Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Ella Frederika Johnsen

The Mass Relocation Problem with Uncertainty

A Robust Approach to the Optimization of a Real-World Road Construction Site

Master's thesis in Applied Physics and Mathematics
Supervisor: Elisabeth Anna Sophia Köbis
Co-supervisor: Torkel Andreas Haufmann
June 2021

NTNU
Norwegian University of Science and Technology

Ella Frederika Johnsen

The Mass Relocation Problem with Uncertainty

A Robust Approach to the Optimization of a Real-World Road Construction Site

Master's thesis in Applied Physics and Mathematics
Supervisor: Elisabeth Anna Sophia Köbis
Co-supervisor: Torkel Andreas Haufmann
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

Abstract

By streamlining operations, a road construction company can potentially save substantial amounts of time and resources. It is therefore interesting to investigate alternative approaches to routing on-site vehicles. This master's thesis aims to formulate and solve the mass relocation aspect of a road construction site by expressing it as a robust optimization problem. The work was conducted in collaboration with SINTEF Digital on assignment from Skanska.

This thesis first explores the field of Vehicle Routing Problems (VRPs) and examines how known variations can be adapted to fit this problem. We find that the Vehicle Routing Problem with Pickup and Delivery (VRPPDTW) closely resembles the given description of our problem, and we modify it into what we call the Mass Relocation Problem (MRP). To include uncertainty into the formulation, we investigate the known robustness techniques commonly used for VRPs; the Soyster approach and the Bertsimas-Sim approach. We also develop a new robustness approach that we call the Removing Outliers Approach (ROA). Using these approaches, we formulate the Mass Relocation Problem with Uncertainty (MRPU). To assess the performance of our MRP and MRPU formulations, we implement them in the optimization software Gurobi. We consider both a simple test case and a real-world case with data provided by Skanska. Our results show that the formulation provides feasible solutions but is currently too computationally expensive for modeling a full work day. Lastly, we present possible improvements to the problem formulation and its implementation to remedy these issues.

Sammendrag

Ved å effektivisere driften kan et veibyggingsselskap potensielt spare mye tid og ressurser. Det er derfor interessant å undersøke alternativer for hvordan de burde rute kjøretøyene sine. Denne masteroppgaven tar sikte på å formulere og løse masseflyttingsaspektet til et veibyggingsanlegg ved å uttrykke det som et robust optimeringsproblem. Arbeidet ble utført i samarbeid med SINTEF Digital på oppdrag fra Skanska.

Denne oppgaven undersøker først kjente kjøreruteproblemer (VRPs) og ser på hvordan de kan brukes til å formulere dette problemet. Vi finner at kjøreruteproblemet med henting og levering og tidsvinduer (VRPPDTW) ligner på den gitte beskrivelsen av anleggsplassen så vi omformulerer dette problemet til det såkalte masseflyttingsproblemet (MRP). Ettersom vi også ønsker å inkludere usikkerhet i formuleringen undersøker vi deretter de to mye brukte robusthetsteknikkene: Soyster-tilnærmingen og Bertsimas-Sim-tilnærmingen. Vi utvikler også vår egen robusthetstilnærming som vi kaller metoden for ekstremverdifjerning (ROA). Ved hjelp av disse formulerer vi masseflyttingsproblemet med usikkerhet (MRPU). For å vurdere ytelsen til våre MRP- og MRPU-formuleringer implementerer vi dem i optimaliseringsprogramvaren Gurobi. Vi undersøker de tilhørende løsningene for et enkelt testforsøk og for ekte data fra ett av Skanskas veibyggingsanlegg. Resultatene våre viser at formuleringen gir gode løsninger, men at beregningstiden er for høy til at metoden kan brukes til å modellere en hel arbeidsdag. Til slutt, presenterer vi mulige forbedringer av problemformuleringen og implementasjonen.

Preface

This thesis is the final work of my master's degree in Industrial Mathematics at the Department of Mathematical Sciences of the Norwegian University of Science and Technology (NTNU). The thesis is written in collaboration with SINTEF for a project by the construction company Skanska, and is a culmination of my work during the spring of 2021.

I would first like to thank my supervisor at SINTEF, Torkel Andreas Haufmann, for his invaluable guidance and for trusting me with this project. I would also like to thank my supervisor at NTNU, Elisabeth Anna Sophia Köbis for her insightful comments and her encouragement. Lastly, without the help and support of my friends and family, I would have never have gotten this far. So thank you - I could not have done this alone.

Trondheim, June 22nd, 2021,

Ella Frederika Johnsen.

Contents

Figures

Tables

Abbreviations

OR Operations Research

TSP Traveling Salesman Problem

VRP Vehicle Routing Problem

CVRP Capacitated Vehicle Routing Problem

VRPTW Vehicle Routing Problem with Time Windows

VRPPD Vehicle Routing Problem with Pickup and Delivery

VRPPDTW Vehicle Routing Problem with Pickup and Delivery and Time Windows

VRPU Vehicle Routing Problem with Uncertainty

VRPUC Vehicle Routing Problem with Uncertain Costs

MRP Mass Relocation Problem

MRPU Mass Relocation Problem with Uncertainty

ROA Removing Outliers Approach

Chapter 1

Introduction

Operations research (OR) techniques have been rising in popularity over the last couple of decades for managing and facilitating the logistics behind goods and services. Using these techniques can save between 5% to 20% of the transportation costs [23]. With 14.7% of the Norwegian mainland's GDP in 2007 directly tied to the cost of logistics [12], the potential savings are substantial

The building and construction sector in Norway provides essential infrastructure and plays a significant role in national employment and value creation. In 2016, this industry employed 227 400 people and had a gross product of NOK 182.2 billion, making it Norway's second-largest mainland industry when excluding the public sector [19]. However, between 2001 and 2016, the total factor productivity of the industry fell by 1.3% annually, while for mainland Norway in general it increased by 1.1% annually [19]. This indicates that the construction industry is falling behind in terms of productivity and could benefit from streamlining productions.

We should also consider the environmental benefits of employing OR techniques. Approximately 24% of carbon dioxide emissions worldwide are directly correlated to distribution processes [18]. In Norway, just the construction site machinery account for 20% of the total emissions released by construction sites [20]. As machines at construction sites are idle for up to 40% of working hours [20], a lot can be saved by employing OR techniques.

This thesis will focus on the Vehicle Routing Problem (VRP). It was introduced by Dantzig and Ramser in 1959 as an extension of the Traveling Salesman Problem (TSP) [8]. It depicts a fleet of homogeneous vehicles that serves a set of customers and tries to find the least total distance travelled by all vehicles while serving all customers. In 1964, Clarke and Wright generalized the problem into what we today recognize as the VRP [7]. The problem has been widely studied and one of the main pillars of Operations Research.

The size and composition of the problem can be adapted to fit the requirements of many different problems. The general formulation describes a fleet of vehicles located at one or more depots that satisfies a set of requests using a given road network such that the transportation costs are minimized, and operational constraints are satisfied. Such constraints can for example describe how the road network can be traversed or the carrying capacity of the vehicles. Thus, the VRP can be modified to fit several problems, such as mail delivery, transportation of passengers, or, as in this thesis, pickup and delivery of mass.

Determining the optimal solution of a VRP is $\mathcal{NP}$-hard. In other words, no algorithm can guarantee optimality for an arbitrary instance in polynomial time of instance size. However, there exists extensive literature on possible solution methods, as the VRP has been widely studied. Exact methods, heuristics, and metaheuristics are some of the most commonly used techniques, where metaheuristics in particular have been widely studied the last couple of decades. Thus, it is easier than ever before to solve or approximate large $\mathcal{NP}$-hard problems.

Today, few, if any, construction sites in Norway use routing tools to schedule the mass relocation on their site [20]. A goal of the research project called "Datadrevet anleggsplass", by the construction company Skanska and the research organization SINTEF, is to make a routing tool of their own. This master's thesis aims to solve part of this problem by formulating the mass relocation aspect of a road construction site as a variation of the VRP. We have called this problem the Mass Relocation Problem (MRP). The idea is that the mass relocation aspect of a road construction site is similar to a VRP as it concerns the pickup and delivery of mass using a fleet of vehicles often subject to time windows. Furthermore, as there is a lot of uncertainty in real world situations, the

routing tool must be able to produce feasible solutions in many different cases. This is why we also formulate the Mass Relocation Problem with Uncertainty (MRPU) as a robust variation of the improved MRP.

This thesis is split into seven stages:

Chapter 1 provides insight into the context, motivation and objectives of this thesis.

Chapter 2 presents the known theory on the VRP, and some of its variations and two of the commonly used robustness approaches for VRPs.

Chapter 3 investigates the potential for improvement of the previous formulation and constructs a new mathematical description of the MRP.

Chapter 4 describes how we can apply robustness approaches to the MRP to formulate the MRPU.

Chapter 5 inspects the performance of both the MRP and the MRPU by implementing and solving them for a simple test case.

Chapter 6 demonstrates how the MRPU can be applied to a real-world problem and analyzes the results.

Chapter 7 suggests future work and concludes the thesis.

Chapter 2

Preliminaries

This chapter introduces known theory on the Vehicle Routing Problem (VRP) and some of its variations, as well as the theory behind some of the robustness approaches often applied for VRPs. Section 2.1 introduces the VRP and the mathematical formulation of its variation; the Vehicle Routing Problem with Pickup and Delivery and Time Windows (VRPPDTW). Section 2.2 introduces two common robustness techniques for VRPs; the Soyster approach and the Bertsimas-Sim approach.

2.1 The Vehicle Routing Problem

The objective of a VRP is to find the optimal route for a set of vehicles, servicing a set of vertices, using a given road network. One must also adhere to a given set of constraints, and the vehicles must start and end at a centralized depot. Usually, each vehicle only serves a subset of the customers. The optimal solution is the route with the lowest total costs that upholds all constraints and that contains no subtours. A subtour is when a vehicle traverses a route that returns it back to where it started, as illustrated in Figure 2.1. The costs can be calculated from the total distance travelled, the fixed cost of each utilized road, the total time spent, among others. A simple case of the VRP is depicted in Figure 2.2. The top image depicts a road network and transportation

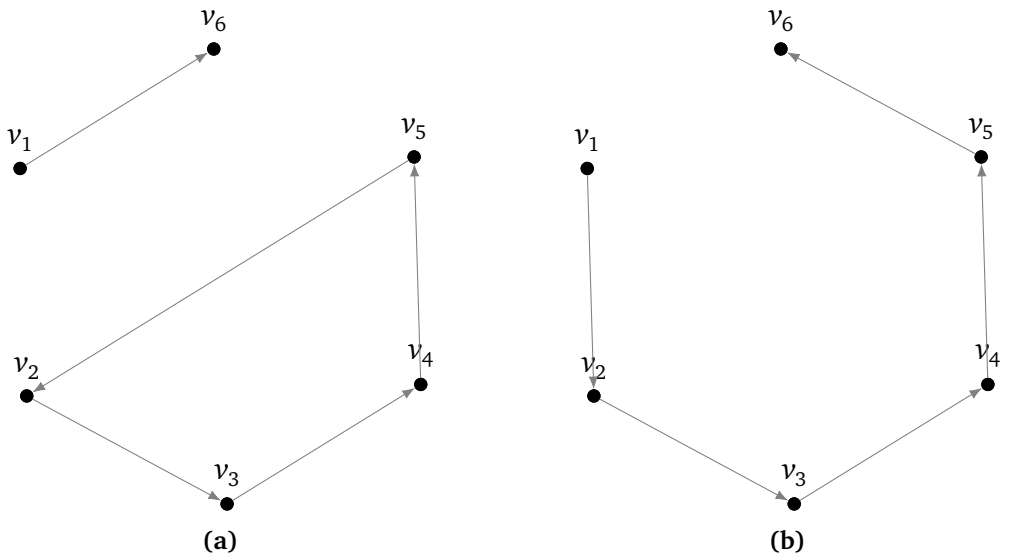
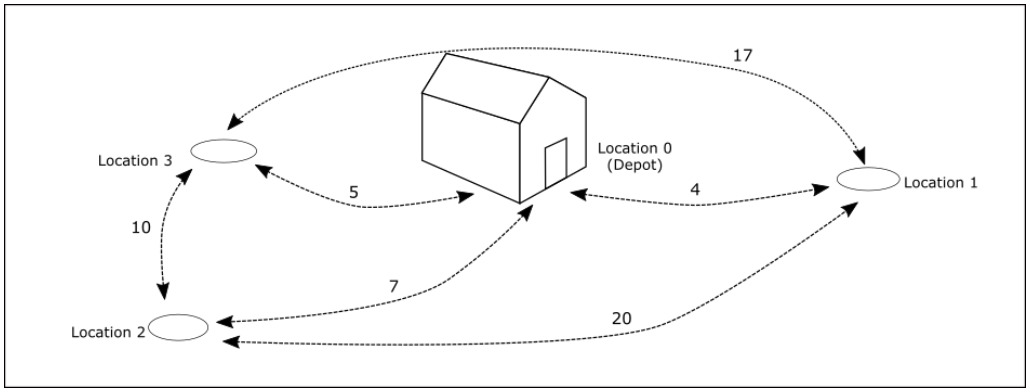


Figure 2.1: An illustration of subtours, where the vertices v_i depict geographic locations and the edges between them depict the calculated route using the VRP. **(a)** depicts a route containing a subtour in v_2, v_3, v_4, and v_5, and **(b)** contains none.

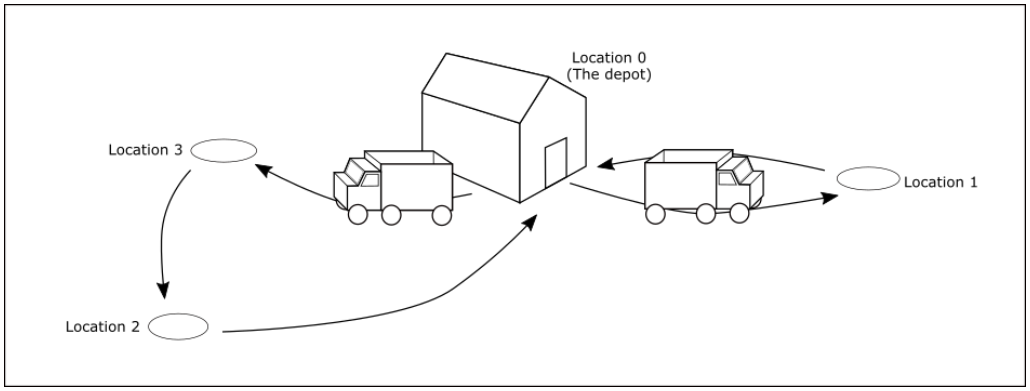
costs, and the two following figures are examples of solutions to the VRP applied to this instance.

The VRP lays the groundwork for many types of problems concerning the transport of goods. The VRP has two main constraints; 1) every vehicle must start and end its route at the depot, and 2) all customers must be served. Beyond that, one must look to its variations. The most common are the Capacitated Vehicle Routing Problem (CVRP), the Vehicle Routing Problem with Time Windows (VRPTW), and the Vehicle Routing Problem with Pickup and Delivery (VRPPD). The CVRP describes a VRP where every vehicle has a specified carrying capacity, i.e. a limit on how many goods it can carry at once. As this is almost always the case in real life, the CVRP has become almost synonymous with the VRP. Furthermore, the VRPTW is, as its name suggests, a VRP with the addition of time windows specifying when vehicles are allowed to visit individual site. This is a common occurrence when one deals with, for example, the transportation of people or other time-sensitive objects. Finally, the VRPPD references a VRP where the vehicles both pickup and deliver items along their route.

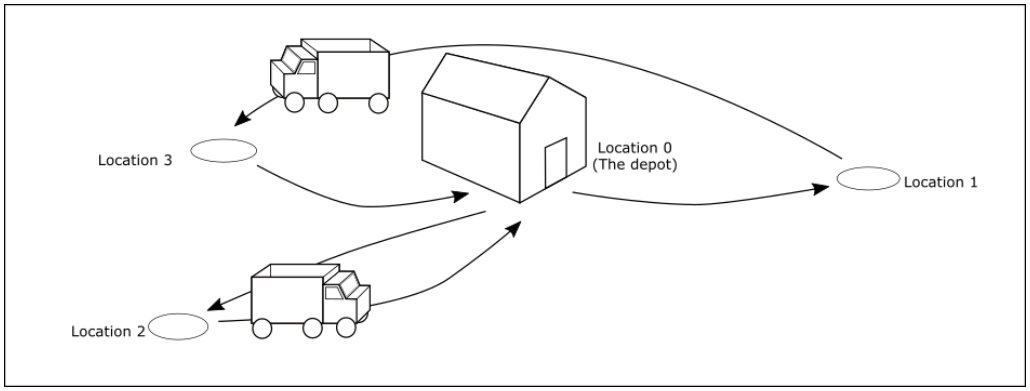
The presented VRP variations can be further combined into the Capacitated Vehicle



(a)



(b)



(c)

Figure 2.2: Illustration of a basic case of the VRP. (a) is an example instance, (b) is a solution with a total cost of 30, and (c) is a suboptimal solution with a total cost of 40.

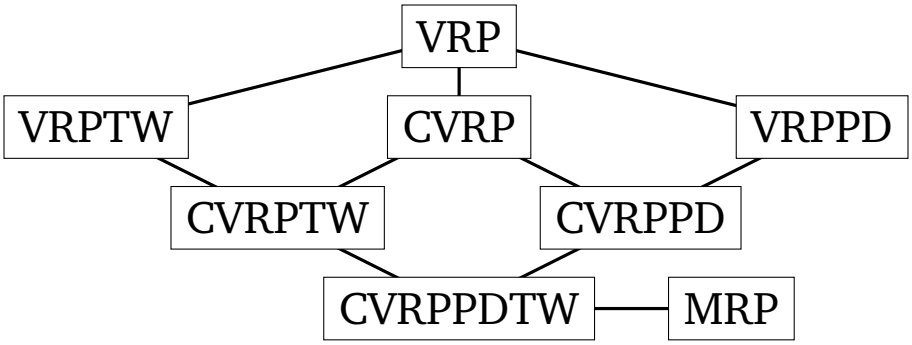


Figure 2.3: Variations of the Vehicle Routing Problem

Routing Problems with Pickup and Delivery and Time Windows (CVRPPDTW) as depicted in Figure 2.3. The CVRPPDTW is as close we get in literature to the Mass Relocation Problem (MRP), which we define in the next chapter and is the main topic of this thesis. Therefore, we will look closer at the CVRPPDTW's mathematical formulation as it lays the foundation for the later formulation of the MRP. For simplicity, we will from here on out omit the "capacitated"-term and assume that all VRPs, unless otherwise specified, are capacitated.

2.1.1 Mathematical Formulation

The VRPPDTW has been previously expressed by Toth and Vigo [23], and Parragh, Doerner, and Hartl [17], among others, and this formulation is largely based on their work. We let $G = (V, E)$ define the directed graph where $V = \{0, 1, \ldots, n, n+1, \ldots, 2n+1\}$ are the vertices of n pickup locations and n delivery locations. Furthermore, vertex 0 and $2n + 1$ denote the start- and end-depot, respectively. E denotes the edges that connect the vertices, i.e. the road network. We also define that a request i represents the mass needed at vertex i. Now, we define the necessary parameters as

K set of vehicles,

c_{ij}^k cost of traversing the edge $(i, j) \in E$ for vehicle $k \in K$, here given as the distance of (i, j),

d_{ij}^k travel time between vertex $i \in V$ and $j \in V$ for vehicle $k \in K$,

C^k capacity of vehicle $k \in K$,

u_i number of units needed for request i,

l_i load to be delivered from vertex i to $n+i$, where $l_i = u_i$ and $l_{n+i} = -u_i$,

s_i service duration of vertex i,

$[a_i, b_i]$ time window in which vertex i must be served,

and the pickup and delivery subsets as

P set of pickup vertices, $P = \{1, \ldots, n\}$,

D set of delivery vertices, $D = \{n+1, \ldots, 2n\}$.

Lastly, we define the decision variables as

x_{ij}^k binary flow variable that equals 1 if edge $(i, j) \in E$ is traversed in the optimal solution by vehicle $k \in K$, and 0 otherwise,

t_i^k beginning of service time at vertex i for vehicle $k \in K$,

L_i^k load of vehicle $k \in K$ after serving vertex i.

We can then formulate the objective function of the VRPPDTW as

$$\texttt{TravelCost(c)} = \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}^k \tag{2.1}$$

which calculates the cost of all traversed edges in the solution and is the standard formulation for VRPs. To find the optimal solution, we wish to minimize this function subject to a set of constraints to find the routes that use the least possible resources. The first constraint we define is

$$\sum_{k\in K}\sum_{j\in V} x_{ij}^k = 1 \;\forall\, i \in V\backslash\{0, 2n+1\}. \tag{2.2}$$

It ensures that all vertices are visited exactly once by establishing that there should only be one edge leading out of the given vertex in the optimal solution with the exception of the depots. This is beneficial as it prohibits multiple vehicles from serving the same request, assuming that every vertex only makes one request. Next, we have the constraints

$$\sum_{j\in P} x_{0j}^k = 1 \;\forall\, k \in K \tag{2.3}$$

and

$$\sum_{i\in V} x_{i,2n+1}^k = 1 \;\forall\, k \in K \tag{2.4}$$

which guarantees that vehicles only can travel to and from the depots once. Note that vehicles can still remain unused since there exists a fictional edge from start-depot to end-depot with zero cost. We also want to ensure the flow of the graph is correct, so we introduce the constraint

$$\sum_{i\in V} x_{ij}^k - \sum_{i\in V} x_{ji}^k = 0 \;\forall\, j \in V\backslash\{0, 2n+1\},\; k \in K. \tag{2.5}$$

We state that in the solution, the number of traversed edges entering a vertex j must equal the number of traversed edges leaving it for all vertices except for the depots. In essence, a vehicle cannot end its route at a vertex that is not a depot. Thus, we have defined how the vehicles can traverse the graph, and we now need to define the time limitations. The first constraint on the decision variable t can be defined as

$$x_{ij}^k(t_i^k + s_i + d_{ij}^k - t_j^k) \leq 0 \;\forall\, i, j \in V, k \in K. \tag{2.6}$$

It says that if edge (i, j) is traversed in the optimal solution, then the arrival time at vertex j, t_j, must be larger or equal to the sum of the arrival time at vertex i, t_i, the service time required at the vertex, s_i, and the travel time between i and j, d_{ij}^k. This constraint also disallows subtours since it prohibits a vehicle from visiting the same vertex more than once by tracking arrival times. As an illustration, if a vehicle travels from vertex i to vertex j, it cannot return as the constraint gives the impossibility $t_i^k \leq t_i^k - s_i - d_{ij}^k - s_j - d_{ji}^k$ with $s_i, s_j, d_{ij}^k, d_{ji}^k > 0$. We note, however, that this is a quadratic constraint.

Furthermore, we wish to require that pickup sites are visited before the paired delivery site. To ensure this, we introduce the constraint

$$t_i^k + s_i + d_{i,n+i}^k \leq t_{n+i}^k \; \forall \, i \in P, k \in K, \tag{2.7}$$

that guarantees the arrival time in the paired delivery vertex, $n + i$, is larger or equal to the arrival time at vertex i and the travel time between them. Next, we define the time windows in which a vertex can be served with the constraint

$$a_i \leq t_i^k \leq b_i \; \forall \, i \in V, k \in K, \tag{2.8}$$

We set that the arrival time at a vertex i must be in the given time window, $[a_i, b_i]$. Lastly, we preserve the carrying capacity by introducing the quadratic constrain

$$x_{ij}^k (L_i^k + l_j - L_j^k) = 0 \; \forall \, i, j \in V, k \in K. \tag{2.9}$$

It guarantees that the whole request is picked up or delivered at its paired vertex. To ensure that we never exceed the vehicle's capacity, we require

$$l_i \leq L_i^k \leq C^k \; \forall \, i \in P, k \in K \tag{2.10}$$

and

$$0 \leq L_{n+i}^k \leq C^k + l_{n+i} \; \forall \; n+i \in D, k \in K, \tag{2.11}$$

for pickup and delivery vertices, respectively. Lastly, we want to establish that vehicles are empty when leaving the depot, which we ensure by introducing the constraint

$$L_0^k = 0 \; \forall \; k \in K. \tag{2.12}$$

Now finally, we only need to define the decision variables. We start by saying that

$$x_{ij}^k \in \{0,1\} \; \forall \; i,j \in V, k \in K, \tag{2.13}$$

which is the binary requirement of x. Furthermore, we set the limits on the continuous variables t and L as

$$0 \leq t_i^k \; \forall \; i,j \in V, k \in K, \tag{2.14}$$

and

$$0 \leq L_i^k \leq C^k \; \forall \; i,j \in V, k \in K. \tag{2.15}$$

We have now defined all the constraints for the VRPPDTW, and thus, we can define the complete problem as

$$\texttt{VRPPDTW}(c, d, C, u, l, s, a, b) = \begin{cases} \min_{x,t,L} & \texttt{TravelCost}(c) \text{ as defined in (2.1)} \\ \text{subject to} & (2.2), (2.3), (2.4), (2.5), (2.6), (2.8), (2.7), (2.9), \\ & (2.10), (2.11), (2.12), (2.13), (2.14) \text{ and } (2.15). \end{cases} \tag{2.16}$$

2.2 Robust Optimization

In traditional optimization problems, the data sets are precisely known, and uncertainty is not considered. However, real-world problems are rarely deterministic as many factors are nearly impossible to predict accurately. The weather, the traffic, or even measuring errors can cause the real coefficients to differ from those used in the mathematical model. The difference, i.e. the perturbation on coefficients, may influence the solutions in an undesirable manner. It is possible that the solution is optimal for the chosen coefficients but infeasible for real life. Therefore, including uncertainty parameters should be considered for real-world optimization problems.

There are two standard practices on how to include uncertainty; stochastic modeling and robust optimization. Stochastic modeling solves the problems by using information about the likelihood of certain events [6]. Conversely, robust optimization is used when the uncertainties have unknown probability distributions [3, 13, 22]. Instead of defining the likelihood of events, the method assumes a degree of conservativeness on the lower and upper bounds of the coefficients to achieve feasible solutions.

The degree of conservativeness is an essential factor in robust optimization and must be chosen with care. Let us say we have uncertainty in a minimization problem, and we have a solution space that contains risky solutions with low objective values. If we chose a low degree of conservativeness, these risky solutions would be optimal. However, there is no guarantee that these solutions are feasible in real life. With a higher degree of conservativeness, the solutions would have a higher objective value but would more likely provide feasible solutions. Thus, the degree of conservativeness

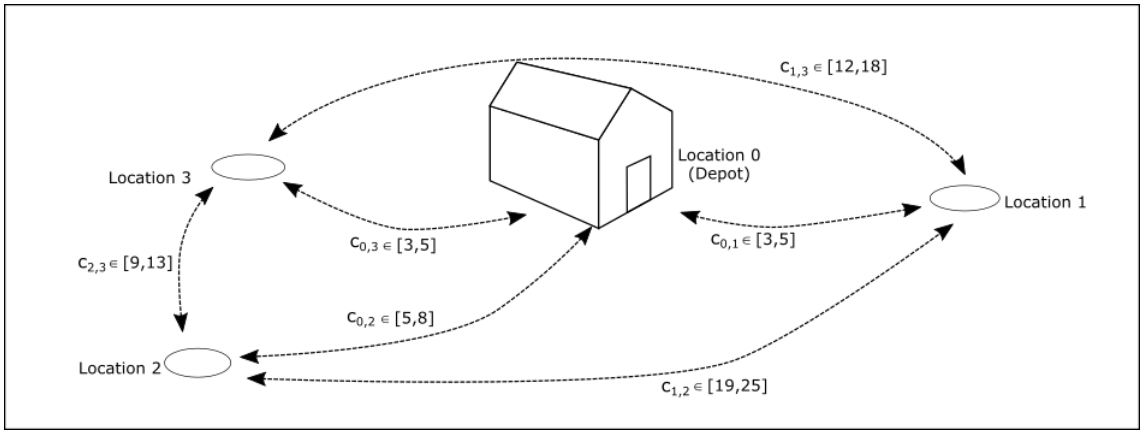


Figure 2.4: Illustration of VRP instance where the uncertainty of travel times is expressed in the interval form.

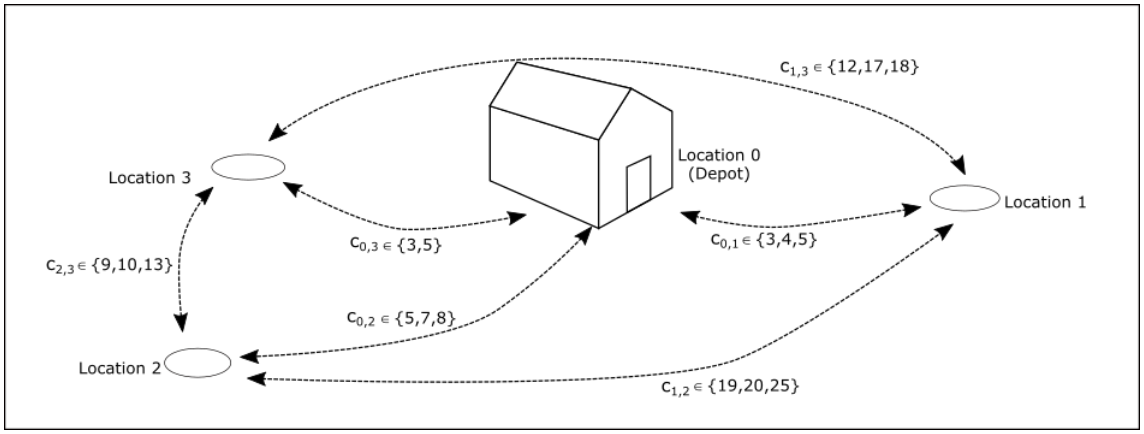


Figure 2.5: Illustration of VRP instance where the uncertainty of travel times is expressed in the discrete form.

should be directly linked to how much risk one is willing to take for better solutions.

In robust optimization, there are mainly two ways of representing uncertainty; as a continuous interval or as a discrete set of values, as illustrated in Figure 2.4 and Figure 2.5, respectively. A continuous interval would arise if we were to look at travel times from A to B and only knew the minimum and maximum possible time travel. On the other hand, a discrete set of values would occur if there existed uncertainty in how many vehicles were available. In both cases, all values within the defined domains are assumed equally probable as we do not have any data stating otherwise.

2.2.1 The Soyster Approach

The Soyster Approach was developed by A. L. Soyster in 1973 [21]. The approach ensures robustness by assuming the worst case scenario of an optimization problem. To better understand the approach, we consider a simple 0-1 linear program, where we introduce the decision variable

x_j binary decision variable that equals 1 if j is chosen in the optimal solution, and 0 otherwise,

the parameters

c_j cost of using x_j,

a_{ij} the j-th left-hand side coefficient of the i-th constraint,

b_i the right-hand side constant of the i-th constraint,

and finally the sets

I set of indices for the constraints,

J set of indices for the coefficients.

We also assume that $a_{ij}, b_i, c_j \geq 0$ for all $i \in I,\ j \in J$. Then we can define the optimization problem as

`LinearProblem(x)` $=$

$$\begin{cases} \text{minimize} \sum_{j \in J} c_j x_j & & \text{(2.17a)} \\ \text{subject to} \sum_{j \in J} a_{ij} x_j \leq b_i & \forall\, i \in I, & \text{(2.17b)} \\ x_j \in \{0,1\} & \forall\, j \in J. & \text{(2.17c)} \end{cases}$$

Furthermore, we consider the uncertainty in interval form where $c_j \in [\underline{c}_j, \bar{c}_j]$ and $a_{ij} \in [\underline{a}_{ij}, \bar{a}_{ij}]$. Thus, the Soyster approach of `LinearProblem` as given in (2.17) becomes

`Soyster(x)` =

$$\begin{cases} \text{minimize} \sum_{j \in J} \bar{c}_j x_j & \\ \text{subject to} \sum_{j \in J} \bar{a}_{ij} x_j \leq b_i & \forall\, i \in I, \\ x_j \in \{0,1\} & \forall\, j \in J. \end{cases}$$

In other words, we minimize the objective function subject to the constraints and based on the maximum possible value of each coefficient. The approach looks at the same when considering a discrete uncertainty set, because then $c_j = \{\underline{c}_j, ..., \bar{c}_j\}$ and $a_{ij} = \{\underline{a}_{ij}, ..., \bar{a}_{ij}\}$. Thus, the Soyster approach ensures feasibility for every solution of `LinearProblem`.

Note that the maximum value of each coefficient does not always correlate with the worst case scenario. For example, five vehicles would generally produce better solutions to a VRP than just one vehicle. Thus, for the Soyster approach to guarantee feasible solutions one would have to include the worst-case coefficients into the problem.

2.2.2 The Bertsimas-Sim Approach

The Bertsimas Sim-Approach is a robust optimization method where one can modify the degree of conservatism [4, 5]. The main idea of this approach is that the uncertainty is bounded by the uncertainty budgets $\Gamma_0, \Gamma_i \geq 0$, which determine the degree of conservativeness for the objective function and the constraint i, respectively. The uncertainty budget represents the number of perturbations allowed, and by determining its value, we are determining the degree of conservativeness. More specifically,

we decrease the associated Γ by one for each coefficient we fully perturb. If $\Gamma = 0.5$, we would perturb the coefficient with half of the maximum perturbation and decrease the uncertainty budget correspondingly.

In a minimization problem, we would allocate perturbations starting with the coefficients that have the worst possible maximum values and continue until the uncertainty budget is empty. The remaining coefficients would assume their best case value. Thus, by setting $\Gamma_0 = 0$ or $\Gamma_i = 0$, we would be non-conservative and assume best case values for all coefficients of the objective function or the given constraint. By setting the associated Γ equal to the number of coefficients, one would get a fully conservative and guaranteed feasible solution.

When applying the Bertsimas-Sim approach to the minimization problem given in (2.17), we get the optimization problem

`Bertsimas_Sim(x)` $=$

$$
\begin{cases}
\text{minimize} \\
\quad \displaystyle\sum_{j \in J} \underline{c}_j x_j + \max \left\{ \underline{c}_j x_j + \hat{\eta}_j^c (\bar{c}_j - \underline{c}_j) x_j \;\middle|\; \sum_{j \in J} \hat{\eta}_j^c \leq \Gamma;\ 0 \leq \hat{\eta}_j^c \leq 1 \ \forall\, j \in J \right\} \\
\text{subject to} \\
\quad \displaystyle\sum_{j \in J} \underline{a}_{ij} x_j + \max \left\{ \underline{a}_{ij} x_j + \hat{\eta}_{ij}^a (\bar{a}_{ij} - \underline{a}_{ij}) x_j \middle| \sum_{j \in J} \hat{\eta}_{ij}^a \leq \Gamma_i;\ 0 \leq \hat{\eta}_{ij}^a \leq 1 \ \forall j \in J \right\} \leq b_i \\
\quad \forall\, i \in I, \\
\quad x_j \in \{0, 1\} \quad \forall\, i \in I,
\end{cases}
$$

where we have defined the parameters $\hat{\eta}_j^c, \hat{\eta}_{ij}^a \in [0, 1]$ as the estimated perturbation of c_j and a_{ij}, respectively.

In summation, the Bertsimas-Sim approach seeks to find the highest perturbation set within a strict uncertainty budget. In other words, this formulation assumes that we perturb the Γ_i number of coefficients so that we ensure that the constraints are as close as possible to their feasibility boundary b_i. The same goes for the Γ_0 number of

coefficients that provide the highest cost.

Chapter 3

The Mass Relocation Problem

This chapter will formulate the Mass Relocation Problem (MRP). More specifically, Section 3.1 introduces the idea behind the MRP, as well as how the MRP can be formulated as an extension of the Vehicle Routing Problem with Pickup and Delivery and Time Windows (VRPPDTW). Then, Section 3.2 will review how we can advance this formulation of the MRP, while Section 3.3 will formulate the refined version, and lastly, Section 3.4 will investigate a reduced version of the MRP.

3.1 Background and Early Formulation

In general, a road construction site consists of a long stretch of terrain. The preliminary objective is to flatten this land to make the road as level as possible. Therefore, the construction company places their excavators by the unwanted mass and has a fleet of dumpers transporting the surplus mass to the delivery sites. This thesis is limited to this preparatory component of the construction process.

The MRP describes the problem of mass relocation at a road construction site. The objective is to maximize the amount of mass moved throughout the day while keeping costs and resources needed at a minimum. Thus, the problem can be recognized as a variation of the VRPPDTW. It shares the characteristics of time windows and limited

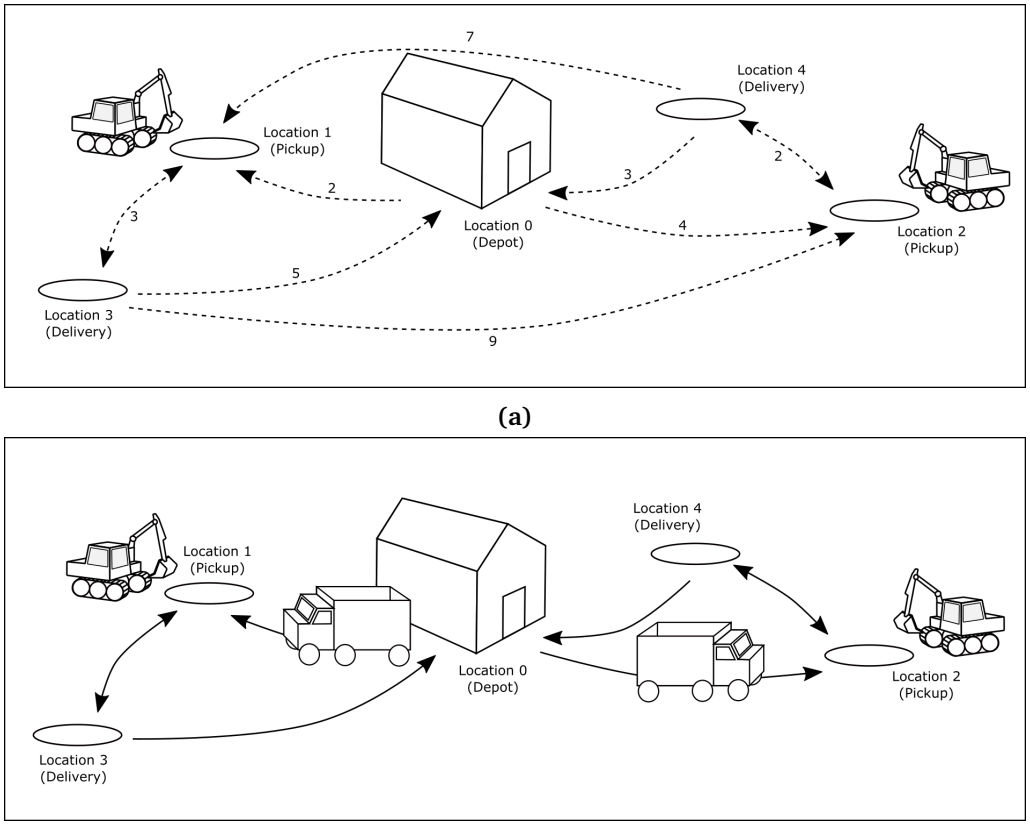


Figure 3.1: Illustration of a basic case of a MRP. (a) is an example instance, and (b) is a solution of the MRP where the dumpers can drive freely between the pickup and delivery sites as many times as necessary within a certain time limit.

capacity limits while performing both pickup and delivery. An illustration of the MRP is depicted in Figure 3.1. The figure visualizes both how a construction site can be imagined as a graph consisting of pickup and delivery sites, and a possible solution.

The MRP differs from the VRPPDTW (2.16) in three principal aspects; load capacity, linearity, and vertex visitation. In the VRPPDTW, we have constraints limiting the amount a given vehicle can transport. For the MRP, we move away from this by assuming that all vehicles can carry the same amount of mass and that they are fully loaded and unloaded at every pickup and delivery vertex, respectively. In real life, it is not fair to assume that all vehicles on-site have the same carrying capacity. But as we do not have any available data on how much a dumper can carry in this thesis, we will proceed regardless. By doing so, we can drop the load capacity constraints (2.9),

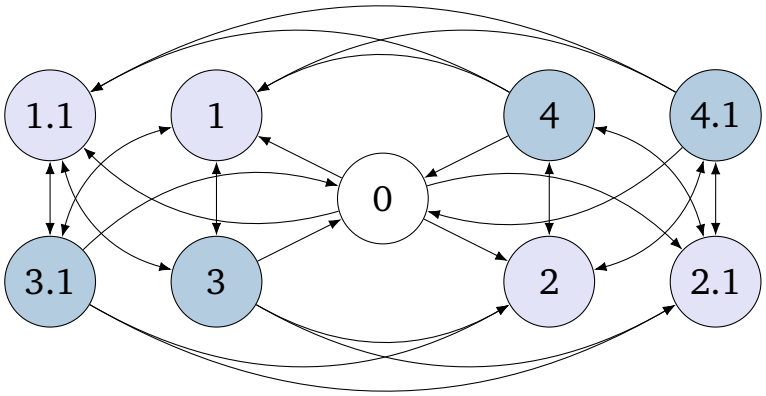


Figure 3.2: An illustration of duplicated vertices, where the decimal number of each vertex denotes the duplication index. This is the graph depicted in Figure 3.1a with two duplicates of each vertex. The purple vertices depict pickup sites, and the blue depict delivery sites.

(2.10), (2.11), (2.12), and (2.15). This makes the model easier to compute.

Furthermore, we want the MRP to be a mixed-integer linear optimization problem. This is because we then can use known exact solution methods such as the Simplex method with branch and bound. The VRPPDTW, as formulated in (2.16), is a quadratically constrained optimization problem and thus needs to be modified appropriately.

Lastly, we introduce the idea of *duplicate vertices* for each pickup and delivery site. In traditional VRPs, all vertices must be serviced once and only once. However, in the MRP, we want to visit the same locations multiple times. To account for this, we *duplicate* the pickup and delivery sites. This means that instead of revisiting the same vertices, we create duplicates representing the same location. This gives the illusion of visiting multiple vertices when in reality, we are revisiting the same one. This way, we can still use the traditional constraints defined for VRPs, such as defining flow conservation in the road network. An illustration of how the idea works is depicted in Figure 3.2. It illustrates the earlier example of an MRP given in Figure 3.1a drawn as a graph with one duplicate of each site. Notice that all the vertices in this new graph are connected the same way the original vertices were since they still represent the same geographic location.

This thesis defines two variations of the MRP that are later used to solve both test

problems and problems based on real data. To better understand how they connect to the VRPPDTW, we first introduce a simple formulation of an MRP as a direct extension of the VRPPDTW. This version, which we will present shortly, will be later dissected in Section 3.2 to investigate how it can be rewritten into a more viable formulation for a real-life setting.

3.1.1 Preliminary Formulation

To define this preparatory formulation of the MRP, we designate the needed parameters as

K set of vehicles,

c_{ij}^k cost of traversing the edge $(i,j) \in E$ for vehicle $k \in K$, here given as the distance of (i,j),

d_{ij}^k travel time between vertex $i \in V$ and $j \in V$ for vehicle $k \in K$,

s_i service duration of vertex i,

$[a_i, b_i]$ time window in which vertex i must be served,

W total amount of working hours per day,

and the pickup and delivery subsets as

P set of pickup vertices, $P = \{1, \ldots, n\}$,

D set of delivery vertices, $D = \{n+1, \ldots, 2n\}$.

where the mass picked up at vertex i must be delivered at vertex $n+i$. Lastly, we define the decision variables as

x_{ij}^k binary flow variable that equals 1 if edge $(i,j) \in E$ is traversed in the optimal solution by vehicle $k \in K$, and 0 otherwise,

t_i^k beginning of service time at vertex i for vehicle $k \in K$.

We are now ready to define the objective function as

$$\texttt{TravelCost(c)} = \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}^k, \tag{3.1}$$

which remains unchanged from the objective function of the previously formulated VRPPDTW. The first constraint of the MRP,

$$\sum_{k \in K} \sum_{j \in V} x_{ij}^k = 1 \; \forall \, i \in V \backslash \{0, 2n+1\}, \tag{3.2}$$

is also equivalent to (2.2) from the formulation of the VRPPDTW. It ensures that only one edge leaving the vertex can be traversed. This applies for all vertices, except the depots. Furthermore, we define that

$$\sum_{j \in P} x_{0j}^k = 1 \; \forall \, k \in K, \tag{3.3}$$

and

$$\sum_{j \in D} x_{0j}^k = 0 \; \forall \, k \in K, \tag{3.4}$$

to replace (2.3) in establishing that all vehicles must start their route at a pickup vertex. By removing the load capacity constraints, we also removed the implicit property that all vehicles must start and end their route at a depot. Therefore, we also need to introduce

$$\sum_{i \in P} x_{i,2n+1}^k = 0 \; \forall \, k \in K, \tag{3.5}$$

and

$$\sum_{i \in D} x^k_{i,2n+1} = 1 \; \forall \; k \in K, \tag{3.6}$$

to replace (2.4). They state that the last vertex a vehicle visits before leaving for the depot must be a delivery vertex. This also ensures that the vehicles end their route empty. Furthermore, we also need a constraint that establishes that the vehicles must move directly between the paired pickup vertices and delivery vertices. Thus, we write that

$$\sum_{k \in K} x^k_{i,n+i} = 1 \; \forall \; i \in P. \tag{3.7}$$

In real life, this constraint may not be necessary. This is because we typically only have one type of mass being excavated in the entire road construction site, and thus, delivery sites are interchangeable. We will investigate this closer in Section 3.2.1. Next, we want to guarantee flow conservation, so we introduce

$$\sum_{i \in V} x^k_{ij} - \sum_{i \in V} x^k_{ji} = 0 \; \forall \; j \in V \backslash \{0, 2n+1\}, \; k \in K. \tag{3.8}$$

As in the VRPPDTW, this makes sure that if a vehicle enters a vertex j, it must also exit vertex j. Now, we are on to defining the constraints for the decision variable t. To do so, we start with a linear version of constraint (2.6) that we define as

$$t^k_i + s_i + d^k_{ij} - t^k_j \leq (W + s_i + d^k_{ij}) \cdot (1 - x^k_{ij}) \; \forall \; i, j \in V, k \in K. \tag{3.9}$$

This states that if an edge from i to j is traversed, then the arrival time at j must be later than the arrival time at i, plus the service time at i, and the travel time to get to j. Else, the time difference cannot be larger than the allocated working hours per day. It is beneficial that this constraint is linear instead of quadratic as then one can use well-

established algorithms such as the Simplex algorithm. Furthermore, this constraint ensures that the MRP contains no subtours, just as (2.6) did for the VRPPDTW. Next, we want to define the time windows as

$$a_i \leq t_i^k \leq b_i \; \forall \; i \in V, k \in K, \tag{3.10}$$

which is the same as (2.8) for the VRPPDTW. The last two constraints of the MRP establishes the domain of the decision variables. We enforce the binary requirement of x by

$$x_{ij}^k \in \{0,1\} \; \forall \; i,j \in V, k \in K, \tag{3.11}$$

and we ensure that the continuous variable t has the correct limits by stating

$$0 \leq t_i^k \leq W \; \forall \; i \in V, k \in K. \tag{3.12}$$

As we have now defined the objective function and all the constraints for MRP, we can define it in its entirety as

$$\texttt{preliminaryMRP}(c,d,s,a,b,W) = \begin{cases} \min_{x,t} & \texttt{TravelCost}(c) \text{ as given in (3.1)} \\ \text{subject to} & (3.2),(3.3),(3.4),(3.5),(3.6),(3.7),(3.8),(3.9), \\ & (3.10),(3.11), \text{ and } (3.12). \end{cases} \tag{3.13}$$

3.2 Potential for Improvement

The mathematical formulation of the MRP as presented in Section 3.1 is not ideal. It is a complex model with many parameters and decision variables. Furthermore, several

underlying assumptions are not realistic, such as knowing exactly how many vertices one can visit during a day or when exactly to set the time windows. Therefore, we will in this section further investigate what the main drawbacks of this model are and how we can resolve them. We start by looking closer at how we have previously defined the cost matrix **c**, and what this entails for how vertices can be paired.

3.2.1 Pairing Vertices

The cost matrix **c** is an integral part of the MRP as it defines what edges a vehicle can traverse and the cost of doing so. For the improved version of the MRP, we disallow the edges between pickup vertices and the edges between delivery vertices, i.e. we force all vehicles to only travel from pickup to delivery or delivery to pickup. To do so, we formulate the cost matrix as

$$
\boldsymbol{c} = \begin{bmatrix}
0 & \overbrace{0 & \dots & 0}^{\text{Pickup vertices}} & \overbrace{M & \dots & M}^{\text{Delivery vertices}} & 0 \\
M & M & \dots & M & c_{1,n+1} & \dots & c_{1,n+m} & M \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
M & M & \dots & M & c_{n,n+1} & \dots & c_{n,n+m} & M \\
M & c_{n+1,1} & \dots & c_{n+1,n} & M & \dots & M & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
M & c_{n+m,1} & \dots & c_{n+m,n} & M & \dots & M & 0 \\
M & M & \dots & M & M & \dots & M & M
\end{bmatrix}
\begin{matrix}
\\
\left.\begin{matrix}\\ \\ \\ \end{matrix}\right\} \text{Pickup vertices} \\
\\
\left.\begin{matrix}\\ \\ \\ \end{matrix}\right\} \text{Delivery vertices} \\
\\
\end{matrix}
$$

where the pickup vertices are defined as the set $P = \{1, \dots, n\}$ and the delivery vertices as $D = \{m+1, \dots, n+m\}$. Here, n is the number of pickup vertices and m is the number of delivery vertices. By forcing $\boldsymbol{c}$ to take this form we can remove the constraint $\sum_{k \in K} x^k_{i,n+i} = 1$ for all $i \in P$ as given in (3.7). We can do so because it is implicitly stated that if (i, j) is an edge in the graph, then $c_{i,j} \neq M$.

One can take this idea even further. We can reformulate the MRP from a matrix formulation to a graph formulation. This would mean that we only incorporate edges that exist in the graph $G = (V, E)$, instead of including all combinations of i and j

for $i, j \in V$. So if $c_{ij} = M$, we would not add this edge in the graph. This makes it possible to pair vertices together, i.e. ensuring that a vehicle servicing a given pickup vertex can only travel to a specific delivery vertex. This reduces the problem size and thus increases efficiency and decreases resource requirements. It also opens for that some sites may have a particular type of mass being transported which can only be delivered at specified locations. We will investigate the graph formulation in Section 3.3 and Section 3.4.

3.2.2 Dimensions of Decision Variable *t*

In the earlier formulation of the MRP, we defined that the continuous decision variable t_i^k represented the beginning of service time for vehicle $k \in K$ at vertex i. To simplify the model, we wish to remove the k-term as it does not add any information to the solution that cannot be calculated from x. Thus, we rewrite the constraint $t_i^k + s_i + d_{ij}^k - t_j^k \leq (W + s_i + d_{ij}^k) \cdot (1 - x_{ij}^k)$ for all $i, j \in V, k \in K$ as given in (3.9), as

$$t_i + s_i + d_{ij}^k - t_j \leq (W + s_i + d_{ij}^k) \cdot (1 - x_{ij}^k) \quad \forall\, i, j \in V, k \in K.$$

We now prove that the new formulation of the constraint still disallows cycles. We imagine a subtour that visits all vertices $\{v_\alpha, v_{\alpha+1}, ..., v_{\beta-1}, v_\beta\} \in V$ before returning to v_α. The arrival time constraint for vertices v_α and $v_{\alpha+1}$ is then

$$t_\alpha + s_\alpha + d_{\alpha,\alpha+1}^k \leq t_{\alpha+1} \quad \forall\, k \in K$$

since $\sum_{k \in K} x_{\alpha,\alpha+1}^k = 1$. Furthermore, the arrival time constraint for $v_{\alpha+1}$ and $v_{\alpha+2}$ is

$$t_{\alpha+1} + s_{\alpha+1} + d_{\alpha+1,\alpha+2}^k \leq t_{\alpha+2} \quad \forall\, k \in K,$$

which is equivalent to

$$t_\alpha + s_\alpha + d_{\alpha,\alpha+1} + s_{\alpha+1} + d^k_{\alpha+1,\alpha+2} \leq t_{\alpha+2} \quad \forall\, k \in K.$$

After continuing calculating the arrival time for each next vertex in the cycle, we finally get that

$$t_\alpha + s_\alpha + d^k_{\alpha,\alpha+1} + \cdots + s_\beta + d^k_{\beta,\alpha} \leq t_\alpha \quad \forall\, k \in K.$$

when we are returning back to the beginning. This statement is a contradiction since $s_i, d^k_{ij} > 0$ for all $i, j \in V$ and $k \in K$. Thus, a solution of the MRP contains no subtours even when removing k from t.

3.2.3 Duplication of Vertices

We must service all vertices in a typical formulation of the VRP. This is also the case for the `preliminaryMRP` as described in (3.13). However, for the MRP this will be highly resource-intensive. We would need to calculate how many times all vehicles would visit the vertices during a given day. This negates the objective of the MRP, as the number of visits is what we wish to maximize. Therefore, we seek to express that all vertices do not have to be visited in this formulation of the MRP.

We start by looking at constraint (3.2) which is given by $\sum_{k\in K}\sum_{j\in V} x^k_{ij} = 1$ for all $i \in V\backslash\{0, n+m+1\}$. It states that all vertices which are not the depots must be visited *exactly* once. We instead formulate the constraint as

$$\sum_{k\in K}\sum_{j\in V} x^k_{ij} \leq 1 \;\forall\, i \in V\backslash\{0, n+m+1\},$$

i.e. all vertices must be visited *at most* once. Now, we only need to specify a sufficiently high number of duplicates to achieve a feasible solution. Keep in mind that for every duplicate we include, we also increase the size of the problem. It is therefore advisable to minimize the amount of excess duplicates in the graph.

We also note that if we modify this constraint and leave the rest of the formulation unchanged, the optimal solution is that the vehicles remain at the depot. To avoid this, we can include a penalty function to the objective function to ensure that the vehicles are incentivized to transport mass.

3.2.4 Objective Function

The objective function of `preliminaryMRP` is defined as $\sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij}^k$ as specified in (3.1). This straightforward function only calculates the total cost of the traversed edges. However, this previous formulation of the objective function is not beneficial for the MRP as we no longer specify that all vertices must be visited. Therefore, we wish to introduce a penalty associated with not visiting a vertex. We establish a parameter μ that represents the sum of all available edges in the graph [9, 16]. Thus, we can write a penalty function as $\mu\sum_{j\in V}\left(1-\sum_{k\in K}\sum_{i\in V} x_{ij}^k\right)$ and the new objective function we wish to minimize becomes

$$\sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij}^k + \mu\sum_{j\in V}\left(1-\sum_{k\in K}\sum_{i\in P} x_{ij}^k\right).$$

Here, we see that, as before, we wish to minimize the total cost by choosing the edges with the least cost. In addition, we add the penalty term μ to the objective value for each unvisited pickup vertex. This will ensure that the solution does not benefit from keeping all the vehicles at the depot because then significant penalties will rack up, and the objective value will become quite large. By only penalizing unvisited pickup vertices, we simplify the objective function. However, as there are no edges from the pickup vertices directly to the depot, we implicitly also punish unvisited delivery vertices.

There is more that we can adjust in the objective function. We can also introduce a priority term, p_i, which we defined as

$$p_i = \begin{cases} 1 & \text{if } i \text{ has low priority} \\ \lambda & \text{if } i \text{ has high priority.} \end{cases} \tag{3.14}$$

where $\lambda > 1$ defines how high priority a given vertex i has. If we set $\lambda = 2$, that means that vertex i is more worth than two visits of a vertex with priority 1. We include this into the objective function by writing

$$\sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij}^k + \mu\sum_{j\in V}\left(1 - \sum_{k\in K}\sum_{i\in P} p_i x_{ij}^k\right).$$

We could also consider adding a time penalty to the objective function. However, this results in problems with scaling the terms, as it is difficult to weigh the benefit of saving time against the penalties of not visiting a vertex. We therefore leave this for another time.

3.2.5 Starting Point of Each Vehicle

To resemble a real-life problem, we also need to define where the vehicles start their routes. Since the depots here are fictional, the model picks where the vehicles start their route independently from where the vehicles are stored. To counteract this, we introduce the set of starting points for each vehicle $k \in K$ that we call $\mathcal{S}_k$. Now, we can introduce the new constraint

$$x_{0\mathcal{S}_k}^k = 1 \quad \forall\, k \in K,$$

which ensures that each vehicle leaves the start-depot to its correct starting position. We assign a zero cost to each of these edges since this is an imagined route.

However, this constraint can be overly restrictive as it forces a vehicle to serve the route's starting vertex before any other vertex. This is only fair if we assume that it

takes less time to wait for its turn at the starting point than leaving to help somewhere else without serving it. This will often not be the case. A remedy is presented as future work in Section 7.1.

Another complication is that we now force all vehicles to be used. This is not always cost beneficial for a construction company. It can be more cost effective to use fewer vehicles, as this requires fewer drivers. A fix would be to introduce a new edge directly from the starting position to the fictional depot. However, it will always be beneficial to use more vehicles from the way we have specified the objective function, as then there would be less driving back and forth between sites. Therefore, we could assign a negative cost to these edges to correct for the subsequent lower objective value resulting from not using an extra vehicle.

3.2.6 Time Windows

In the `preliminaryMRP` as described in (3.13), we relied on knowing exact time windows for when to service each vertex. This was formulated in (3.10) as $a_i \leq t_i^k \leq b_i$ for all $i \in V, k \in K$ which is problematic. It is difficult to accurately predict when vertices are ready to be serviced beforehand as it depends on many uncertain factors, such as when the vertex last was visited. Therefore, in this new formulation, we introduce the parameter *dig rate* as r_i and the sets P_D and D_D, which are the sets of pickup duplicates and delivery duplicates, respectively. The dig rate denotes how long time an excavator uses to excavate a truckload worth of mass.

Now, we can define the time windows of the vertices by setting that the vertices can only be served if the earlier duplicates that have already been served. In other words, vertex i_{j+1} can only be served if vertex i_j already has been serviced, and i_j and i_{j+1} are duplicates of the same vertex. The purpose of this is to ensure that no two vehicles service the same vertex simultaneously.

3.3 Mathematical Formulation

With these preliminary refinements explained, we are now ready to express the improved version of the MRP. We start by defining the weighted and directed graph $G = (V, E)$ as the set of edges and vertices at a given construction site, like we did before. We set that $n+m+2$ is the total number of vertices, where vertices 0 and $n+m+1$ are the start- and end-depot, respectively. We set that n is the number of pickup vertices and m is the number of delivery vertices, such that $V = \{0, 1, \ldots, n, n+1, \ldots, n+m, n+m+1\}$.

Next, we define the sets

P	set of pickup vertices, $P = \{1, \ldots, n\}$,
D	set of delivery vertices, $D = \{n+1, \ldots, n+m\}$,
P_D	set of duplicates of pickup vertices, where $P_D \subset P$,
D_D	set of duplicates of delivery vertices, where $D_D \subset D$,
$e^-(i)$	edges out of vertex $i \in V$ in graph $G = (V, E)$,
$e^+(i)$	edges into vertex $i \in V$ in graph $G = (V, E)$,
i_p	the previous duplicate of vertex $i \in P_D \cup D_D \subset V$,

and the parameters

K	set of vehicles,
c_{ij}	cost of traversing the edge $(i, j) \in E$ given in distance between i and j,
d_{ij}	duration of time needed to traverse the edge $(i, j) \in E$,
s_i	service duration of vertex i,
r_i	time needed to prepare a truckload of mass at pickup vertex $i \in P$,
μ	penalty term calculated as the sum of the cost of all edges in G,

W total allowed amount of working hours per day,

$\mathcal{S}_k$ starting position of vehicle $k \in K$,

p_i priority of vertex i as defined in (3.14).

Notice that we remove the k-term from travel time d. We do this because we do not have any data on the different speeds vehicles use to traverse certain edges. Thus, including the k-term would only complicate the formulation. Lastly, we express our decision variables as

x_{ij}^k binary flow variable that equals 1 if edge $(i,j) \in E$ is traversed in the optimal solution by vehicle $k \in K$, and 0 otherwise,

t_i beginning of service time at vertex i.

Then we are ready to formulate the MRP. We start with the objective function

$$\texttt{MostVertices}(c,\mu,p) = \sum_{k\in K}\sum_{i\in V}\sum_{j\in e^-(i)} c_{ij}x_{ij}^k + \mu \sum_{j\in e^-(i)}\left(1 - \sum_{k\in K}\sum_{i\in P} p_i x_{ij}^k\right). \tag{3.15}$$

The function `MostVertices` is what we wish to minimize in the MRP. It calculates the cost of the chosen route and punishes for every unvisited pickup vertex according to its priority as previously explained in Section 3.2.4. Next, we define our constraints dealing with flow conservation. We introduce

$$\sum_{k\in K}\sum_{j\in e^-(i)} x_{ij}^k \leq 1 \quad \forall\, i \in V\setminus\{0, n+m+1\}. \tag{3.16}$$

which establishes that every vertex should be visited at most once. This prohibits multiple vehicles from serving the same vertex while also specifying that not all vertices must be visited. Next, we have the constraint

$$\sum_{j \in e^-(0)} x_{0j}^k = 1 \quad \forall\, k \in K \tag{3.17}$$

which guarantees that every vehicle must leave the start-depot. On the same note, constraint

$$\sum_{i \in e^+(n+m+1)} x_{i,n+m+1}^k = 1 \quad \forall\, k \in K \tag{3.18}$$

ensures that all vehicles end their route at the end-depot. Next, we wish to specify the starting position of each vehicle so we introduce the constraint

$$x_{0\mathcal{S}_k}^k = 1 \quad \forall\, k \in K. \tag{3.19}$$

It requires a vehicle $k \in K$ to start its route at vertex $\mathcal{S}_k$. Note that $(n+m+1) \in e^-(i)$ for all $i \in \mathcal{S}_k$, so that vehicles can remain unused. Furthermore, we wish to ensure flow conservation by the constraint

$$\sum_{j \in e^+(i)} x_{ji}^k - \sum_{j \in e^-(i)} x_{ij}^k = 0 \quad \forall\, i \in V, k \in K. \tag{3.20}$$

This guarantees that a vehicle cannot suddenly appear at a vertex, but must follow the given road network. It also guarantees that a vehicle cannot start or end its route anywhere except the depots. Now, onto the time limit constraint. We set that

$$t_i + s_i + d_{ij} - t_j \leq (W + s_i + d_{ij}) \cdot \left(1 - \sum_{k \in K} x_{ij}^k\right) \quad \forall\, i \in V, j \in e^-(i), \tag{3.21}$$

which ensures that a vehicle is limited by the travel and service times of the graph. This constraint, as we have previously proved, will also prevent subtours. Furthermore, we also set time limitations on when these vertices can be visited by asserting that

$$t_i \geq s_{i_p} + r_{i_p} + \left(\sum_{k \in K} \sum_{j \in e^-(i_p)} x^k_{i_p j} - 1 \right) \left(s_{i_p} + r_{i_p} \right) + t_{i_p} \quad \forall\, i \in P_D. \tag{3.22}$$

This constraint guarantees that we cannot visit the next duplicate of a pickup location, before the previous duplicate is serviced and the excavator at the vertex has had time to prepare a new truckload. Similarly,

$$t_i \geq s_{i_p} + \left(\sum_{k \in K} \sum_{j \in e^-(i_p)} x^k_{i_p j} - 1 \right) s_{i_p} + t_{i_p} \quad \forall\, i \in D_D \tag{3.23}$$

ensures the same for delivery vertices. We recognize that the formulation as of now is a highly symmetrical problem, i.e. many solutions will give the same objective value. This is because the duplicated vertices of the same location are interchangeable. Therefore, we want to introduce a symmetry-breaking constraint to reduce the solution space of the model [10, 15]. We introduce the constraint

$$\sum_{k \in K} \sum_{j \in e^-(i)} x^k_{ij} \geq \sum_{k \in K} \sum_{j \in e^-(i')} x^k_{i'j} \quad \forall\, i, i' \in \{P_i | \delta_i < \delta_{i'}\}, \tag{3.24}$$

where δ_i denotes the duplicate index of vertex i and P_i denotes the set of duplicates of pickup vertex i. It states that we must visit the duplicates in lexicographical order. In other words, if i_0 is the first duplicate of vertex i, then i_0 must be visited before i_1, i_2, and so on. This way we remove a lot of the solutions that would have given identical objective values. Lastly, we state that

$$x^k_{ij} \in \{0, 1\} \quad \forall\, i, j \in V, k \in K \tag{3.25}$$

and

$$0 \leq t_i \leq W \quad \forall\, i \in V, k \in K \tag{3.26}$$

define the domain of the problem. Thus, the MRP is defined as

$$\texttt{MRP}(c,\mu,s,r,W,p) = \begin{cases} \min_{x,t} & \texttt{MostVertices}(c,\mu,p) \text{ as given in (3.15)} \\ \text{subject to} & (3.16),(3.17),(3.18),(3.19),(3.20),(3.21),(3.22), \\ & (3.23),(3.24),(3.25), \text{ and } (3.26). \end{cases} \tag{3.27}$$

3.4 Reduced Mathematical Formulation

In this section, we will formulate a reduced version of the MRP where we assume a homogeneous fleet of vehicles. We investigate this case because we, in this thesis, do not have data on the types of vehicles the construction sites have available. Thus, we can formulate a more computationally efficient model by assuming the fleet is homogeneous. Note that if the fleet is heterogeneous but the vehicles have approximately the same speed and carrying capacity, the solutions could still be relevant.

For this model, we redefine our decision variable, x, as

x_{ij} binary flow variable that equals 1 if edge $(i,j) \in E$ is traversed in the optimal solution, and 0 otherwise,

or in other words, we remove the k-dimension from x. The objective function will then become

$$\texttt{modifiedMostVertices}(c,\mu,p) = \sum_{i\in V}\sum_{j\in e^-(i)} c_{ij}x_{ij} + \mu \sum_{j\in e^-(i)}\left(1 - \sum_{i\in P} p_i x_{ij}\right), \tag{3.28}$$

which has the same purpose as before, i.e. minimizing costs and penalizing unserviced pickup vertices. The only difference is that we do not sum over $k \in K$ anymore. The

same goes for the constraint

$$\sum_{j \in e^-(i)} x_{ij} \leq 1 \quad \forall\, i \in V \backslash \{0, n+m+1\}, \tag{3.29}$$

that details how many times each vertex can be visited. For constraint

$$\sum_{j \in e^-(0)} x_{0j} = K, \tag{3.30}$$

and

$$\sum_{i \in e^+(n+m+1)} x_{i,n+m+1} = K, \tag{3.31}$$

we ensure that the number of vehicles leaving and entering the depot equals the number of available vehicles. Moreover, constraint

$$x_{0\mathcal{S}_k} = 1 \quad \forall\, k \in K, \tag{3.32}$$

states that every starting point for each vehicle must be visited, but we do not specify which vehicle starts where, as it is no longer relevant. The flow conservation,

$$\sum_{j \in e^+(i)} x_{ji} - \sum_{j \in e^-(i)} x_{ij} = 0 \quad \forall\, i \in V, \tag{3.33}$$

still assumes that the same amount of vehicles must both enter and exit a given vertex i. Furthermore, the constraints detailing the arrival time,

$$t_i + s_i + d_{ij} - t_j \leq (W + s_i + d_{ij}) \cdot \Big(1 - x_{ij}\Big) \quad \forall\, i \in V, j \in e^-(i), \tag{3.34}$$

$$t_i \geq s_{i_p} + r_{i_p} + \Bigg(\sum_{j \in e^-(i_p)} x_{i_p j} - 1 \Bigg) \big(s_{i_p} + r_{i_p} \big) + t_{i_p} \quad \forall\, i \in P_D, \tag{3.35}$$

and

$$t_i \geq s_{i_p} + \Bigg(\sum_{j \in e^-(i_p)} x_{i_p j} - 1 \Bigg) s_{i_p} + t_{i_p} \quad \forall\, i \in D_D, \tag{3.36}$$

are the same as stated previously, except that we no longer sum over $k \in K$. The symmetry breaking constraint is defined as

$$\sum_{j \in e^-(i)} x_{ij} \geq \sum_{j \in e^-(i')} x_{i'j} \quad \forall\, i, i' \in \{P_i | \delta_i < \delta_{i'}\}, \tag{3.37}$$

which is also nearly unchanged as we have only removed the k-term. Lastly, we denote the domain of the decision variables as

$$x_{ij} \in \{0, 1\} \quad \forall\, i, j \in V \tag{3.38}$$

and

$$0 \leq t_i \leq W \quad \forall\, i \in V. \tag{3.39}$$

Thus, the reduced formulation of the MRP is

$$\texttt{reducedMRP}(c,\mu,s,r,W,p) = \begin{cases} \min_{x,t} & \texttt{modifiedMostVertices}(c,\mu,p) \text{ as given in (3.28)} \\ \text{subject to} & (3.29), (3.30), (3.31), (3.32), (3.33), (3.34), (3.35), \\ & (3.36), (3.37), (3.38), \text{ and } (3.39). \end{cases} \tag{3.40}$$

Chapter 4

The Mass Relocation Problem with Uncertainty

The goal of this thesis is to solve the Mass Relocation Problem (MRP) for a real-world problem. Thus, we will in this chapter investigate how to include uncertainty in the travel times d, and the dig rates, r, to achieve the Mass Relocation Problem with Uncertainty (MRPU). Section 4.1 introduces how the Soyster approach can be applied to `reducedMRP`, while Section 4.2 presents the Bertsimas-Sim approach and Section 4.3 proposes a new approach specifically for the MRPU.

4.1 The Soyster Approach

The Soyster approach is a method of ensuring the strict robustness of a solution and is introduced in Section 2.2.1. The approach entails using the maximum value of the uncertain parameters to guarantee feasibility. This method is appropriate for the MRP as we want to limit the uncertainty in travel times and dig rates, which worst values directly correlate to the worst possible objective value. If we increase the travel time or dig rate, the vehicles will have less time to visit vertices and may not have time to service as many vertices as they would in a best-case scenario. Note that there is

uncertainty in other parameters than d and r, such as the service time s. However, in this thesis we assume that d and r are the most significant terms which makes the others negligible.

4.1.1 Mathematical Formulation

To formulate the reduced MRP as the MRPU, we need to apply the Soyster approach to the constraints that include uncertainty; (3.34) and (3.35). As such, the new, robust constraints are denoted as

$$t_i + s_i + \bar{d}_{ij} - t_j \leq (W + s_i + \bar{d}_{ij}) \cdot \left(1 - x_{ij}\right) \quad \forall\, i \in V, j \in e^-(i), \tag{4.1}$$

and

$$t_i \geq s_{i_p} + \bar{r}_{i_p} + \left(\sum_{j \in e^-(i_p)} x_{i_p j} - 1\right)\left(s_{i_p} + \bar{r}_{i_p}\right) + t_{i_p} \quad \forall\, i \in P_D, \tag{4.2}$$

where we have assumed the worst possible values for all d_{ij} and r_i, i.e. $\bar{d}_{ij}$ and $\bar{r}_i$. We have $\bar{d}_{ij}$ on each side of the equation (4.1) so that if there is no path (i, j) in the optimal solution, the maximum time limit W will not be affected. Thus, the MRPU can be defined as

$$\texttt{MRPU\_Soyster}(c, s, d, r, W, p) = \begin{cases} \min_{x,t} & \texttt{modifiedMostVertices}(c, \mu, p) \text{ as given in (3.28)} \\ \text{subject to} & (3.29), (3.30), (3.31), (3.32), (3.33), (3.36), (3.37), \\ & (3.38), (4.1), \text{ and } (4.2). \end{cases} \tag{4.3}$$

where we have replaced constraints (3.34) and (3.35) with (4.1) and (4.2), respectively. Similarly, if we would want to apply the Soyster approach to the full MRP given

in (3.27), we would do the same and replace d_{ij} and r_i with $\bar{d}_{ij}$ and $\bar{r}_i$ in the relevant constraints.

4.1.2 Drawbacks

The Soyster approach is an easy way of guaranteeing the feasibility of a solution, but it has some drawbacks. It will always provide feasible solutions, given that the uncertainty sets are accurate. The uncertainty sets given for the MRPU will not be able to take into account every possible scenario. For example, a driver could get sick or a vehicle could break down. Such events are usually not included in the uncertainty sets as they happen so rarely.

Furthermore, it is not the ideal way of ensuring robustness. Let us say that the uncertainty in specific parameters mainly revolve around the weather conditions. By using the Soyster approach, one would then get the parameters corresponding to the worst possible weather. We can safely assume it will not storm every day, and it would therefore be wasteful to schedule the site based on the assumption that it does. We will therefore also investigate other options for including robustness.

4.2 The Bertsimas-Sim Approach

The Bertsimas-Sim approach is a way of configuring the degree of conservativeness in a solution [4]. The idea is presented in Section 2.2.2. It introduces an uncertainty budget, Γ, that represents how conservative we want the solution to be. If Γ equals the number of edges in the graph, then the method will be strictly robust as all edges would take the worst possible value. If Γ is smaller than the number of edges, then the Γ edges with the worst values in the worst-case scenario will assume these values, while the rest will take the values of the best-case scenario. In such a way, we construct a robust solution by applying the worst-case scenarios to the parameters that make the largest negative impact on the arrival time decision variable.

We remark that the Bertsimas-Sim method will not guarantee feasibility for the MRPU. This is because this method is based on solving for the best case scenario and then modifying the given solution to handle travel time uncertainty. This means that we no longer adhere to the constraints restricting multiple vehicles visiting the same geographic location at once. Nevertheless, we will investigate it as it lays the foundation for the new robustness approach presented later in Section 4.3.

4.2.1 Mathematical Formulation

The formulation of the MRPU using the Bertsimas-Sim approach is based on the works of Agra et al. [1], Lee et al. [14], and Toklu et al. [22]. We start by introducing the following notation:

Γ	number of edges in the solution that we choose to have the worst possible value for the objective function.
$sol[k]$	route of vehicle $k \in K$ of the optimal solution.
$\|sol[k]\|$	number of visits by vehicle $k \in K$ of the optimal solution when not counting the depot.
$sol[k,n]$	nth visit of vehicle $k \in K$ of the optimal solution.

To use the Bertsimas-Sim approach, we first solve the MRPU for the best-case scenario. We introduce the new constraints

$$t_i + s_i + \underline{d}_{ij} - t_j \leq \left(W + s_i + \underline{d}_{ij}\right) \cdot (1 - x_{ij}) \quad \forall\, i \in V, j \in e^-(i) \tag{4.4}$$

to replace constraint (3.34) and

$$t_i \geq s_{i_p} + \underline{r}_{i_p} + \left(\sum_{j \in e^-(i_p)} x_{i_p j} - 1 \right)\left(s_{i_p} + \underline{r}_{i_p} + t_{i_p}\right) \quad \forall\, i \in P_D \tag{4.5}$$

to replace (3.35). The next step is to apply the uncertainty budgets with two recursive functions $\texttt{d\_MaximumLatency}(sol[k], n, \Gamma^d)$ and $\texttt{r\_MaximumLatency}(sol[k], n, \Gamma^r)$. They were first proposed by Agra et al. [1] and are adapted here like so:

$$
\texttt{d\_MaximumLatency}(sol[k], n, \Gamma^d) =
$$

$$
\begin{cases}
0 \quad \text{if } n = 0 \\
\\
\max\{t_j, \\
\qquad \texttt{d\_MaximumLatency}(sol[k], n-1, 0) + t_i + s_i + \underline{d}_{i,j}\} \\
\qquad \text{if } \ 1 \le n \le |sol[k]| \ \text{ and } \ \Gamma^d = 0 \\
\\
\max\{t_j, \\
\qquad \texttt{d\_MaximumLatency}(sol[k], n-1, 0) + s_i + \underline{d}_{i,j} + \Gamma^d(\bar{d}_{i,j} - \underline{d}_{i,j}), \\
\qquad \texttt{d\_MaximumLatency}(sol[k], n-1, \Gamma^d) + s_i + \underline{d}_{i,j}\} \\
\qquad \text{if } \ 1 \le n \le |sol[k]| \ \text{ and } \ 0 < \Gamma^d < 1 \\
\\
\max\{t_j, \\
\qquad \texttt{d\_MaximumLatency}(sol[k], n-1, \Gamma^d - 1) + s_i + \bar{d}_{i,j}, \\
\qquad \texttt{d\_MaximumLatency}(sol[k], n-1, \Gamma^d) + s_i + \underline{d}_{i,j}\} \\
\qquad \text{if } \ 1 \le n \le |sol[k]| \ \text{ and } \ 1 \le \Gamma^d \le n-1 \\
\\
-\infty \ \text{ if } \ 1 \le n \le |sol[k]| \ \text{ and } \ \Gamma^d \ge n
\end{cases}
$$

and

$$\texttt{r\_MaximumLatency}(sol[k], n, \Gamma^r) =$$

$$\begin{cases} 0 \quad \text{if } n = 0 \\ \\ \max\{t_i, \\ \qquad \texttt{r\_MaximumLatency}(sol[k], n-1, 0) + s_j + \underline{r}_j\} \\ \qquad \text{if } \ 1 \le n \le |sol[k]| \ \text{ and } \ \Gamma^r = 0 \\ \\ \max\{t_i, \\ \qquad \texttt{r\_MaximumLatency}(sol[k], n-1, 0) + s_j + \underline{r}_j + \Gamma^r(\bar{r}_j - \underline{r}_j), \\ \qquad \texttt{r\_MaximumLatency}(sol[k], n-1, \Gamma^r) + s_j + \underline{r}_j\} \\ \qquad \text{if } \ 1 \le n \le |sol[k]| \ \text{ and } \ 0 < \Gamma^r < 1 \\ \\ \max\{t_i, \\ \qquad \texttt{r\_MaximumLatency}(sol[k], n-1, \Gamma^r - 1) + s_j + \bar{r}_j, \\ \qquad \texttt{r\_MaximumLatency}(sol[k], n-1, \Gamma^r) + s_j + \underline{r}_j\} \\ \qquad \text{if } \ 1 \le n \le |sol[k]| \ \text{ and } \ 1 \le \Gamma^r \le n-1 \\ \\ -\infty \ \text{ if } \ 1 \le n \le |sol[k]| \ \text{ and } \ \Gamma^r \ge n \end{cases}$$

where we the uncertainty budgets Γ^d and Γ^r are assigned to d and r, respectively. These functions recursively loop through all vertices used in the solution $sol[k]$ and update the arrival time, t, with respect to the Bertsimas-Sim approach. It does not matter in which order one employs them because we have not included the maximum arrival time, W, which they must adhere to. We will instead do this at a later stage. Thus, the result is the same regardless of the order in which they are employed.

Next, we wish to introduce a new term for configuring the robustness as

$$\xi = \frac{\Gamma}{\sum_{k \in K} |sol[k]|}$$

which was first done by Toklu et. al [22]. The parameter ξ will always be between 0 and 1, as we compare our uncertainty budget Γ against the total amount of traversed edges in the solution. In other words, ξ will be a term we configure on the basis of how conservative we wish the solution to be. If $\xi = 1$, we have a strictly robust solution, and if $\xi = 0$, we assume the best case values for all edges. We introduce this term as it is more intuitive than Γ, since we no longer need to know how many edges there are in the graph.

Now, we can calculate a robust solution based on the Bertsimas-Sim approach by Algorithm 1, where `MRPU_BestCase` is defined as

$$\texttt{MRPU\_BestCase}(c,s,d,r,W,p) = \begin{cases} \min_{x,t} & \texttt{modifiedMostVertices}(c,\mu,p) \text{ as given in (3.28)} \\ \text{subject to} & (3.29),(3.30),(3.31),(3.32),(3.33),(3.36),(3.37), \\ & (3.38),(4.4), \text{ and } (4.5). \end{cases} \tag{4.8}$$

Algorithm 1 Calculating a Robust Solution with Bertsimas-Sim

```
1:  function RobustSolution_BertsimasSim(ξ^d, ξ^r, c, s, d, r, W)
2:      sol ← MRPU_BestCase(c, s, d, r, W)
3:      Γ^d ← ξ^d Σ_{k∈K} |sol[k]|
4:      Γ^r ← ξ^r Σ_{k∈K} |sol[k]|
5:      for each vehicle k in K do
6:          d_MaximumLatency(sol[k], |sol[k]|, Γ^d)
7:          r_MaximumLatency(sol[k], |sol[k]|, Γ^r)
8:          for each visit sol[k, n] in sol[k] do
9:              if t_n > W then
10:                 sol[k] ← sol[k, 0 : n]
11:                 break
12:             end if
13:         end for
14:     end for
15:     return sol
16: end function
```

4.2.2 Drawbacks

There are some pitfalls regarding this approach. First and foremost, we can no longer guarantee feasible solutions as the solutions does not necessarily comply with the constraints for the MRP. It is also more resource-intensive than the other approaches we investigate as we have to manipulate the given initial solution. This results in a longer computation time and more resources being occupied with solving this problem. Furthermore, this method is, similarly to the Soyster approach, very dependent on the values in the uncertainty set. If the uncertainty set mainly concerns the weather conditions and the worst-case scenario is gathered from a once-in-a-decade storm, then even just a few of the worst case values would obscure a meaningful solution. The last approach we will consider resolves this issue.

4.3 The Removing Outliers Approach

By removing outliers in the uncertainty sets, we can achieve feasible solutions without overzealous uncertainty assumptions. The idea is to introduce a parameter that configures the degree of conservativeness by removing the specified amount of outliers. It is inspired by both the Soyster approach and the Bertsimas-Sim approach. We keep the simplicity of the Soyster approach by optimizing based on the maximum allowed values. We also want to be able to configure the degree of conservativeness, as we could with the Bertsimas-Sim approach. However, instead of calculating the uncertainty by only adjusting the uncertainty coefficients that have the largest negative impact on the objective value, we adjust all uncertainty coefficients but to a lesser degree. The thought is that this will result in a comparable conservativeness of the solution, while not overvaluing potential outliers in the data sets. This approach is what we will denote as the Removing Outliers Approach (ROA). It is beneficial for cases where there exists anomalies in the uncertainty sets, such as with the MRPU.

4.3.1 Mathematical Formulation

We introduce $\xi \in [0,1]$ which is a parameter that determines the degree of conservativeness by picking the $\lceil \xi \cdot |\mathcal{U}| \rceil$ element of uncertainty in the discrete uncertainty set $\mathcal{U}$, where $\lceil x \rceil$ denotes the rounding up of a parameter x and $|\mathcal{U}|$ denotes the size of $\mathcal{U}$, i.e. the number of elements in $\mathcal{U}$. We set ξ as the parameter that determines how many percent of the values in the discrete uncertainty sets we want to consider. This is beneficial for the MRP with real data as the travel time is given as a set $\mathcal{U}^{d_{ij}} = \{d_{ij}^1, d_{ij}^2, \ldots, d_{ij}^n\}$ where we have n different possible values of the travel time of edge (i,j). We assume all such sets will be sorted from shortest to highest time, $d_{ij}^1 \leq d_{ij}^2 \leq \ldots \leq d_{ij}^n$. As such, by using this method, one excludes the highest values. For the MRP, this can be advantageous as the highest travel times are often exceptions that could stem from, for example, the tracking of the vehicles' positions being wrong or the vehicle getting a flat tire. The resulting constraints with this method will be

$$t_i + s_i + \hat{d}_{ij} - t_j \leq \left(W + s_i + \hat{d}_{ij}\right) \cdot \left(1 - x_{ij}\right) \quad \forall\, i \in V, j \in e^-(i) \tag{4.9}$$

where we set that $\hat{d}_{ij}$ is the ν-th element of the uncertainty set for d_{ij} as such

$$\hat{d}_{ij} = \mathcal{U}_\nu^{d_{ij}}, \;\; \nu = \lceil \xi^d \cdot |\mathcal{U}^{d_{ij}}| \rceil.$$

Furthermore, we also have

$$t_i \geq s_{i_p} + \hat{r}_{i_p} + \left(\sum_{j \in e^-(i_p)} x_{i_p j} - 1 \right) \left(s_{i_p} + \hat{r}_{i_p} + t_{i_p} \right) \quad \forall\, i \in P_D \tag{4.10}$$

where

$$\hat{r}_i = \mathcal{U}_\nu^{r_i}, \;\; \nu = \lceil \xi^r \cdot |\mathcal{U}^{r_i}| \rceil$$

and ξ^d and ξ^r denote the uncertainty configuration parameter for d and r, respectively. Now, we can write the MRPU with the ROA as

$$\texttt{MRPU\_Outliers}(c,s,d,r,W,p) = \begin{cases} \min_{x,t} & \texttt{modifiedMostVertices}(c,\mu,p) \text{ as given in (3.28)} \\ \text{subject to} & (3.29),(3.30),(3.31),(3.32),(3.33),(3.36),(3.37), \\ & (3.38),(4.9), \text{ and } (4.10). \end{cases} \tag{4.11}$$

4.3.2 Drawbacks

The ROA only works well if one has some knowledge of what the uncertainty sets contain. By recognizing approximately how large part of the sets are outliers, one would get rather good robust solutions that do not encourage excessive downtime. Furthermore, the method may also work better if the uncertainty sets are large. With small sets and a high ξ, the approach would often be equivalent to Soyster approach as we would then likely pick the maximum value. This is not a problem in itself but would entail that we would potentially include outliers that do not represent actual events, such as data errors for example.

Chapter 5

Performance of Model

This chapter will investigate the performance of the Mass Relocation Problem (MRP) and the Mass Relocation Problem with Uncertainty (MRPU) as defined previously. Section 5.1 looks into how the MRP functions for a test case and investigates potential faults. Section 5.2 looks closer at the MRPU and the different robustness approaches.

5.1 The Mass Relocation Problem

To test the performance of our formulation of the MRP, we first implement it for a test case to simplify the interpretation of the results. We keep the test case in the same format as the data provided by Skanska, which we will investigate closer in Chapter 6, to simulate a real-life solution.

The test case we will use from here on out is presented in Figure 5.1. In Figure 5.1a we depict of the graph $G' = (V', E')$, as it would look in real life. A crossroad is depicted in white, the pickup vertices in purple and the delivery vertices in blue. In Figure 5.1b we have the test case in the desired format for the MRP with two duplicates of each pickup and delivery vertex. We define the graph depicted here as $G = (V, E)$ where

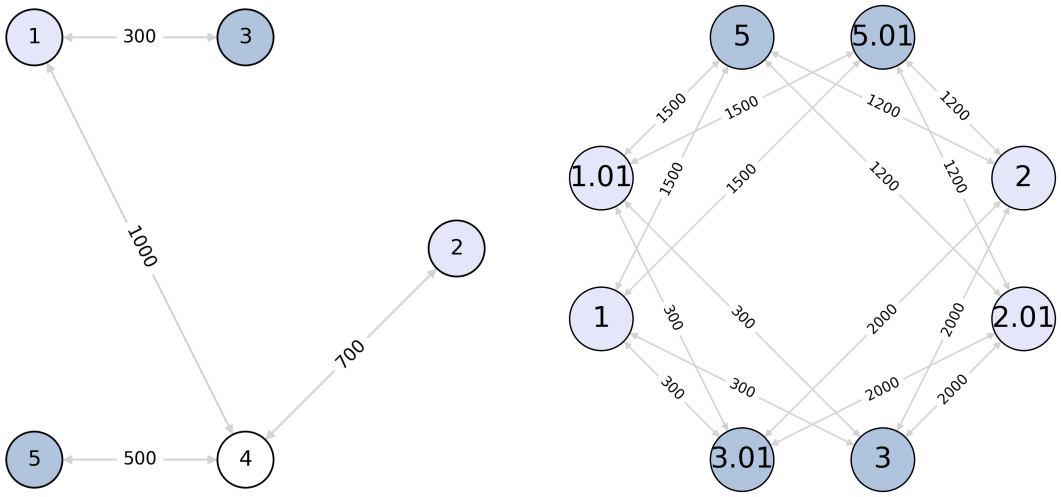


Figure 5.1: (a) A depiction of the geographic location of the vertices and connecting road network for a test case, and **(b)** a depiction of the road network after formatting with two duplicates of each vertex. Purple vertices depict pickup sites, blue vertices are delivery sites, and white vertices are crossroads. The cost of edges is given in meters.

$$E = \{(i,j) \mid (i \in P \wedge j \in D) \vee (i \in D \wedge j \in P)\}$$

and V is the set of vertices. Furthermore, P and D are the sets of pickup and delivery vertices in the graph G, respectively. The cost of each edge, c_{ij} for all $(i,j) \in E$, is calculated from the shortest path algorithm Dijkstra of G'. We assume there will never be road congestion so that vehicles can traverse the graph freely.

Next, we set the parameters of the model. We define the travel times as $d_{ij} = c_{ij}/\text{speed}$, where we assume that all vehicles hold a constant speed of 16 km/h. Furthermore, we set that the service time of all vertices is 5 minutes, and that all excavators prepare a truckload worth of materials every 10 minutes, meaning that $s_i = 5$ for all $i \in V \backslash \{0, n+m+1\}$ and $r_i = 10$ for all $i \in P$. We have assumed for this test case that these values are constant and independent. This will not be the case for a real construction site. However, as we do not have the data available on how they relate to each other, we will proceed with this assumption for the rest of this thesis. Moreover, we do not include any priority vertices, or in other words $p_i = 1$ for all $i \in V$. We also do not

include any negative cost on the edges leading from the starting positions to the depot unless stated otherwise.

To solve the MRP, we have used the optimization software Gurobi Optimizer version 9.1.1 with build v9.1.1rc0 for mac64 with the Gurobi Python Interface, gurobipy. It is one of the fastest solvers available [2] and can be used for linear, quadratic, and mixed-integer linear programming, among others [11]. Furthermore, the solver was used on a Macbook Air (M1, 2020) running Python 3.9.1 with 8 physical cores and 8 logical processors that could use up to 8 threads.

5.1.1 Model Choice

The first assessment of the solutions for the MRP will investigate the differences between the full MRP formulation given in (3.27) and the reduced version as given in (3.40). To refresh, `MRP` facilitates a heterogeneous vehicle fleet while `reducedMRP` assumes a homogeneous one. In this thesis, there will not be any differences between the vehicles, so we expect the models to produce the same objective value.

To perform this test, we must define the remaining parameters. We set that we have 10 duplicates of each pickup and delivery site and that the fleet size, K, is equal to 3. The vehicles will start in the vertices 1.0, 1.01, and 2.0. Lastly, we set the allowed working time, W to 100 minutes.

By solving `MRP` and `reducedMRP` we get the solutions presented in Figure 5.2 and Figure 5.3. Figure 5.2 presents the optimal routes for both models while Figure 5.3 illustrates the optimal schedules. The objective value of both solutions is $7.01760 \cdot 10^6$, meaning that both models provide equally good solutions, as expected. However, we can see that the suggested routes are not identical. This points to an underlying symmetry in the problem. Because we have included duplicates, many of the edges will have the same exact cost. This leads to some cases, where it does not make a difference for the objective function exactly which edges are traversed. There will be no preference on whether an edge (i_0, j_0) or (i_1, j_1) should be traversed if we are able to service just as many vertices with both choices. As such, there will be symmetry in the

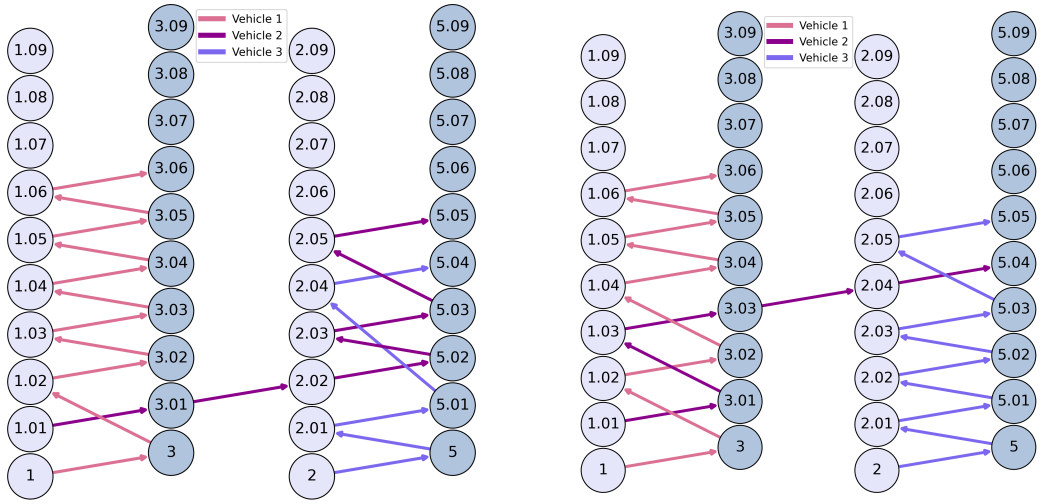
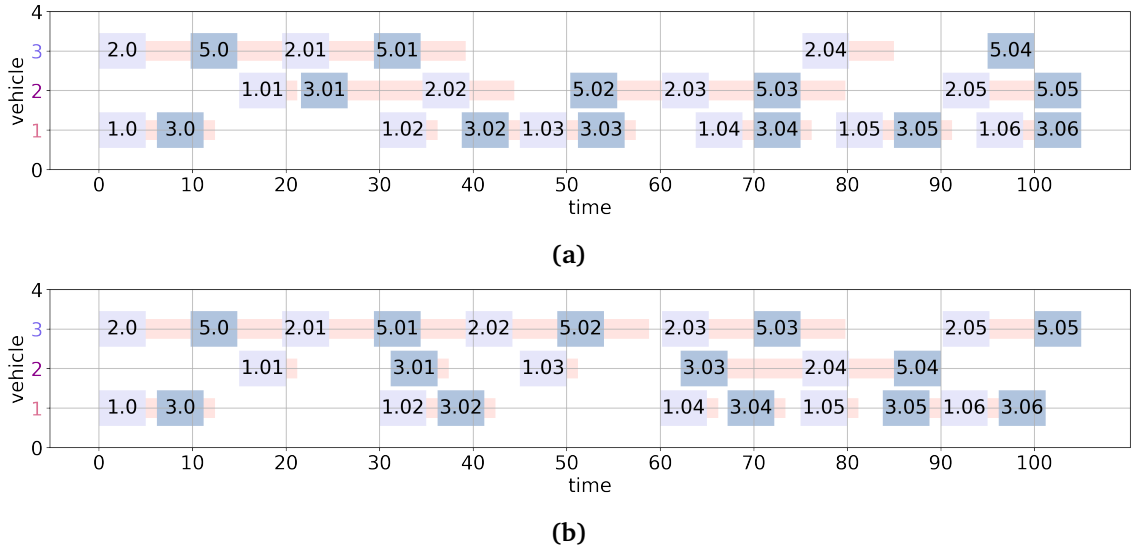


Figure 5.2: The calculated optimal routes for the test case. The purple vertices depict pickup sites while the blue depict delivery sites. The edges between the vertices show the optimal route of each vehicle. **(a)** is the optimal route calculated from the full model, while **(b)** is calculated from the reduced model.

model that can result in multiple solutions with the same objective value. We will see more examples of this later. To further reduce the computation time of the MRP, we could remove this symmetry but then we would have to have more data on how we wish to traverse the graph.

Moreover, `MRP` used 99.54 seconds and `reducedMRP` used 13.88 seconds to find the optimal solution, meaning they have vastly different computation times. This is clearly a trend as seen in Table 5.1. Thus, if the assumption of a homogeneous vehicle fleet is fair, there is a lot of time and computation resources to be saved by utilizing `reducedMRP`. Furthermore, notice that we have a computation time that is quite a lot smaller than W for all cases of W. If the computation time had been larger than, or of equal size to W that would mean we would use more time on planning the schedule than executing it, making the model inadequate.

Next, we also remark that we do not visit all vertices in Figure 5.2. This is a critical aspect of the MRP, as we do not wish the vehicles to be able to visit all vertices within the given W. If we were to do so, that would mean that the solution includes spare time for the drivers. By not visiting all vertices and penalizing every missed vertex,



(a)

(b)

Figure 5.3: The calculated optimal schedule for the test case. The purple and blue blocks describe when the given vehicle on the y-axis is servicing the corresponding pickup and delivery vertex, respectively. The pink stripes depict when the vehicle is travelling between vertices, and where there are no blocks the vehicle is standing still and waiting. **(a)** is the optimal schedule calculated from the full model, while **(b)** is calculated from the reduced model.

W	Duplicates	Computation time of `MRP`	Computation time of `reducedMRP`
100 min	10	100s	14s
150 min	15	463s	21s
200 min	20	1102s	215s

Table 5.1: The computation times of the full MRP given in (3.27), and the reduced MRP, given in (3.40), for varying parameters. All other parameters are held constant.

we ensure that the vehicles traverse the graph as efficiently as possible. However, this means that we need to be vigilant when choosing how many duplicates each vertex should have, as too many slows down the computation time, and too few makes the solution redundant.

Lastly, we also wish to address the scheduled downtime between servicing vertices in Figure 5.3. As we see in Figure 5.3a for example, Vehicle 3 has approximately half an hour of waiting between servicing vertex 5.01 and 2.04. This gap occurred as there are too few jobs in the test case. We can, by definition, not service 2.04 at the same time as 2.03 as they represent the same on-site location. Thus, because we have three vehicles but only two pickup sites, there will be time wasted.

Since the MRP and the reducedMRP obtained the same objective value, we will from here on out only use the reducedMRP from (3.40). This is because as we do not have any data on the vehicles to account for using the MRP, we wish to use the more computationally efficient version.

5.1.2 Size of Fleet

Next, we wish to assess how the size of the vehicle fleet alters the optimal solution and running time. To do so, we keep the problem size fixed with the exception of the fleet size, K. As before, we set that we have $W = 100$, and that we have 10 duplicates of each pickup and delivery site. We wish to test for $K = 1$, $K = 5$, and $K = 10$, so we need also to define the corresponding starting positions. For $K = 1$, we set that the vehicle starts in vertex 1, and for $K = 5$, we use the vertices 1.00-1.02 and 2.00-2.01 as starting positions. Lastly, for $K = 10$, we set 1.00-1.04 and 2.00-2.04 as the starting positions. These were chosen to distribute the vehicles equally between pickup sites.

The resulting optimal solutions for these cases are depicted in Figure 5.4 and Figure 5.5. The resulting objective values were $1.30039 \cdot 10^7$ for $K = 1$, $6.01770 \cdot 10^6$ for $K = 5$, and $6.01350 \cdot 10^6$ for $K = 10$, i.e. the more vehicles the better the solution. This is because one has to drive less back and forth between vertices, and therefore one also has the possibility of servicing more vertices.

In these solutions, we can see the consequence of how we have defined the constraints describing the starting position of each vehicle, (3.32). The same goes for the constraint (3.19) in the full MRP. We made these constraints with the requirement that the vertex a vehicle starts its route must be served before leaving. This, therefore, results in a stunted start for every vehicle and ensures excess downtime at the beginning. We also see that it would be beneficial in a real-life setting not to use all the vehicles as many are mostly waiting around. Therefore, it may be an idea to introduce a negative cost for the edge leading from the starting position to the fictional depot. We investigate this next.

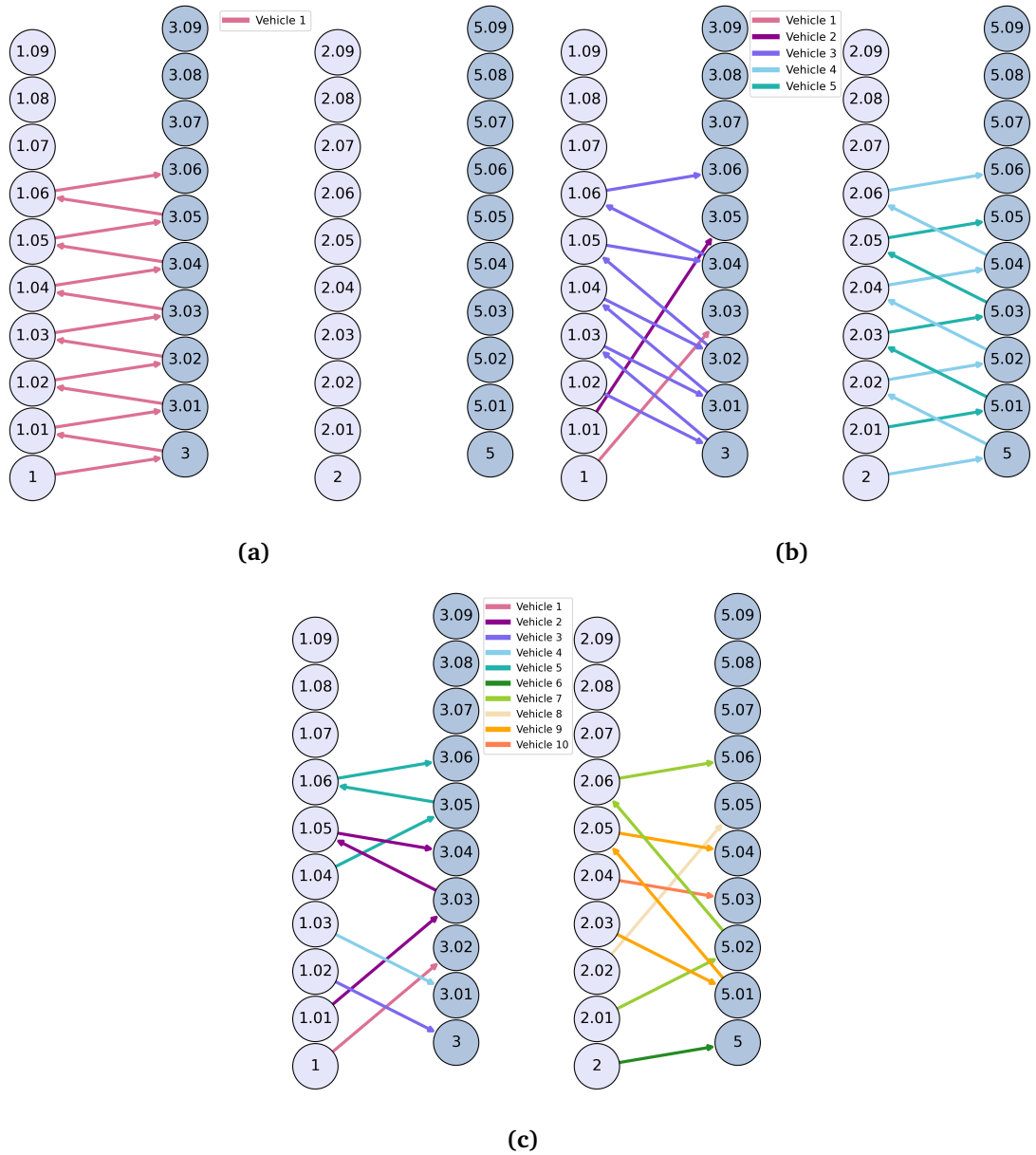


Figure 5.4: The calculated optimal routes for the test case. The purple vertices depict pickup sites while the blue depict delivery sites. The edges between the vertices show the optimal route of each vehicle. **(a)** is the optimal route for $K = 1$, **(b)** is for $K = 5$, and **(c)** is for $K = 10$.

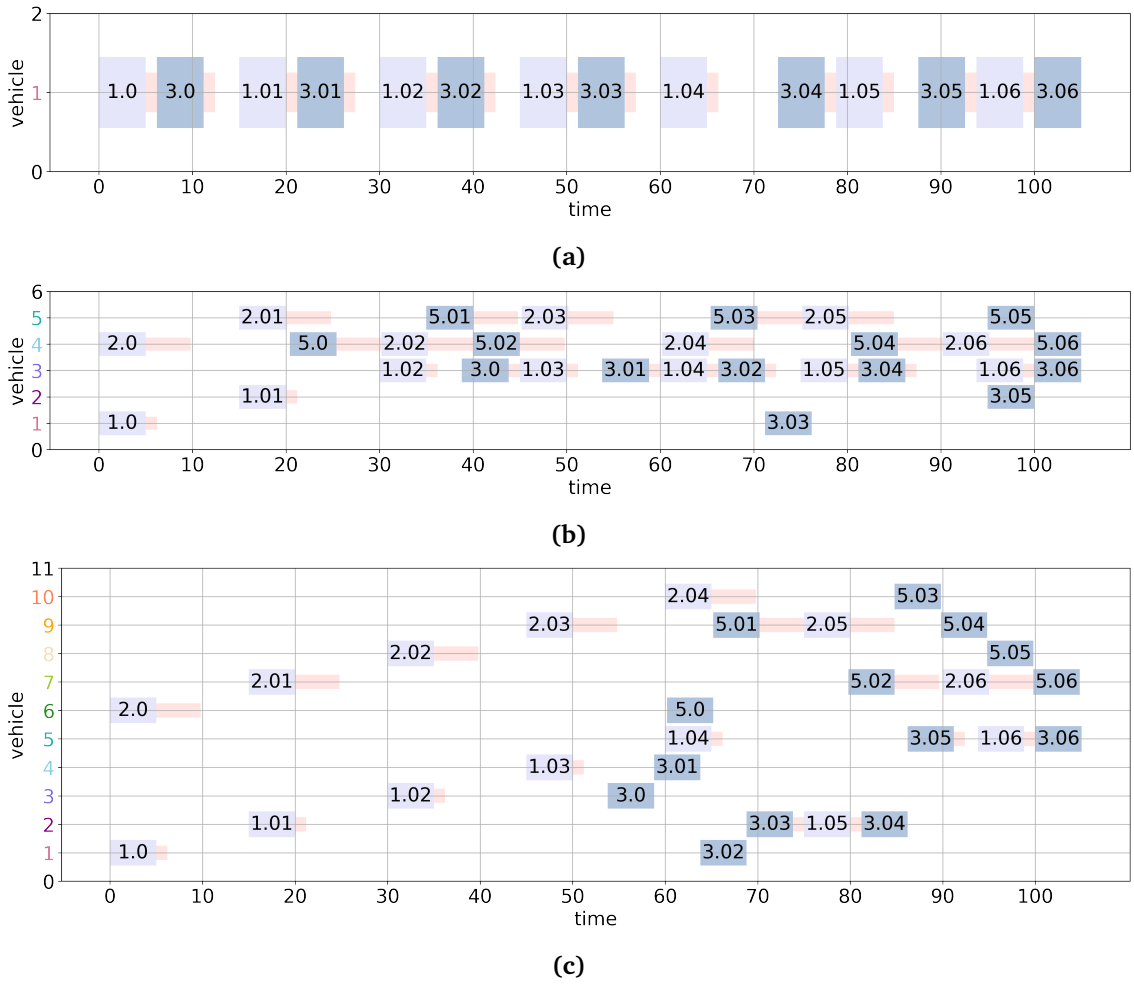
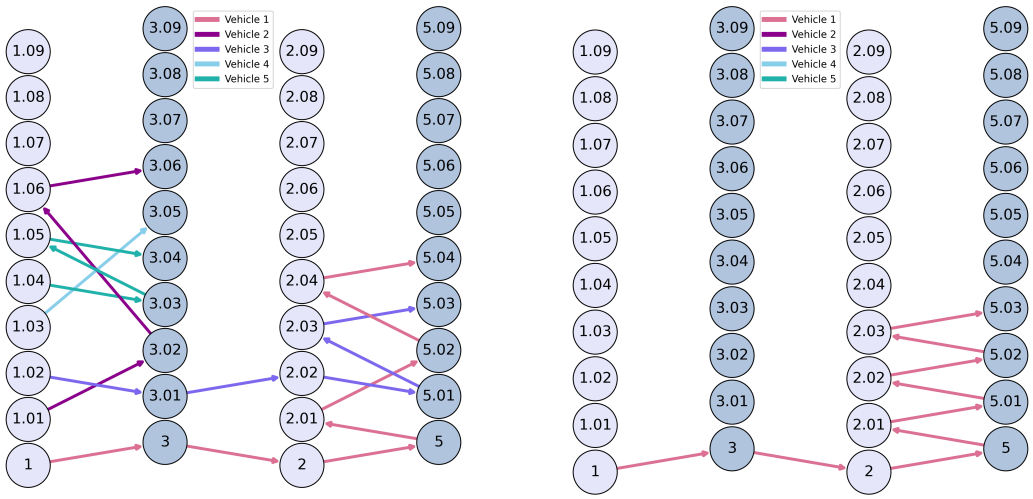


Figure 5.5: The calculated optimal schedule for the test case. The purple and blue blocks describe when the given vehicle on the y-axis is servicing the corresponding pickup and delivery vertex, respectively. The pink stripes depict when the vehicle is travelling between vertices, and where there are no blocks the vehicle is standing still and waiting. **(a)** is the optimal schedule for $K = 1$, **(b)** is for $K = 5$, and **(c)** is for $K = 10$.



(a) **(b)**

Figure 5.6: The calculated optimal routes for the test case. The purple vertices depict pickup sites while the blue depict delivery sites. The edges between the vertices show the optimal route of each vehicle. **(a)** is the optimal route calculated when there is no benefit of returning directly to the depot, while **(b)** sets this benefit to -μ

5.1.3 Negative Costs

We have observed that the formulation of the MRP is not ideal when there are few sites and many vehicles. Thus, we now wish to explore if this can be remedied by introducing a negative cost to the edges leading from the starting position to the fictional end-depot. We set this negative cost to equal the penalty parameter, i.e. $c_{S_k,n+m+1} = -\mu$ for all $k \in K$. This means that we are asserting that it is not beneficial to use an extra vehicle to reach one more vertex, which is a reasonable assumption for real life. This parameter can, however, be adjusted freely based on how strict we want to be. The rest of the parameters are defined as $K = 5$, $W = 100$, and ten duplicates of each site. We set that the starting positions are vertices 1.00-1.05. The results are shown in Figure 5.6 and Figure 5.7.

As seen in the solutions, we use only one of the available vehicles instead of the entire fleet when including a negative cost. However, we can also see a pitfall with how we have defined our starting positions. As we have stated that every vertex can only be visited once and that every vehicle needs a starting position, we prohibit any other

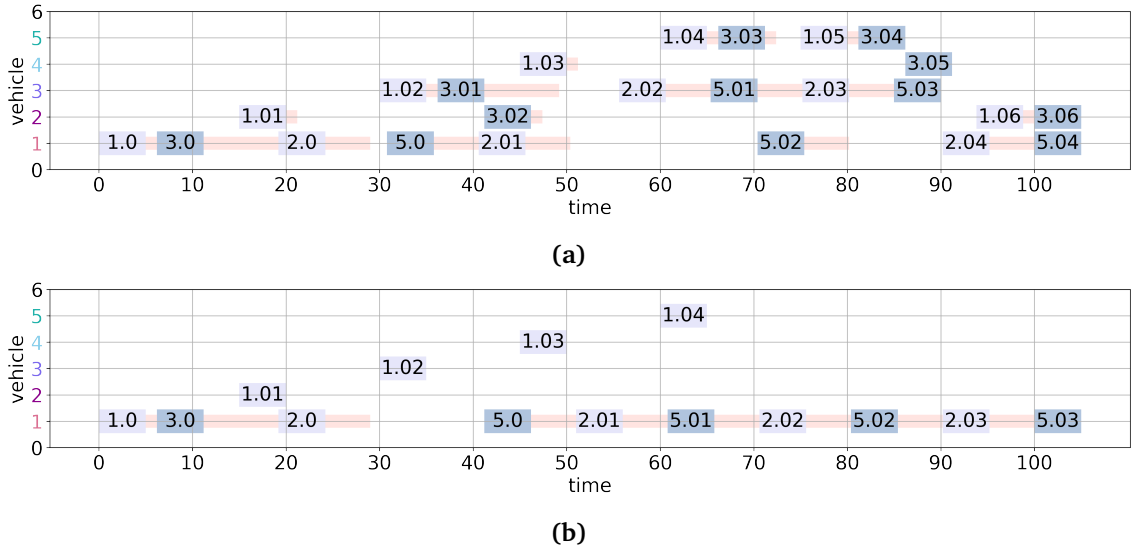


Figure 5.7: The calculated optimal schedule for the test case. The purple and blue blocks describe when the given vehicle on the y-axis is servicing the corresponding pickup and delivery vertex, respectively. The pink stripes depict when the vehicle is travelling between vertices, and where there are no blocks the vehicle is standing still and waiting. **(a)** is the optimal schedule when there is no benefit of returning directly to the depot, while **(b)** sets this benefit to -μ

vehicle from servicing this starting position. This is exemplified in Figure 5.7b, where location 1 is unavailable throughout almost the entire day because of the starting positions. Thus, further work on this model should correct this. A suggestion is shown later in Section 7.1.

5.1.4 Priority Vertices

Prioritized sites are common at real-life construction sites, as often, some processes need to be finished quicker than others. We, therefore, have a priority parameter, p, that determines the importance of each vertex. We wish to investigate how this parameter influences a solution of the MRP. To do so, we use the previously defined test case, with $K = 2$, $W = 100$, and 15 duplicates of each site. Furthermore, we also change the dig rate of the excavators from 10 to 6 and define the starting positions as vertex 1.0 and 2.0. We wish to test how the solution changes if we set $p_i = 5$ for all 2-indexed vertices and compare it to a solution with no prioritized vertices. These

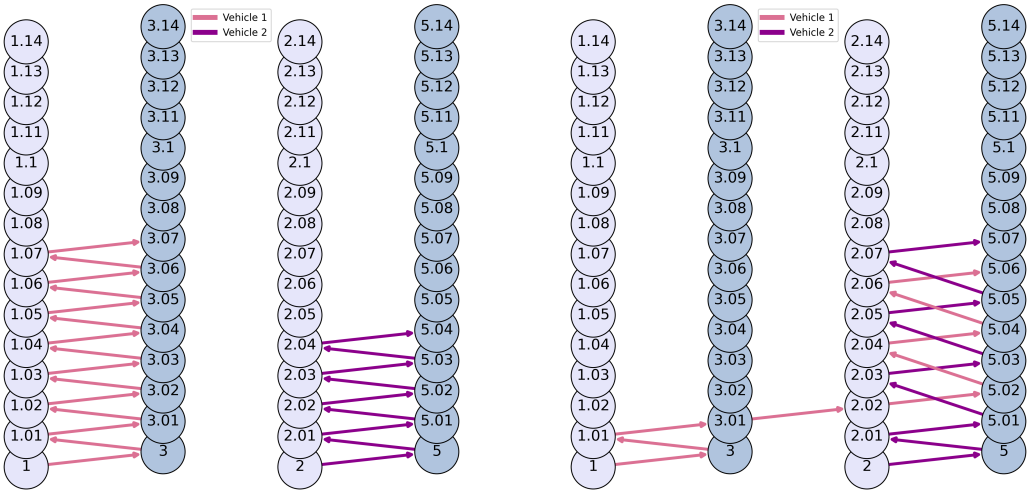
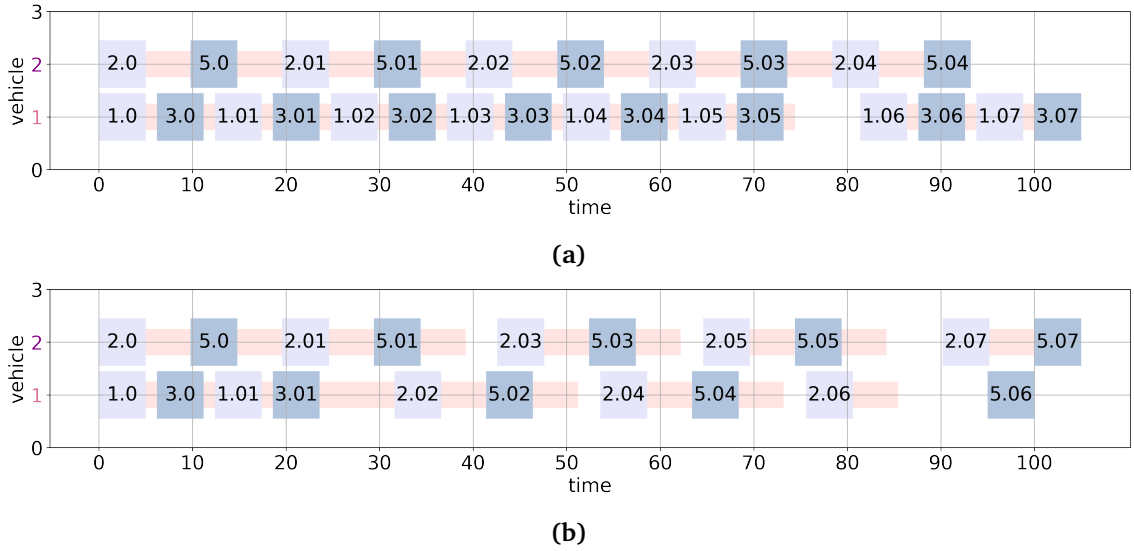


Figure 5.8: The calculated optimal routes for the test case. The purple vertices depict pickup sites while the blue depict delivery sites. The edges between the vertices show the optimal route of each vehicle. **(a)** is the optimal route calculated when there is no prioritized vertices, while **(b)** has a priority of 5 in vertex 2.

solutions are depicted in Figure 5.8 and Figure 5.9.

The objective values of these solutions were $3.82653 \cdot 10^7$ for the case without prioritization and $-2.69803 \cdot 10^7$ for the other. Note that the route's costs and the objective value are not equal as we include the penalizing factor for unvisited vertices in the objective value. The total cost of each solution is 15300 and 18700, respectively, meaning that the solution with no prioritization is more cost and time-efficient. This is because it is free to choose the most cost efficient route. However, since we have set the parameter $p_i = 5$, that means that one of the 2-indexed vertices are worth five of the 1-indexed ones. In other words, we reward the visited prioritized vertices much more to make up for this cost difference. Thus, the objective value of the solution with prioritization is negative as the benefit of visiting these vertices outweighed the cost of the process. We remark that we have to be careful when setting the priority parameter as it far offsets the travel costs.



(a)

(b)

Figure 5.9: The calculated optimal schedule for the test case. The purple and blue blocks describe when the given vehicle on the y-axis is servicing the corresponding pickup and delivery vertex, respectively. The pink stripes depict when the vehicle is travelling between vertices, and where there are no blocks the vehicle is standing still and waiting. **(a)** is the optimal schedule when there is no prioritized vertices, while **(b)** depicts the optimal schedule when vertex 2 has a priority of 5.

5.2 The Mass Relocation Problem with Uncertainty

We have now investigated how the MRP performs, so next, we wish to explore how the different robustness methods behave on a test case. To do so, we need to introduce uncertainty into the model. For the Skanska construction site we will be investigating, we only have data on the uncertainty in travel times d. Therefore, we will only investigate this uncertainty for the test case as well. We present the new discrete uncertainty sets for d_{ij} in Table 5.2.

i/j	**1**	**2**	**3**	**4**	**5**
1	(M)	(16,17,18,36)	(2,3,4,8)	(9,10,11,22)	(14,15,16,32)
2	(16,17,18,36)	(M)	(19,20,21,42)	(6,7,8,16)	(11,12,13,26)
3	(2,3,4,8)	(19,20,21,42)	(M)	(6,7,8,16)	(17,18,19,38)
4	(9,10,11,22)	(6,7,8,16)	(12,13,14,28)	(M)	(4,5,6,12)
5	(14,15,16,32)	(11,12,13,26)	(17,18,19,38)	(4,5,6,12)	(M)

Table 5.2: The discrete uncertainty sets for the travel times d_{ij}.

The uncertainty sets are structured to be similar to the composition of the real data. We have four discrete values for all edges in the graph, where three are similarly valued, and one is much larger. In the real data, there are more values in each set, but they follow a similar pattern. These larger values likely signify that the vehicle stopped for gas or the driver had a lunch break, for example, or a data inaccuracy. The rest of the set is the approximate actual travel time.

To illustrate the robustness methods we use the same test case as before, only now with the presented uncertainty in d. We use the same service rate, $s_i = 5$ for all $i \in V \backslash \{0, n+m+1\}$, and dig rate, $r_i = 10$ for all $i \in P$, as well as $W = 100$, $K = 3$, and the starting positions are the vertices 1, 1.01, and 2. Lastly, we also define that we have ten duplicates of each site and for the Bertsimas-Sim approach and the Removing Outliers Approach (ROA), we set $\xi^d = 0.7$. The results are depicted in Figure 5.10 and Figure 5.11.

The results show the optimal route and schedule given the Soyster approach presented in equation (4.3), the Bertsimas-Sim approach, formulated in Algorithm 1, and lastly the ROA as presented in equation (4.11). To recollect, the Soyster approach uses the maximum value of the uncertainty sets to ensure strict robustness. The Bertsimas-Sim approach assumes worst-case values for the coefficients that have the worst impact on the objective value, within an uncertainty budget, ξ. Lastly, the ROA optimizes the problem based on the maximum value of the coefficients after removing the outliers.

If we look closer at the suggested optimal routes for each robustness method, we can see some of the differences between the approaches. We see that the ROA is the most efficient, as it services the largest number of vertices. The Soyster approach is just behind in terms of performance. This is expected as the Soyster has taken into account all the worst-case outlier values, while the ROA used the second largest coefficients, which are potentially more realistic. The Bertsimas-Sim approach produces an infeasible solution. As seen in Figure 5.11b for the Bertsimas-Sim approach, Vehicle 1 and Vehicle 3 both service a 2-indexed vertex in the time interval between 50 and 60 minutes. This means that there is no time for the excavator placed here to prepare a new truckload of mass, making the solution infeasible. This can be remedied, but would likely result in an even worse solution. One could potentially apply the

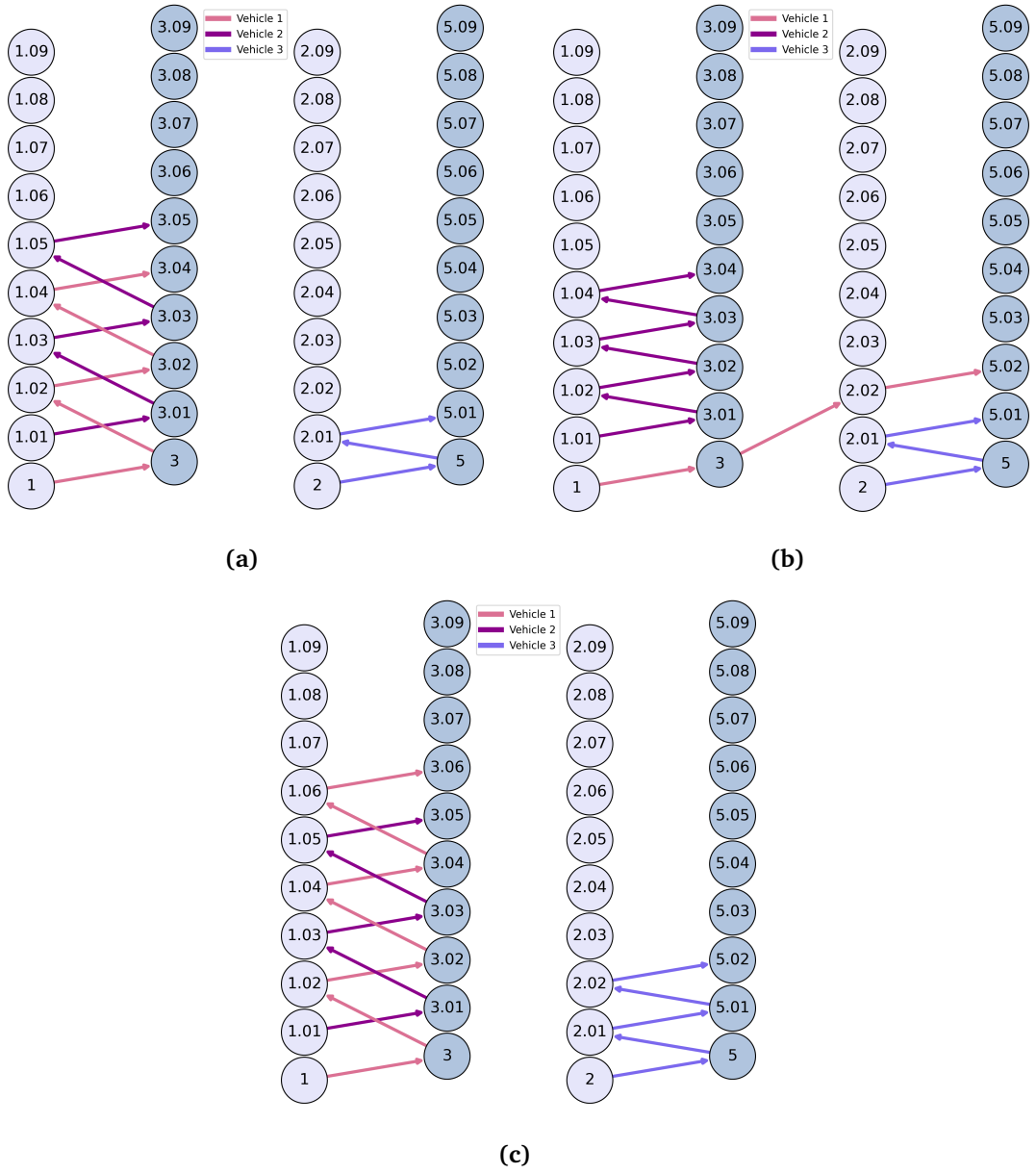


Figure 5.10: The calculated optimal routes for the test case. The purple vertices depict pickup sites while the blue depict delivery sites. The edges between the vertices show the optimal route of each vehicle. **(a)** is the robust route with the Soyster approach, **(b)** is with the Bertsimas-Sim approach, and **(c)** is with the ROA.

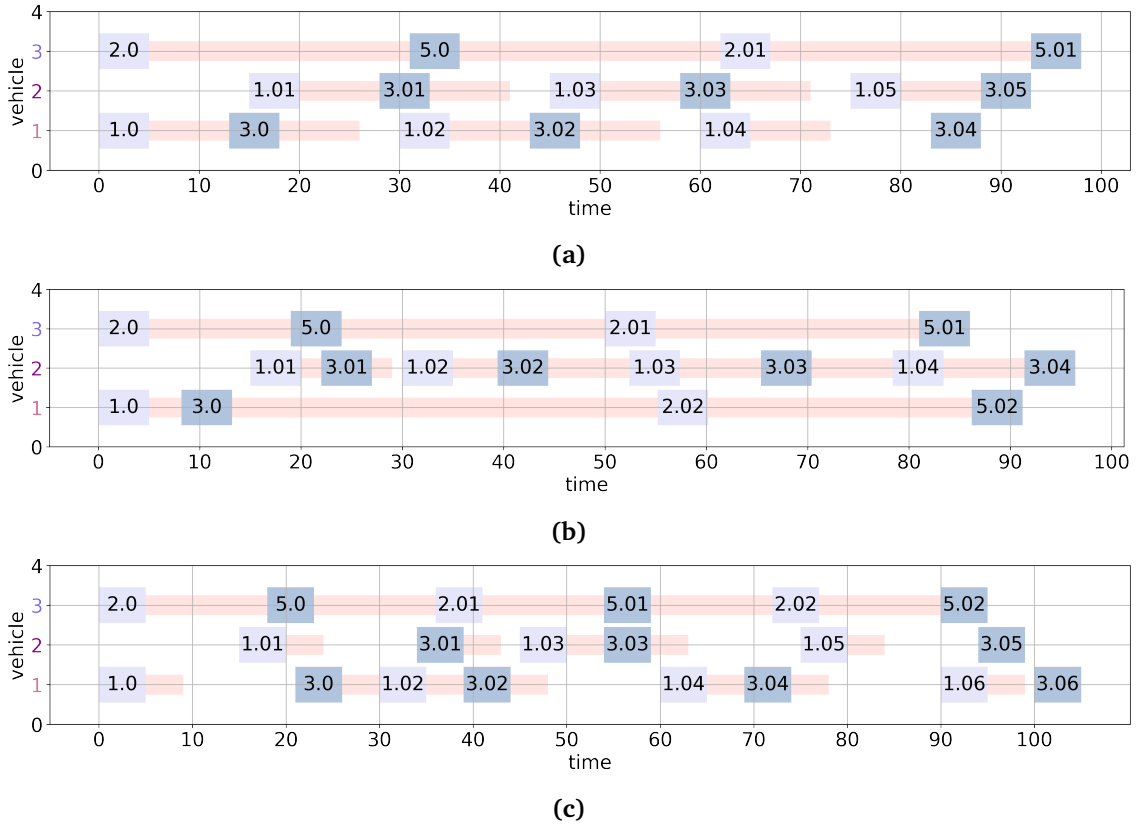


Figure 5.11: The calculated optimal schedule for the test case. The purple and blue blocks describe when the given vehicle on the y-axis is servicing the corresponding pickup and delivery vertex, respectively. The pink stripes depict when the vehicle is travelling between vertices, and where there are no blocks the vehicle is standing still and waiting. **(a)** is the robust schedule with the Soyster approach, **(b)** is with the Bertsimas-Sim approach, and **(c)** is with the ROA.

W	Duplicates	Best case	Soyster	Removing outliers
100 min	10	16	24	20
150 min	15	24	36	28
200 min	20	32*	48*	38*

Table 5.3: The number of unvisited vertices for each feasible approach given different parameters and $\xi = 0.7$ for the ROA. If number is marked with "*", that means that the case was stopped by time limit on computation time, W, i.e. optimal solution was not found.

constraints of the MRP into the Bertsimas-Sim algorithm. This would however result in having to further offset the service times, making the model more inefficient. We therefore deem this model insufficient for calculating the MRPU.

Lastly, we investigate the *price of robustness* for the MRPU. This term was coined by Bertsimas and Sim [5] and is commonly thought of as the sacrifice made by the decision-maker to ensure robustness. In thesis thesis, we analyze the price of robustness by examining the number of unvisited vertices for both the Soyster and the ROA and compare them to the best case scenario for the test case. The comparison is shown in in Table 5.3, where we have modified W and the number of duplicates but kept all other parameters as before. We have not included the Bertsimas-Sim approach as it is not sensible comparing infeasible solutions to feasible ones.

As seen, this table shows that the ROA performs consistently better than the Soyster approach, and they both perform worst than the best-case scenario. However, as the best case solution is unrealistic, we decide that the ROA is best suited for the MRPU, since we can achieve a sufficiently robust solution by fine-tuning the uncertainty budget, ξ. The Soyster is overly cautious with our uncertainty sets and works better for models where the uncertainty is more compounded. Therefore, we will use the ROA for further work with real data.

We also observe that for $W = 200$, none of the instances were able to provide optimal solutions before the computation time surpassed W. This is not ideal, as it means that once the problem has increased to a certain size, it is no longer computationally efficient to solve. We will discuss this closer in the next section. We also remark that when investigating of the differences between `MRP` and `reducedMRP` we were able to find the optimal solution for $W = 200$ and 20 duplicates, as seen in Table 5.1. This is

likely because then the vehicles traveled at higher speeds, meaning that it was easier to pick which vehicles not to service. The exact reasoning should be investigated closer in future work.

Chapter 6

The Mass Relocation Problem with Uncertainty on Real-Life Data

We will now finally investigate how the Mass Relocation Problem with Uncertainty (MRPU) works on real data provided by the construction company Skanska. Section 6.1 summarizes the assumptions we have made about the data, Section 6.2 presents the data and how we have chosen to structure it, and lastly, Section 6.3 discusses the results.

6.1 Assumptions

Before we start solving the MRPU for actual data, we want to describe the assumptions we have made. First, since we use the reduced MRP as given in (3.40), we assume a homogeneous vehicle fleet. This will mean that we assume that all vehicles travel at the same speed and have the same carrying capacity. This will usually not be true in real life, but as we do not have any data on the different vehicles, we will proceed regardless.

Next, we assume that if multiple vehicles start their route at the same position, that it takes less time to wait for the vehicle's turn than to traverse to a different pickup

vertex. This is also not entirely reasonable and should be amended in future work. A suggestion to how is presented in Section 7.1.

Furthermore, we assume that we have constant and independent service times and dig rates. In real life, this is a crude assumption. Weather, for example, will rarely only affect one vertex, thus making the uncertainty in both of these parameters interconnected.

Lastly, we also assume there will be no congestion on the road network. The way we have formatted the graph means that we have not considered that roads can be occupied and thus not available at all times for any vehicle to traverse. If we were to take this into account, we could no longer use a shortest path algorithm and create an edge between each pickup and delivery pair.

6.2 Solving the MRPU

As this thesis is done in collaboration with SINTEF on behalf of the construction company Skanska, we have solved the MRPU on the data gathered from their road construction site in Jevnaker, Norway. The coordinates of the pickup and delivery sites make up the vertices in the graph. The edges are calculated from the coordinates of the paths the vehicles have driven. We transform this information into a graph as depicted in Figure 6.1. The travel times are given as a set of times, where each instance represents the actual duration of a vehicle traversing the edge. As mentioned previously, there is a considerable range of uncertainty in these sets. We can show this with an example. We look closer at the travel times between vertex 1 and vertex 2 in the graph depicted in Figure 6.1 and see that $d_{12} = [203.4s, 207.8s, 221.8s, 229.0s, 310.5s, 600.8s, 2100.0s]$. Notice that there is a significant variation in the travel times as in most cases, it takes approximately 4 minutes to traverse, but it can also take 10 minutes or 35 minutes. This may be due to data inaccuracies or random occurrences such as stopping for vehicle maintenance. However, as this has not been the subject of this thesis, we will assume that all data is a correct portrayal of real life.

We now wish to depict the road network as a graph on which we can solve the MRPU.

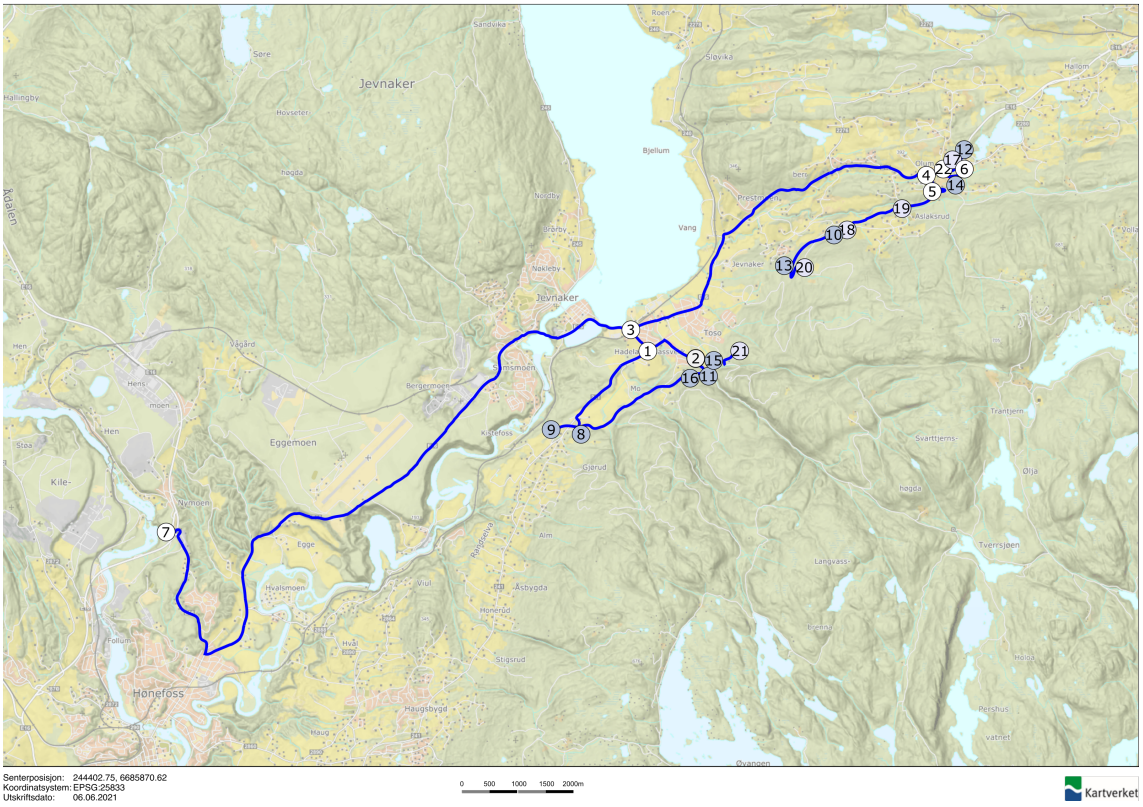
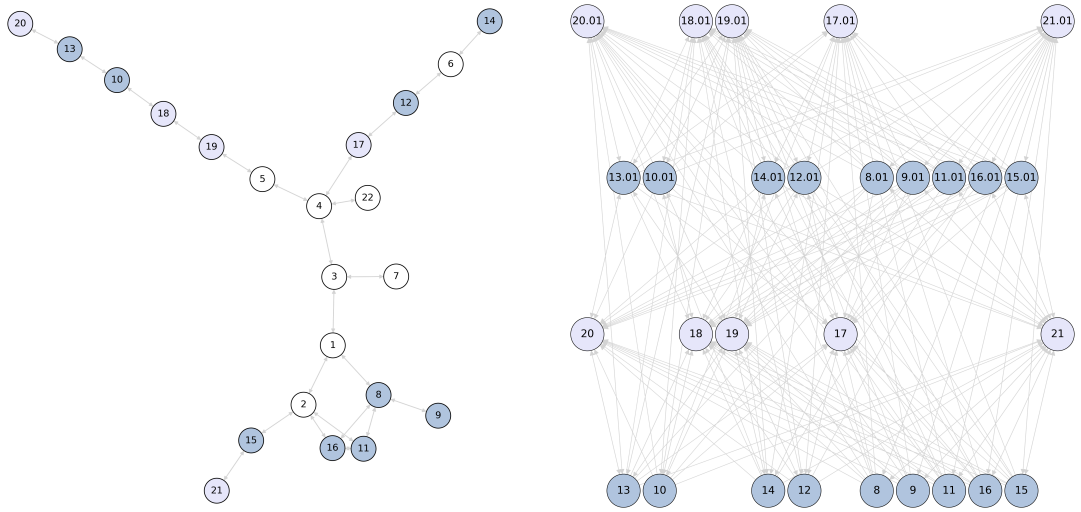


Figure 6.1: The real road network of a construction site, marked with purple pickup locations, blue delivery locations, white crossroads, and blue roads between them, which is drawn on top of a map from ©Kartverket



(a) **(b)**

Figure 6.2: The formatting of the real-life data as a graph. The purple vertices depict pickup sites while the blue depict delivery sites, while the edges between them depict the road network. **(a)** is the actual road network depicted as a graph, while **(b)** illustrates the graph on which we will solve the MRPU with two duplicates per site.

The translation from the graph depiction of the actual construction site into a correctly formatted graph with two duplicates is depicted in Figure 6.2. Previously, we have defined that there exists edges between all pickup and delivery vertices. Here, however, we have paired some pickup and delivery vertices to reduce the complexity of the problem. These pairings are given in Table 6.1.

From vertex:	To vertex:
17	12, 14
18	10
19	10, 12, 13, 14
20	13
21	8, 9, 11, 15, 16

Table 6.1: The only allowed paths out of the pickup vertices after pairing.

As seen in this table, we have only connected vertices that already were close to each other geographically. For example, we pair pickup site 21 with the delivery vertices 8, 9, 11, 15, and 16, as they are clustered together in real life. Notice that we have

not set any restrictions on the edges out of pickup vertices. This ensures that pairing vertices does not give a predefined solution to the MRPU. Ideally, we would only pair vertices if certain types of mass could only be delivered at specific locations. For this case, we pair vertices to reduce the computation time since we do not have any data on what type of mass is being excavated where.

An outline of the implementation of the MRPU with the ROA is given in Algorithm 2.

Algorithm 2 Solving the MRPU

1: Set parameters
2: Collect graph information
3: Define pickup- and delivery sites
4: Define all edges with `Dijkstra`
5: Define travel times of edges given robustness approach
6: Duplicate edges and vertices
7: Pair relevant vertices
8: Solve `MRPU_outliers` as given in (4.11)
9: Collect solutions

To solve the MRPU, we use the same computer as in Chapter 5.

6.3 Results

We do not have data for all parameters in this real-life case, and must therefore choose some ourselves. We set a constant service time of 5 minutes for every vertex, and a constant dig rate of 10 minutes for every pickup vertex. It could be interesting to look at the uncertainty in these parameters, but we will leave it for another time as we do not have any data on this. Furthermore, we set that $K = 5$ where all vehicles start in different pickup vertices, namely vertex 17.0, 18.0, 19.0, 20.0, and 21.0. We also determine that we have no prioritized vertices and that the penalty term, μ, is defined as before as the sum of all edges in the graph. We also set a cost of $-\mu$ on the edges leading from the starting positions to the end-depot to discourage vehicles from being used unnecessarily.

We use the Removing Outliers Approach (ROA) to ensure robustness in the MRPU as

it is the best-suited method for the MRPU, as we have shown previously. In this real-data case, we set $\xi = 0.9$ to give a high likelihood of a feasible solution. However, this should be adjusted more precisely when we have more knowledge about the reasons behind the uncertainty in future work.

6.3.1 Solving for a Half Working Day

The first case we will look at is the calculated route for a half-day. Thus, we set that $W = 360$ minutes, which is six hours. We only look at a half-day as the computation time needed grows exponentially for larger models. We also assume that in real life, we can only accurately predict what will happen for approximately the next six hours before something unexpected happens that throws off the schedule. Furthermore, we need to duplicate each site. Since we want to reduce the problem size as much as possible, we wish to include an as small as possible surplus of duplicates that will never be reached. The amount of duplicates for each site in this graph is specified in Table 6.2.

Vertex:	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Duplicates:	5	5	22	5	20	15	5	5	5	20	22	5	15	10

Table 6.2: The number of duplicates for every vertex in the real case.

These numbers were found by trial and error by running the case for different values and seeing how many duplicates were needed for each site. In a real life setting, this would not be efficient as we would waste a lot of resources on solving test runs. However, a real construction site would probably have more preferences on which sites should be visited and how often, which would suggest the number of necessary duplicates. We would therefore not have to guess blindly.

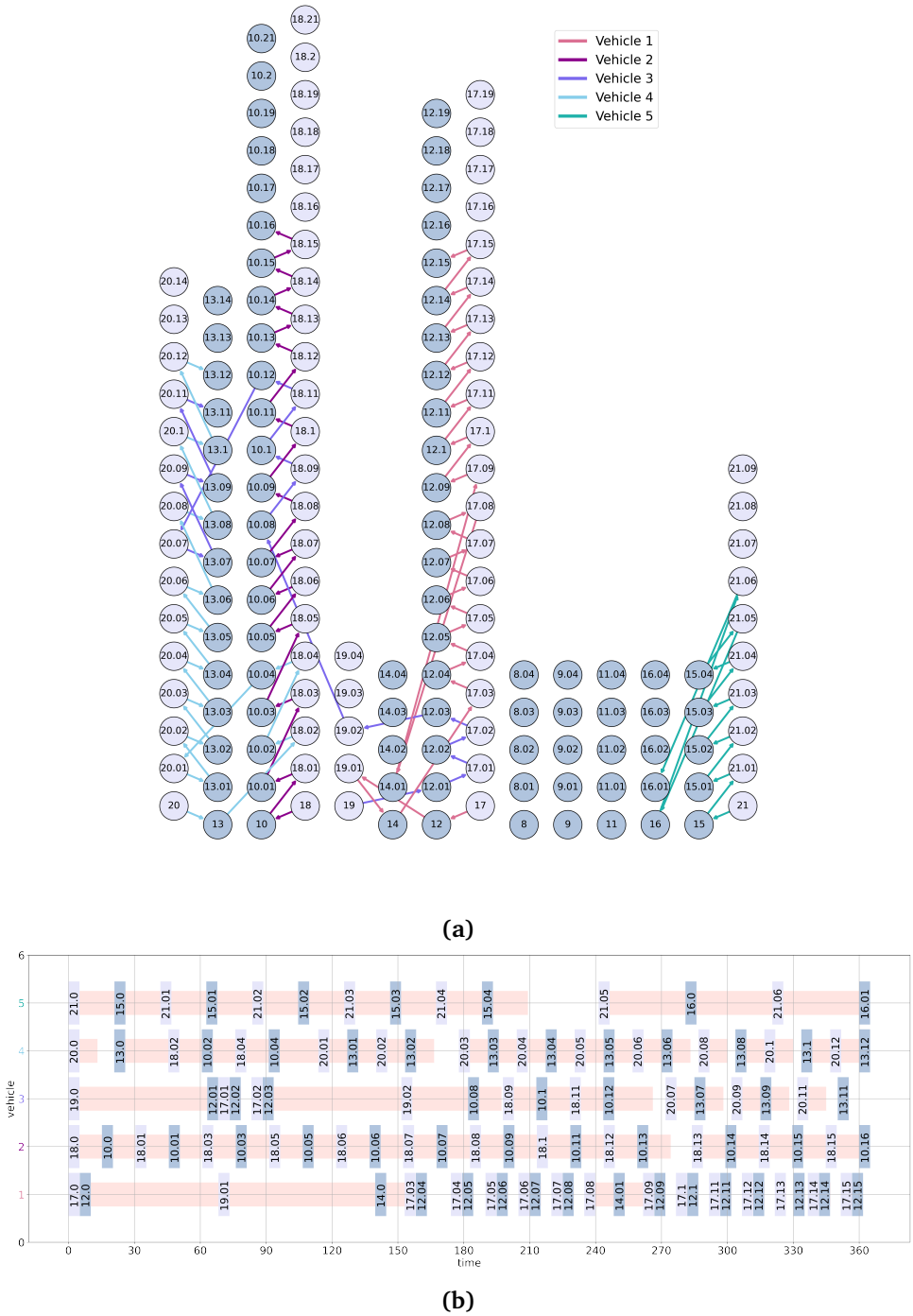
Notice that in this case, we specifically set that pickup vertex 21 has more duplicates than all of its paired delivery vertices. If we set them to equal amounts, we would get only pairing between vertex 21 and 15 as they are the closest to each other. In real life, we could potentially only need five truckloads to a specific delivery vertex. Thus, by modeling it this way, we can also investigate what it means for the solution.

We also set a time limit on the computation time since the solver can potentially run for many days. We set this limit to be W, as the model is not practical if it runs any longer than this. If we wish to solve the MRPU for six-hour intervals, it would be unreasonable for it to take six hours as we would then have to start running the next time interval once the previous ended. This would result in us assume nothing unexpected happened as we would have to determine the starting position of each vehicle.

The following results of this case are shown in Figure 6.3. These solutions are not the guaranteed optimal solutions, as the solver was terminated because it passed the time limit. Despite that, we get a rather interesting solution. We note that the symmetry in the problem plays a prominent role in this solution and may be why the computation is inefficient. We see, for example, that Vehicle 5 traverses between pickup site 15 and the delivery sites 15 and 16. Since we have no preference on when the vehicles should visit the closest vertex, the objective is the same no matter the order in which we service these delivery vertices. This may be the reason it is challenging for the solver to prove the optimality of a solution. The same happens with pickup vertex 16 and the delivery vertices 12 and 14.

Furthermore, the choice that Vehicle 1 should go from vertex 17.08 to vertex 14.01 is rather odd as there is no reason we should not visit the closer vertex 12.09 instead. This is probably due to the fact that this is not the optimal solution. The objective of this solution is $7.516692 \cdot 10^8$ and the penalty factor is $\mu = 44212470$. That means that the total cost of the solution is approximately 57 000, or rather, the vehicles in total drove 57 kilometers during the modeled six hours. Thus, wasting two kilometers on driving to vertex 14 is not critical in the big picture but it is not the purpose of the MRP. We wish to find the absolute shortest allowed path while visiting as many vertices as possible.

The reported optimality gap of this solution was 99.9893 percent when the solver was interrupted. This means that we were far off the optimal solution, and we can conclude that the formulation of the MRPU with the ROA as given in (4.11) for this real-data problem is not sufficient. However, we cannot conclude that the MRPU with the ROA will not be sufficient for other data sets. But this would have to be investigated further.



(a)

(b)

Figure 6.3: The calculated route and schedule for the real data after 6 hours computation time. The purple vertices and blocks represent the pickup sites, while the blue represent the delivery sites. **(a)** is the route suggested for the vehicles, where the edges between the vertices represent the paths for each vehicle.**(b)** is the calculated schedule for each vehicle, where the pink stripes depict travel time for each vehicle, and the white spaces means the vehicle is waiting its turn.

6.3.2 Comparing Computation Time for Different W-Parameters

Since we did not find an optimal solution of a realistic instance for a real-life situation, we will analyze when an optimal solution is no longer able to be found within reasonable time limits. To do so, we will examine the real-life case for different W-parameters and comparing the needed computation time. The results of this analysis are visualized in Table 6.3

W:	60 min	90 min	120 min
Computation time:	0.8s	5.5s	120 min*

Table 6.3: The computation time of a real-life depiction of a construction site with varying total allowed working hours, W. If number is marked with "*", that means that the case was stopped by time limit on computation time, W, i.e. optimal solution was not found.

We can observe that there is a large jump in computation time from $W = 90$ to $W = 120$. For $W = 120$, the solver had to terminate itself as the time limit was surpassed. This is an interesting observation because that means that once we exceed $W = 90$ for this real case, we get too many options for Gurobi to solve efficiently. Thus, we cannot schedule the vehicles for more than 90 minutes at a time and guarantee that we find an optimal solution within the time limit of W. Therefore, it is not surprising that we previously did not find the optimal solution for the realistic depiction of a real-world site for six hours.

However, this is bad news for the formulation of the MRP as it will likely not be efficient enough for any real-world cases with a realistic time horizon. It would not be sufficient to only compute for 90 minute intervals, as we cannot guarantee that it would result in the optimal solution for a full working day. Thus, in future work, one should try to improve the computation time. We discuss this further in Section 7.1.

6.3.3 Comparing Different ξ-Parameters

We also want to investigate how the ROA works with different uncertainty budgets for real-life data. In this case, we will compare $\xi = 0.25$, $\xi = 0.5$, and $\xi = 0.75$ for a smaller version as the same real-life case. We only change W and the number of

duplicates by setting that $W = 60$ and duplicating the vertices as specified in Table 6.4. The results are shown in Figure 6.4 and Figure 6.5.

Vertex:	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Duplicates:	5	5	10	5	10	10	10	5	5	10	10	10	10	10

Table 6.4: The number of duplicates for every vertex in the real case when we are comparing ξ-values.

We observe that the resulting schedules service 32, 30, and 26 vertices within the time limit for $\xi = 0.25$, $\xi = 0.5$, and $\xi = 0.75$, respectively. As expected, the lower the ξ value, the less time is used on traveling between vertices. What is not entirely expected, is that the vehicles service different sets of vertices for the different ξ-parameters. This means that it is not necessarily straightforward that the closest vertex in distance is the wisest to serve next. For example, we see that the vehicles are scheduled to service site 19 three times for $\xi = 0.25$ and $\xi = 0.5$, and only once for $\xi = 0.75$. This is probably due to that it takes less time to travel to vertex 19 for the lower-valued ξ's, than it does to wait for a new truckload to be prepared at the vertex the vehicles are currently placed at. This way, they are able to service more vertices than we do for $\xi = 0.75$, even though they are travelling a larger distance.

Furthermore, we can observe that the travel time between site 12 and site 17 is approximately the same for all investigated ξ-values. Let us look closer at the uncertainty set for this edge. We have 48 instances of a vehicle taking around seven seconds to traverse this edge, and then three instances where it takes respectively 325 seconds, 582 seconds, and 3774 seconds. Thus, we exclude these outliers by not using a ξ-value of more than 0.95, and as such, we get similar travel times for the different cases. This is a testament to the fact that the ROA seems to be very beneficial for the MRPU.

Lastly, we can also notice the vehicles wait around before ending their schedule. Since we have not added minimizing the makespan into the objective function, the optimal solution is not dependent on finishing the jobs quickly. Also, note that since vehicles have to end their routes at a delivery vertex, they will not be able to make use of their remaining time if they cannot reach both a pickup and delivery vertex before the day is passed. This could also explain why Vehicle 5 in our previous case for a half-day, shown in Figure 6.3b, waited around for half an hour at 210 minutes.

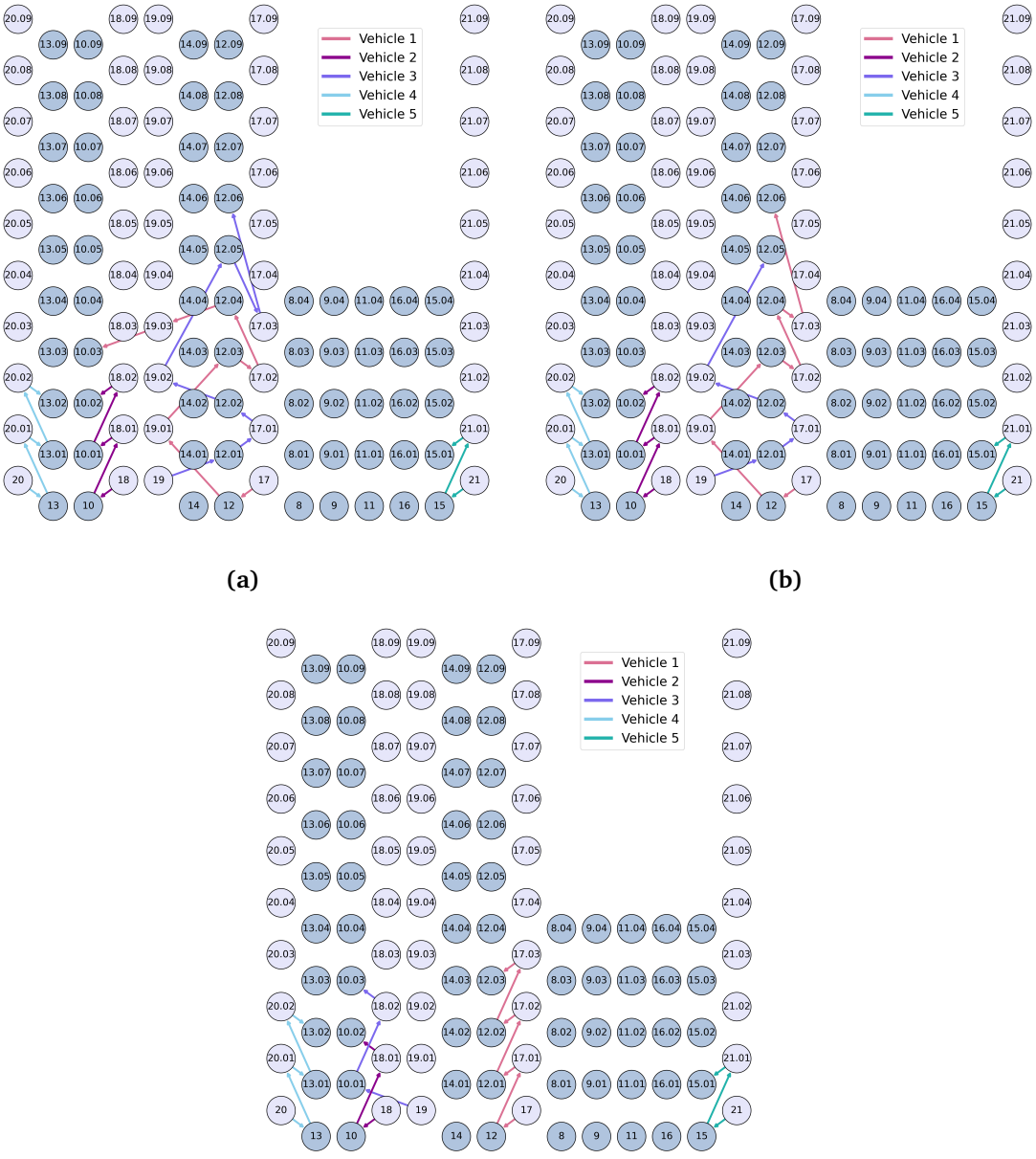
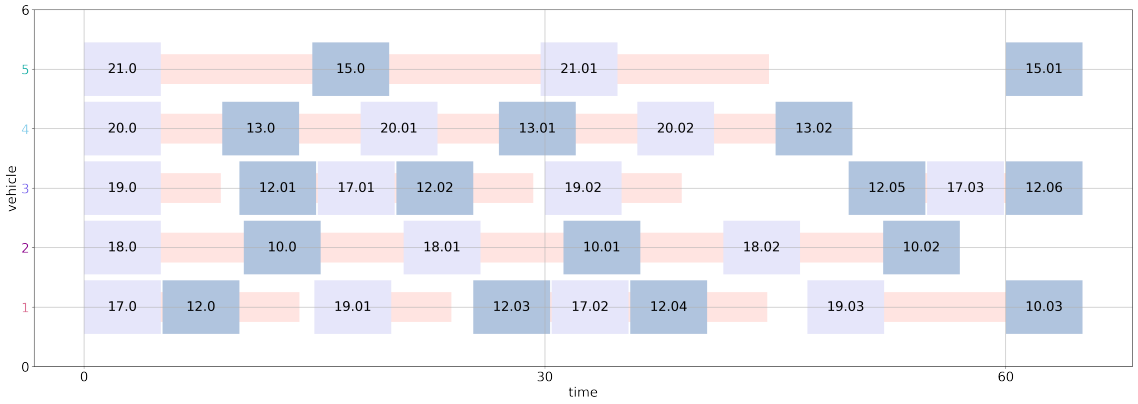
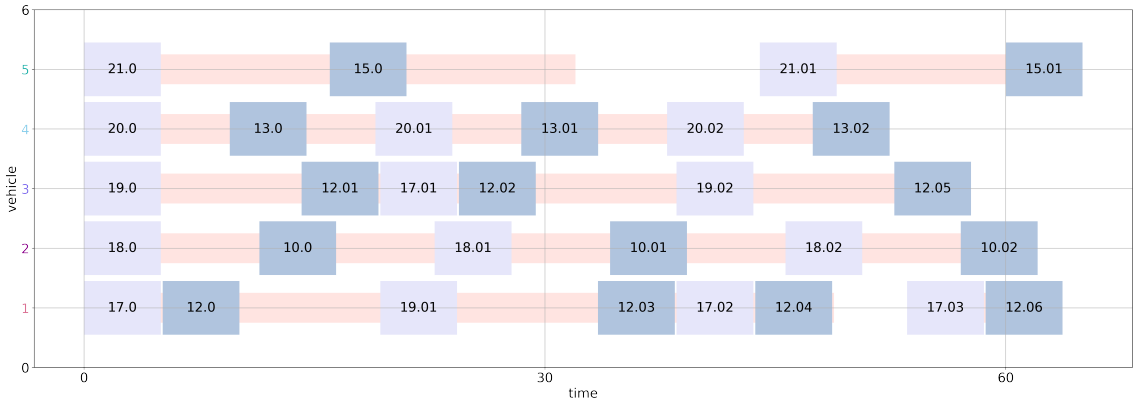


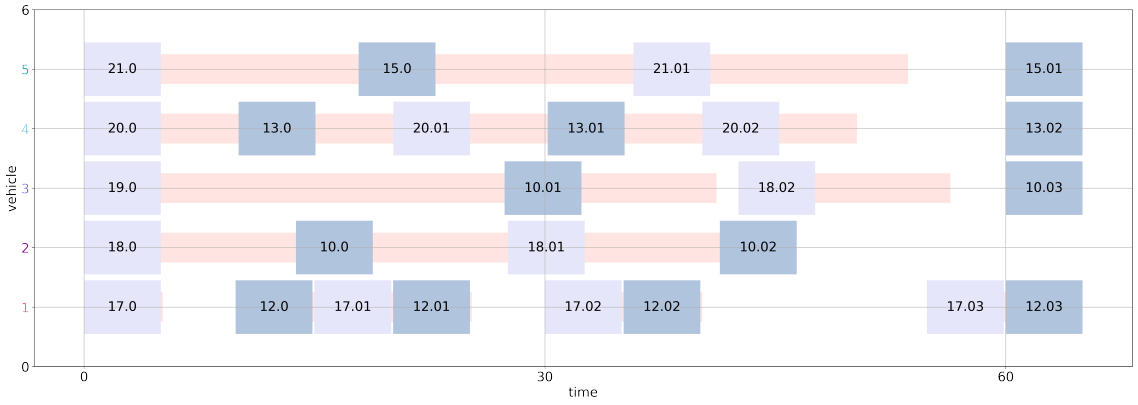
Figure 6.4: The calculated optimal routes for the test case. The purple vertices depict pickup sites while the blue depict delivery sites. The edges between the vertices show the optimal route of each vehicle. **(a)** is the optimal route for $\xi = 0.25$, **(b)** is for $\xi = 0.5$, and **(c)** is for $\xi = 0.75$.



(a)



(b)



(c)

Figure 6.5: The calculated optimal schedule for the test case. The purple and blue blocks describe when the given vehicle on the y-axis is servicing the corresponding pickup and delivery vertex, respectively. The pink stripes depict when the vehicle is travelling between vertices, and where there are no blocks the vehicle is standing still and waiting. **(a)** is the optimal schedule for $\xi = 0.25$, **(b)** is for $\xi = 0.5$, and **(c)** is for $\xi = 0.75$.

Chapter 7

Conclusion

7.1 Future Work

As we have seen, the formulation of the MRP is not optimal. We have taken some crude assumptions, and the computation time for actual data is still too slow since it surpasses W. Therefore, we will review some of the potential improvements to make the formulation and implementation more functional for real-life situations.

7.1.1 Computation Time

First and foremost, we want to reduce the computation time to make the MRP applicable to real life. There are a couple of ways we could do this. The most obvious solution would be to use a computer with more computing power. Another suggestion would be to remove the remaining symmetries in the problem to make the Branch & Bound algorithm more efficient. We could also introduce cuts to tighten the relaxations of the problem.

Lastly, to reduce computation time, we could have implemented some sort of learning algorithm. The data set will rarely change, as a construction company works on the same site for many months and possibly years. Thus, if we construct an algorithm that

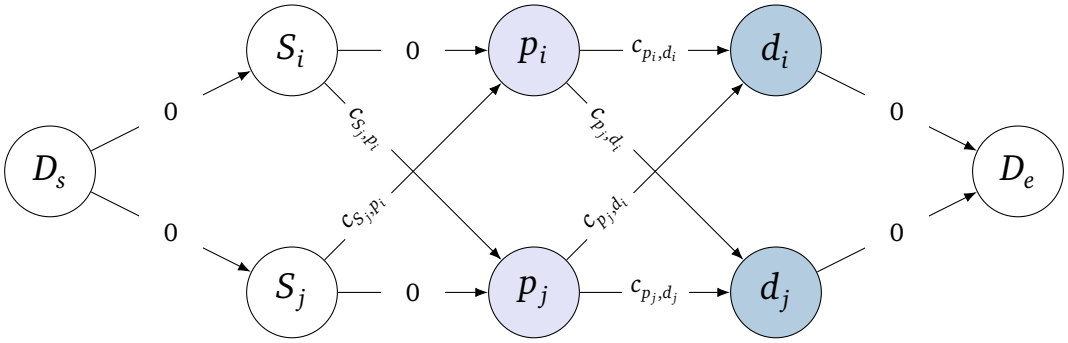


Figure 7.1: An illustration of an alternative for the starting position constraint in the MRP. The vertices D_s and and D_e are the fictional start- and end-depots. Vertices S_i^f and S_j^f are the fictional starting position depots, for starting position p_i and p_j respectively. Lastly, d_i and d_j represent the delivery sites in the graph.

we train on previous optimal solutions, we could produce new ones faster with fewer resources. This would not guarantee optimal solutions, but could be an acceptable alternative if we relax the requirements on solution accuracy.

7.1.2 Assumptions

Future work should not only reduce the computation time but also make better assumptions about the data. As we have mentioned previously, one should improve the formulation of the starting positions. One way to remedy this constraint is to introduce a fictional depot for each of the vehicles. This depot can only be entered from the start-depot, and the edges out of the depot lead to all pickup locations and the end-depot. We set the cost of the edges between the starting depot and the starting location to zero. The rest of the edges cost the distance from the starting position to the other pickup vertices. This way, a starting position would not be categorized as serviced, and thus other vehicles would be able to service it. A simple illustration of the idea is depicted in Figure 7.1.

Furthermore, we should move away from the assumption of a homogeneous fleet of vehicles. We could potentially still use `reducedMRP` formulated in (3.40) as a foundation. We would then have to reintroduce k into the travel time parameter d and reformulate the associated set of constraints. We would also need to adjust the starting

position constraint to take into account vehicle-specific locations. However, this would still assume that all vehicles have the same carrying capacity. Thus, if we wanted to eliminate this assumption, we would have to back to the full model given in (3.27).

Lastly, an accurate formulation of the MRP should include potential congestion. In real life, there will not be unlimited space on every edge in the graph. Some graphs include one-way streets or one-lane streets, and as such, they cannot fit as many vehicles as we may want. To account for this, we need more constraints restraining how the road network can be traversed and more data about the given roads. Alternatively, one could assess if it would be easier to include congestion by not using a VRP as the groundwork for our formulation of the MRP.

7.1.3 Data Sets

Another improvement would be to have better data on the uncertainty in the graph. The range between the smallest and largest element in the potential travel times of an edge is significant in our available data. Since we do not know the reasoning behind this, we cannot know for sure that it is not due to data errors. If it is, then the error would propagate throughout the solution, giving possibly infeasible or inefficient solutions.

Furthermore, if we had more data on what the uncertainty sets contain, we could refine our robustness approaches based on the likelihood of certain travel times. We could potentially even model feasible solutions given the weather forecasts for the specified day. Ideally, we would also include uncertainty to even more parameters to ensure the feasibility of the solution. However, as this data is not available, this will have to be done in future work.

7.2 Concluding Remarks

This thesis has succeeded in formulating a robust variant of the Vehicle Routing Problem (VRP) that models a real world road construction site, the Mass Relocation Problem with Uncertainty (MRPU). With data collected from Skanska, we have been able to make a routing tool that provides good and feasible solutions. Unfortunately, the computation time when solving the MRPU is too large for the formulation to be useful for real-world scenarios. However, we have shown that it could be possible and it is worthy of further exploration. We therefore remain optimistic about this formulation and anticipate large improvements with, potentially, only minor adjustments such as those suggested previously.

Bibliography

[1] A. Agra, M. Christiansen, R. Figueiredo, L. M. Hvattum, M. Poss and C. Requejo, 'The robust vehicle routing problem with time windows,' *Computers & Operations Research*, vol. 40, pp. 856–866, 2013.

[2] R. Anand, D. Aggarwal and V. Kumar, 'A comparative analysis of optimization solvers,' *Journal of Statistics and Management Systems*, vol. 20, no. 4, pp. 623–635, 2017.

[3] A. Ben-Tal and A. Nemirovski, 'Robust solutions of linear programming problems contaminated with uncertain data,' *Mathematical Programming*, vol. 88, pp. 411–424, 2000.

[4] D. Bertsimas and M. Sim, 'Robust discrete optimization and network flows,' *Mathematical Programming*, vol. 98, pp. 49–71, 2003.

[5] D. Bertsimas and M. Sim, 'The price of robustness,' *Operations Research*, vol. 52, pp. 35–53, 2004.

[6] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*. Springer, 1997.

[7] G. Clarke and J. Wright, 'Scheduling of vehicles from a central depot to a number of delivery points,' *Operations Research*, vol. 12, pp. 568–581, 1964.

[8] G. B. Dantzig and J. H. Ramser, 'The truck dispatching problem,' *Management Science*, vol. 6, pp. 1–140, 1959.

[9] Google OR-Tools. (n.d.). 'Penalties and dropping visits,' [Online]. Available: `https://developers.google.com/optimization/routing/penalties` (visited on 30/03/2021).

[10] A. H. Gundersen, M. Johansen, B. S. Kjær, H. Andersson and M. Stålhane, 'Arc routing with precedence constraints: An application to snow plowing operations,' *Lecture Notes in Computer Science*, vol. 10572, 2017.

[11] Gurobi Optimization LLC. (n.d.). 'Gurobi optimizer,' [Online]. Available: `https://www.gurobi.com/products/gurobi-optimizer/` (visited on 08/01/2021).

[12] I. B. Hovi and W. Hansen, 'Logistikkostnader i norske vareleverende bedrifter,' Transportøkonomisk institutt, 2010.

[13] P. Kouvelis and G. Yu, *Robust Discrete Optimization and Its Applications*. Springer, 1997.

[14] C. Lee, K. Lee and S. Park, 'Robust vehicle routing problem with deadlines and travel time/demand uncertainty,' *Journal of the Operational Research Society*, vol. 63, pp. 1294–1306, 2012.

[15] F. Margot, 'Symmetry in integer linear programming,' in *50 years of Integer Linear Programmin 1958-2008: From the Early Years to the State-of-the-Art*, M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi and L. Wolsey, Eds., Springer Berlin Heidelberg, 2010, ch. 17, pp. 647–686.

[16] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 2006.

[17] S. N. Parragh, K. F. Doerner and R. F. Hartl, 'A survey on pickup and delivery problems,' *Journal für Betriebswirtschaft*, vol. 58, pp. 81–117, 2008.

[18] J.-P. Rodrigue, *The Geography of Transport Systems*. Routledge, 2013.

[19] K. L. Rødseth, R. B. Holmen, F. R. Førsund and S. A. Kittelsen, *Effektivitet og produktivitet i norsk veibygging 2007-2016*. Ex ante akademiske forlag, 2019.

[20] Skanska. (n.d.). 'Skanska vil kutte utslipp med kunstig intelligens,' [Online]. Available: `https://www.skanska.no/hvem-vi-er/media/aktuelt/skanska-vil-kutte-utslipp-med-kunstig-intelligense/` (visited on 18/06/2021).

[21] A. L. Soyster, 'Convex programming with set-inclusive constraints and applications to inexact linear programming,' *Operations Research*, vol. 21, pp. 1154–1157, 1973.

[22] N. E. Toklu, 'Matheuristics for robust optimization - application to real-world problems,' Ph.D. dissertation, Università della Svizzera Italiana, 2014.

[23] P. Toth and D. Vigo, *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002.

Appendix

A Implementation of MRP in gurobipy

```
def solve_MRP(c, K, s, r):
    m = gp.Model('fullMRP')

    m.Params.SolutionNumber = 10
    m.Params.TimeLimit = W*60

    # Decision variables
    x = m.addVars(K, Gd.edges(), vtype=GRB.BINARY, name='x')
    t = m.addVars(nnodes, vtype=GRB.CONTINUOUS, lb=0, ub=W, name='t')

    # If we want to warm-start the solver
    #m.update()
    #m.read('previous_solution.sol')

    # Constraints
    # Constraint (3.16)
    m.addConstrs(((gp.quicksum(x[k,i, out_edge[1]] for k in range(K) for
     ↪  out_edge in Gd.out_edges(i)) <= 1) for i in G.nodes()), name =
     ↪  '3.16')

    # Constraint (3.17)
```

```
m.addConstrs(((gp.quicksum(x[k,0,out_edge[1]] for out_edge in
 ↪  Gd.out_edges(0)) == 1) for k in range(K)), name='3.17')

# Constraint (3.18)
m.addConstrs(((gp.quicksum(x[k,in_edge[0],end_depot] for in_edge in
 ↪  Gd.in_edges(end_depot)) == 1) for k in range(K)), name='3.18')

# Constraint (3.19)
m.addConstrs((x[k,0,S[k]] == 1 for k in range(K)), name='3.19')

# Constraint (3.20)
m.addConstrs(((gp.quicksum(x[k,in_edge[0], i] for in_edge in
 ↪  Gd.in_edges(i)) - gp.quicksum(x[k,i, out_edge[1]] for out_edge
 ↪  in Gd.out_edges(i)) == 0) for k in range(K) for i in G.nodes()),
 ↪  name='3.20')

# Constraint (3.21)
for i in G.nodes():
    for out_edge in Gd.out_edges(i):
        if out_edge[1] != end_depot:
            m.addConstr((t[dup_to_ind[i]] + s[dup_to_ind[i]] +
             ↪  d[dup_to_ind[i]][dup_to_ind[out_edge[1]]] -
             ↪  t[dup_to_ind[out_edge[1]]] <= (W + s[dup_to_ind[i]]
             ↪  + d[dup_to_ind[i]][dup_to_ind[out_edge[1]]]) * (1 -
             ↪  x.sum('*',i,out_edge[1]))), name = '3.21')

# Constraint (3.22)
m.addConstrs((t[dup_to_ind[i]] >= (s[dup_to_ind[i]-1] +
 ↪  r[dup_to_ind[i]-1] + (gp.quicksum(x[k,i,out_edge[1]] for
 ↪  out_edge in Gd.out_edges(i) for k in range(K)) - 1) *
 ↪  (s[dup_to_ind[i]-1] + r[dup_to_ind[i]-1]) + t[dup_to_ind[i]-1])
 ↪  for i in pick_duplicates), name = '3.22')
```

```
# Constraint (3.23)
m.addConstrs((t[dup_to_ind[i]] >= (s[dup_to_ind[i]-1] +
 ↪  (gp.quicksum(x[k,i,out_edge[1]] for out_edge in Gd.out_edges(i)
 ↪  for k in range(K)) - 1) * (s[dup_to_ind[i]-1]) +
 ↪  t[dup_to_ind[i]-1]) for i in deli_duplicates), name = '3.23')

# Constraint (3.24) for pickup vertices
for it in range(len(pi)):
    for im in i_pick_duplicates[it]:
        for i in i_pick_duplicates[it][:dup_to_ind[im]-1-
         ↪  pick_lengths[it]]:
            m.addConstr(gp.quicksum(x[k,i,out_edge[1]] for out_edge
             ↪  in G.out_edges(i) for k in range(K)) >=
             ↪  gp.quicksum(x[k,im,out_edge[1]] for out_edge in
             ↪  G.out_edges(im) for k in range(K)), name = '3.24
             ↪  pickup')

# Constraint (3.24) for delivery vertices
for it in range(len(de)):
    for im in i_deli_duplicates[it]:
        for i in i_deli_duplicates[it][:dup_to_ind[im]-1-
         ↪  deli_lengths[it]-len(pick_all)]:
            m.addConstr(gp.quicksum(x[k,in_edge[0],i] for in_edge in
             ↪  Gd.in_edges(i) for k in range(K)) >=
             ↪  gp.quicksum(x[k,in_edge[0],im] for in_edge in
             ↪  Gd.in_edges(im) for k in range(K)), name = '3.24
             ↪  delivery')

# Objective function (3.15)
```

```
    m.setObjective(gp.quicksum(c[dup_to_ind[j]][dup_to_ind[out_edge[1]]]
    ↪  * x[k,j,out_edge[1]] for j in G.nodes() for out_edge in
    ↪  G.out_edges(j) for k in range(K)) +
    ↪  gp.quicksum(c[dup_to_ind[i]+1][-1] * x[i,end_depot] for i in S)
    ↪  + mu * gp.quicksum(1 - gp.quicksum(p[dup_to_ind[i]] *
    ↪  x[k,i,out_edge[1]] for k in range(K) for out_edge in
    ↪  Gd.out_edges(i)) for i in pick_all), GRB.MINIMIZE)

    m.optimize()
    m.write('solution.sol')

    return m
```

B Implementation of `reducedMRP` in gurobipy

```
def solve_reducedMRP(c, K, s, r):
    m = gp.Model('reducedMRP')

    m.Params.SolutionNumber = 10
    m.Params.TimeLimit = W*60

    # Decision variables
    x = m.addVars(Gd.edges(), vtype=GRB.BINARY, name='x')
    t = m.addVars(nnodes, vtype=GRB.CONTINUOUS, lb=0, ub=W, name='t')

    # If we want to warm-start the solver
    #m.update()
    #m.read('previous_solution.sol')

    # Constraints
    # Constraint (3.29)
    m.addConstrs(((gp.quicksum(x[i, out_edge[1]] for out_edge in
     ↪  Gd.out_edges(i)) <= 1) for i in G.nodes()), name='3.29')

    # Constraint (3.30)
    m.addConstr(((gp.quicksum(x[0,out_edge[1]] for out_edge in
     ↪  Gd.out_edges(0)) == K)), name='3.30')

    # Constraint (3.31)
    m.addConstr(((gp.quicksum(x[in_edge[0],end_depot] for in_edge in
     ↪  Gd.in_edges(end_depot)) == K)), name='3.31')

    # Constraint (3.32)
    m.addConstrs((x[0,S[k]] == 1 for k in range(K)), name='3.32')

    # Constraint (3.33)
```

```
m.addConstrs(((gp.quicksum(x[in_edge[0], i] for in_edge in
 ↪  Gd.in_edges(i)) - gp.quicksum(x[i, out_edge[1]] for out_edge in
 ↪  Gd.out_edges(i)) == 0) for i in G.nodes()), name='3.33')

# Constraint (3.34)
for i in G.nodes():
    for out_edge in Gd.out_edges(i):
        if out_edge[1] != end_depot:
            m.addConstr((t[dup_to_ind[i]] + s[dup_to_ind[i]] +
             ↪  d[dup_to_ind[i]][dup_to_ind[out_edge[1]]]  -
             ↪  t[dup_to_ind[out_edge[1]]] <= (W + s[dup_to_ind[i]]
             ↪  + d[dup_to_ind[i]][dup_to_ind[out_edge[1]]]) * (1 -
             ↪  x[i,out_edge[1]])), name = '3.34')

# Constraint (3.35)
m.addConstrs((t[dup_to_ind[i]] >= (s[dup_to_ind[i]-1] +
 ↪  r[dup_to_ind[i]-1] + (gp.quicksum(x[i,out_edge[1]] for out_edge
 ↪  in Gd.out_edges(i)) - 1) * (s[dup_to_ind[i]-1] +
 ↪  r[dup_to_ind[i]-1]) + t[dup_to_ind[i]-1]) for i in
 ↪  pick_duplicates), name = '3.35')

# Constraint (3.36)
m.addConstrs((t[dup_to_ind[i]] >= (s[dup_to_ind[i]-1] +
 ↪  (gp.quicksum(x[i,out_edge[1]] for out_edge in Gd.out_edges(i)) -
 ↪  1) * (s[dup_to_ind[i]-1]) + t[dup_to_ind[i]-1]) for i in
 ↪  deli_duplicates), name = '3.36')

# Constraint (3.37) for pickup vertices
for it in range(len(pi)):
    for im in i_pick_duplicates[it]:
        for i in i_pick_duplicates[it][:dup_to_ind[im]-1-
         ↪  pick_lengths[it]-len(deli_all)]:
```

```
                m.addConstr(gp.quicksum(x[i,out_edge[1]] for out_edge in
                ↪  G.out_edges(i)) >= gp.quicksum(x[im,out_edge[1]] for
                ↪  out_edge in G.out_edges(im)), name = '3.37 pickup')

    # Constraint (3.37) for delivery vertices
    for it in range(len(de)):
        for im in i_deli_duplicates[it]:
            for i in i_deli_duplicates[it][:dup_to_ind[im]-1-
            ↪  deli_lengths[it]]:
                m.addConstr(gp.quicksum(x[in_edge[0],i] for in_edge in
                ↪  Gd.in_edges(i)) >= gp.quicksum(x[in_edge[0],im] for
                ↪  in_edge in Gd.in_edges(im)), name = '3.37 delivery')

    # Objective function (3.28)
    m.setObjective(gp.quicksum(c[dup_to_ind[j]][dup_to_ind[out_edge[1]]]
    ↪  * x[j,out_edge[1]] for j in G.nodes() for out_edge in
    ↪  G.out_edges(j)) + gp.quicksum(c[dup_to_ind[i]][-1] *
    ↪  x[i,end_depot] for i in S) + mu * gp.quicksum(1 -
    ↪  gp.quicksum(p[dup_to_ind[i]] * x[i,out_edge[1]] for out_edge in
    ↪  Gd.out_edges(i)) for i in pick_all), GRB.MINIMIZE))

    m.optimize()
    m.write('solution.sol')

    return m
```

C Gurobi Logs Excerpts

All Gurobi log excerpts for the cases I have run in this thesis are to find at:

https://github.com/ellafj/Repo-Masters-Thesis/tree/main/Gurobi-Log-Excerpts

Ella Frederika Johnsen

The Mass Relocation Problem with Uncertainty

NTNU
Norwegian University of
Science and Technology