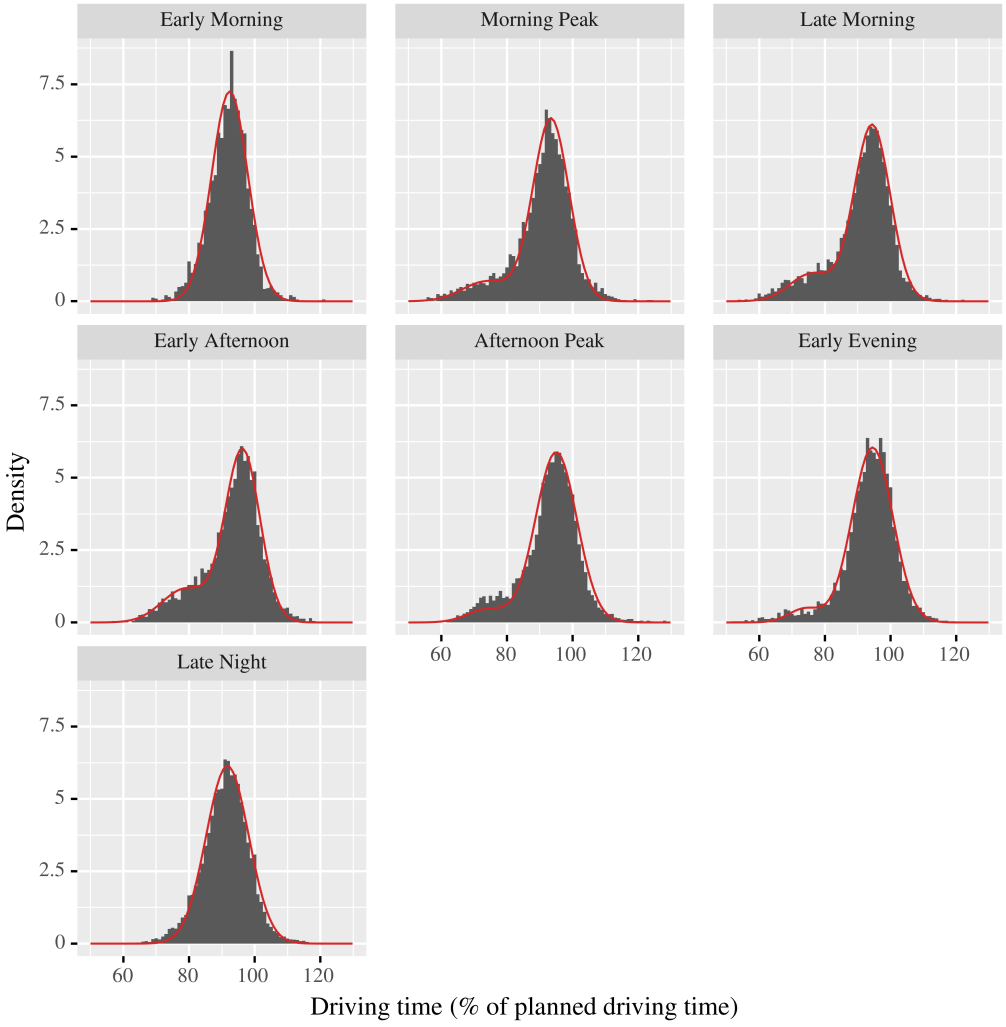

Figure 7.6: Histograms and the fitted probability density function of the driving time distribution for each time period. Here the density is the probability of a certain driving time occurring.

Table 7.4: Parameters of the fitted driving time distributions.

	p	μ_1	σ_1	μ_2	σ_2
Early Morning	1.00	0.924	0.055		
Morning Peak	0.87	0.934	0.055	0.740	0.075
Late Morning	0.84	0.944	0.055	0.760	0.067
Early Afternoon	0.79	0.963	0.053	0.790	0.072
Afternoon Peak	0.93	0.950	0.063	0.740	0.061
Early Evening	0.94	0.945	0.062	0.740	0.050
Late Night	1.00	0.917	0.065		

generate driving times which are 5% higher with a probability of 20%, and with a probability of 10% we generate driving times that are 10% higher.

Chapter 8

Experiments and Results

8.1 General Setup

In this chapter we will compare the performance of our stochastic E-VSP model with the deterministic one. We will also compare the different methods for calculating the number of simulations required, which are described in Chapter 6. For this we use several instances from various regions in The Netherlands. These instances are from the regions of Dordrecht, Groningen, and Utrecht. The instances are provided by Qbuzz, which is the bus company that serves these areas. We implemented our algorithms in C# .NET 6.0, and we use IBM ILOG CPLEX version 22.1 for solving our ILPs. We run the algorithms on a computer with an Intel® Core™ i5-6400 quadcore processor, with 16 GB of RAM.

Table 8.1: Overview of the used datasets and their parameters.

Dataset	#Trips	#Lines	Battery Capacity (kWh)
`dmg`	631	8	232
`gn345`	463	3	184
`qlink`	590	3	160
`zst`	317	2	232

For all our simulated annealing runs, we use a starting temperature $T = 10\,000$ and a cooling rate $\alpha = 0.0001$. To get the scores for the columns in the recombination, we simulate each simulated annealing 1 000 times, where we use CRN such that everything is evaluated on the same driving times.

To compare the different final solutions found by our algorithm, we simulate each solution 1 000 times. We do not employ CRN in these simulations. Thus, each solution is simulated on different driving times. However, due to the number of simulations we use, we do not expect this to result in unfair comparisons, as the number of simulations should reduce the variance in score. When comparing solutions created with deterministic driving times, we also simulate these 1 000 times using stochastic driving times.

8.2 Required Number of Simulations

To compare the methods for calculating the required number of simulations described in Chapter 6, we mainly look at the running times and scores of different simulated annealing runs. The running time is measured inside our program as the wall-clock time of the simulated annealing

run. This only includes work that is done within our simulated annealing run and thus excludes something like data reading and parsing.

An overview of the tested methods and their parameters is provided in Table 8.2. Note here that for the "Equal" methods when the maximum number of simulation is, for example, 100, each solution is simulated half that number, thus 50 times. Furthermore, we included a method where simulations are equally divided and CRN is disabled, in order to understand the effect of CRN on the simulated annealing. Lastly, with "TTest2" we refer to our extended t-test method, thus the t-test method that potentially performs an additional t-test based on the acceptance probability. These parameters are chosen such that for each method they should provide a good tradeoff between runtime performance and the final score.

Table 8.2: Parameters for the different methods for calculating the required number of simulations we are comparing. For an explanation of these parameters see Chapter 6.

Method	n_0	Δ	N	α
Equal			20	
Equal			50	
Equal			100	
Equal (no CRN)			100	
OCBA	20	5	100	
IZ	10	10	100	0.1
TTest	10	10	200	0.2
TTest2	10	10	200	0.2

To get a fair comparison between the different methods for calculating the required number of simulations, all our runs use the same parameters for our simulated annealing. But, more importantly, we run our simulated annealing for a constant number of iterations, meaning that faster methods do not get a time advantage. This is also described in Section 4.2, but in short, we run the simulated annealing until the temperature is below 1, after which we run a set number of iterations in hillclimb mode.

8.2.1 Performance

In Figure 8.1, we show the runtime and score of the different simulated annealing runs for all different methods for calculating the required number of simulations in Table 8.2. Here we see a big difference when it comes to the scores of the methods that employ CRN compared to the methods that do not employ this technique, which are OCBA and the equal method without CRN. While we do think that the runtime of some of these methods that do not employ CRN could be improved, we do not see how this could also improve the score, as the number of iterations is constant.

To get a better overview of how the methods that do employ CRN compare against each other, we only show these methods in Figure 8.2. In here, we see that the use of IZ or t-tests can lead to runtime improvements while not scoring worse. From this figure we see that the IZ and the different t-test methods are generally a bit faster, while scoring similar or even better compared to the equal distribution with $N = 100$. In general, we see that the IZ and t-test methods perform the best in terms of their score, where the t-test methods seem slightly faster.

8.2.2 Number of simulations

In the previous section we saw that the IZ and t-test methods perform very similarly both runtime- and score-wise, where the single t-test might be a slight bit quicker. We also sampled

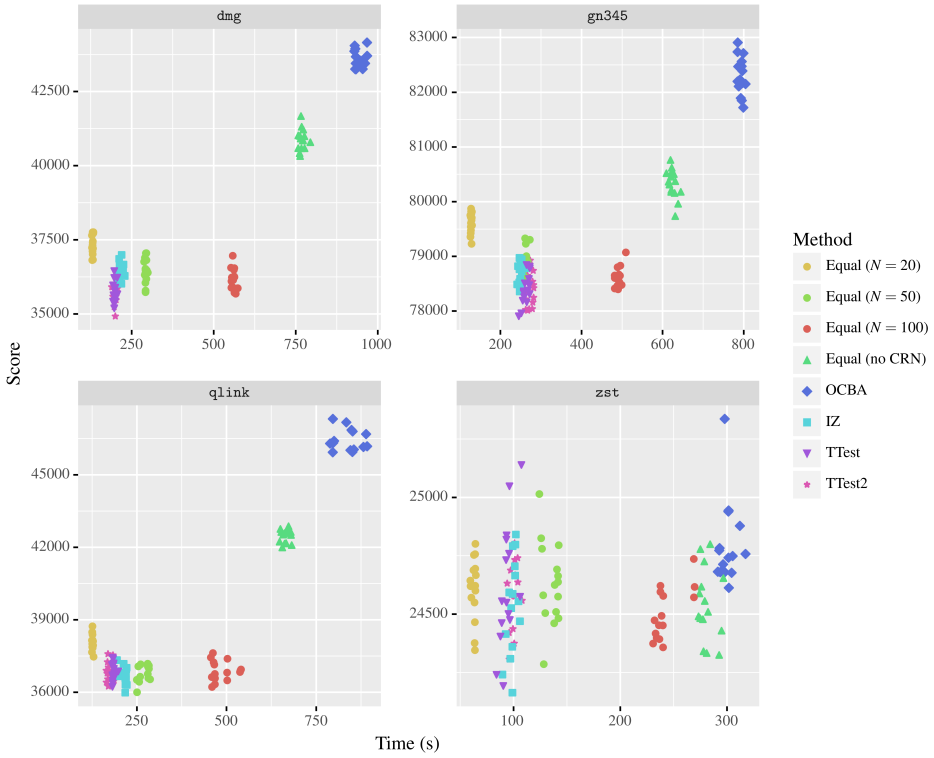

Figure 8.1: Comparison of the score and runtime of various simulated annealing runs for different methods for calculating the required number of simulations on different datasets.

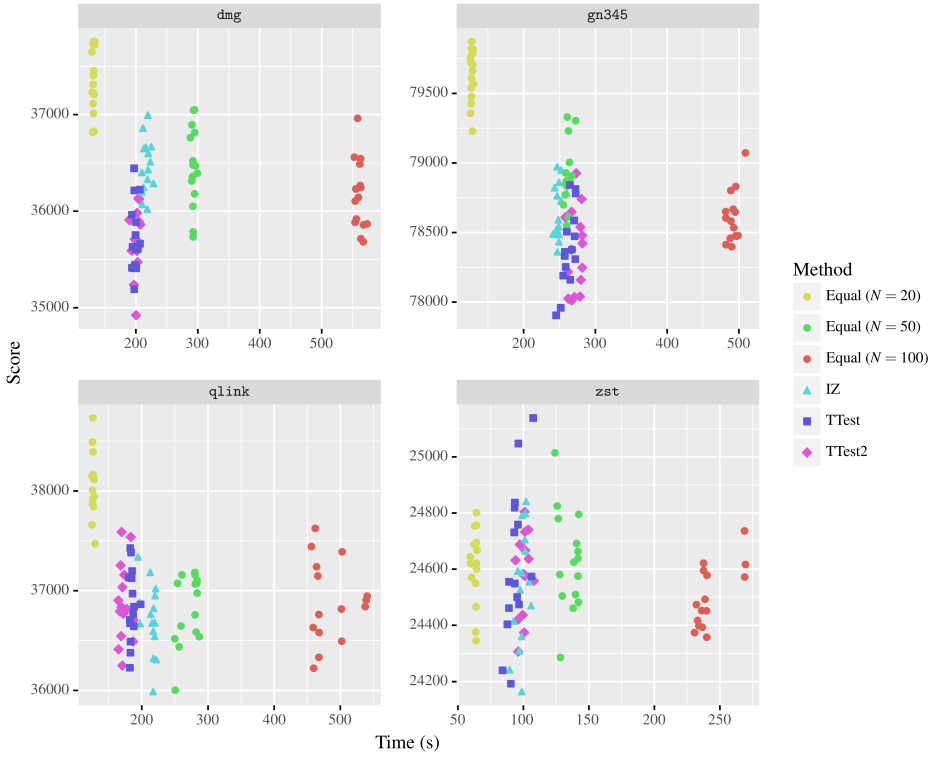

Figure 8.2: Comparison of the score and runtime of various simulated annealing runs for different methods for calculating the required number of simulations on different datasets. Here we exclude all the methods that do not employ CRN.

some iterations to see how many simulations a method used in that iteration. The results for these three methods are shown in Figure 8.3. Note that we still use the parameter from Table 8.2, thus the t-test methods are allowed more simulations in an iteration than the IZ method. Here, we see that our extended t-test method uses the least number of simulations, while the IZ method has a lot of situations where it uses its maximum number of simulations.

Considering the runtime performance of these methods in Figure 8.2, it seems that, compared to the IZ method, the t-tests are quite a bit slower in determining wether more simulations are required, although this might also be due to the fact that the IZ method is limited in its maximum number of simulations. Furthermore, we also note that while the "TTest2" method uses the least number of simulations of these three methods, it is not necessarily the fastest overall.

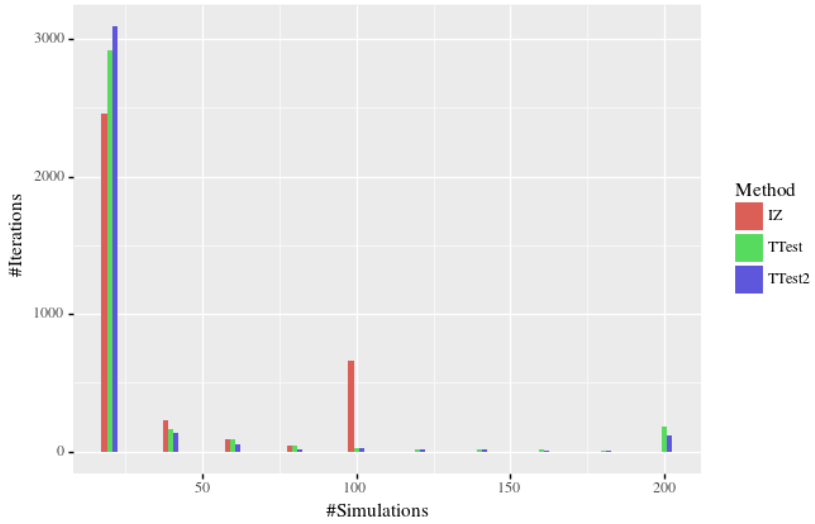

Figure 8.3: Histogram of the number of simulations used in an iteration for different methods for calculating the required number of simulations.

8.2.3 Hillclimb

In Section 6.2 we suggested that the use of simulated annealing allows us to be a bit more 'sloppy' when it comes to deciding which solution is better when deciding between two solutions that are close to each other. That is, since there is a probability that the simulated annealing does not select the better solution, it might not be necessary to be completely sure about which solution actually is the better one. To test this, we compare our simulated annealing runs with only doing hillclimb. Using an equal number of simulations for each solution as a baseline, we would expect that the hillclimb with, for example, the IZ method to perform relatively better. We use the equal distribution with $N = 200$ as a baseline, and test the IZ method and the t-test method which performs a single t-test. In these tests we make sure that both the simulated annealing and the hillclimb take about the same time. Futhermore, just as with the simulated annealing runs, we perform multistart with the hillclimb runs, meaning that different hillclimb runs have a different seed for the random number generator that is used for generating neighbours.

In Table 8.3 we compare the scores between various simulated annealing and hillclimb runs. Here, a negative difference means that the hillclimb improves on the simulated annealing. From these results it is not clear that the hillclimb with either the IZ or t-test method performs relatively better than the hillclimb with the equal distribution. Thus, showing that we do need to be somewhat sure about which solution is better. Although, it is also not clear if the methods perform relatively worse, showing that our current 'sloppiness' when dealing with these close

solutions might already be good enough.

Table 8.3: Mean scores of various simulated annealing runs compared to the scores of various hillclimb runs.

Dataset	Method	Hillclimb	Simulated annealing	Difference
dmg	Equal ($N = 200$)	35 686	36 144	−1.27%
	IZ	35 730	36 462	−2.01%
	TTest	35 322	35 724	−1.13%
gn345	Equal ($N = 200$)	79 698	78 391	1.67%
	IZ	80 148	78 654	1.90%
	TTest	79 845	78 391	1.85%
qlink	Equal ($N = 200$)	36 344	37 237	−2.40%
	IZ	35 995	36 716	−1.96%
	TTest	35 818	36 856	−2.82%
zst	Equal ($N = 200$)	25 162	24 540	2.53%
	IZ	25 120	24 535	2.38%
	TTest	25 161	24 619	2.20%

8.2.4 Best solution comparison

As we mentioned in Section 6.4, we also test wether we can use fewer simulations when comparing the newly accepted neighbour with the best known solution. For this, we test two configurations. One where we use the same method that is also used in the other comparison, and one where we use just 10 simulations for each solution. We call this last configuration "E10". We test these configurations for the t-test methods, the IZ method, and for the equal distribution with $N = 50$.

Comparisons of the scores and runtime performance for these configurations are shown in Figures 8.4 and 8.5 respectively. In here, we also show the 95% confidence interval for the means. These comparisons show that switching to the E10 configuration reduces the runtime performance, while scoring very similar.

8.2.5 Recombination

In Section 8.2.1, we have seen the performance of the different simulated annealing runs for different methods for calculating the required number of simulations. However, as described in Section 4.2, we use a recombination technique to create a final schedule with these simulated annealing runs. For some of the methods, we compare the results from different runs to see if the used method has an effect on the recombination performance. In Figure 8.6 we show these recombination scores for the IZ method, the t-test methods and the equal distribution with $N = 50$. These results are similar to the results in Section 8.2.1. We will look more into the effects of recombination in Section 8.3.1.

8.3 Stochastic E-VSP

We also test the robustness of our final solutions. For this, we will compare the use of stochastic driving times with the deterministic driving times in our input data. These deterministic driving times already contain some slack in order to make the schedule more robust. For the simulation

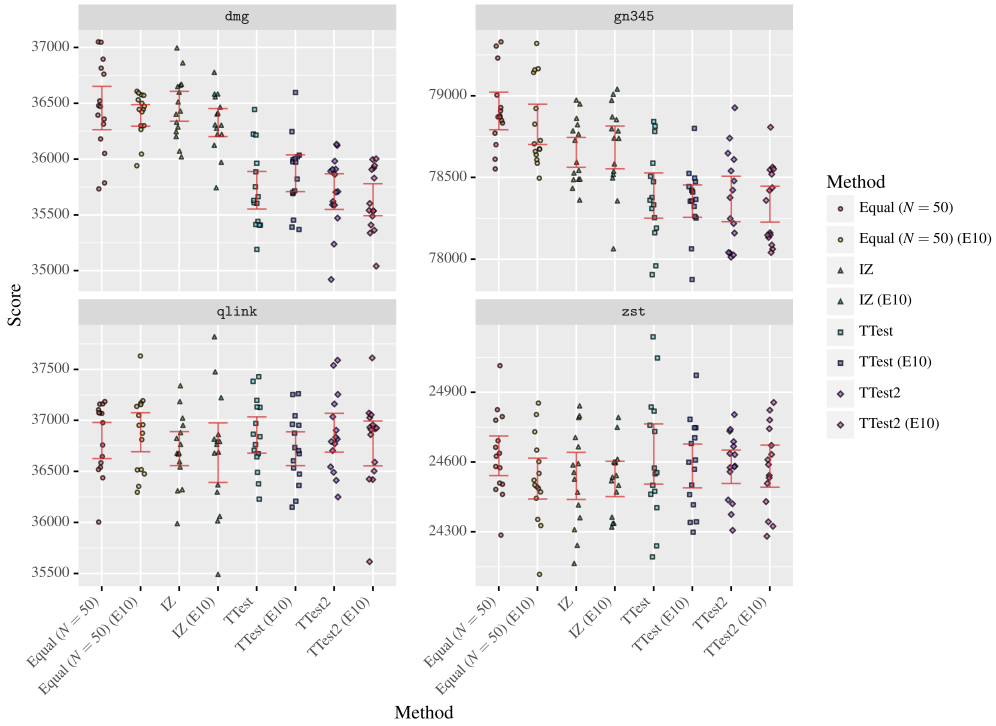

Figure 8.4: Comparison of the scores of various simulated annealing runs for different methods for calculating the required number of simulations and different configurations for comparing the newly accepted neighbour with the current best solution.

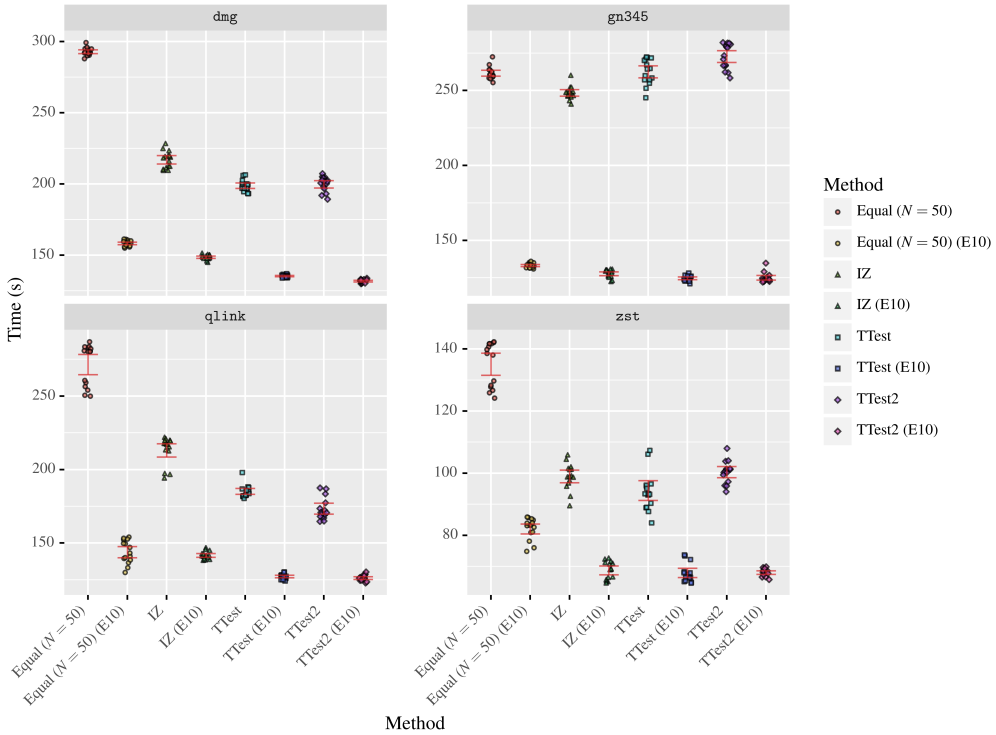

Figure 8.5: Comparison of the runtimes of various simulated annealing runs for different methods for calculating the required number of simulations and different configurations for comparing the newly accepted neighbour with the current best solution.

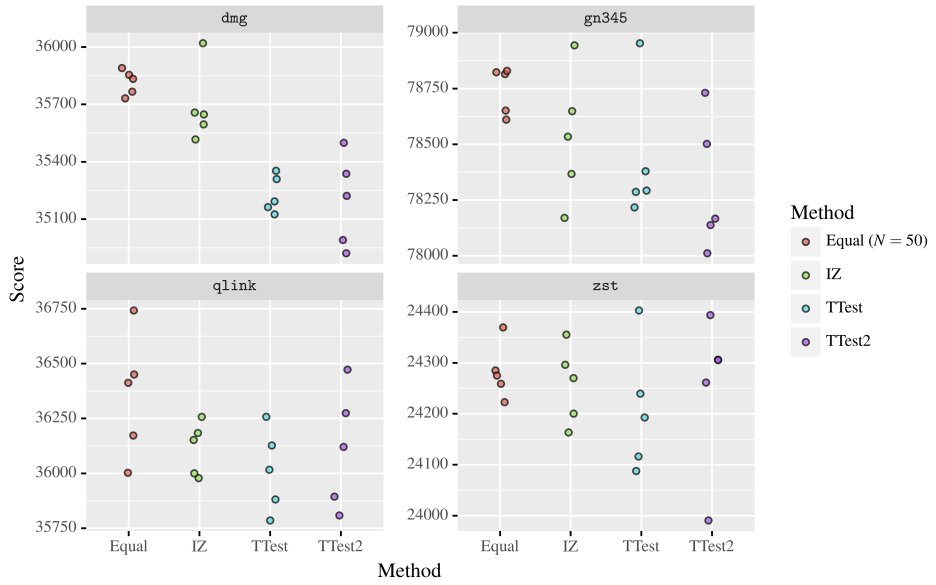

Figure 8.6: Scores of the solutions created with recombination, where we use the specified method for the methods for calculating the required number of simulations in our simulated annealing.

with stochastic driving times, we use the "TTest" method with the same parameters as in Table 8.2.

First we compare the lateness, maximum DoD, and number of vehicles used. Averages of these statistics over multiple runs are shown in Table 8.4. In this table, we use L to denote the set containing the lateness values for each trip. We define the lateness as the difference between the planned starting time of a trip and the earliest time a bus could depart for this trip, where a positive value means the trip started late. Furthermore, these values are in minutes. This means that $\bar{L}$ denotes the mean lateness of all the trips, and we use L_{95} to denote the 95th percentile of the lateness. We define the punctuality to be the percentage of trips that started on time. Lastly, the column "Mean Late" denotes the mean of the set $\{x \in L \mid x > 0\}$. Thus, it is the average number of minutes a bus starts late, given that it starts late. From these results we see some reductions in the lateness of a trip, and also a reduction in the "Mean Late" statistic. Furthermore, the other statistics seem more or less similar. We also checked these results with our contact person at Qbuzz, confirming that these lateness values are similar to what they encounter in practice.

8.3.1 Recombination

We also compare our simulated annealing runs with the results from the recombination. For this, we compare the scores of the schedules. We use 15 simulated annealing runs for the recombination. During our simulated annealing we keep multiple of our best solutions, which are used for the recombination. From a certain point in our simulated annealing we will collect the new best solutions, however after collecting a new best solution, we wait a few iterations before collecting the next one. This is done in order to not collect solutions that just differ in one neighbour. This means that we collect about 20 to 40 solutions per simulated annealing run, which results in about 7 000 to 56 000 columns depending on the dataset we use. Note that there may be duplicate columns in here, as we collect multiple solutions from the same simulated annealing run. Lastly, we set the time limit of the ILP to 20 minutes in order to reduce the total computation time.

Table 8.4: Various statistics regarding the final solutions calculated with either deterministic or stochastic driving times.

Dataset	Driving times	#Vehicles	$\bar{L}$	L_{95}	Mean Late	Punctuality	Max DoD
dmg	Deterministic	45.0	-3.8	2.0	2.9	89.5%	53.2%
	Stochastic	52.2	-4.5	1.0	2.0	92.7%	58.3%
gn345	Deterministic	92.0	-2.3	1.2	4.7	93.4%	95.1%
	Stochastic	94.0	-2.9	1.0	3.3	94.7%	95.2%
qlink	Deterministic	37.8	0.8	5.0	5.1	83.2%	52.9%
	Stochastic	45.8	-3.0	1.6	2.7	91.8%	62.3%
zst	Deterministic	38.6	-5.9	0.0	2.4	98.4%	69.8%
	Stochastic	37.4	-6.2	0.0	2.2	98.5%	70.0%

The results of this are shown in Table 8.5, where we note the average results for different statistics. Note that the reported time includes both the recombination and the score calculation of the columns, which is why some instances report a time that is above 20 minutes. Also, the 'Improvement' denotes the percentage improvement compared to the best simulated annealing score, where a negative value means that the recombination did not improve compared to the simulated annealing. Here we see that a few of our datasets run into the time limit of 20 minutes. Furthermore, these are also the only tests with fairly big integrality gaps, and they do not show an improvement compared to the simulated annealing. However, the results on the other tests are quite promising as they show 1 to 3 percent improvements compared to simulated annealing.

Table 8.5: Various statistics regarding the performance of the recombination. Here 'Gap' denotes the gap to the LP relaxation, and 'Improvement' denotes the percentage improvement compared to the best simulated annealing score.

Dataset	Driving Times	#Columns	Gap	Improvement	Time (s)
dmg	Deterministic	20 523.7	5.889%	−2.377%	1 200.41
	Stochastic	27 566.7	7.411%	−3.425%	1 251.85
gn345	Deterministic	33 320.8	0.013%	1.545%	467.80
	Stochastic	55 770.7	0.038%	2.372%	760.99
qlink	Deterministic	21 931.6	0.005%	3.645%	18.70
	Stochastic	25 805.3	4.672%	−1.237%	1 238.82
zst	Deterministic	7 037.4	0.009%	2.713%	74.39
	Stochastic	23 271.2	0.010%	3.206%	245.42

8.3.2 Lateness

First we looked at the histogram of the lateness values (the set L) we encountered in our simulations. This is displayed in Figure 8.7. There are a few things to notice in this figure. First, we see that for most trips the bus is about 2 minutes or less too early, which is normal and expected behaviour. We also see some peaks at -10 and -15 minutes. This is especially clear in the qlink dataset. These peaks correspond to the frequency of some of the lines in these datasets. We suspect that these peaks are due to the dataset not containing many lines, thus the only way to increase robustness is to arrive one trip early. Furthermore, for every dataset

there is also a big peak at 0 minutes. This is due to the buses charging until their trip starts.

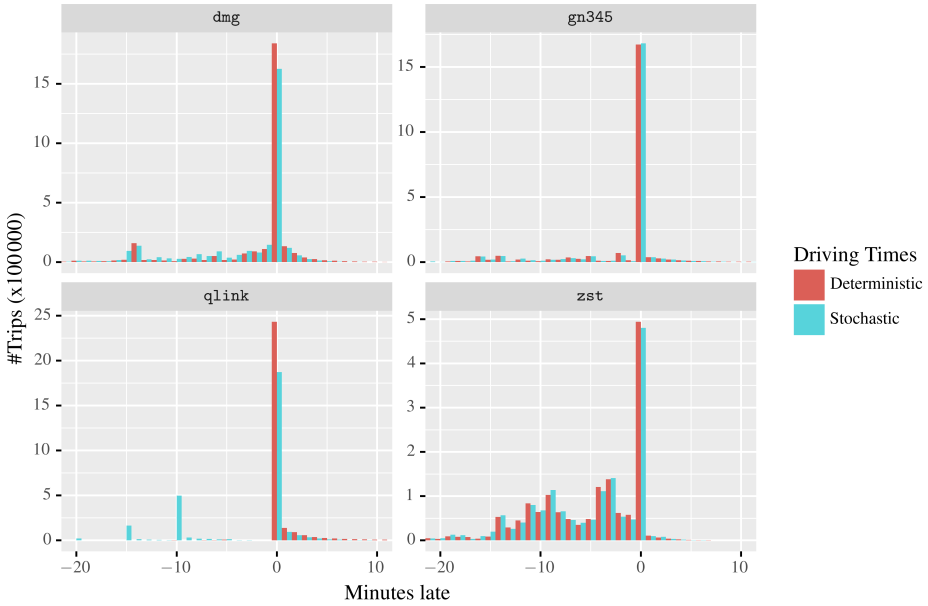

Figure 8.7: Histogram of the set L.

We ran our algorithm with different penalty factors for the lateness, in order to get a better overview of how the stochastic driving times compare to the deterministic driving times. Note that in these tests we only change the value α in our objective function (Equation (4.1)).

For this, we first compare the punctuality in these solutions to the operating cost and to the number of vehicles used. Here the punctuality is the percentage of trips that started on time. These comparisons are shown in Figures 8.8 and 8.9 respectively. In these figures we see that using stochastic driving times, we generally get solutions with a better punctuality, but they use a bit more vehicles. This is also reflected in Figure 8.8, where we see higher operating costs for the same punctuality. Note that the operating cost also contains a time component. Thus, buses needed to wait for their trip to start increases the operating cost. However, in the case of stochastic driving times, not waiting may induce a penalty for the lateness on the next trip.

We also compare the lateness itself. For this we look at the average lateness compared to the operating cost and to the number of vehicles used. The results of these comparisons are shown in Figures 8.10 and 8.11 respectively. These figures show results that are similar to the punctuality, where we decrease the average lateness for a bit more vehicle usage, which is reflected in the operating costs.

8.3.3 Depth of Discharge

We also ran our algorithm with different penalty factors for the DoD. Here, the lateness penalties stayed constant. In Figure 8.12, we see the maximum DoD for different number of vehicles used in the final solution. For most of the datasets, we see very similar results between the use of stochastic and deterministic driving times. Meaning that the maximum DoD is more or less the same for both deterministic and stochastic driving times. The main differences here are in the number of vehicles a solution requires.

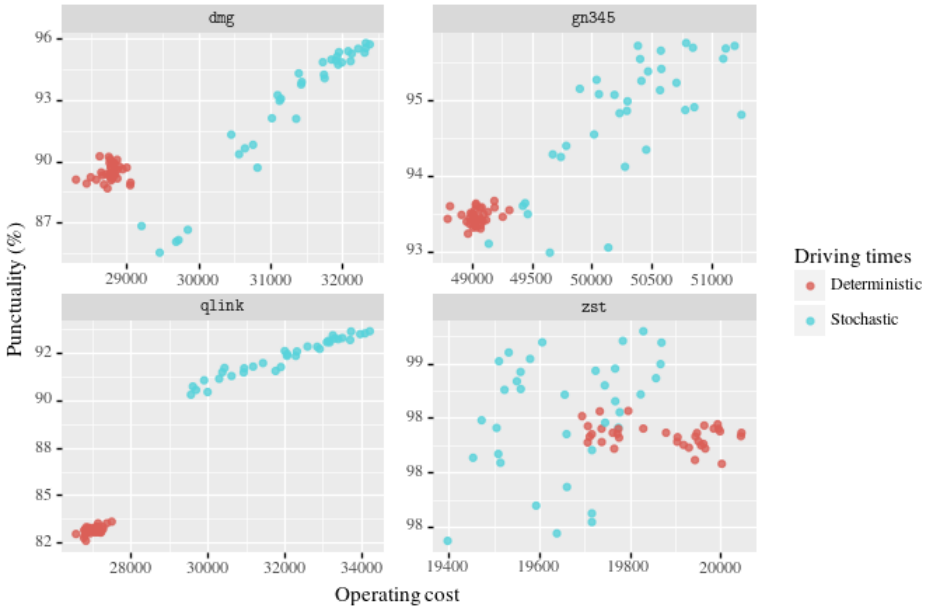

Figure 8.8: Punctuality compared to the operating cost.

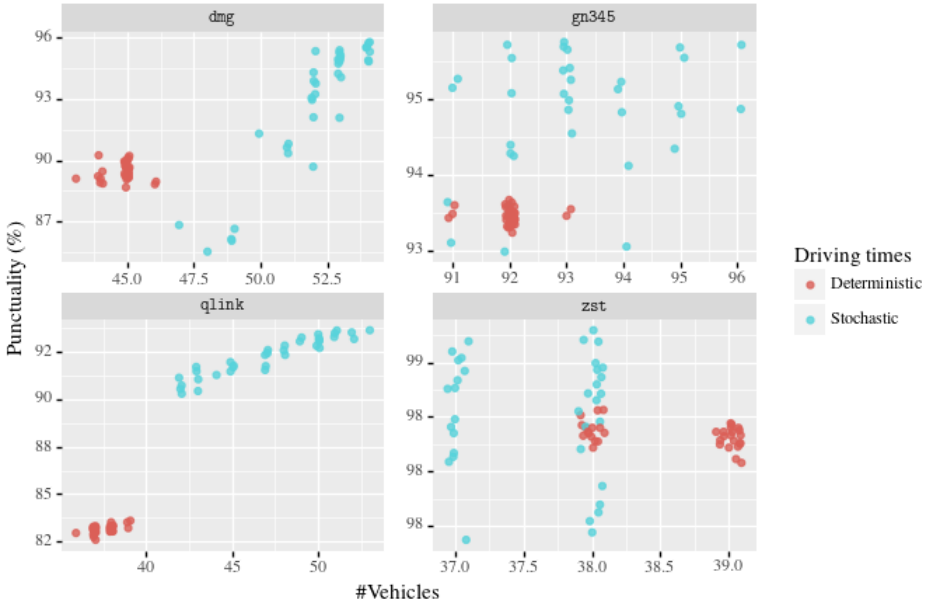

Figure 8.9: Punctuality compared to the number of vehicles used.

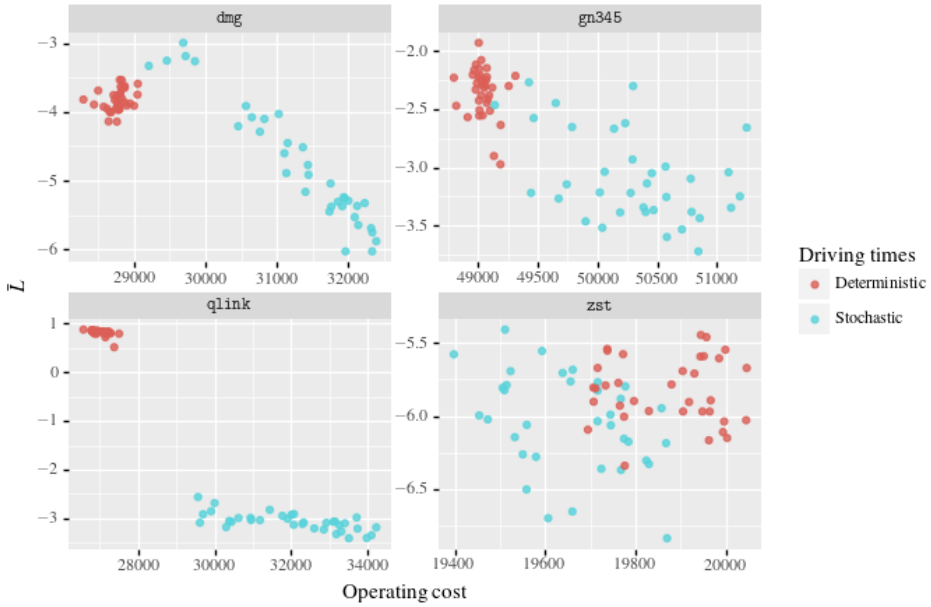

Figure 8.10: Mean minutes late compared to the operating cost.

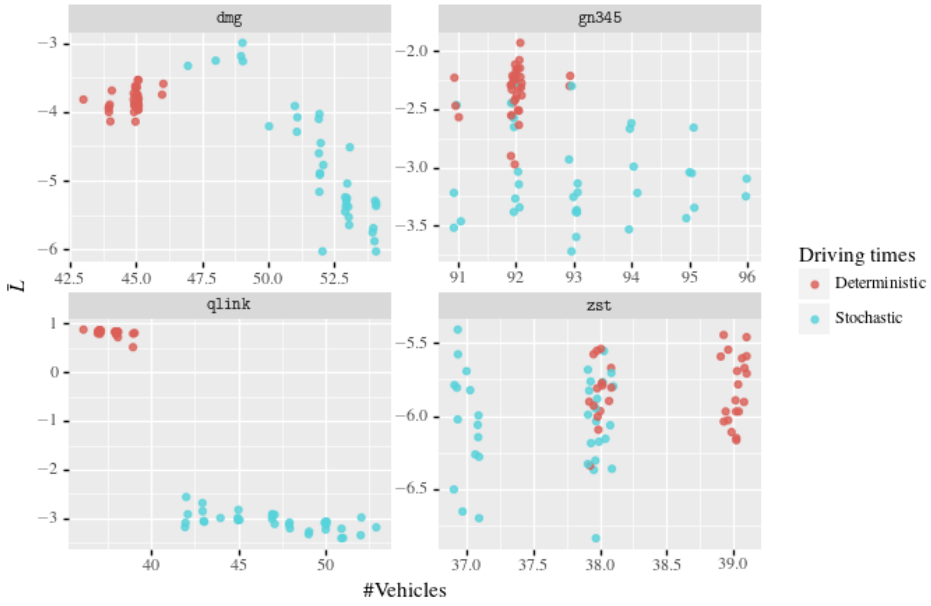

Figure 8.11: Mean minutes late compared to the number of vehicles used.

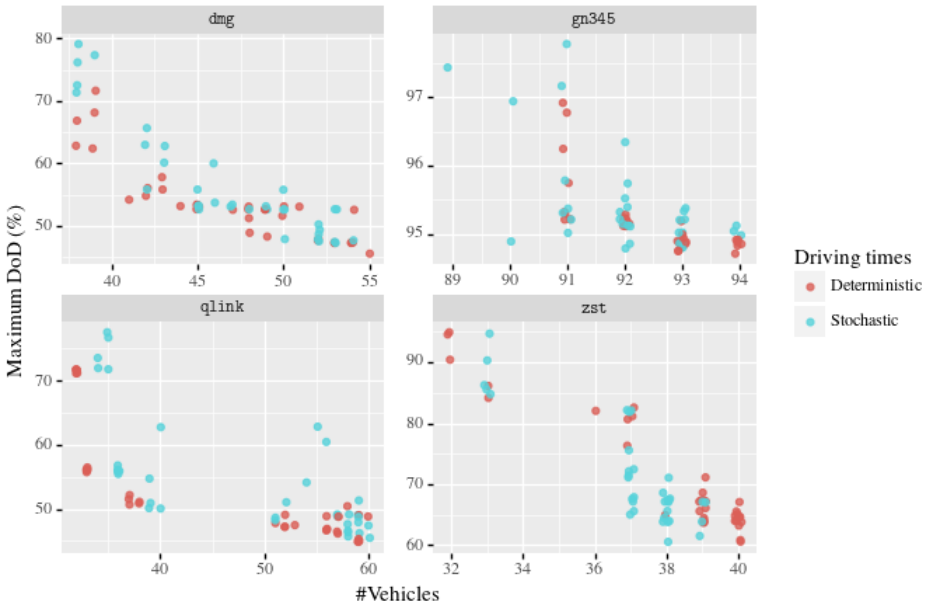

Figure 8.12: Maximum DoD for different number of vehicles used.

Chapter 9

Conclusion

9.1 Summary

In this thesis we created a model to solve E-VSP with stochastic driving times. For this we used a local search approach in the form of simulated annealing. We extended an existing simulated annealing approach for the case of deterministic driving times with simulations, such that we could use it with stochastic driving times. The use of these simulations forms a bottleneck in the runtime performance of our local search, thus we explored different methods for calculating the required number of simulations. With some of these methods we could keep number of simulations and runtime low, while being confident about which solution to select.

Furthermore, we analyzed historic driving times to find out which distributions to use in our model. For this, we explored different external factors that should be considered in these distributions. These factors are the time of day, the weather, and number of passengers. Here, we determined that it is unclear how the weather influences the driving times. We used the other two factors to create our final distributions.

9.2 Conclusion

9.2.1 Required Number of Simulations

When it comes to our simulation setup, we saw in Section 8.2.1 that the use of CRN is very effective. In our case, it even seems the use of CRN is more or less required in order for the simulated annealing to converge to a better score. Furthermore, when comparing the different methods for calculating the required number of simulations, we saw in Figure 8.2 that both the t-test methods and the IZ method are quite similar in terms of score and runtime performance. They are also the best performers of all the tested methods. In Section 8.2.2, we note that the t-test method uses the least number of simulations, meaning that, depending on the runtime efficiency of the simulations, these could be the best choice. In our case, the simulations seem quite efficient, and we seem more limited by how fast we determine if we need additional simulations. Here, the IZ method seems to be much faster, but this is cancelled out by the need of more simulations.

Comparing our t-test method to our extended t-test method, we saw a similar performance in both score in runtime. Although our extended t-test method uses fewer simulations overall, the required runtime for this was offset by the calculations for the extra t-test. So, in our case, the use of the extended t-test did not improve the runtime. However, it could benefit other problems where the simulations are less efficient in terms of computation time.

Overall we can conclude that the use of these different methods can increase the runtime performance while not giving up on solution quality. We have shown this in both the simulated annealing performance and the recombination performance. Furthermore, we can achieve an even better runtime performance by using relatively few simulations when comparing for the best solution. In our case just doing 10 simulations per schedule did not reduce the overall score, while running faster.

Lastly, we experimented with using only hillclimb in order to get a better understanding of how we should handle solutions that are very close to each other. However, these tests did not yield conclusive results. Futhermore, we only tested if we can be a bit 'sloppy' when it comes to these solutions, but there are other options that could be explored. For example, one could take the variance of the simulated solutions into account when deciding which solution is better.

9.2.2 Stochastic E-VSP

For the stochastic E-VSP problem, we see in Table 8.4 and Figure 8.7 a decrease in lateness compared to using deterministic driving times. This is not only true for the average lateness, but more importantly in the extreme cases. That is, we see reduction of the 95th percentile of the lateness. From this we can conclude that the use these stochastic driving times indeed increases robustness of our schedules. However, this comes at a small cost. In general these solutions could require about the same number of vehicles, although on average solutions created with stochastic driving times require a bit more vehicles. The additional number of vehicles required in combination with the extra waiting time that is sometimes needed, is also reflected in the operating costs, which increases compared to the use of deterministic driving times.

9.2.3 Recombination

From the results of the recombination of different simulations runs, we see improvements on the solution quality of up to 3%. However, for some of our experiments the ILP ran into the time limit of 20 minutes, where it didn't find any improvements compared to the simulated annealing. This is especially visible in the integrality gaps reported in Table 8.5, where the experiments that ran into the time limit have a fairly big gap compared to the experiments that did not run into this limit. It could be that, given more time, these instances also improve compared to the simulated annealing, but they could also already be at their optimum. This would need to be further verified with longer experiments. However, for the other instances, we can improve our best solution quite quickly. Thus, although it is not always improving our best result, this extra recombination step seems a good addition to our simulated annealing.

9.3 Future Research

We showed that our model for stochastic E-VSP is quite successful in creating more robust schedules. This approach could be further enhanced to increase the usefulness of our results. One of the assumptions we made for our stochastic driving times is that we use the distribution for every line in every direction. This is not necessarily realistic, as there are lines that solely cross city centers, but also lines over longer distances. Buses on these lines encounter very different traffic conditions and thus could end up with different distributions for their driving time. Our work could be extended to include a distinction between these different line types, but would require more research into how these distinctions could be made and also the distributions that are required.

Our model itself could also be further enhanced. For the simulated annealing we used the same neighbours as ten Bosch et al. [24]. These neighbours are very simple and seem to deliver good

results, but maybe more sophisticated neighbours could be added in order to find better solutions. For this, it would also be interesting to see if adding these more sophisticated neighbours has an effect on the recombination step. This recombination step also left us with some further research questions. For example, we did not experiment with how we select our columns from the simulated annealing. Simply always picking the best solution, might leave the ILP with not much room to improve, but selecting a lot of 'bad' columns might unnecessarily increase the total runtime.

Lastly, further research could also be done on the integration of the investigated methods for calculating the number of simulations required into the simulated annealing. We did some experiments on this with our extended t-test method, using the fact that simulated annealing does not always select the better solution. However, further research could be done to answer the question on what to do when we have solutions that are close to each other, and one solution is not significantly better than the other. Although it is also the question if we need to do anything about that situation.

Bibliography

[1] M. A. Ahmed and T. M. Alkhamis. 'Simulation-based optimization using simulated annealing with ranking and selection'. In: *Computers & Operations Research* 29.4 (2002), pp. 387–402. DOI: 10.1016/S0305-0548(00)00073-3.

[2] Y. Bie, J. Ji, X. Wang and X. Qu. 'Optimization of electric bus scheduling considering stochastic volatilities in trip travel time and energy consumption'. In: *Computer-Aided Civil and Infrastructure Engineering* 36.12 (2021), pp. 1530–1548. DOI: 10.1111/mice.12684.

[3] Z. Chao and C. Xiaohong. 'Optimizing Battery Electric Bus Transit Vehicle Scheduling with Battery Exchanging: Model and Case Study'. In: *Procedia - Social and Behavioral Sciences* 96 (2013), pp. 2725–2736. ISSN: 1877-0428. DOI: 10.1016/j.sbspro.2013.08.306.

[4] C. Chen, J. Lin, E. Yücesan and S. E. Chick. 'Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization'. In: *Discret. Event Dyn. Syst.* 10.3 (2000), pp. 251–270. DOI: 10.1023/A:1008349927281.

[5] Z. Chen, Y. Yin and Z. Song. 'A cost-competitiveness analysis of charging infrastructure for electric bus operations'. In: *Transportation Research Part C: Emerging Technologies* 93 (2018), pp. 351–366. ISSN: 0968-090X. DOI: 10.1016/j.trc.2018.06.006.

[6] C. E. Clark. 'The Greatest of a Finite Set of Random Variables'. In: *Operations Research* 9.2 (1961), pp. 145–162. DOI: 10.1287/opre.9.2.145.

[7] G. Ghiani, P. Legato, R. Musmanno and F. Vocaturo. 'A combined procedure for discrete simulation-optimization problems based on the simulated annealing framework'. In: *Computational Optimization and Applications* 38.1 (2007), pp. 133–145. DOI: 10.1007/s10589-007-9010-7.

[8] Y.-C. Ho. 'An explanation of ordinal optimization: Soft computing for hard problems'. In: *Information Sciences* 113.3 (1999), pp. 169–192. ISSN: 0020-0255. DOI: 10.1016/s0020-0255(98)10056-7.

[9] S.-H. Kim and B. L. Nelson. 'Chapter 17 Selecting the Best System'. In: *Simulation*. Ed. by S. G. Henderson and B. L. Nelson. Vol. 13. Handbooks in Operations Research and Management Science. North-Holland, 2006, pp. 501–534. DOI: 10.1016/s0927-0507(06)13017-0.

[10] KNMI. *Hourly weather station readings*. Dutch. URL: https://daggegevens.knmi.nl/klimatologie/uurgegevens (visited on 24/05/2022).

[11] A. M. Law. *Simulation Modeling and Analysis*. 5th ed. McGraw-Hill, 2015. ISBN: 978-0-07-340132-4.

[12] L. H. Lee, C.-H. Chen, E. P. Chew, J. Li, N. A. Pujowidianto and S. Zhang. 'A Review of Optimal Computing Budget Allocation Algorithms for Simulation Optimization Problem'. In: *International Journal of Operations Research* 7.2 (2010), pp. 19–31.

[13] J.-Q. Li. 'Battery-electric transit bus developments and operations: A review'. In: *International Journal of Sustainable Transportation* 10.3 (2016), pp. 157–169. DOI: 10.1080/15568318.2013.872737.

[14] N. Olsen and N. Kliewer. 'Scheduling Electric Buses in Public Transport: Modeling of the Charging Process and Analysis of Assumptions'. In: *Logist. Res.* 13.1 (2020), p. 4. DOI: 10.23773/2020_4.

[15] S. N. Parragh and V. Schmid. 'Hybrid column generation and large neighborhood search for the dial-a-ride problem'. In: *Computers & Operations Research* 40.1 (2013), pp. 490–497. DOI: 10.1016/j.cor.2012.08.004.

[16] G. J. P. N. Passage, J. M. van den Akker and J. A. Hoogeveen. 'Local search for stochastic parallel machine scheduling: improving performance by estimating the makespan'. In: *European Conference on Stochastic Optimization*. 2017.

[17] J. Patnaik, S. Chien and A. Bladikas. 'Estimation of Bus Arrival Times Using APC Data'. In: *Journal of Public Transportation* 7.1 (2004), pp. 1–20. DOI: 10.5038/2375-0901.7.1.1.

[18] S. S. G. Perumal, R. M. Lusby and J. Larsen. 'Electric bus planning & scheduling: A review of related problems and methodologies'. In: *European Journal of Operational Research* 301.2 (2022), pp. 395–413. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2021.10.058.

[19] S. Pirkwieser and G. R. Raidl. 'Multiple Variable Neighborhood Search Enriched with ILP Techniques for the Periodic Vehicle Routing Problem with Time Windows'. In: *Hybrid Metaheuristics, 6th International Workshop, HM 2009, Udine, Italy, October 16-17, 2009. Proceedings*. Ed. by M. J. Blesa, C. Blum, L. D. Gaspero, A. Roli, M. Sampels and A. Schaerf. Vol. 5818. Lecture Notes in Computer Science. Springer, 2009, pp. 45–59. DOI: 10.1007/978-3-642-04918-7_4.

[20] S. J. Raff. 'Routing and scheduling of vehicles and crews : The state of the art'. In: *Computers & Operations Research* 10.2 (1983), pp. 63–67. DOI: 10.1016/0305-0548(83)90030-8.

[21] Y. Rinott. 'On two-stage selection procedures and related probability-inequalities'. In: *Communications in Statistics - Theory and Methods* 7.8 (1978), pp. 799–811. DOI: 10.1080/03610927808827671. eprint: https://doi.org/10.1080/03610927808827671. URL: https://doi.org/10.1080/03610927808827671.

[22] A. Subramanian, E. Uchoa and L. S. Ochi. 'A hybrid algorithm for a class of vehicle routing problems'. In: *Computers & Operations Research* 40.10 (2013), pp. 2519–2531. DOI: 10.1016/j.cor.2013.01.013.

[23] X. Tang, X. Lin and F. He. 'Robust scheduling strategies of electric buses under stochastic traffic conditions'. In: *Transportation Research Part C: Emerging Technologies* 105 (2019), pp. 163–182. ISSN: 0968-090X. DOI: 10.1016/j.trc.2019.05.032.

[24] W. ten Bosch, J. A. Hoogeveen and M. E. van Kooten Niekerk. 'Scheduling electric vehicles by Simulated Annealing with recombination through ILP'. 2021.

[25] M. van den Akker, K. van Blokland and H. Hoogeveen. 'Finding Robust Solutions for the Stochastic Job Shop Scheduling Problem by Including Simulation in Local Search'. In: *Experimental Algorithms, 12th International Symposium, SEA 2013, Rome, Italy, June 5-7, 2013. Proceedings*. Ed. by V. Bonifaci, C. Demetrescu and A. Marchetti-Spaccamela. Vol. 7933. Lecture Notes in Computer Science. Springer, 2013, pp. 402–413. DOI: 10.1007/978-3-642-38527-8_35.

[26] M. E. van Kooten Niekerk, J. M. van den Akker and J. A. Hoogeveen. 'Scheduling electric vehicles'. In: *Public Transp.* 9.1-2 (2017), pp. 155–176. DOI: 10.1007/s12469-017-0164-0.

[27] H. Wang and J. Shen. 'Heuristic approaches for solving transit vehicle scheduling problem with route and fueling time constraints'. In: *Applied Mathematics and Computation* 190.2 (2007), pp. 1237–1249. DOI: 10.1016/j.amc.2007.02.141.

[28] M. Wen, E. Linde, S. Ropke, P. Mirchandani and A. Larsen. 'An adaptive large neighborhood search heuristic for the Electric Vehicle Scheduling Problem'. In: *Computers & Operations Research* 76 (2016), pp. 73–83. ISSN: 0305-0548. DOI: 10.1016/j.cor.2016.06.013.

[29] R. R. Wilcox. 'A Table for Rinott's Selection Procedure'. In: *Journal of Quality Technology* 16.2 (1984), pp. 97–100. ISSN: 0022-4065. DOI: 10.1080/00224065.1984.11978896.

[30] H.-a. Yang, Y. Lv, C. Xia, S. Sun and H. Wang. 'Optimal Computing Budget Allocation for Ordinal Optimization in Solving Stochastic Job Shop Scheduling Problems'. In: *Mathematical Problems in Engineering* 2014 (Mar. 2014), pp. 1–10. ISSN: 1024-123X. DOI: 10.1155/2014/619254.

[31] M. Yoon and J. Bekker. 'Considering sample means in Rinott's procedure with a Bayesian approach'. In: *European Journal of Operational Research* 273.1 (2019), pp. 249–258. DOI: 10.1016/j.ejor.2018.06.040.