Master's thesis

2022

Camilla Idina Jensen Elvebakken

Master's thesis

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

NTNU
Norwegian University of
Science and Technology

Camilla Idina Jensen Elvebakken

Resource-Constrained Project Scheduling for the Mass Relocation Problem

February 2022

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of
Science and Technology

Resource-Constrained Project Scheduling for the Mass Relocation Problem

Camilla Idina Jensen Elvebakken

Applied Physics and Mathematics
Submission date: February 2022
Supervisor: Markus Grasmair, NTNU
Co-supervisor: Torkel Haufmann, SINTEF

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abstract

Construction machinery is responsible for one fifth of the greenhouse gas emissions from the construction industry in Norway, and spends up to 40 % of the workday idle with the engine running due to conflicting schedules. Reducing idle time by project scheduling and route optimization is thus an important step towards creating a more efficient and sustainable construction process.

This thesis is concerned with the relocation of mass on a road construction site. We aim at finding optimal schedules for mass pickup and delivery activities, given a fleet of available dump trucks. We formulate the problem as an extension of the Resource-Constrained Project Scheduling Problem (RCPSP), with the objective of maximizing the number of pickups and deliveries completed within a fixed time horizon. Due to the intractability of the RCPSP formulation, finding optimal schedules for projects with many activities is computationally infeasible. Thus, we develop several inexact methods to schedule the pickup and delivery activities more efficiently. We introduce a location mapping to narrow the solution space of the RCPSP instances, and create a schedule concatenation algorithm to find feasible schedules with long time horizons by merging shorter plans. In addition, we develop a greedy scheduling heuristic to build schedules iteratively by adding new activities one by one until infeasibility is proven. We solve the RCPSP instances using the optimization tool Gurobi and with a custom implementation of Benders decomposition. A simulation study and a real-world case study show promising results for the inexact methods.

Sammendrag

Anleggsmaskiner står for en femtedel av klimagassutslippene i bygg- og anleggsbransjen i Norge, og tilbringer opptil 40 % av arbeidsdagen på tomgang som følge av venting og dårlig planlegging. Bruk av prosjektplanlegging og ruteoptimering for å redusere tomgangskjøring er dermed et viktig steg mot å skape en mer effektiv og bærekraftig byggeprosess.

Denne masteroppgaven handler om transportering av masse på anleggsplasser. Vi tar sikte på å finne optimale tidsplaner for henting og levering av masse mellom lokasjoner med et begrenset antall tilgjengelige dumpere. Vi formulerer optimeringsproblemet som en utvidelse av Resource-Constrained Project Scheduling-problemet (RCPSP), med mål om å maksimere mengden masse transportert innen en gitt tidshorisont. På grunn av den kompliserte RCPSP-formuleringen er det beregningsmessig krevende å finne optimale tidsplaner for prosjekter over lange tidsrom. Vi utvikler derfor flere ikke-eksakte metoder for å planlegge hente- og leveringsaktivitetene mer effektivt. Vi introdusere en lokasjonsparing for å redusere løsningsrommet til probleminstansene, og utvikler en plansammenslåingsalgoritme for å raskt kunne finne lange tidsplaner ved å slå sammen kortere planer. Vi utvikler også en grådig planleggingsheuristikk for å finne tidsplaner iterativt ved å legge til nye aktiviteter en etter en. Vi løser optimeringsproblemene ved hjelp av optimeringsverktøyet Gurobi, og med en egen implementasjon av Benders dekomponeringsalgoritme. En simuleringsstudie og en casestudie med ekte data viser lovende resultater for de ikke-eksakte metodene.

Preface

This thesis is a result of my work in the subject *TMA4900 Industrial Mathematics, Master's Thesis* as a student at the Norwegian University for Science and Technology (NTNU). The thesis is written in cooperation with Sintef Digital and is part of the project *Datadrevet anleggsplass* which aims at developing a solution for automatic real-time management of construction machinery at road construction sites. I would like to direct a thank you to my supervisors, Markus Grasmair (NTNU) and Torkel Haufmann (Sintef), for many helpful meetings and discussions. I would also like to thank Silius Mortensønn Vandeskog for his support and encouragements.

Camilla Idina Jensen Elvebakken
February 2022
Trondheim

Table of Contents

1 Introduction **1**

2 Mass Relocation Problem **7**

3 Resource-Constrained Project Scheduling Problem **13**

3.1 General problem formulation 14

3.2 Mathematical formulations 16

3.2.1 Time-indexed formulation 17

3.2.2 Schedule-based formulation 18

3.3 Modifications 20

3.3.1 Multiple modes 21

3.3.2 Time lags 21

3.3.3 Setup times 22

3.4 Solution methods 23

4 Mass Relocation RCPSP **25**

4.1 Modelling the MRP 25

4.2 Full model 30

4.3 Reduced model 34

4.4 Flipped model 38

4.4.1 Priorities 42

5 Benders decomposition **45**

5.1 Standard Benders decomposition 45

5.2 Benders decomposition for model 1 and model 2 51

5.3 Algorithmic improvements 54

5.3.1 Feasibility cuts 54
5.3.2 Combinatorial cuts 55
5.3.3 Optimality cuts 57
5.3.4 Lazy constraints 58
5.3.5 Other improvements 59

6 Inexact methods **61**
6.1 Location mapping 62
6.2 Schedule concatenation 67
6.3 Greedy scheduling heuristic 71

7 Simulation study **77**
7.1 Model features 77
7.2 Benders decomposition 84
7.3 Inexact methods 90

8 Case study: Skanska construction site **105**
8.1 Data 105
8.2 Schedules without priorities 109
8.3 Schedules with priorities 114

9 Closing remarks **119**

Bibliography **123**

Abbreviations

LP	Linear program
MIP	Mixed-integer program
MILP	Mixed-integer linear program
MRP	Mass Relocation Problem
VRP	Vehicle Routing Problem
RCPSP	Resource-Constrained Project Scheduling Problem
AON	Activity-on-node
BD	Benders decomposition
MP	Benders master problem
SP	Benders subproblem
DSP	Dual of Benders subproblem
IIS	Irreducible inconsistent system
MIS	Minimal infeasible subsystem
IQR	Interquartile range

Chapter 1

Introduction

The construction industry in Norway plays an important role in providing infrastructure, creating employment, and generating value. In 2016, the industry had a gross product of NOK 182.2 billion and employed 227 400 persons [1]. Alarmingly, the construction industry is also responsible for about 15 % of Norway's total greenhouse gas emissions [2]. It is estimated that about one fifth of the emissions come from the construction machinery alone and that these can spend up to 40 % of their time idle as a consequence of conflicting schedules [3], where idle time is the time spent inactive with the engine still running. Additionally, it is approximated that 70 % of the cost of road construction projects are related to fuel and machine operations [3]. Reducing the idle time of construction machinery by project scheduling and route optimization is thus an important step towards creating a more efficient and sustainable construction process.

The use of operations research for project management has grown rapidly in recent years as a response to the need for efficient project scheduling, also in the construction industry [4, 5]. Project management involves the planning and scheduling of activities to reach goals associated with project performance, cost, and duration while using resources efficiently. Today, project management is essential for businesses to stay competitive, and efficient planning can have both economic and environmental impact. With the current availability of computing power and advanced algorithms, mathematical methods for project scheduling have become a vital research area [5, 4, 6].

In this thesis, we are concerned with the specific problem of optimizing the relocation of mass on a road construction site, henceforth named the Mass Relocation Problem (MRP). The movement of mass is an essential preparatory component to flatten the land before construction. The MRP involves routing dump trucks between loading and unloading sites with the aim of moving mass as quickly and efficiently as possible, given a fixed time horizon. Today, the routing of dump trucks is determined on-site based on short-term goals and convenience. This can lead to excessive work, unnecessary waiting, and extended project durations. Using optimization methods to plan the movement of dump trucks and schedule activities can increase project productivity and provide valuable insight into mathematical programming methods for vehicle routing and scheduling.

The Mass Relocation Problem is part of a project aimed at using data-driven methods to create a more efficient and sustainable construction industry. The project, aptly named *Datadrevet anleggsplass* (eng: *Datadriven construction site*), is a collaboration between Skanska, Sintef, Volvo, and Ditio aimed at developing a solution for real-time management of construction machinery to streamline the road construction process [7]. The project has a goal of cutting at least 10 % of the emissions from the construction industry by use of machine learning and route optimization and describes itself as a front-runner when it comes to using artificial intelligence for construction machinery [3]. Finding an efficient solution to the MRP can be an important contribution toward the ultimate goal of automatic real-time machine routing.

A previous attempt at solving the MRP with exact methods was implemented in the master's thesis *The Mass Relocation Problem with Uncertainty* by Ella Johnsen [8] using the Vehicle Routing Problem (VRP) [9]. The approach aimed to find the optimal routes of dump trucks between pickup and delivery locations, with the objective of minimizing the cost of travelling. Uncertainty in the travelling times was accounted for with different robustness techniques. The resulting MRP formulation managed to provide feasible routes and schedules but became too computationally demanding for large real-world scenarios. The slowness of the VRP method was attributed to symmetries in the problem formulation, in addition to the difficulty of providing reasonable estimates of the number of activities for which there was time to complete.

Instead of using the VRP, we model the Mass Relocation Problem as an extension of the Resource-Constrained Project Scheduling Problem (RCPSP). The RCPSP is a general problem formulation suitable for solving a variety of scheduling problems, and several books [10, 4, 6] and surveys [11, 12, 13] treat the problem and its extensions thoroughly. The objective of the RCPSP is to provide a schedule of activities with varying durations, taking into account precedence relations between the activities and resource constraints. With this formulation, we shift the focus of the MRP from vehicle routing to explicit scheduling of pickup and delivery activities, and the objective of the MRP translates into completing as many activities as possible within a time limit. The scheduling approach allows for maximizing the number of activities we can complete, which contributes to reducing the idle time of the dump trucks while optimizing the mass relocation. The MRP is modelled by several extensions of the RCPSP formulation, namely multiple modes, time lags, and setup times.

The Resource-Constrained Project Scheduling Problem is $\mathcal{NP}$-hard in the strong sense [10]. Due to the intractability of the formulation and the symmetries that arise when modelling, solving the MRP exactly with the RCPSP can be very time-consuming when dealing with many activities. To solve the Mass Relocation Problem more efficiently, we develop several inexact solution methods tailored to find feasible schedules quickly. To limit the solution space of the problem instances, pickup and delivery locations are paired ahead of time. This allows us to redefine an activity as a pickup and a corresponding delivery, thus reducing the number of binary variables in the problems. An algorithm for schedule concatenation is developed to merge schedules while preserving as many activities as possible, thus providing an efficient tool for quickly building feasible schedules with long time horizons. A greedy scheduling heuristic is also created by using the exact mathematical formulation of the RCPSP within a heuristic framework. Schedules are made by iteratively adding activities in a greedy manner until infeasibility is proven or a predefined time limit is reached. By prioritizing each activity based on its urgency, the heuristic can ensure that the most time-sensitive activities are given precedence in the resulting schedules.

The RCPSP instances are solved using a mixed-integer programming (MIP) solver implemented in the commercial optimization tool Gurobi [14]. We also implement a solver based on Benders decomposition [15]. Benders decomposition is a decomposition algorithm that takes advantage of a mixed-integer linear program's (MILP) structure and splits it into smaller independent subproblems. These simpler subproblems are iteratively solved to produce an exact solution to the original MILP. To make our implementation efficient enough to compete with the MIP solver in Gurobi, several algorithmic improvements to the decomposition algorithm are discussed and implemented.

A simulation study is carried out to test the models developed with the RCPSP formulation using both Gurobi's MIP solver and our implementation of Benders decomposition. A case study using real-world data from Skanska is also conducted in order to consider the practicability of the scheduling methods. Results show that the implementation of Benders decomposition is not able to solve MRP instances more efficiently than the Gurobi MIP solver. However, schedules produced with the inexact scheduling methods perform better than schedules approximated with the exact solution methods considering computation time, vehicle idle time, and number of activities included. The RCPSP provides a flexible starting point for modelling the Mass Relocation Problem, but the exact mathematical formulation is challenging to solve to optimality. Thus, going forward, the focus should be on incorporating the mathematical model into a heuristic framework.

The remainder of the thesis is structured as follows. In Chapter 2 the Mass Relocation Problem is properly defined and discussed, and an example problem and solution is presented. In Chapter 3 we introduce the Resource-Constrained Project Scheduling Problem formulation and explore the variants and extensions which are relevant when modelling the MRP. In Chapter 4 the MRP is formulated in terms of the RCPSP, and two separate models are developed; model 1 with the objective of minimizing the project duration given a fixed number of activities, and model 2 with the objective of maximizing the number of activities within a fixed project makespan. In Chapter 5 we propose the use of Benders decomposition for solving the RCPSP instances and discuss algorithmic enhancements necessary for an efficient implementation. Chapter 6 proposes the use of several inexact methods to find feasible schedules more efficiently. The models and methods are tested in a simulation study in Chapter 7, and on real-world data in Chapter 8. Our results and conclusions are summed up in Chapter 9.

Chapter 2

Mass Relocation Problem

The movement of mass is an essential component of the road construction process and involves routing dump trucks between mass loading and unloading sites. Currently, routing is determined on-site based on short-term goals and convenience, resulting in delayed projects and unnecessary idle time for the construction machinery. To streamline operations, routing and scheduling optimization using data-driven methods can be beneficial. The idea of using route optimization was explored in the master's thesis by Ella Johnsen [8], and the problem of mass relocation was suitably named the Mass Relocation Problem (MRP). In this thesis, we change the wording of the MRP to free it from the specific implementation by Johnsen and formally define it as:

Definition 1. *The Mass Relocation Problem is the problem of scheduling the movement of mass between loading and unloading sites with a fleet of dump trucks, given a fixed time horizon. The objective is to maximize the amount of mass moved while minimizing the idle time of the dump trucks.*

The Mass Relocation Problem encompasses two separate but intertwined problems. We need to find both the optimal travel patterns of the dump trucks between the pickup and delivery locations and the optimal schedule of the specific pickup and delivery activities. The MRP can therefore be approached by both routing and scheduling techniques.

When modelling the MRP, we consider the scheduling of activities at pickup and delivery locations. Mass is excavated and loaded into dump trucks at the pickup locations and unloaded and dumped at the delivery locations. Thus, dump trucks must alternatively travel between the pickup and delivery locations to move the mass. On real construction sites, the pickup and delivery locations can include different types of mass, like sand, dirt, and gravel. However, in this preliminary approach to solving the MRP, we do not consider the implications of different mass types.

Each pickup and delivery location has an associated service time which relates to the time it takes for a dump truck to load or unload mass. A preparation time is also associated with each pickup location to account for the time it takes to dig up new mass for a pickup. As the dump trucks have to be loaded by an excavator at the pickup locations, we impose that only one vehicle can visit these locations at a time. For the delivery locations, the capacity for simultaneous deliveries will differ from location to location. To simplify the modelling, we assume that only one vehicle can visit the delivery locations at a time as well. This means that the individual activities at all pickup locations must be separated by at least the service and preparation time, while activities at delivery locations must be separated by the service time. When solving the MRP, we make no impositions on where the dump trucks begin or end their workday, as long as they respect the road network on the construction site during their travel. We assume that there are no one-way or one-lane roads, and we ignore the possibility of queues and road congestion. In this thesis, we also assume that the fleet of dump trucks has the same capacity, as was assumed in [8], but allow the vehicles to operate at different speeds. We are usually interested in solving the Mass Relocation Problem for a full or half workday, equivalent to 12 or 6 hours.

The outline of the Mass Relocation Problem given above gives a very simplified version of the real problem. On real construction sites, different dump trucks can have different capacities. Hence, the number of pickup and delivery activities needed would vary based on which dump trucks move the mass, complicating the MRP significantly. We have also assumed that only one vehicle can service a location at a time, but this can vary from site

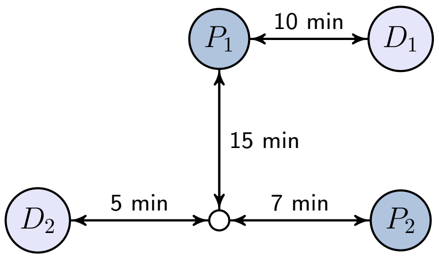


Figure 2.1: Example of a simple construction site to illustrate the Mass Relocation Problem. The colored nodes represent the pickup (P_1, P_2) and delivery (D_1, D_2) locations. The edges represent the legal travel patterns between the locations. The white node illustrates a road intersection.

to site; the number of simultaneous pickups at a location depends on the number of excavators present, and the number of simultaneous deliveries depends on the size of the specific delivery locations. The simplification that we can have no road congestion, queues, one-way or single-lane roads is also unlikely and can cause considerable delays if not accounted for. Letting the dump trucks choose freely where to both start and end their journeys is also unrealistic, especially for half-day schedules where there might be long distances between where the vehicles finish in one schedule and start in another. Lastly, the assumption that we need a preparation time before every individual pickup at a location is also improbable. A more natural way to model this would be to include a preparation of, for example, four loads of mass at a time. Once an efficient way to model and solve the simplified Mass Relocation Problem is obtained, the more complicating aspects of the problem can be considered.

To give an example of an MRP instance, we use the simple construction site shown in Figure 2.1. The colored nodes show the pickup and delivery locations, while the edges show the network of roads. With the assumption that each vehicle has the same capacity, the objective of the MRP reduces to maximizing the number of pickup and delivery activities possible within a fixed time horizon. Given two dump trucks operating at the same speed

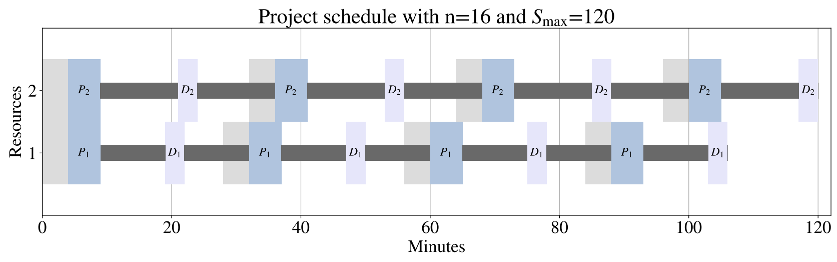


Figure 2.2: Gantt chart of a solution to the Mass Relocation Problem on the construction site shown in Figure 2.1, given a project duration of 120 minutes. The y-axis shows which activities are processed by which vehicles, and the x-axis shows the timing of each activity.

and a time horizon of 120 minutes, an example of an optimal schedule of pickup and delivery activities can be seen in Figure 2.2. We have assumed that each pickup and delivery takes 5 and 3 minutes, respectively, regardless of location and dump truck used and that the preparation of mass by an excavator takes 4 minutes.

The schedule is given as a Gantt chart and shows what activities are processed by the two dump trucks at different times. The colored areas indicate the duration of the pickup and delivery activities, while the dark grey areas represent the time spent driving between the locations. The light grey areas show the mass preparation time at each pickup location. The two dump trucks can complete a total of eight pickups and eight deliveries within the project makespan. Vehicle 1 drives between P_1 and D_1, while vehicle 2 drives between P_2 and D_2. This constructed example makes it easy to verify that the provided schedule is a feasible and optimal solution to the MRP. Solving the Mass Relocation Problem in more involved examples with many locations, dump trucks, and long scheduling horizons can be very challenging.

The previous approach to modelling the MRP by Johnsen was formulated using the Vehicle Routing Problem (VRP) [8]. We propose a new model formulation using the Resource-Constraint Project Scheduling Problem (RCPSP). Instead of minimizing the cost associated with the routing, the RCPSP allows us to focus on scheduling a given number of activities as efficiently as possible, given precedence and resource constraints.

Chapter 3

Resource-Constrained Project Scheduling Problem

The Resource-Constrained Project Scheduling Problem (RCPSP) is a general problem formulation suitable for solving a variety of scheduling problems. The objective of the RCPSP is to provide a schedule of a set of activities with varying durations, taking into account precedence relations and resource constraints. The key difference between the RCPSP and other well-known scheduling approaches, like the Critical Path Method [5], is the presence of limited resource capacities. This makes the RCPSP one of the most intractable scheduling problems in practice [10]. Because of the challenging nature of the problem, as well as its industrial relevance, solving and modelling the RCPSP and its extensions has become an important research area. Several books [10, 6, 4] and surveys [11, 12, 13] treat the problem and its variations thoroughly.

This chapter is concerned with the general RCPSP formulation and the extensions necessary to model the Mass Relocation Problem presented in Chapter 2. In Section 3.1 the classical Resource-Constrained Project Scheduling Problem is defined, and in Section 3.2 two different mathematical formulations to model the RCPSP as a mixed-integer linear program (MILP) are introduced. Section 3.3 explores three different extensions of the general RCPSP formulation, namely multiple modes, time lags, and setup times. Lastly, Section 3.4 discusses the computational complexity of the RCPSP and exact solution methods.

3.1 General problem formulation

The standard Resource-Constrained Project Scheduling Problem can be formulated as a combinatorial optimization problem and is defined by a set of activities, precedence relations, and resource capacity constraints [10]. The goal of the RCPSP is to find a feasible schedule of the activities which is optimal in regard to some objective. The n activities we want to schedule are denoted by $A_1, \ldots, A_n$. We introduce two dummy activities, A_0 and A_{n+1}, to represent the start and end of the schedule, and denote the set of all activities by $V = \{A_0, \ldots, A_{n+1}\}$. Each activity A_i is given a duration d_i, defined as the time needed to complete the activity. The duration of the dummy activities is defined to be zero. The activities are linked by precedence relations that define in what order they must be completed. The precedence relations can be represented by an activity-on-node (AON) graph $G(E, V)$ where the nodes correspond to the activities and the edges to the precedence relations [10]. The pair $(A_i, A_j) \in E$ indicates that activity A_i must precede A_j in a feasible schedule. All non-dummy activities are preceded by the source A_0, while the dummy activity A_{n+1} can only start once all other activities are completed, such that (A_0, A_i) and $(A_i, A_{n+1}) \in E \quad \forall\ A_i \in V \setminus \{A_0, A_{n+1}\}$. To complete the activities we have K renewable resources, defined by the set $\mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_K\}$. The resources are renewable in the sense that a resource allocated to an activity is unavailable while the activity is being processed but freed at full capacity once the activity is finished. The availability of resource $\mathcal{R}_k$ is denoted by B_k, while its demand for activity A_i is denoted by b_{ik}. The demand represents how much of resource $\mathcal{R}_k$ is needed to service an activity [10].

A schedule is usually defined by a vector $S \in \mathbb{R}^{n+2}$, where each coordinate $S_i \geq 0$ denotes the start time of activity A_i [10]. We define the start time of dummy activity A_0 as $S_0 = 0$, and the start time of A_{n+1} as the end of the project. In many applications, the objective of the RCPSP is chosen such as to minimize the makespan of the project. This is equivalent to minimizing the start time of the last dummy activity as this activity succeeds all others. Other objectives also exist, like minimizing the tardiness of activities or the cost of resources [4].

A feasible schedule must respect the precedence and resource constraints imposed by the precedence graph, resource capacities, and demands. The general precedence constraints are usually of the form [10]:

$$S_j - S_i \geq d_i \qquad \forall\, (A_i, A_j) \in E. \tag{3.1}$$

The constraint states that the start time of activity A_j must be greater or equal to the start time and duration of activity A_i, if A_i precedes A_j. The general resource constraint is given as:

$$\sum_{A_i \in V_t} b_{ik} \leq B_k \qquad \forall\, \mathcal{R}_k \in \mathcal{R}, \quad \forall\, t \geq 0, \tag{3.2}$$

where $V_t = \{A_i \in V \mid S_i \leq t < S_i + d_i\}$ represents the set of activities in process at time t [10]. The constraint simply states that the sum of demands for all activities processed at a time t, given a fixed resource $\mathcal{R}_k$, cannot surpass the availability of the resource, B_k. With the two general constraints we define a feasible schedule [16]:

Definition 2. *A feasible schedule S of a set of activities V is given as a schedule that respects the precedence constraints* (3.1) *and resource constraints* (3.2).

Choosing the objective of the RCPSP as to minimize the total project duration, we also define an optimal schedule:

Definition 3. *An optimal schedule S^* of a set of activities V is a feasible schedule S with the minimal makespan S_{n+1}.*

Depending on the characteristics of the activities and resources, several optimal schedules might exist.

Generally, we prefer the schedules with the smallest sum of start times:

$$\sum_{i=0}^{n+1} S_i. \tag{3.3}$$

This is usually called an *active* schedule and refers to a schedule where none of the activities can be globally left-shifted [16]. This means that all the activities are completed as soon as possible and that no rearrangement of the activities can decrease the start times S_i. An active schedule is guaranteed to minimize the idle time of the resources between the processing of the activities, but might be expensive to compute. Instead we settle for a *semi-active* schedule. This is a feasible schedule where none of the activities can be locally left-shifted, meaning that the start time of an activity cannot be decreased without changing the start time of any other activity [16]. A semi-active schedule can be found in a post-processing step by simply shifting all activities to the left while ensuring that the time constraints are still respected.

3.2 Mathematical formulations

We restrict our attention to mathematical formulations of the RCPSP to mixed-integer linear programs. Formulating an MILP with constraints (3.1) and (3.2) directly becomes difficult because of the set V_t, which can not trivially be incorporated into a linear constraint. We must therefore find alternative formulations to express the resource constraints. Several MILP formulations exist in the literature, and they are generally divided into two classes based on their decision variables [10]. The *time-indexed* formulations assign resources to activities at each time unit based on a time discretization. In contrast, the *schedule-based* formulations focus on the ordering of the activities in relation to each other. It has been shown that when exact commercial solvers are used, no model class is superior to the other when used on a wide variety of problems [17]. Therefore, an appropriate mathematical formulation must be chosen based on the specific problem characteristics.

Below we present both the time-indexed formulation of Pritsker et al. [18] and the schedule-based formulation of Artigues et al. [19]. Additional noteworthy MILP formulations can be found in [4]. For the remainder of the thesis we shorten the mathematical notation introduced in Section 3.1 and simply denote activities A_i and resources $\mathcal{R}_k$ by their indices i and k.

3.2.1 Time-indexed formulation

The time-indexed formulations are based on an early model by Pritsker et al. in 1969 [18]. Given a fixed planning horizon, T, the decision variables y_{it} are defined as Boolean variables equal to 1 if activity i starts at time $t \in \mathcal{T} = \{0, \dots, T\}$, and 0 otherwise [10]. Thus, we have the following relation:

$$S_i = \sum_{t=0}^{T} t \cdot y_{it} \qquad \forall\, i \in V, \tag{3.4}$$

and can define the MILP as [4]:

$$\begin{aligned}
\min \quad & \sum_{t=0}^{T} t \cdot y_{(n+1)t} && && (3.5)\\
\text{s.t.} \quad & \sum_{t=0}^{T} y_{it} = 1, && \forall\, i \in V, && (3.6)\\
& \sum_{t=0}^{T} t \cdot (y_{jt} - y_{it}) \geq d_i, && \forall\, (i,j) \in E, && (3.7)\\
& \sum_{i \in V \setminus \{0,n+1\}} b_{ik} \sum_{\tau = t - d_i + 1}^{t} y_{i\tau} \leq B_k, && \forall\, k \in \mathcal{R},\ \forall\, t \in \mathcal{T}, && (3.8)\\
& y_{it} \in \{0,1\}, && \forall\, i \in V,\ \forall\, t \in \mathcal{T}. && (3.9)
\end{aligned}$$

The objective (3.5) is similar to the one defined in Section 3.1, and aims at minimizing the makespan of the project. The first constraint (3.6) imposes that each activity must be started exactly once over the planning horizon T. Constraint (3.7) imposes the precedence relations in a similar manner to the precedence constraint defined in (3.1) and states that an activity can only be processed after the completion of the preceding activities. Constraint (3.8) makes sure the resource constraints are respected by imposing that the amount of resource k used at time t cannot exceed the resource availability B_k. The resource conflicts are thus avoided by adding a linear constraint for each time period [10]. The last constraint defines the domain of the decision variables y_{it}.

The disadvantage of this formulation is the potential for many binary variables and constraints when using a large scheduling horizon T. The formulation allows for up to nT binary variables and will attain another resource constraint for every time unit. With the time-indexed formulation, we are also limited to using only integer activity durations or choosing a very fine time grid to avoid wasting time. An advantage of the formulation is that a simple preprocessing algorithm can be used to tighten the time windows of the activities, which allows us to remove the superfluous binary variables [4]. A notable time-indexed formulation that also uses the idea of time discretization is the formulation by Mingozzi et al. [20].

3.2.2 Schedule-based formulation

The sequence-based models are usually defined on two sets of decision variables: a Boolean linear ordering variable to determine the relative ordering of the activities and a date variable to determine the start or completion time of the activities [10]. The linear ordering variables make sure the sequence of activities respects the precedence and resource constraints, while the date variable makes sure the sequence of activities does not violate the time restrictions. The main difference between the time-indexed and schedule-based formulations is that the latter do not discretize the time.

An early schedule-based formulation by Alvarez et al. [21] is based on the notion of forbidden sets to avoid resource overuse. As the computation of the forbidden sets can be difficult and time-consuming [10], a more recent formulation by Artigues et al. [19] based on the notion of resource flow has gained popularity. Renewable resources can naturally be modelled as a flow in and out of activities; the resource is occupied when an activity is processed but freed again once the activity is complete. In this formulation, an additional decision variable f_{ijk} is introduced to model the flow and represents the amount of resource k transferred from activity i to activity j [10]. Linear resource constraints can now easily be formulated by imposing that the total resource flow into an activity must meet the activity's resource demands b_{ik}.

The formulation by Artigues et al. also includes a linear ordering variable x_{ij} and a date variable S_i [10]. The Boolean variable x_{ij} equals 1 if activity j starts after activity i in the final schedule and 0 otherwise. The variable S_i gives the start time of each activity. The MILP of Artigues' flow formulation is thus given as [10]:

$$\min \quad S_{n+1} \tag{3.10}$$

$$\text{s.t.} \quad x_{ij} = 1, \; x_{ii} = 0, \qquad \forall\, (i,j) \in E, \tag{3.11}$$

$$x_{ij} + x_{ji} \leq 1, \qquad \forall\, (i,j) \in V^2, \; i < j, \tag{3.12}$$

$$x_{ik} \geq x_{ij} + x_{jk} - 1, \qquad \forall\, (i,j,k) \in V^3, \tag{3.13}$$

$$S_j - S_i \geq -M + (d_i + M)x_{ij}, \qquad \forall\, (i,j) \in V^2, \tag{3.14}$$

$$\sum_{j \in V} f_{ijk} = b_{ik}, \qquad \forall\, i \in V, \; \forall\, k \in \mathcal{R}, \tag{3.15}$$

$$\sum_{j \in V} f_{jik} = b_{ik}, \qquad \forall\, i \in V, \; \forall\, k \in \mathcal{R}, \tag{3.16}$$

$$0 \leq f_{ijk} \leq \min(b_{ik}, b_{jk})x_{ij}, \qquad \forall\, (i,j) \in V^2, \; \forall\, k \in \mathcal{R}, \tag{3.17}$$

$$x_{ij} \in \{0,1\}, \qquad \forall\, (i,j) \in V^2, \tag{3.18}$$

$$f_{ijk} \in \mathbb{R}^+, \qquad \forall\, (i,j) \in V^2, \; \forall\, k \in \mathcal{R}, \tag{3.19}$$

$$S_i \in \mathbb{R}^+, \qquad \forall\, i \in V. \tag{3.20}$$

The first three constraints are concerned with the linear ordering variables. Constraint (3.11) makes sure the precedence relations are respected, while constraints (3.12) and (3.13) make sure we get no cycles and that the linear ordering variables are transitive. Constraint (3.14) links the linear ordering variables and the date variables with a big-M constraint to make sure the ordering of the activities respects the time restrictions. The value of M must be large enough to avoid cutting of legitimate solutions when x_{ij} is equal to 0, which means that $M > S_{n+1}$. Finding a tight lower bound on M to avoid a bad linear relaxation can be challenging. However, by computing the latest start time of activity A_{n+1}, an estimate can be obtained. Constraints (3.15), (3.16) and (3.17) make sure the correct amount of resources service each activity and are transferred between the activities. To make sure that the availability of the resources is not exceeded, the demands of the dummy activities b_{0k} and $b_{(n+1)k}$ are set to B_k for every resource k [10].

The advantage of the schedule-based formulations is that the size of the MILP is not directly related to the project duration, such that schedules with long planning horizons can be made without increasing the number of binary variables. Without the time discretization, we can also use non-integer activity durations since S_i can take on any positive, real value. The disadvantages include the increased number of decision variables and the big-M constraints, which have a bad linear relaxation. Using the flow-based formulation also has the disadvantage of inducing many symmetries, making it more computationally demanding to find optimal solutions [10].

3.3 Modifications

With only the simple precedence and resource constraints presented in Section 3.1, the classical Resource-Constrained Project Scheduling Problem formulation does not allow for accurate modelling of many practical situations. Consequently, various extensions of the basic RCPSP have been developed to model a variety of problems. We present three extensions that seem natural to include when modelling the Mass Relocation Problem: multiple modes, time lags, and setup times. A comprehensive study of the various variants can be found in [12, 13].

3.3.1 Multiple modes

The standard RCPSP assumes each activity can only be performed in one predefined way, defined by the activity's duration and resource requirements. The activity concept can be extended by allowing an activity to be performed in several different ways, specified by a specific duration and resource demand [12]. Each such duration-resource combination is called a *mode* [22]. Each activity has a given number of modes M_i and must be completed in one of them. With only one mode per activity, we are reduced to the standard RCPSP. We define a new duration and resource demand for each mode $m \in \{1, \ldots, M_i\}$ [6]:

d_{im}	Duration of activity i in mode m.
b_{ikm}	Resource demands required by activity i when completed by resource k in mode m.

An example of the time-discretized formulation by Pritsker et al. with multiple modes can be found in [22].

3.3.2 Time lags

Time lags are usually included to model time restrictions between the activities. If a certain number of time units must elapse between the end of an activity and the beginning of another, this can be modelled with time lags [12]. A *minimal time lag* is defined as a constraint of the form:

$$S_j - S_i \geq T_{ij}^{\min}, \tag{3.21}$$

and imposes that the start of activity j can only happen $T_{ij}^{\min}$ time units after the beginning of activity i [23]. If activity j can begin before the completion of activity i, this can be expressed by setting $T_{ij}^{\min} < d_i$. Minimal time lags are often used to model release dates or ready times [23], where we want to specify at what time since the beginning of the project, or since the last activity, a new activity can be processed. A *maximal time lag* on the other hand, can model the maximal time units between two activities:

$$S_j - S_i \leq T_{ij}^{\max}. \tag{3.22}$$

Maximal time lags are often used to model deadlines, where $T_{ij}^{\max}$ can express the longest time we can wait from the start of the project until activity i must be processed. Time lags can be modelled as an AON graph where the edges between the activities are weighted by the time lags [23]. Time lags focus on the activities and their ordering in relation to each other but are usually independent of the resources used to complete the activities.

3.3.3 Setup times

Setup times are the times needed for preparation or transportation of resources between activities [12]. In scheduling models with setup times, resources are unavailable for certain periods due to being changed, charged, or transported from one location to another [6]. If t_{ijk} denote the setup time between activity i and j for resource k, a scheduling model with setup times must satisfy:

$$S_i + d_i + t_{ijk} \leq S_j, \tag{3.23}$$

if activity j is processed directly after activity i on resource k [13]. The difference between setup times and time lags is that the resources are unavailable during the setup time, while the time lags focus solely on the ordering of the activities and do not affect the resource availability.

Setup times connected to transportation of resources are often called transfer times [13]. In this case, t_{ijk} represents the travel time between activity i and j with resource k. Many different approaches to modelling transfer times and routing exist [24, 25, 26]. Lacomme et al. [26] investigates the integration of routing in RCPSP instances where resources have to be transported between activities with a limited fleet of vehicles. The problem consists of finding a feasible schedule of minimal duration and assigning a vehicle to each transport of resources. If the transportation of the resources does not have to be completed by a separate fleet of vehicles, we can instead use approaches proposed by Quilliot and Toussaint [24] and Poppenberg and Knust [25]. In these approaches, the possible resource flows are represented by a graph where each activity is modelled as a node, and each resource transfer is represented by a directed arc. To incorporate the transfer times, the edges are weighted with the values $d_i + t_{ijk}$ [25].

3.4 Solution methods

The Resource-Constrained Project Scheduling Problem is one of the most complex scheduling formulations and belongs to the class of $\mathcal{NP}$-hard problems in the strong sense. A proof can be found in [4]. The introduction of limited renewable resources makes the RCPSP more intractable than other similar scheduling formulations, and exact solution methods become slow and time-consuming once the number of activities and resources increases.

Exact solution methods for the RCPSP are usually based on the branch-and-bound algorithm [10], which is used to solve mixed-integer linear programs [27]. The branch-and-bound algorithm enumerates candidate solutions of an MILP by creating a search tree [27]. At the root node, the linear relaxation of the MILP is solved. A recursive splitting of the solution space into smaller regions is performed in a process called branching to systematically discover new incumbents. Upper and lower solution bounds are used to prune branches until the optimal solution is found. Specialized branch-and-bound methods take advantage of the problem structure of the

RCPSP, and enhancements like cut generation and constraint propagation are often used [24]. Even the best exact solution schemes for solving the RCPSP struggle when the problem instances become too big.

An alternative to using exact methods for solving the RCPSP instances is to use heuristics and metaheuristics. A heuristic is an algorithm that yields a solution to an optimization problem efficiently, even though the solution might not be optimal [4]. Heuristics are often used in cases where exact methods are too time-consuming and computationally demanding to solve to optimality. The approximate solutions are often good enough in practice, but it is challenging to verify how close to optimal they are. Heuristics are therefore problematic to use in many applications. Even though exact solution methods are often too computationally demanding, they can contribute greatly to understanding the characteristics of problems and help the development of other solution techniques [10].

Chapter 4

Mass Relocation RCPSP

When modelling the Mass Relocation Problem with the Resource-Constrained Project Scheduling Problem, we want to schedule pickup and delivery activities given a fixed time horizon using the dump trucks as resources. The schedules must respect resource constraints imposed by the number of dump trucks, as well as precedence relations between the activities. The implicit routing of the dump trucks is natural when viewed as a resource flow between activities; hence we use the schedule-based formulation by Artigues et al. [19] as a starting point. With this problem formulation, the program size is not directly related to the scheduling horizon, and we can use real-valued durations and travel times with no adjustments.

In Section 4.1 we model the different aspects of the MRP in terms of the schedule-based mathematical formulation of the RCPSP. In Section 4.2 we formally define the full Mass Relocation RCPSP formulation, while in Section 4.3 a reduced model is presented. Section 4.4 introduces a flipped model where we change the objective to maximizing the number of activities instead of minimizing the makespan.

4.1 Modelling the MRP

Each individual pickup and delivery at a location is considered an activity, and the total number of activities is denoted by n. Like in the standard RCPSP in Section 3.1, we include two dummy activities, A_0 and A_{n+1}. These act as a source and a sink where all dump trucks have to start

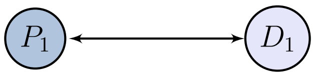


Figure 4.1: Illustration of construction site with only one pickup and one delivery location, P_1 and D_1.

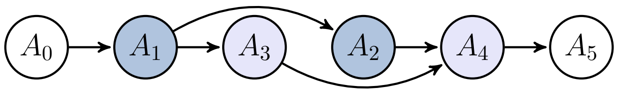


Figure 4.2: Precedence graph for two pickups and two deliveries on the construction site in Figure 4.1. Nodes A_1 and A_2 represent the pickup activities, while A_3 and A_4 represent the delivery activities. The white nodes illustrate the dummy activities, and the directed edges represent the precedence relations.

and end their journeys. Precedence relations are defined to ensure that the time restrictions between adjoining activities can be enforced. We define the precedence relations between the activities such that only one activity can be simultaneously processed at any given location. We also include precedence relations between the source and the first pickup at each location and between the sink and the last deliveries. These relations ensure that the schedules always start with activity A_0 and end with activity A_{n+1}. If the ordering of the individual pickup and delivery activities are unambiguous, precedence relations between these can also be included.

To illustrate the precedence relations, we include two examples. The simple construction site in Figure 4.1 only has one pickup and one delivery location where we want to schedule two pickups and two deliveries. The pickups are denoted by A_1 and A_2, while the delivery activities are denoted by A_3 and A_4. The ordering of the activity numbers indicates which pickup and delivery activities must be completed first. Figure 4.2 shows the precedence relations between the activities as an AON graph, henceforth called the *precedence graph*. Each node represents an activity, and each edge represents a precedence relation. The first pickup A_1 is preceded by the source A_0. The first delivery activity A_3 cannot be finished before

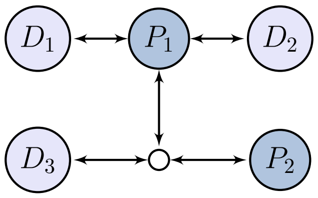


Figure 4.3: Illustration of construction site with two pickup (P_1, P_2) and three delivery (D_1, D_2, D_3) locations.

the first pickup is completed, thus A_1 precedes A_3. Since we cannot do two pickups simultaneously at the same location, activity A_1 also precedes the second pickup activity A_2. The same is true for the deliveries, thus A_3 precedes the second delivery activity A_4. The last delivery A_4 precedes the sink A_5. As it is possible for the second pickup activity A_2 to start while the first delivery A_3 is still in progress, there is no need for an edge between A_3 and A_2.

The more complicated construction site in Figure 4.3 includes two pick-up and three delivery locations. As it is no longer trivial which deliveries follow which pickups, we cannot model any precedence relations between these without narrowing the solution space and potentially excluding the optimal solution. The precedence graph thus only includes edges between activities at the same locations, in addition to edges from the source and sink. Figure 4.4 shows the precedence graph assuming we want to schedule three pickups at P_1 and P_2, two deliveries at D_1, one delivery at D_2 and three deliveries at D_3. To avoid a disconnected graph, precedence relations between the source and the first deliveries and the sink and the last pickups are also included. These are not strictly necessary but remove any ambiguity as to which activities should be scheduled first and last.

We model the service and preparation times of the activities using minimal time lags. The service times can differ based on location, while the preparation time is assumed to be constant at all pickup sites. We define the time lags for the pickup activities as $d_i + p$, where d_i is the duration of activity i and p is the mass preparation time. For the delivery locations,

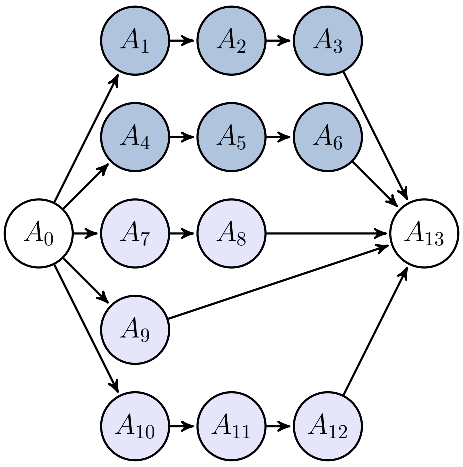


Figure 4.4: Precedence graph for three pickups at P_1 and P_2, two deliveries at D_1, one delivery at D_2 and three deliveries at D_3 on the construction site in Figure 4.3. Nodes A_1-A_6 represent pickup activities, while A_7-A_{12} represent delivery activities. The white nodes illustrate the dummy activities, and the directed edges represent the precedence relations.

the time lag will include only the service time of an activity d_i.

The transportation of dump trucks between pickup and delivery locations is implemented using transfer times. The legal travel patterns on the construction sites are modelled by a *travel graph*, where each node represents an activity and each directed edge a possible travel path. The travel graph incorporates the restrictions left out by the precedence graph, namely that the vehicles must alternatingly travel between the pickup and delivery locations. An example of a travel graph for the construction site in Figure 4.1 is shown in Figure 4.5. From a pickup activity, the dump trucks can travel to any delivery activity and vice versa. Thus, edges between the pickups and deliveries will always be bidirectional and weighted by the travel times t_{ij} and t_{ji}. As the dump trucks can start from an arbitrary pickup activity, the travel paths from the source to each pickup are weighted by 0 and can be travelled along with no time delay. The same is true for the paths leading from the delivery activities to the sink. To include the option

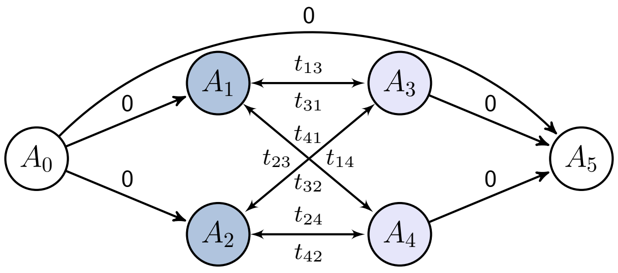


Figure 4.5: Travel graph illustrating the legal travel patterns on the construction site in Figure 4.1, assuming we want to complete two pickup (A_1, A_2) and two delivery (A_3, A_4) activities. The weights on the edges show the travel times between the activities, and the white nodes illustrate the dummy activities.

of vehicles remaining unused, a zero-delay edge directly between the source and the sink is also included. The unavailability of a vehicle during travel is modelled by imposing that the start time of two activities processed by the same vehicle must be separated by at least $d_i + t_{ij}$.

The fleet of K dump trucks available to complete the activities creates the resource constraints. In Chapter 2, we made the assumption that the dump trucks have the same capacity. Hence, we define an activity's demand for a resource as $b_{ik} = 1$ for all $i \in V \setminus \{0, n+1\}$ and $k \in \mathcal{R}$. In addition, since each dump truck can only visit one location at a time, $B_k = 1$ for all resources $k \in \mathcal{R}$. As the speed of the vehicles can vary, the service times of the activities and the travel times between the locations can change based on which dump truck is used. To model this, we use the extension of multiple modes to choose which resource-mode to complete an activity in. We define the set of resources $\mathcal{R} = \{1, \ldots, K\}$ to include the K different dump trucks. The resource $\mathcal{R}$ is now cumulative in the sense that it may process up to K activities at a time [10], and gives us the option to complete an activity in K different modes. The duration of activity $i \in V$ completed with vehicle $k \in \mathcal{R}$ is given by d_{ik}, and the travel time between activity i and j using vehicle $k \in \mathcal{R}$ is given by t_{ijk}.

4.2 Full model

With the precedence graph, the travel graph, and the time delays defined, we can model the Mass Relocation Problem as a mixed-integer linear program. We define the following sets and variables:

V	Set of activities, $V = \{A_0, \dots, A_{n+1}\}$. Contains two dummy activities, A_0 and A_{n+1}.
E	Set of pairs (A_i, A_j) defining the precedence relations between the activities. The pair $(A_i, A_j) \in E$ indicates that activity A_i precedes activity A_j.
F	Set of pairs (A_i, A_j) defining the legal travel patterns between the activities. The pair $(A_i, A_j) \in F$ indicates that a resource can travel from activity A_i to activity A_j.
$\mathcal{R}$	Set of K resources, $\mathcal{R} = \{1, \dots, K\}$.
d_{ik}	Duration of activity $A_i \in V$ when processed by resource $k \in \mathcal{R}$. For A_0 and A_{n+1}, it holds that $d_{0k} = d_{(n+1)k} = 0 \ \forall \ k \in \mathcal{R}$.
p_i	Preparation time of activity $A_i \in V$. The preparation time is equal to p for all pickup activities, and 0 otherwise.
t_{ijk}	Travel time between activity A_i and A_j with resource $k \in \mathcal{R}$ where $(A_i, A_j) \in F$.

The resource flow-based MILP formulation defined in (3.10)-(3.20) includes three types of decision variables: x_{ij}, f_{ijk} and S_i. Since the availability of the resources and the activity demands are equal to 1, the variable f_{ijk} is now a binary variable. Either a vehicle traverses the edge $(i, j) \in F$, or it does not. We can thus omit the linear ordering variable x_{ij} from the formulation, as constraint (3.17) is no longer needed to make sure the correct resource flows are transferred between activities. Information about which activities directly precede each other is defined in the precedence graph, hence the presence of x_{ij} in constraint (3.14) is also superfluous. The date variable S_i remains unchanged. Thus, the decision variables are defined as:

S_i	Start time of activity $A_i \in V$, $S_0 = 0$.
f_{ijk}	Boolean variable equal to 1 if edge $(A_i, A_j) \in F$ is traversed with resource $k \in \mathcal{R}$.

As in the classical RCPSP, we define the objective of the MILP as:

$$\min \quad S_{n+1}, \tag{4.1}$$

which is equivalent to minimizing the makespan of the schedule. We start by defining the constraints concerned with the resource allocation, which are variations of constraints (3.15) and (3.16). To make sure all activities are visited by only one dump truck, we impose that:

$$\sum_{j \in V_i^{\text{in}}} \sum_{k \in \mathcal{R}} f_{jik} = 1, \qquad \forall\, i \in V \setminus \{0, n+1\}, \tag{4.2}$$

where $V_i^{\text{in}} = \{j \in V \mid (j,i) \in F\}$ is the set consisting of all activities with a travel edge into activity i. The constraint enforces that only one of the travel paths leading into an activity can be used, regardless of the mode k, and thus only one vehicle can complete an activity. As all vehicles start at the source and end at the sink, this is not true for the dummy activities, and thus they are omitted. Similarly, to make sure only one vehicle can leave an activity, we impose that:

$$\sum_{j \in V_i^{\text{out}}} \sum_{k \in \mathcal{R}} f_{ijk} = 1, \qquad \forall\, i \in V \setminus \{0, n+1\}, \tag{4.3}$$

where $V_i^{\text{out}} = \{j \in V \mid (i,j) \in F\}$ is the set consisting of all activities with a travel edge out of activity i. Constraints (4.2) and (4.3) make sure each activity is visited only once, but we have not guaranteed that it is the same vehicle that enters and leaves an activity. To do this we include a flow conservation constraint:

$$\sum_{j \in V_i^{\text{in}}} f_{jik} - \sum_{j \in V_i^{\text{out}}} f_{ijk} = 0, \qquad \forall\, i \in V \setminus \{0, n+1\},\ \forall\, k \in \mathcal{R}. \tag{4.4}$$

The constraint imposes that if an edge into an activity is traversed by a resource k, the same resource must also traverse an edge out of the activity to make sure the travel of the vehicles is consistent. Since we impose that each activity must be visited and can only be visited once, we have to ensure that we have the same amount of pickups and deliveries, and therefore an even total number of activities n. Otherwise, it will be impossible to complete all the activities, and the problem becomes infeasible.

Since we have omitted the dummy activities from the previous constraints, we have to make sure the resources behave as we want them to also when travelling out of the source and into the sink. To enforce that the vehicles can only travel from the source or to the sink once, we impose that:

$$\sum_{j \in V_0^{\text{out}}} f_{0jk} = 1, \qquad \forall\, k \in \mathcal{R}, \tag{4.5}$$

and

$$\sum_{j \in V_{n+1}^{\text{in}}} f_{j(n+1)k} = 1, \qquad \forall\, k \in \mathcal{R}. \tag{4.6}$$

The constraints simply state that each vehicle must traverse exactly one edge out from the source and one edge into the sink. Since we have included an edge directly from the source to the sink in the travel graph, these constraints are still valid when the optimal use of the vehicles is to remain unused. This is relevant in cases where we have more vehicles than activities to complete. The precedence constraint (3.14) can be implemented without much change as:

$$S_j - S_i \geq d_{ik} + p_j, \qquad \forall\, (i, j) \in E,\ \forall\, k \in \mathcal{R}, \tag{4.7}$$

where we have included a minimal time lag to account for the mass preparation time at each pickup location. Since the set E is explicitly defined, we no longer need to include the variable M. The constraint imposes that the start time of activity j must be greater than the sum of the start time of activity i, the duration of activity i and preparation time of activity j, given that $(i, j) \in E$.

To make sure the travel times are respected, we need an additional time constraint:

$$S_j - S_i \geq -M + (d_{ik} + t_{ijk} + M) f_{ijk}, \qquad \forall\ (i,j) \in F,\ \forall\ k \in \mathcal{R}. \tag{4.8}$$

The constraint imposes that the start times of activities connected by a travel path that is being traversed must be separated by the time it takes to travel between the activities with resource k and the duration of the activity. We do not have to include the preparation time in this constraint as we cannot travel directly between pickup activities. As f_{ijk} is binary and can be turned on and off, we need to include the variable M to make sure the constraint still holds when $f_{ijk} = 0$. As discussed in Section 3.2.2, M must be chosen such that $M > S_{n+1}$.

With the constraints defined above, we define the full Mass Relocation Resource-Constrained Project Scheduling Problem as:

$$\begin{aligned} \min \quad & S_{n+1} \\ \text{s.t.} \quad & (4.2), (4.3), (4.4), (4.5), \\ & (4.6), (4.7), (4.8), \\ & f_{ijk} \in \{0,1\}, \qquad && \forall\ (i,j) \in F,\ \forall\ k \in \mathcal{R}, \\ & S_i \in \mathbb{R}^+, && \forall\ i \in V. \end{aligned} \tag{4.9}$$

The program has a total of $K|F|$ binary variables, where $|F|$ is the number of edges in the travel graph. The majority of the binary variables will simply become zero as all edges, apart from possibly those connecting to A_0 or

A_{n+1}, will be traversed in at most one mode. For every pickup-delivery pair we add to the problem, the number of binary variables grows. Increasing the number of activities from n to $n+2$ increases the number of binary variables by $K(2n+4)$. This quickly makes the problem instances large and computationally demanding to solve. Therefore, for this preliminary effort to solve the MRP using the RCPSP, we define a reduced program where we assume we are dealing with a homogeneous fleet of vehicles to reduce the problem size.

4.3 Reduced model

The full model defined in (4.9) is necessary when we are dealing with dump trucks with different travel speeds and service times. Assuming we have a homogeneous fleet of dump trucks where the vehicles do not differ significantly in terms of travel or processing speeds, we can omit the k-index from all variables:

d_i	Duration of activity $A_i \in V$. For A_0 and A_{n+1}, it holds that $d_0 = d_{n+1} = 0$.
t_{ij}	Travel time between activity A_i and A_j where $(A_i, A_j) \in F$.

The flow decision variable is redefined as:

f_{ij}	Boolean variable equal to 1 if edge $(A_i, A_j) \in F$ is traversed.

Removing the multiple modes extension reduces the number of binary variables in the problem by $(K-1)|F|$, and we are left with $|F|+n+2$ decision variables. We define the reduced Mass Relocation Resource-Constrained Project Scheduling Problem, henceforth named model 1, by the following

MILP:

$$\min \quad S_{n+1} \tag{4.10}$$

$$\text{s.t.} \quad \sum_{j \in V_i^{\text{in}}} f_{ji} = 1, \qquad \forall\, i \in V \setminus \{0, n+1\}, \tag{4.11}$$

$$\sum_{j \in V_i^{\text{out}}} f_{ij} = 1, \qquad \forall\, i \in V \setminus \{0, n+1\}, \tag{4.12}$$

$$\sum_{j \in V_0^{\text{out}}} f_{0j} \leq K, \tag{4.13}$$

$$\sum_{j \in V_{n+1}^{\text{in}}} f_{j(n+1)} \leq K, \tag{4.14}$$

$$S_j - S_i \geq d_i + p_j, \qquad \forall\, (i,j) \in E, \tag{4.15}$$

$$S_j - S_i \geq -M + (d_i + t_{ij} + M) f_{ij}, \qquad \forall\, (i,j) \in F, \tag{4.16}$$

$$f_{ij} \in \{0,1\}, \qquad \forall\, (i,j) \in F, \tag{4.17}$$

$$S_i \in \mathbb{R}^+, \qquad \forall\, i \in V. \tag{4.18}$$

The objective (4.10) of model 1 is unchanged from the full model. We still want to minimize the time spent completing the activities. In constraints (4.11) and (4.12) we have simply removed the sum over k. Only one vehicle can still enter and leave an activity, and all activities must be visited once. As it no longer matters which specific vehicle visits each activity, the flow conservation constraint (4.4) is redundant. Constraints (4.5) and (4.6) are changed to (4.13) and (4.14), and rewritten to such a form that not more than K vehicles can leave the source or enter the sink. Since the travel graph only allows for one path from the source directly to the sink, we cannot impose that all vehicles must travel out from the source as there might be no available paths. Therefore, the vehicles have the choice to remain unused by never leaving the source, and the path between the source and the sink is removed from the travel graph. The restrictions imposed by the travel graph will make sure that if a vehicle leaves the source, it must also end at the sink. Constraints (4.15) and (4.16) are unchanged, except for the fact that we have removed the k-index from the variables. Without the multiple modes, constraint (4.14) is redundant given the constraints above it and can in principle be removed.

Even though we have drastically reduced the number of binary variables in model 1, finding an optimal schedule is still demanding in terms of computing power and time once the number of activities increases. As discussed in Chapter 3, the RCPSP formulation is $\mathcal{NP}$-hard, and the flow-based formulation is prone to inducing symmetries [10]. Symmetries arise when multiple optimal solutions exist with different binary variables turned on and off. They often appear in scheduling problems with identical machines, like model 1 [28]. Problem instances can become expensive to compute with the branch-and-bound algorithm when symmetries are present, as computing power is wasted on solving isomorphic solutions in the search tree [28]. Methods to combat symmetries include adding symmetry-breaking constraints to reduce the number of optimal solutions. This has proved successful in some RCPSP formulations [29].

Most of the symmetries in model 1 arise due to the ordering of the activities. Constraint (4.15) makes sure that the activities at each location must be completed in order but does not impose in what order the pickup and delivery locations should be visited. The vehicles thus have the freedom to choose which locations to visit first. This induces many symmetries, especially with few dump trucks to schedule the activities. With only one vehicle, there exists a multitude of different routes between the activities, and several of these will lead to equivalent project makespans. When increasing the number of resources, the number of symmetries of this type decreases as there is less freedom in the ordering of the activities. This would not be the case if the vehicles were not homogeneous. An artificial example is illustrated in Figures 4.6 and 4.7. Assuming the distances between locations P_1, P_2, D_1 and D_2 are all equal, all routes in Figure 4.6 yield an optimal solution for model 1 with one vehicle. With two vehicles, the number of optimal solutions is reduced to two, as shown

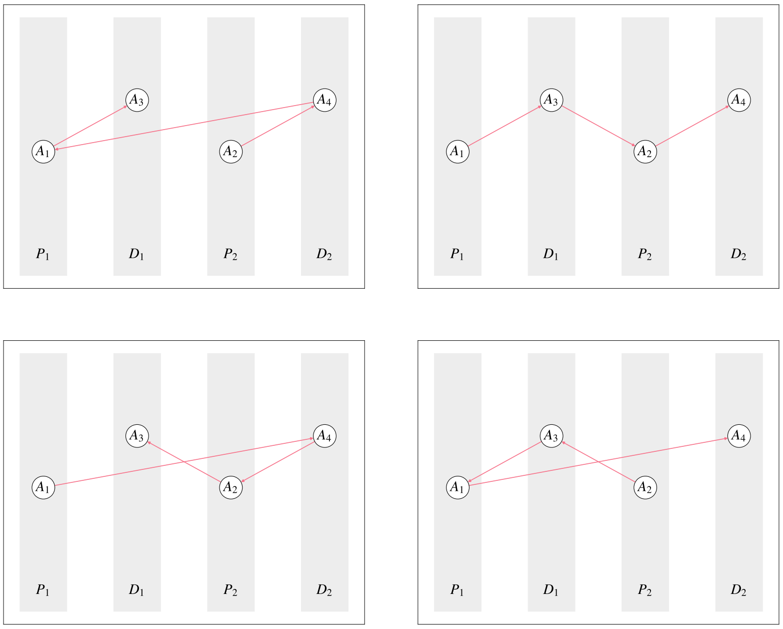


Figure 4.6: Illustration of four solutions from an MRP instance with one vehicles. The graphs show the vehicle's path through four activities on a construction site. The travel times between the locations are equal, and thus the four different paths lead to the same optimal value.

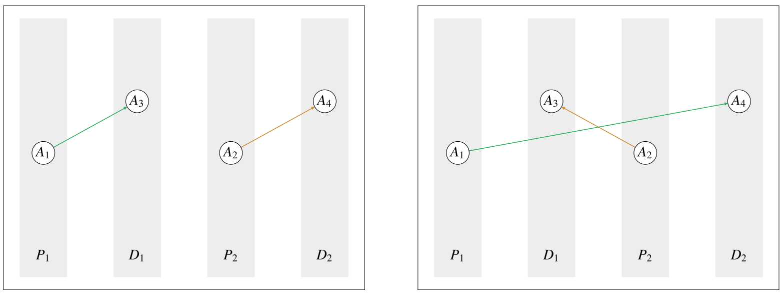


Figure 4.7: Illustration of two solutions from an MRP instance with two vehicles. The graphs show the vehicles' paths through four activities on a construction site. The travel times between the locations are equal, and thus the two different paths lead to the same optimal value.

in Figure 4.7. This simple example illustrates that with more vehicles, the number of symmetries can be reduced as each vehicle's path choice limits the movement of the other vehicles.

Symmetries and computation time aside, model 1 can successfully find optimal schedules given a fixed number of activities at each pickup and delivery location. Unfortunately, this does not solve the Mass Relocation Problem as posed in Chapter 2. The model assumes that we know exactly how many activities we want to complete at each location and that the number of pickups and deliveries have to match. However, to solve the MRP, we have to maximize the number of activities we have time to complete before a fixed time horizon. In this case, we do not know exactly how many deliveries and pickups it is most efficient to complete at each location. Model 1 could still be used in a trial and error manner to find schedules of an approximate length, but there is still the issue of picking the exact number of activities at each location. Thus, we must change the model to find a more precise formulation of the MRP.

4.4 Flipped model

The standard Resource-Constrained Project Scheduling Problem finds the ordering of a given set of activities with the goal of completing them in the shortest amount of time. The goal of the Mass Relocation Problem is to maximize the number of activities we can complete within a fixed scheduling horizon, and thus we do not know how many activities to schedule at each location ahead of time. Hence, to solve the MRP, we have to change the objective of the RCPSP; instead of searching for the smallest project makespan, we search for the maximal number of activities we have time to complete. We denote the fixed scheduling horizon by $S_{\max}$ and the number of activities we have time to complete within $S_{\max}$ by q. The variable q

can be defined in terms of the flow variable f_{ij} in the following manner:

$$q = \sum_{(i,j)\in F'} f_{ij}, \tag{4.19}$$

where $F' = \{(i,j) \in F \mid j \neq n+1\}$. Since every traversed path $(i,j) \in F$ signifies that activity j is visited, the sum of the traversed edges equals the number of completed activities. To avoid counting the sink as an activity, the paths leading into it are omitted. Changing the objective of model 1 to maximizing q, we get the following MILP, henceforth called model 2:

$$\begin{aligned}
\max\ & q && && (4.20)\\
\text{s.t.}\ & \sum_{j\in V_i^{\text{in}}} f_{ji} \leq 1, && \forall\, i \in V \setminus \{0, n+1\}, && (4.21)\\
& \sum_{j\in V_i^{\text{out}}} f_{ij} \leq 1, && \forall\, i \in V \setminus \{0, n+1\}, && (4.22)\\
& \sum_{j\in V_0^{\text{out}}} f_{0j} \leq K, && && (4.23)\\
& \sum_{j\in V_{n+1}^{\text{in}}} f_{j(n+1)} \leq K, && && (4.24)\\
& \sum_{j\in V_i^{\text{in}}} f_{ji} - \sum_{j\in V_i^{\text{out}}} f_{ij} = 0, && \forall\, i \in V \setminus \{0, n+1\}, && (4.25)\\
& \sum_{k\in V_i^{\text{in}}} f_{ki} - \sum_{k\in V_j^{\text{in}}} f_{kj} \geq 0, && \forall\, (i,j) \in E'', && (4.26)\\
& S_{n+1} \leq S_{\max}, && && (4.27)\\
& S_j - S_i \geq d_i + p_j, && \forall\, (i,j) \in E', && (4.28)\\
& S_j - S_i \geq -M + (d_i + t_{ij} + M) f_{ij}, && \forall\, (i,j) \in F, && (4.29)\\
& f_{ij} \in \{0,1\}, && \forall\, (i,j) \in F, && (4.30)\\
& S_i \in \mathbb{R}^+, && \forall\, i \in V, && (4.31)
\end{aligned}$$

where $E' = \{(i,j) \in E \mid j \neq n+1\}$ and $E'' = \{(i,j) \in E \mid i \neq 0, j \neq n+1\}$. In model 2, the objective is to complete as many activities as possible before $S_{\max}$. Hence, the model has to allow for some activities not to be completed. This is implemented in constraints (4.21) and (4.22) by imposing that at most one path in or out of an activity can be traversed, allowing the dump trucks to omit activities. Constraints (4.23) and (4.24) are unchanged from model 1, while the flow conservation constraint (4.25) has to be reintroduced to ensure the vehicle routes are continuous. With the addition of the flow conservation constraint, constraints (4.22) and (4.24) become redundant and can, in principle, be removed from the model.

Since the vehicles no longer have to visit all activities, the start time S_i of activities that are not completed can take on arbitrary values. Thus, enforcing the precedence relations is not enough to ensure that activities that come first in the precedence chains are completed before the others. To avoid the symmetries related to allowing the vehicles to choose freely among the identical activities at a location, we add the symmetry breaking constraint (4.26). The constraint ensures that a vehicle can only travel to an activity if the preceding activities have been travelled to. We cannot impose this on the source or the sink, as the source has no incoming edges, and the sink must be completed at the end of the schedule, even if not all other activities have been completed.

To make sure the project duration does not exceed the scheduling horizon $S_{\max}$, we include constraint (4.27). Considering we might not have time to complete all the activities, we can no longer impose that the last activity at each location has to be finished before S_{n+1}, and thus the set E is changed in precedence constraint (4.28). Travel time constraint (4.29) is unchanged from model 1.

The advantage of model 2, as opposed to model 1, is that we can find the maximal number of activities and the optimal schedule in one step, but the formulation has several disadvantages. Firstly, we have to define an initial overhead of activities at each location beforehand. If too many activities are defined, the MILP becomes large and difficult to solve. If too few activities are defined, the solution is not necessarily optimal in the sense that we could have included more activities. The initial number of activities n must therefore be chosen based on trial and error, which can be time-consuming for long schedules.

Model 2 also includes many symmetries. As in model 1, we have the symmetries related to the ordering of the activities. In addition, the model includes symmetries linked to the choice of which activities to include in the schedule. The objective simply requires that the number of activities is maximized, but several activity subsets yielding feasible schedules can have the same size. These symmetries are challenging to remove without including a prioritization on which activities we prefer to be included in the schedules.

Since model 2 is focused simply on completing as many activities as possible, the ordering of the activities in the resulting schedule might not be optimal. A possible solution is using model 2 to identify which activities we should schedule and model 1 to schedule them efficiently. Since both models can be time-consuming, using them both to produce a schedule can be very inefficient. Instead, a semi-active schedule can be found in a post-processing step by locally left-shifting all the activities.

Curiously, even with different starting points, the final formulation of model 2 is very similar to the VRP formulation of the Mass Relocation Problem developed in [8]. The only significant difference in the MILPs is the objective. The objective of the VRP formulation is to minimize the cost related to travelling while penalizing the activities not completed. As minimizing the travelling of the dump trucks is a natural byproduct of completing as many activities as possible within a fixed time horizon, the objective of model 2 is perhaps more intuitive when solving the MRP. Because of the similar formulations, the models face the same challenges concerning symmetries and estimation of initial activities.

4.4.1 Priorities

With the single objective of maximizing the number of activities, we might encounter cases where only some activity types are completed, and certain locations are ignored because of long travelling times. To prevent this, we can include individual activity priorities into the objective of model 2:

$$q_w = \sum_{(i,j)\in F'} w_j f_{ij}. \tag{4.32}$$

Instead of maximizing q, we maximize q_w where each activity is weighted by a unique priority. By prioritizing time-sensitive or strenuous activities, the model will be prone to completing these even if it means reducing the total number of activities in the schedule. The priorities can be chosen such that an equal amount of activities at each location is favorable or according to a pre-existing plan.

Construction sites usually have a plan of how much mass they want moved to and from the different locations within a due date. With such a plan, we can define a priority w_j for each activity by calculating:

$$w_j = \frac{\text{remaining activities}_j}{\text{due date}_j} + 1 \qquad \forall\, j \in V \setminus \{0, n+1\}. \tag{4.33}$$

A higher value of w_j signifies a higher priority. The variable "remaining activities$_j$" indicates how many remaining activities we have to complete at the location of activity j. The number of activities can be calculated based on the mass volume needed to be moved to/from the location by taking into account the capacity of the dump trucks. The variable "due date$_j$" indicates how much time is left before the activity must be completed. The due date is measured in "schedule lengths". If we make a six-hour schedule, the due date indicates how many six-hour periods we have left to finish an activity. Given a month to complete the activities at a location, and assuming a workday is twelve hours, this would translate into a due date of 60 "six-hour schedules". Holidays and Sundays could be subtracted from the due date. Activities not part of the plan are given priority 1.

When defining the priorities in this manner, we avoid completely ignoring certain activity types. We also remove some of the symmetries in the model, as feasible schedules with the same number, but different types, of activities can be differentiated based on their prioritized sum. There will be a trade-off between completing as many activities as possible and completing the highest prioritized activities.

Chapter 5

Benders decomposition

As discussed in Chapters 3 and 4, exact solutions to large RCPSP instances can be expensive and time-consuming to compute with methods based on the branch-and-bound algorithm, such as the MIP solver in the optimization tool Gurobi [30]. Benders decomposition (BD) is a solution method that exploits the structure of large MILPs to "decentralize the overall computational burden" [31]. The method was first introduced by J. F. Benders in 1962 [15] and has been applied to solve large MILPs in many fields, including planning and scheduling [31]. Approaches to solving the Resource-Constrained Project Scheduling Problem with Benders decomposition have also proved successful, for example, in maritime trafficking [32], for projects with multi-skilled personnel [33], and for projects with many resources [34].

In this chapter, we apply Benders decomposition to model 1 and model 2 with the aim of finding exact solutions to the MRP efficiently. In Section 5.1 the standard Benders decomposition algorithm is introduced, and in Section 5.2 the decomposition method is applied to model 1 and model 2. Section 5.3 discusses improvements to the standard algorithm necessary for an efficient implementation.

5.1 Standard Benders decomposition

Benders decomposition is an iterative algorithm used to find exact solutions to mixed-integer linear programs [15]. The decomposition works by separating the original problem into a master problem (MP) and a cut-generating problem, often called the subproblem (SP) [35]. The solution of the master problem is used to generate the subproblem, and the solution of the subproblem is used to tighten the master problem by adding new constraints, often called *cuts*. The master and subproblem are alternatingly and iteratively solved until the master problem is sufficiently bounded and a solution can be found [31]. The BD algorithm is most often used in domains like stochastic optimizations, where the problems take on specific forms that are advantageous for the decomposition [31].

To use Benders decomposition we assume we have an MILP of the following form:

$$\begin{aligned} \min\ & d^T y + c^T x \\ \text{s.t. } & Ax + By \geq b, \\ & y \in \mathcal{Y}, \\ & x \geq 0, \end{aligned} \tag{5.1}$$

where $y \in \mathbb{R}^{n_1}$ is an integer variable and $x \in \mathbb{R}^{n_2}$ is continuous. The remaining variables are given as follows: $d \in \mathbb{R}^{n_1}$, $c \in \mathbb{R}^{n_2}$, $A \in \mathbb{R}^{m \times n_2}$, $B \in \mathbb{R}^{m \times n_1}$ and $b \in \mathbb{R}^m$. The constraints are written in the general form $Ax + By \geq b$. Benders decomposition decomposes the MILP into a master problem containing the integer variable and a subproblem formulated in the space of the remaining variables [35]. The master problem is thus an MILP, while the subproblem becomes a linear program (LP).

To derive the linear SP we fix the integer variable y in the original MILP, and obtain the following:

$$\begin{aligned} \min\ & d^T \bar{y} + c^T x \\ \text{s.t. } & Ax \geq b - B\bar{y}, \\ & x \geq 0, \end{aligned} \tag{5.2}$$

where $\bar{y}$ denotes a fixed y-value. The term $d^T \bar{y}$ is now a constant and can be removed from the objective. The resulting program is called Benders Subproblem and is a linear program concerned only with the continuous variable x. For certain $\bar{y}$'s, the SP might become infeasible. To avoid infeasibility and to easily obtain dual multipliers which are used to generate cuts, it is usual to solve the Dual of the SP (DSP) instead of the SP itself [31]. The DSP is formulated as:

$$\begin{aligned} \max\ & (b - B\bar{y})^T u \\ \text{s.t. } & A^T u \leq c, \\ & u \geq 0, \end{aligned} \tag{5.3}$$

where we have introduced the continuous variable $u \in \mathbb{R}^m$. From duality theory it is known that if the Primal is infeasible, the Dual is unbounded. By strong duality we also know that if the Dual is feasible for a given u, the Primal and the Dual have the same optimal solutions. Thus, an unbounded DSP indicates an infeasible SP, and in the case where the DSP is feasible, we know that $(b - B\bar{y})^T u = c^T x$.

The Benders Master Problem is concerned with the integer variable y and is formulated as follows [35]:

$$\begin{aligned} \min\ & d^T y + z \\ \text{s.t. } & \{\text{Benders cuts}\}, \\ & y \in \mathcal{Y}, \\ & z \in \mathbb{R}. \end{aligned} \tag{5.4}$$

The addition of the z variable works as a surrogate for the $c^T x$ term in the objective function of the original problem. To begin with, the MP contains no constraints, except in the case where the original problem includes constraints concerning only the integer variable. In this case, the integer variable constraints will be directly included in the MP. The set of *Benders cuts* is initially empty, but a new constraint is added to the set every time we solve the DSP to tighten the MP formulation. The form of the cut depends on the feasibility of the subproblem. When solving the DSP, we encounter extreme points u^p if the problem is feasible and extreme rays u^r if the problem is unbounded [31]. In the case of feasibility, an *optimality cut* will be added to the master problem. The optimality cuts are of the form [31]:

$$(b - By)^T u^p \leq z, \tag{5.5}$$

where u^p is the solution variable of the DSP. The cut will tighten the lower bound on z in the MP. If the DSP is unbounded, a *feasibility cut* is generated. The feasibility cuts are of the form [31]:

$$(b - By)^T u^r \leq 0, \tag{5.6}$$

where u^r is an unbounded ray. The unbounded ray indicates the direction of unboundedness where the SP is infeasible for the given $\bar{y}$. The addition of the feasibility cut restricts further movement in the unbounded direction [31].

We have reached optimality when the optimal value of z from the MP equals the objective value of the corresponding DSP (or SP), or when they are closer than a given tolerance ε:

$$c^T x^* - z^* = OBJ_{\text{DSP}} - z^* < \varepsilon. \tag{5.7}$$

We can calculate the optimality gap at each iteration by keeping track of the objective values of the MP and SP. The objective value of the master problem yields a valid lower bound on the optimal solution, as the master problem is a relaxation of the original problem. A valid upper bound is given by the combination of the objective value of the DSP/SP and the contribution of the fixed y-value:

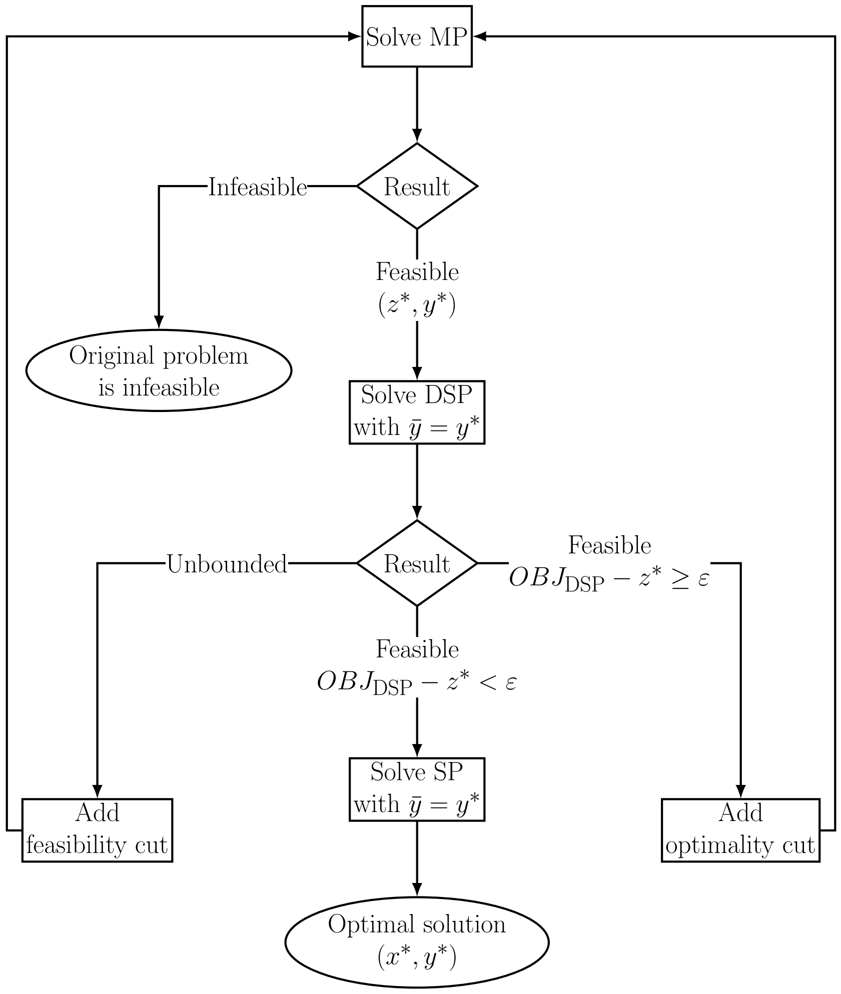
$$LB = OBJ_{\text{MP}}, \quad UB = d^T \bar{y} + OBJ_{\text{DSP}}. \tag{5.8}$$

Algorithm 1: Benders decomposition

```
Initialize $\varepsilon$, $LB = -\infty$, $UB = \infty$;
while $UB - LB \geq \varepsilon$ do
    Solve MP;
    if infeasible then
        Stop algorithm;
    end
    Update $LB = OBJ_{\text{MP}}$;
    Solve the DSP with $\bar{y} = y^*$;
    if unbounded then
        Find unbounded ray $u^r$;
        Add feasibility cut $(b - By)^T u^r \leq 0$ to MP;
    end
    if feasible then
        Add optimality cut $(b - By)^T u^p \leq z$ to MP;
        Update $UB = d^T \bar{y} + OBJ_{\text{DSP}}$;
    end
end
Solve SP with $\bar{y} = y^*$;
```

The Benders decomposition algorithm thus iteratively adds optimality and feasibility cuts to the master problem until the upper and lower bounds meet. Geometrically, the cuts generated from the DSP shrink the polyhedron of the MP and allow us to find the optimal solution to the MILP when the solution space is sufficiently decreased.

A flowchart of the Benders decomposition process can be seen in Figure 5.1, and the general algorithm is given in Algorithm 1. We initialize the algorithm by defining the bounds and choosing a tolerance ε. We start by solving the MP. If the master problem is infeasible, the original problem is also infeasible, and the algorithm is terminated. If the MP is feasible, we update the lower bound and solve the DSP with the optimal y-value from the MP. Based on the result of the DSP, either a feasibility or optimality cut is generated, and the upper bound is updated. The algorithm contin-



Solve MP
Result
Infeasible
Original problem is infeasible
Feasible (z^*, y^*)
Solve DSP with $\bar{y} = y^*$
Result
Unbounded
Feasible $OBJ_{\text{DSP}} - z^* \geq \varepsilon$
Feasible $OBJ_{\text{DSP}} - z^* < \varepsilon$
Add feasibility cut
Solve SP with $\bar{y} = y^*$
Add optimality cut
Optimal solution (x^*, y^*)

Figure 5.1: A flowchart of the standard algorithm for Benders decomposition. Figure is inspired from [36], with some modifications.

ues until the difference between the upper and lower bounds is smaller than the chosen tolerance. When these are sufficiently close, we have found the optimal y, and we use this to find the optimal x by solving the SP.

5.2 Benders decomposition for model 1 and model 2

When applying Benders decomposition to model 1 and model 2 derived in Chapter 4, the MILPs are decomposed by separating the f_{ij} and S_i variables. In the original model formulations, we attempt to compute both the optimal driving patterns of the vehicles and the optimal schedule of the activities in one step. Computing both of these simultaneously yields a complex problem when the number of activities, and thus the number of binary variables, increases. Splitting the models into smaller parts where each problem focuses on one of the two aspects can therefore be advantageous and contribute to solving the RCPSP instances faster. With Benders decomposition, the master problem will be concerned with finding feasible driving patterns for the vehicles, while the subproblem will ensure that the time restrictions between the activities are respected. If we can find a driving pattern in the master problem that does not violate the time restrictions in the subproblem, we have a feasible solution.

Both model 1 and model 2 fit into the standard MILP form (5.1). For model 1, constraints (4.15) and (4.16) yield mixed constraints of the form $Ax + By \geq b$. The objective function of model 1 is given as S_{n+1} and can be written as $c^T S$, where c is a vector with zeroes everywhere except at the last coordinate and $S \in \mathbb{R}^{n+2}$ is a vector where the ith entry equals S_i. The subproblem can thus be written in the form:

$$\begin{aligned} \min\ & c^T S \\ \text{s.t. } & AS \geq b - B\bar{f}, \\ & S \geq 0. \end{aligned} \tag{5.9}$$

Constraint $AS \geq b - B\bar{f}$ is a combination of constraints (4.15) and (4.16), where A is the constraint matrix of S_i and B is the constraint matrix of f_{ij}. The Dual of the SP is given as:

$$\begin{aligned} &\max\ (b - B\bar{f})^T u \\ &\text{s.t. } A^T u \leq c, \\ &u \geq 0. \end{aligned} \tag{5.10}$$

The remaining constraints from model 1, only concerned with the binary variables f_{ij}, can be directly included in the master problem. The MP for model 1 is thus given as:

$$\begin{aligned} \min\ & z \\ \text{s.t. } & \sum_{j \in V_i^{\text{in}}} f_{ji} = 1, && \forall\ i \in V \setminus \{0, n+1\}, \\ & \sum_{j \in V_i^{\text{out}}} f_{ij} = 1, && \forall\ i \in V \setminus \{0, n+1\}, \\ & \sum_{j \in V_0^{\text{out}}} f_{0j} \leq K, \\ & \sum_{j \in V_{n+1}^{\text{in}}} f_{j(n+1)} \leq K, \\ & \{\text{Benders cuts}\}, \\ & f_{ij} \in \{0, 1\}, && \forall\ (i, j) \in F, \\ & z \in \mathbb{R}, \end{aligned} \tag{5.11}$$

where the variable z acts as a surrogate for the continuous variables S_i in the objective function. Since z is initially unbounded, the MP will remain unbounded until we add an optimality cut to bound the lower value of z. To prevent this, we impose a lower bound on z during the implementation of the BD algorithm. Given a feasible schedule, the value of z can never be smaller than 0, so any bound below this is valid.

Model 2 is a maximization problem but can easily be turned into a minimization problem by multiplying the objective with a negative one. The master problem is thus given as:

$$\begin{aligned} \min \quad & -q+z \\ \text{s.t.} \quad & (4.21), (4.22), (4.23), \\ & (4.24), (4.25), (4.26), \\ & \{\text{Benders cuts}\}, \\ & f_{ij} \in \{0,1\}, \qquad \forall\, (i,j) \in F, \\ & z \in \mathbb{R}. \end{aligned} \tag{5.12}$$

The subproblem will include constraints (4.27), (4.28) and (4.29), and the DSP is defined in the same manner as in (5.10). A notable difference from Benders decomposition for model 1 is that the vector of weights c is now equal to the zero vector. Hence, the SP has no objective to minimize. This means that once the fixed $\bar{f}$ from the master problem provides a feasible solution to the DSP, an optimal solution is already found. Because of this, no optimality cuts will be generated during the iterative solution process, and the lower bound of z will remain unchanged. We can thus remove the variable z from the MP completely as it does not provide any information about the convergence. The algorithm is terminated once the DSP becomes feasible.

With the programs defined above, Benders decomposition can be implemented for both model 1 and model 2. In addition to separating the search for valid driving patterns and feasible schedules, we have moved the big-M constraint from the MP to the SP, which will yield a better linear relaxation when solving the master problem [10]. For the numerical experiments in Section 7.2, the MPs and SPs are solved using implemented solvers in Gurobi.

5.3 Algorithmic improvements

A straightforward implementation of Benders decomposition can often result in excessive computing time and memory usage [31]. The algorithm is prone to many pitfalls, including weak cut generation and time-consuming iterations. Thus, to implement a competitive algorithm, some enhancements have to be considered [35]. A plethora of algorithmic improvements exist [31, 35, 37], but the effectiveness of each method for a specific program is difficult to assess beforehand. As the number of iterations, and by extension the computing time, is strongly related to the strength of the feasibility and optimality cuts [31], we focus on enhancements aimed at strengthening the cuts, as well as dealing with the growing number of constraints in the master problem more efficiently.

5.3.1 Feasibility cuts

Feasibility cuts do not directly assist us in finding optimal solutions but rather help us avoid the infeasible ones. For an MRP instance, many valid driving patterns yield infeasible schedules. If we generate weak feasibility cuts at each such occurrence, the Benders decomposition algorithm will converge very slowly. To strengthen the cuts, we must choose the unbounded rays u^r with care [35]. Instead of choosing an arbitrary u^r every time the DSP is unbounded, it has been shown that adding a normalization on the unbounded ray can be advantageous [35, 38]. One such normalization is the $L1$-normalization, also called the Standard Normalization Condition [38]. If the SP is infeasible, an L_1-normalized unbounded ray is found by solving the auxiliary LP [35]:

$$\begin{aligned} &\min\ v \\ &\text{s.t. } Ax + v \geq b - B\bar{y}, \\ &\qquad x, v \geq 0. \end{aligned} \tag{5.13}$$

The constant v is called the penalty parameter, and acts as a normalization condition in the dual space [35]. If the SP is infeasible, the constraint $Ax \geq b - B\bar{y}$ is impossible to satisfy given the fixed value of y. The solution of problem (5.13) gives us the smallest v which makes the inequality $Ax + v \geq b - B\bar{y}$ true. By strong duality we know that $v = (b - B\bar{y})^T u^r > 0$. Hence the cut $(b - By)^T u^r \leq 0$ is violated by $\bar{y}$, and can be used as a feasibility cut.

The $L1$-normalization has been shown to exhibit many good properties, including sparsity, but does not give any theoretical guarantees on the strength of the resulting cuts [38, 35]. Since the normalization includes the optimization of an additional LP, this can increase the computing time of each iteration. There is a trade-off between the increased time and the potential decrease in the total number of iterations.

5.3.2 Combinatorial cuts

An alternative to strengthening the feasibility cuts is to generate Combinatorial Benders cuts [31]. Combinatorial cuts were developed as a method for dealing with MILPs with binary variables and big-M constraints [31]. The addition of the big-M constraints weakens the feasibility cuts, independent of the normalization of the unbounded rays. Thus, for these kinds of problems, purely combinatorial cuts can be generated instead by using the minimal infeasible subsystem of the subproblems [39]. We adopt a hybrid version of the method proposed in [39] by generating optimality cuts when the SP is feasible and combinatorial cuts when the SP is infeasible. A similar approach was implemented in [40] and [41].

The key to understanding the combinatorial cuts is to recognize that if a subproblem is infeasible, some of the binary variables in the fixed $\bar{y}$ have to change for the problem to become feasible. This can be stated as a constraint by imposing that [41]:

$$\sum_{\bar{y}_i=0} y_i + \sum_{\bar{y}_i=1} (1 - y_i) \geq 1. \tag{5.14}$$

The constraint states that at least one of the binary variables must change from 0 to 1, or vice versa. It yields a purely combinatorial constraint, not dependent on the big-M constraints. We notice that in our models, we can never go from an infeasible to a feasible solution by only changing variables from 0 to 1. Thus, to further strengthen the cuts, we impose that at least one of the binary variables equal to 1 has to change:

$$\sum_{\bar{y}_i=1} (1 - y_i) \geq 1. \tag{5.15}$$

To further strengthen the combinatorial cuts, we can identify the subset of constraints in the SP which has to be changed for the problem to become feasible [39]. This subset is usually called the minimal infeasible subsystem (MIS) or the irreducible inconsistent system (IIS) of an infeasible problem. An IIS is a subset of constraints with the properties that: (1) it is infeasible, (2) if a single constraint or bound is removed, the subsystem becomes feasible [41]. The IIS can therefore provide us with a subset of the variables $\bar{y}$ that have to be changed. We denote the binary variables associated with the IIS $\hat{y}$ and define the combinatorial cut by [39]:

$$\sum_{\hat{y}_i=1} (1 - y_i) \geq 1. \tag{5.16}$$

If the IIS is small, this will provide a strong combinatorial cut.

Commercial optimization solvers like Gurobi can generate an IIS from an infeasible subproblem, but this will only provide us with one combinatorial cut each iteration. To generate inconsistent systems fast, we solve the following linear problem [39]:

$$\begin{aligned} \min \ & \sum_{i=1}^{m} u_i \\ \text{s.t } & u^T A = 0, \\ & u^T(b - B\bar{y}) = 1, \\ & u \geq 0. \end{aligned} \tag{5.17}$$

The problem is designed to find a linear combination of the constraints in the SP that yields a minimal infeasible subsystem [39]. The program finds a dual multiplier u such that $u^T Ax \geq u^T(b - B\bar{y})$ is false, and by extension the assertion $Ax \geq (b - B\bar{y})$ is infeasible. This is achieved by imposing that the left-hand side of the equality has to equal zero, while the right-hand side must be positive [39]. The objective makes sure we obtain the u with the lowest cardinality, as this will generate the strongest cuts [41]. The program (5.17) can be solved multiple times by setting the non-zero elements of the previous u to zero. This is continued until no more feasible solutions can be found [41].

With (5.17) we can quickly generate many strong combinatorial cuts each iteration and thus decrease the occurrence of infeasible subproblems later in the solution process. The disadvantages of generating combinatorial cuts include solving several linear programs every time we encounter an infeasible SP and the management of the numerous cuts generated at each iteration.

5.3.3 Optimality cuts

For the implementation of Benders decomposition for model 1, it can also be beneficial to strengthen the optimality cuts. The strength of these cuts is directly related to the quality of the lower bound and thus to the speed of convergence. Any feasible solution to the DSP generates an optimality cut. In our subproblems, multiple optimal solutions S^* can exist as we only seek to minimize S_{n+1}. The different optimal solutions might not produce cuts of equal strength, and consequently, we aim to find the solutions that generate the best cuts at each iteration [37]. This can be achieved by solving an auxiliary subproblem to find a dual solution u that dominates other cuts in terms of Pareto-optimality [31].

To find the Pareto-optimal solutions, we must first find a core point. A core point y^o is defined as a point in the relative interior of the convex hull of the domain $\mathcal{Y}$ of y [31]. In our case, the domain $\mathcal{Y}$ consists of binary variables, and a valid core point will be any vector with $0 < y_i^o < 1$. We therefore choose the vector $y^o = [0.5, 0.5, \ldots, 0.5]$ as the core point. The dual variable u for the Pareto-optimal cut can be found by solving [37]:

$$\begin{aligned} \max\ & (b - By^o)^T u \\ \text{s.t. } & (b - B\bar{y})^T u = v(\bar{y}), \\ & A^T u \leq c, \end{aligned} \tag{5.18}$$

where $v(\bar{y})$ is the objective value of the DSP. The formulation finds an extreme point u which yields the same objective value as the solution of the original DSP and respects the same constraints, in addition to providing the maximal value at the core point. The optimal u^* is inserted into (5.5) to produce the Pareto-optimal cut.

Pareto-optimal cuts have proven effective in practice [31] but include the optimization of an additional program at each iteration. Alternative approaches to finding good cuts without solving an auxiliary problem exist but are usually more complicated to implement [31].

5.3.4 Lazy constraints

Many implementations of Benders decomposition include the addition of so-called *lazy constraints* [35, 40, 41]. Instead of repeatedly solving the master problem and generating new cuts, the addition of lazy constraints allows us to work with a single search tree during the optimization of the MP [41].

Using a callback function, the branch-and-bound solution process of the master problem can be paused every time we encounter a new incumbent solution. During the pause, the related subproblem is solved, and new cuts are added to the master problem as lazy constraints before the optimization is continued [40].

The advantage of this solution process is that we do not encounter the same incumbents several times, as we might do in the standard implementation of Benders decomposition. The lazy constraints also make sure only cuts violated by previous solutions are included in the program, making the cut management in the MP easier. The approach is not necessarily faster when dealing with small problem instances but can be beneficial when many iterations are needed to reach an optimal solution.

5.3.5 Other improvements

In addition to the enhancements discussed above, countless other improvements exist [31]. While it is common to attribute slowness in the BD algorithm to bad cut generation [31], some also claim that there is more to be gained by stabilizing the initial cut loop of the master problem in the root node [42]. To help the master problems find feasible solutions faster, the author of [40] also proposed a two-stage Benders decomposition. The first stage involves solving a relaxed version of the original MILP to generate cuts fast, while the second stage involves solving the original problem with the cuts generated in the first stage. The goal of the process is to tighten the master problem faster.

Chapter 6

Inexact methods

Given the intractability of the RCPSP formulation, the implementation of Benders decomposition in Chapter 5 is not guaranteed to find optimal solutions to the MRP within a reasonable computation time. For model 1 and model 2, providing a suitable initial number of activities at each location also presents a challenge, as the chosen n can affect the efficiency of the solution methods and the resulting optimal schedules. To solve these problems, we shift our focus from finding optimal schedules to finding feasible schedules efficiently by introducing elements of inexactness. The use of inexact methods can reduce the computation time by narrowing the solution space, and can make initial activity estimation easier. However, the methods come at the cost of not being able to prove the resulting schedules' optimality.

To restrict the freedom of the dump trucks, a predefined mapping between pickup and delivery locations is introduced in Section 6.1. The mapping allows us to reduce the number of binary variables in the models. It also makes it easier to estimate the number of activities we have time to complete at each location. In Section 6.2 we introduce the idea of schedule concatenation. With schedule concatenation, short schedules can be merged into longer ones, thus allowing for more efficient scheduling of projects with long time horizons. Lastly, Section 6.3 uses the mathematical formulation of the RCPSP as a basic tool within a heuristic framework. The result is a greedy scheduling heuristic that adds activities to a schedule one by one in an iterative process.

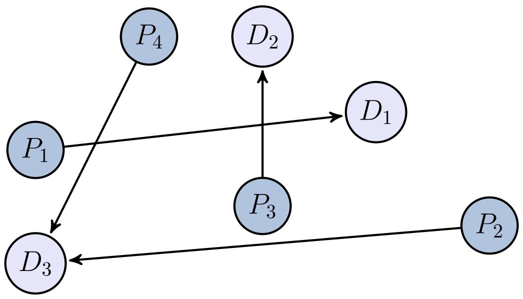
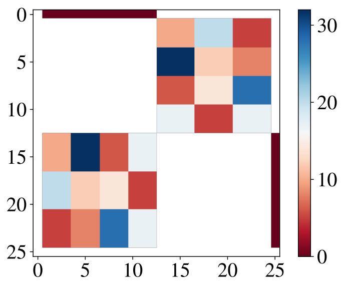


Figure 6.1: Illustration of a construction site where pickup and delivery locations are mapped together ahead of the scheduling. Every time we do a pickup at P_1, we have to do a delivery at D_1. The same is true for the pairs P_2-D_3, P_3-D_2 and P_4-D_3.

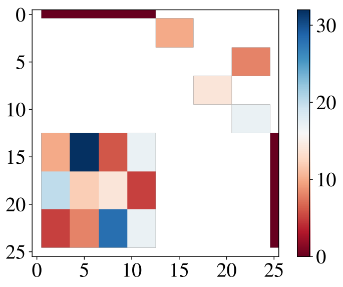
6.1 Location mapping

The vehicles' freedom to choose which pickup and delivery locations to travel between induces many symmetries in the Mass Relocation Problem, as discussed in Sections 4.3 and 4.4. To remove some of these symmetries, we pair pickup and delivery locations ahead of time. This entails deciding which delivery locations vehicles must travel to after completing a specific pickup activity prior to the scheduling. Figure 6.1 shows an example of a location mapping in the case where we have four pickup and three delivery locations. Pickup location P_1 is mapped to D_1, meaning that every time we do a pick up at P_1 the mass must be dumped at D_1. The same is true for the pairs P_2-D_3, P_3-D_2 and P_4-D_3. Several pickup locations can be mapped to the same delivery location, but the travel path from any pickup activity must be unambiguous. The vehicles can choose freely among the different pickup locations when leaving a delivery location.

The location mapping shrinks the solution space of the MRP considerably. Consequently, we might not find the optimal scheduling of the activities with this method. However, the mapping allows us to remove several paths from the travel graph and thus reduce the number of binary



(a) Without location mapping.

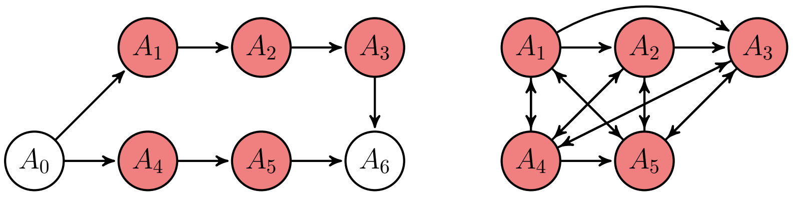


(b) With location mapping.

Figure 6.2: Comparison of adjacency matrices from travel graphs with and without the location mapping from the construction site in Figure 6.1. Without the location mapping, the problem includes 312 binary variables. The problem size is reduced to 216 binary variables with the location mapping. Travel times are randomly generated for illustration.

variables. This reduction is necessary for solving the MRP more efficiently. Even though the location mapping adds a considerable restriction to the vehicles' travel paths, the assumption that specific locations are connected based on mass type or proximity is not unlikely on a real construction site. Since a pickup location can only be paired with one delivery location, we might not be able to service all delivery locations when using the location mapping.

The location mapping can be implemented directly into model 1 and model 2 without significant modifications. Vehicles can be prevented from travelling between certain activities by removing edges from the travel graph. Hence, the location mapping can be implemented by enforcing that vehicles can only travel from a pickup activity to a delivery activity at a predefined location. Figure 6.2 shows a comparison between the adjacency matrices of travel graphs with and without the location mapping from Figure 6.1, in the case where we want to schedule three activities at each pickup location and four activities at each delivery location. Activities A_1-A_{12} represent the pickups and A_{13}-A_{24} represent the deliveries. In



(a) Precedence graph. **(b)** Travel graph.

Figure 6.3: Examples of precedence and travel graphs with the new activity definition. Each activity A_i represents a pickup and a delivery, as well as the travel time between them. The white nodes illustrate the dummy activities.

Figure 6.2(a), the vehicles can travel freely from any pickup to any delivery activity and vice versa. This induces 312 edges, and thus equally many binary variables. With the location mapping, we reduce the number of binary variables by 96, as shown in Figure 6.2(b). Since the location mapping only pairs the locations, not the activities, we do not know which specific activities from each location must be adjoined. Hence, the vehicles still have some freedom in which deliveries to pair with each pickup.

To reduce the number of binary variables further, the location mapping can also be used to redefine the concept of an activity. Instead of defining an activity as an individual pickup or delivery, we can now define it as both. An activity's duration will thus encompass the pickup duration, the delivery duration, and the travel time between the locations. With this definition, we halve the number of activities in the MRP instances and significantly reduce the number of binary variables. With the same 24 activities as in Figure 6.2, a problem with the new activity definition will only include 144 binary variables.

An example of a precedence and travel graph for the new activity concept is illustrated in Figure 6.3. The example is based on a construction site with two pickup (P_1, P_2) and two delivery (D_1, D_2) locations. The locations P_1 and D_1 are mapped together, as well as P_2 and D_2. We want to complete three activities (A_1, A_2, A_3) at P_1-D_1, and two activities (A_4, A_5)

at P_2-D_2. As the precedence graph shows, the activities at the same locations still precede each other, and the precedence chains still have to start and end with the dummy activities. In the travel graph, vehicles can now travel freely between all activities at different locations. In accordance with the precedence relations, vehicles can only travel between activities at the same location if they are travelling to an activity further on in the precedence chain. The travel graph also includes edges from the source to all activities and from all activities to the sink, even though these are omitted from the figure.

The models defined in Chapter 4 can still be used with the new activity formulation with a small adjustment. To ensure that two activities that precede each other do not have to be separated by the first activity's entire duration, we have to change the precedence constraints. We only want other vehicles to wait for the duration of a pickup, as well as the mass preparation time, before starting a new activity at the same location. We thus have to rewrite constraint (4.15) in model 1 and (4.28) in model 2 as:

$$S_j - S_i \geq d_i^p + p_j \quad \forall\ (i,j) \in E, \tag{6.1}$$

where d_i^p is the duration of the pickup associated with activity i.

The new activity definition comes at the cost of losing the precedence relations on the delivery activities. This only becomes an issue when several pickup locations are mapped to the same delivery location. When this happens, we cannot ensure that several deliveries are not simultaneously scheduled at the same location. The precedence relations between the deliveries thus have to be added in a post-processing step. When doing this, we cannot be sure the length of the schedules stays below $S_{\max}$. However, as it is challenging to fill a schedule precisely to the end when no vehicles can work past $S_{\max}$, we expect long schedules to contain enough slack to stay feasible even after the precedence relations are enforced. This is shown to be true in our numerical experiments in Sections 7.3 and 8.2. How big of a problem the conflicting delivery activities become depends on the topology of the construction sites and the chosen location mapping. On some construction sites, there will be room for several deliveries simultaneously, and thus the lack of precedence relations will not be an issue.

The new and old activity definitions can be combined to allow some pickup and delivery locations to be paired while others are not. This is relevant for construction sites with different types of mass, where some types can be transported to almost any delivery location, but some can only be dumped at a specific location. By combining the two activity concepts, we can restrict the travel paths between locations with delivery restrictions but retain the freedom of the vehicles between the remaining locations. Although interesting, this approach is not explored further in this thesis.

The advantage of the new activity definition, in addition to reducing the problem size, is being able to roughly estimate how many activities can be completed at each location before $S_{\max}$. Given the distances between the paired pickup and delivery locations, we can estimate how many rounds of pickups and deliveries one vehicle can complete within the fixed time horizon. The duration of a round between pickup location i and delivery location j is given by:

$$T_{ij} = d_i + t_{ij} + d_j + t_{ji} + \gamma. \tag{6.2}$$

The parameter γ is added for the possibility of including some slack to the round durations. In the cases where we have many vehicles and might have to wait for the mass preparation at a pickup location, $\gamma = p_i$ could for example be used. The number of rounds r_{ij} we can complete at each pickup-delivery pair is defined as:

$$r_{ij} = \left\lfloor \frac{S_{\max} - p_i}{T_{ij}} \right\rfloor + \begin{cases} 1 & \text{if } \mathrm{res}_{ij} \geq d_i + t_{ij} + d_j + \gamma, \\ 0 & \text{otherwise}, \end{cases} \tag{6.3}$$

where res_{ij} is defined as:

$$\text{res}_{ij} = S_{\max} - T_{ij} \cdot \left\lfloor \frac{S_{\max} - p_i}{T_{ij}} \right\rfloor . \tag{6.4}$$

The number of rounds one vehicle can complete at each location pairing can be used to provide an estimate of how many activities we have time to do. The addition of the term γ ensures that the estimation can be both an under- and overestimation based on the needs. If r_{ij} equals zero for some pickup-delivery pair, we do not have time to service the locations within $S_{\max}$.

6.2 Schedule concatenation

The location mapping helps reduce the size of the mixed-integer linear programs in model 1 and model 2. Nevertheless, finding long optimal schedules with exact solutions methods is still computationally demanding and impractical for real-world applications. Hence, we develop an approach for quickly building feasible schedules with large scheduling horizons. The schedule concatenation method works by merging short schedules into longer ones while guaranteeing that the resulting schedule's length does not surpass the sum of the smaller schedules makespans. This is not an exact approach as we cannot prove that the concatenated schedules are optimal, even when they consist of optimal components. However, the method allows for long schedules to be built quickly. Schedules must be concatenated two by two, but the procedure can be continued in an iterative manner to merge several plans.

We denote the first schedule we want to concatenate by S^1 and the second by S^2. The schedules can have different lengths and number of activities but must use the same number of vehicles. The fixed scheduling horizons are denoted by $S^1_{\max}$ and $S^2_{\max}$. When concatenating the schedules, we have to take into account the travel times between the locations we end at in S^1 and start at in S^2. As there might not be enough time to travel between the end of one schedule and the start of another, we cannot guarantee that the resulting schedule S will include as many activities as S^1 and S^2 combined. To merge the schedules, each vehicle's activity path in S^1 has to be paired with an activity path in S^2. An activity path a_k is defined as the sequence of activities vehicle $k \in \mathcal{R}$ traverses through in a schedule. Assuming that the activity path for vehicle k_1 in schedule S^1, denoted by $a^1_{k_1}$, is to be merged with the activity path for vehicle k_2 in schedule S^2, $a^2_{k_2}$, we have to make sure that the following relation holds:

$$S^1_{\max} - S^1_i - d^1_i + S^2_j \geq \operatorname{dist}(i, j). \tag{6.5}$$

Activity i denotes the last delivery activity in $a^1_{k_1}$ and j the first pickup activity in $a^2_{k_2}$. The variable d_i denotes the duration of activity i in S^1 and $\operatorname{dist}(i, j)$ represents the distance between the activities. The inequality states that the time left in S^1 before $S^1_{\max}$, added to the start time of the first activity in S^2, has to be larger or equal to the travel time between the activities. If the assertion does not hold, the first pickup in $a^2_{k_2}$ is discarded from S^2 along with its corresponding delivery activity, and activity j is set to the next pickup activity in $a^2_{k_2}$. This process is continued until (6.5) holds. The makespan of the resulting schedule is then guaranteed to be below, or equal to, $S_{\max} = S^1_{\max} + S^2_{\max}$.

When concatenating the schedules, we must choose which activity paths should be coupled. The easiest way to merge the paths is according to vehicle number, such that a^1_k and a^2_k are merged for all $k \in \mathcal{R}$. Concatenating schedules in this manner might lead to unnecessarily many activities being discarded if the travel times between the activity paths are long. A better approach is thus creating a pairing that considers the distances between the activity paths and minimizes the number of activities discarded during the concatenation. With K vehicles, there are $K!$ different pairings to choose from. The best mapping can be calculated efficiently by utilizing graph matching algorithms. In this thesis, we have implemented an alternative approach based on a greedy pairing for simplicity. For a_1^1, we choose the path in S^2 with the shortest positive residual time:

$$S_{\max}^1 - S_i^1 - d_i^1 + S_j^2 - \text{dist}(i, j). \tag{6.6}$$

Once the path for vehicle 1 is chosen, the pairing for activity path a_2^1 is chosen among the remaining available paths, and so on. As we choose the optimal choice at each step without considering the overall solution, the activity path pairing will not necessarily lead to the optimal coupling. It is shown in Section 7.3 that the simple algorithm produces good results in practice. Still, if the greedy scheduling heuristic is to be developed further, it can be worthwhile to find an improved method based on an existing graph matching algorithm.

The pseudocode for the schedule concatenation algorithm is given in Algorithm 2. We start by finding the activity paths a_k^1 and a_k^2 for the different vehicles in both schedules and choose an activity path mapping h. The mapping details which specific paths in each schedule will be concatenated. For each vehicle $k \in \mathcal{R}$, a_k^1 and $a_{h[k]}^2$ is merged to create a new activity path a_k. The new activity paths are used to construct the flow variables f for the concatenated schedule. To find the updated start times of the activities, f is used as a warm start in model 1 with the objective:

$$\min \sum_{i=0}^{n+1} S_i, \tag{6.7}$$

and the added constraint:

$$S_{n+1} \leq S_{\max}^1 + S_{\max}^2. \tag{6.8}$$

The model aims at ordering the activities such that they are globally left-shifted while still making sure that the start times S_i stay below $S_{\max}$. Even with a warm start, model 1 can be time-consuming, and thus the optimization is only run for a short time to obtain an approximate activity

Algorithm 2: Schedule concatenation

```
Define schedules $S^1$ and $S^2$;
Find activity paths $a_k^1$ and $a_k^2$ $\forall\ k \in \mathcal{R}$;
Choose activity paths map $h$;
for $k \leftarrow 1$ to $K$ do
    Set $i$ to last delivery activity in $a_k^1$;
    Set $j$ to first pickup activity in $a_{h[k]}^2$;
    while $S_{\max}^1 - S_i^1 - d_i^1 + S_j^2 <$ dist$(i, j)$ do
        Discard first pickup and delivery in $a_{h[k]}^2$;
        Set $j$ to next activity in $a_{h[k]}^2$;
    end
    Create new vehicle path $a_k$ by merging $a_k^1$ and $a_{h[k]}^2$;
end
Create flow-variable $f$ based on $a_k$ $\forall\ k \in \mathcal{R}$;
Use $f$ as warm start in model 1 with a time limit;
```

ordering. The computation time of the algorithm depends on the efficiency of the method used to find the activity paths map h and on how long model 1 is run.

The schedule concatenation algorithm can be used to make long schedules with prioritized activities by consecutively merging small schedules made with continuously updated priorities. When making short schedules, we must ensure that the makespans are long enough to include all activity types. Otherwise, certain activities will never be included in the resulting concatenated schedule. Another consideration when concatenating is the boundary effect a schedule can exhibit towards the end. A schedule might behave differently close to $S_{\max}$ as a result of being close to finishing. This can result in activity patterns we would not observe in longer schedules. To alleviate the consequences of the boundary effects, we can remove the last m activities in each activity path in schedule S^1 before concatenation. In our scheduling problems, we do not impose that the vehicles have to start or end at any specific locations. Hence, we assume that any boundary effects are negligible.

6.3 Greedy scheduling heuristic

The way model 2 is formulated, feasible schedules are quickly produced if the n activities initially defined can all be completed before $S_{\max}$. However, the model slows down considerably when having to prove that a feasible ordering of all the activities is impossible and choose which activities to remove. We exploit this characteristic of model 2 to create a greedy scheduling heuristic. The heuristic iteratively adds activities to a schedule one by one until additional activities make the schedule infeasible or a time limit is reached. This method is thus not dependent on a predefined overhead n but adaptively adds activities based on the feedback from the current schedule.

The heuristic's efficiency depends on the method used to determine if a schedule is feasible or not. Instead of simply using model 2, which is slow to prove that a schedule is infeasible, we combine features from model 1 and model 2 to create model 3, defined as:

$$\max \quad q = \sum_{(i,j)\in F'} f_{ij} \tag{6.9}$$

$$\text{s.t.} \quad \sum_{j\in V_i^{\text{in}}} f_{ji} = 1, \qquad \forall\, i \in V \setminus \{0, n+1\}, \tag{6.10}$$

$$\sum_{j\in V_i^{\text{out}}} f_{ij} = 1, \qquad \forall\, i \in V \setminus \{0, n+1\}, \tag{6.11}$$

$$\sum_{j\in V_0^{\text{out}}} f_{0j} \le K, \tag{6.12}$$

$$\sum_{j\in V_{n+1}^{\text{in}}} f_{j(n+1)} \le K, \tag{6.13}$$

$$S_{n+1} \le S_{\max}, \tag{6.14}$$

$$S_j - S_i \ge d_i^p + p_j, \qquad \forall\, (i,j) \in E', \tag{6.15}$$

$$S_j - S_i \ge -M + (d_i + t_{ij} + M) f_{ij}, \qquad \forall\, (i,j) \in F, \tag{6.16}$$

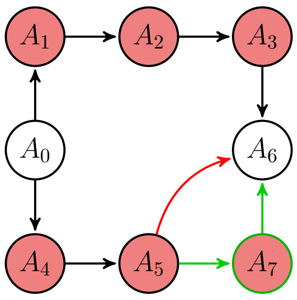
$$f_{ij} \in \{0,1\}, \qquad \forall\, (i,j) \in F, \tag{6.17}$$

$$S_i \in \mathbb{R}^+, \qquad \forall\, i \in V. \tag{6.18}$$

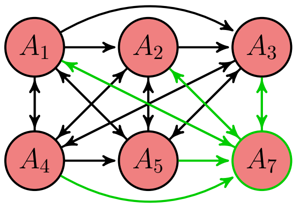
The goal of model 3 is to decide if a schedule is feasible or infeasible considering the precedence and resource constraints. The model's objective (6.9) is thus inconsequential for the optimization. Constraints (6.10) and (6.11) impose that all activities have to be completed, as in model 1, and constraint (6.14) from model 2 makes sure all activities must be finished within $S_{\max}$. The remaining constraints are common for both model 1 and model 2 and ensure that the schedules respects the resource allocations and time restrictions. We expect model 3 to be faster than model 2, both when it comes to proving feasibility and infeasibility, as the model knows beforehand that all activities must be completed.

To build schedules with the heuristic, activities are iteratively added to an initially empty plan. In order to add activities one by one, the activity definition from Section 6.1 must be used. Activities are added greedily by alternating between the different activity types created by the location mapping. Once a schedule is proven infeasible with the addition of a specific activity type, we refrain from adding activities of this type in the remainder of the algorithm. The process is terminated when the inclusion of no further activity types is feasible. When making schedules with prioritized activities, the activity types with the highest priorities are added to the schedule first. In this way, we can guarantee that the most time-sensitive activities are prioritized. As proving infeasibility can be time-consuming when the schedules are close to being complete, a cut-off point τ is chosen to avoid spending too much time on this step. As there is a trade-off between adding the most activities and reducing the time spent on proving infeasibility, the time limit τ must be chosen with care.

When adding new activities to a schedule, we must update the precedence and travel graphs. Instead of redefining the problem, we simply add the new activities into the current formulation. This makes it easy to add and remove activities quickly. To illustrate the process, we make an example by adding a new activity to the graphs in Figure 6.3. New activities always get the highest activity number in the graphs; thus, the activity will



(a) Precedence graph.



(b) Travel graph.

Figure 6.4: Illustration of the changes in the precedence and travel graph when a new activity is added in the example shown in Figure 6.3. The red and green edges indicate the edges we have to remove and add, respectively.

be denoted A_7. Assuming A_7 is a pickup and delivery at P_2-D_2, the activity is included in the precedence graph by simply appending it to the last non-sink activity at the location precedence chain. The activity must also be connected to the sink, and the previous connection from the penultimate activity to the sink has to be removed. This is illustrated in Figure 6.4(a). To add the activity to the travel graph, we do not have to remove any edges but simply add edges from the new activity to all activities at other locations and edges from all other activities to the new activity. This is shown in Figure 6.4(b).

The pseudocode for the greedy scheduling heuristic is given in Algorithm 3. We start by defining the project makespan $S_{\max}$ and the time limit τ for model 3. We construct a list ℓ of activities in the order we want to add them to the schedule. If there are no priorities on the activities, the different activity types are alternatingly added to ℓ. In the case of priorities, the activities in ℓ are ordered based on w. While the activity list is not empty, the algorithm adds the topmost activity in ℓ to the schedule. Model 3 is used to determine if the schedule with the added activity is feasible or not, either by proving this exactly or by reaching the time limit τ. If an activity is not accepted, all activities of the specific type are removed

Algorithm 3: Greedy scheduling heuristic

```
Define the project makespan $S_{\max}$;
Set the time limit $\tau$;
Construct list of activities $\ell$;
while $\ell$ is not empty do
    Add next activity in $\ell$ to schedule;
    Use model 3 to determine feasibility/infeasibility;
    if infeasible or timeout $\tau$ reached then
        Remove activity type from $\ell$;
    else
        Update parameters to include new activity;
    end
end
Solve model 1 with a time limit;
```

from ℓ. The algorithm terminates when ℓ is empty. After having added all the activities we can, we try to find an active schedule by globally left-shifting all the activities with model 1, similar to what was implemented in the schedule concatenation in Section 2.

The advantage of the greedy scheduling heuristic is that we have complete control over which activities are added to the schedules. This is especially important when long and time-consuming activities must be prioritized. Even though the algorithm is not exact, we have the chance of proving that the addition of more activities is infeasible for short schedules. If this is the case, the schedules are "optimal" in the sense that we can prove we do not have room for more activities. From experiments, schedules with around 30 activities can be proven "optimal" in this regard. Thus, the heuristic can be used in combination with the scheduling concatenation. Small schedules quickly made with the heuristic can be merged into feasible schedules with large scheduling horizons.

The heuristic is challenging to use for creating long schedules because of the difficulty of choosing the time limit τ. If we meet the time limit at the addition of every new activity at the end of the algorithm, the computation time will be at least τ times the number of activity types. Thus, the time limit cannot be chosen too large to avoid spending too much time proving infeasibility. Setting the time limit too low, we risk discarding feasible schedules.

A key component to keeping the heuristic efficient is the greedy activity addition. However, instead of simply adding activities one by one, we can think of situations where it would be profitable to remove certain activities to maximize q or q_w. Achieving this would include comparing several schedules made with different activities to determine which should be removed and added and would significantly increase the computation time.

Chapter 7

Simulation study

We conduct a simulation study to test the different models and methods we have developed to solve the Mass Relocation Problem. Section 7.1 starts by examining the features of model 1 and model 2, showing how the models become inefficient and computationally demanding when we are scheduling long projects. In Section 7.2 we test the implementation of Benders decomposition and compare it to the standard branch-and-cut MIP solver implemented in Gurobi in terms of computation speed and solution quality. Lastly, we investigate the effectiveness and correctness of the inexact methods developed in Chapter 6 and compare the results to schedules made with the exact methods. The computations are performed on NTNU's computation server Markov, using 10-20 threads for parallel computing. We limit ourselves to studying schedules of a maximum of 6 hours.

7.1 Model features

We start the simulation study by testing how the computation time of model 1 and model 2 vary with the number of activities n, the number of vehicles K, and the fixed time horizon $S_{\max}$ when using exact solution methods. We simulate random construction sites of different sizes and complexities to estimate the average performance. A problem instance is generated by determining the number of pickup and delivery locations, the number of activities at each location, and the distance between the locations. A random discrete uniform distribution on the interval $[2, 5]$ is used to determine the number of pickup locations, n_1, and delivery locations, n_2. Given a total number of activities n, we use a multinomial distribution to distribute $\frac{n}{2}$ activities among the n_1 pickup locations and the n_2 delivery locations. Each location has an equal probability of being assigned an activity. Lastly, to simulate the distances between the pickup and delivery locations, we draw from a uniform distribution on the interval $[3, 20]$, such that the driving times range from 3 to 20 minutes. For every problem size, we generate and solve ten different problem instances with different topography. The goal is to explore model characteristics, and thus, the experiments are completed using the built-in MIP solver in Gurobi. We set the pickup and delivery durations to 5 and 3 minutes, respectively, and the mass preparation time to 4 minutes.

We start by investigating the computation time of model 1 as a function of n, using a fixed number of vehicles $K = 3$. The results are presented in Figure 7.1. The solid line shows the average computation time of an optimal schedule on ten randomly generated problem instances with n activities. The small marks indicate each problem's particular computation time, and the shaded area shows the span between the fastest and slowest problem instances to be solved. The grey dotted line shows the 2-hour time limit used for each problem instance. The instances where no solutions are found within the time limit contribute to the average with 2 hours. We see that the computation time increases roughly exponentially with n. Problem instances with many activities quickly grow in size and complexity, and as the solution space increases, so does the time to find an optimal schedule. The expansion of the shaded area for larger n indicates that the complexity of the topography of the construction site impacts the difficulty of the scheduling. At $n = 24$, some problem instances are solved in minutes, while others are not solved within the 2-hour time limit. This tells us that even though a model might work well for a particular MRP instance, it might not converge as fast on other construction sites. Instances that

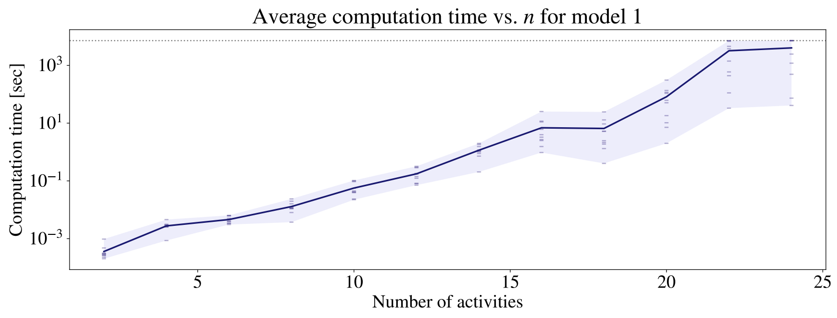


Figure 7.1: Average computation time as a function of number of activities for model 1 with $K = 3$. Calculated using ten randomly generated problem instances at each n.

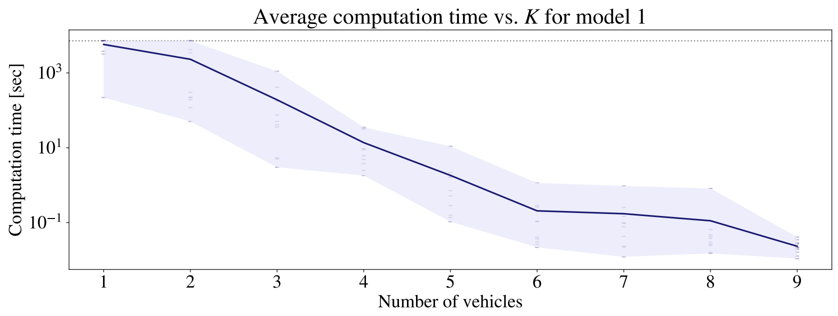


Figure 7.2: Average computation time as a function of number of vehicles K for model 1 with $n = 20$. Calculated using ten randomly generated problem instances at each K.

are solved fast usually contain fewer pickup and delivery locations, which makes the routing of the dump trucks easier.

We also investigate the effect the number of dump trucks has on the computation time for model 1. Figure 7.2 shows the average computation time as a function of K with $n = 20$. The figure indicates that increasing the number of resources decreases the computation time. Scheduling $n = 20$ activities with one dump truck can take over 2 hours while scheduling the same activities with nine dump trucks takes less than a second. The reason for this was discussed in Section 4.3; with few vehicles, many different orderings of the activities can provide feasible and optimal schedules, and these symmetries can slow down the optimization. With many similar dump trucks, where we do not differentiate between which vehicle does what activity, the solution room is smaller, and multiple optimal solutions are not as abundant. This means that, generally, a bigger fleet of homogeneous vehicles will provide feasible schedules faster. However, scheduling with many resources will also become time-consuming once the number of activities grows.

For model 2, we want to see how the chosen scheduling horizon and activity overhead affect the program's performance. Figure 7.3 shows the average computation time as a function of $S_{\max}$, when $n = 30$ and $K = 3$. Green instances represent schedules where all 30 activities can be completed within $S_{\max}$. Problem instances that have not finished within the 2-hour time limit are marked with orange. The figure shows that all instances where the optimal solution is found before the time limit are green, indicating that the model is having problems proving that only a subset of the n activities can be completed within $S_{\max}$. If all activities can be scheduled with a good margin, like for $S_{\max}$ greater than 200 minutes, optimal schedules are quickly found. For problems with a smaller scheduling horizon, like 50 minutes, it is obvious that every activity cannot be completed in time, but model 2 still struggles to prove this. The branch-and-cut method in Gurobi manages to increase the lower bound on the objective

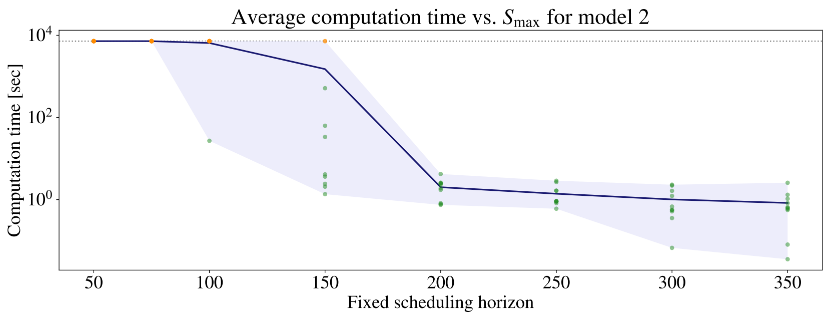


Figure 7.3: Average computation time as a function of the fixed time horizon $S_{\max}$ for model 2 with $n = 30$ and $K = 3$. Calculated using ten randomly generated problem instances at each $S_{\max}$.

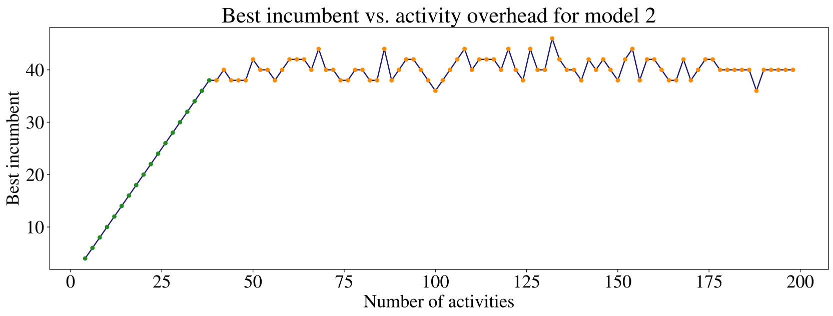


Figure 7.4: Best incumbent solution as a function of number of activities with $S_{\max} = 100$, $K = 3$ and a time limit of 10 minutes for each problem instance.

$q = \sum_{(i,j) \in F'} f_{ij}$ reasonably fast but is unable to lower the upper bound. This feature makes it difficult to utilize model 2 to maximize the number of activities. However, the model can still be used as a heuristic by using the lower bound on q as a solution after a certain amount of time.

Figure 7.4 shows how the best incumbent solution of model 2 changes as a function of n. A single construction site with a fixed number of pickup and delivery locations was used to generate the figure. For each n, the activities were randomly distributed between the locations. With $S_{\max} = 100$ and $K = 3$, the best solutions of the problem instances after 10 minutes are plotted. Instances with fewer than approximately 40 activities are quickly solved to optimality, but instances with a large activity overhead are not solved within the 10 minutes time limit. We see that even when the activity overhead increases, which produces larger and more complex scheduling problems, roughly the same solution is reached for problems with up to $n = 200$. This result indicates that the initial activity overhead does not change the solution significantly when using model 2 as a heuristic by stopping the optimization early. Choosing a suitable activity overhead is thus most crucial when using model 2 to find exact solutions.

Figure 7.5 shows the average computation time as a function of K for model 2, with $S_{\max} = 150$ and $n = 30$, and verifies that maximizing q is faster with more vehicles to complete the activities. This is not unnatural, as we can process activities much faster with more vehicles available. None of the schedules where we can do fewer than 30 activities are solved to optimality. Thus, the figure does not indicate if the model is faster at finding new incumbents with more vehicles present. However, as this has been shown to remove some of the symmetries in model 1, we assume this is also the case for model 2.

Based on the previous analysis of model 2, showing that we can schedule n activities before $S_{\max}$ is fast, but proving that we cannot is slow. To determine the infeasibility of a schedule faster, model 3 was developed in Section 6.3. The model simply checks if scheduling n activities within a

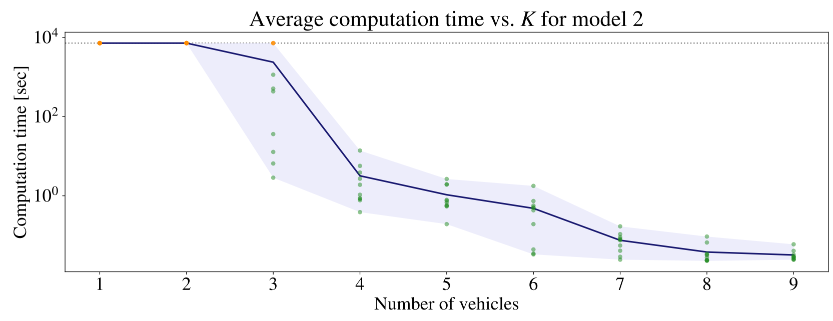


Figure 7.5: Average computation time as a function of number of vehicles K for model 2 with $S_{\text{max}} = 150$ and $n = 30$. Calculated using ten randomly generated problem instances at each K.

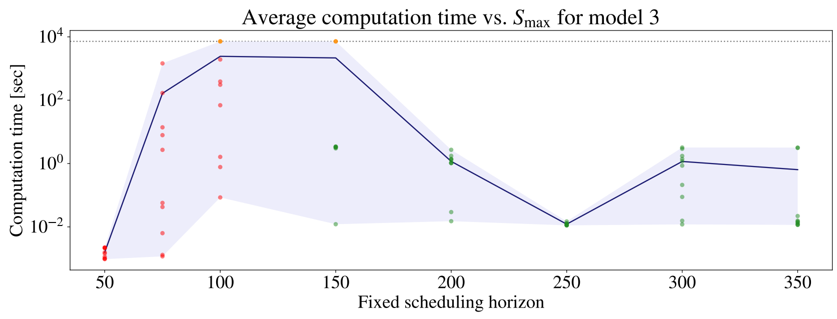


Figure 7.6: Average computation time as a function of the fixed time horizon S_{max} for model 3 with $n = 30$ and $K = 3$. Calculated using ten randomly generated problem instances at each S_{max}.

fixed scheduling horizon is feasible or infeasible and can be used to more quickly determine if n is too large. Figure 7.6 shows the average computation time of model 3 as a function of $S_{\max}$ with $n = 30$ and $K = 3$. As before, green instances represent schedules where all 30 activities are completed within $S_{\max}$, while red instances show schedules where $n = 30$ is infeasible. The figure shows that problems are easily classified when the activity margin is large but takes longer when the activities are close to being a perfect fit. Compared to Figure 7.3, model 3 manages to show that many instances below $S_{\max} = 100$ are infeasible very quickly, which we were not able to determine with model 2. This indicates that model 3 is better suited for the greedy scheduling heuristic.

7.2 Benders decomposition

The results from the previous section indicate that the branch-and-cut MIP solver in Gurobi is unable to solve large MRP instances to optimality within a reasonable time. In this section, we test if the implementations of Benders decomposition for model 1 and model 2 can find optimal schedules more efficiently by decomposing the large mixed-integer linear programs. To assess the effectiveness of the different algorithmic enhancements studied in Section 5.3, four Benders decomposition models are developed: Standard BD, Enhanced BD, Combinatorial BD, and Lazy BD. The Standard BD model is implemented using the basic Benders decomposition algorithm outlined in Algorithm 1. The Enhanced BD algorithm is implemented with cut strengthening techniques; feasibility cuts are strengthened by normalizing the unbounded rays, and optimality cuts are strengthened by using Pareto-optimal cuts. The Combinatorial BD generates combinatorial cuts instead of feasibility cuts, in addition to Pareto-optimal optimality cuts. Lastly, Lazy BD is implemented as the standard BD algorithm, except with the addition of lazy constraints. The different models are created to recognize and isolate the improvements beneficial for the optimization.

We start by investigating the average computational time vs. n for model 1 with the four BD algorithms. We also include results from the Gurobi MIP solver for comparison. The results are shown in Figure 7.7. The figure shows the average computation time of three randomly generated problem instances for every even $n \in \{2, \dots, 20\}$ with $K = 3$. A time limit of 1 hour is used for each problem instance during the optimization. We observe that the MIP solver is considerably faster than either of the BD implementations, which are unable to find optimal solutions for instances with $n > 12$ within the time limit. The difference in computation time between the four BD algorithms is minimal, though the Combinatorial BD seems slightly slower than the other methods.

To evaluate which of the BD algorithms perform best, we compare the upper bounds of the methods at termination. The upper bound of a minimization problem represents the best feasible solution found thus far and indicates how close the methods are to reaching the optimal solution. Figure 7.8 shows the upper bounds for the individual problem instances with $n \geq 12$. The individual problem instances, ordered by size, are shown on the x-axis, while the upper bounds of the problems are shown on the y-axis. The red line, corresponding to the solutions from the MIP solver, shows the optimum for each instance. We see that even though the Combinatorial BD algorithm was the slowest, it is also the one with upper bounds generally closest to the optimal solutions, closely followed by the Lazy BD. The upper bounds of the Standard BD algorithm get larger as the problem instances grow. The same is true for the Enhanced BD, which struggles to find any upper bounds once the problems become too big.

A selection of the results for $n = 14$ and $n = 20$ is shown in the first two sections of Table 7.1. The columns UB and LB indicate the upper and lower bounds at termination, while FC and OC represent the number of feasibility and optimality cuts, respectively. We see that the MIP solver finds optimal solutions in less than a minute while all BD algorithms run until the timeout. The Enhanced BD generally produces fewer optimality

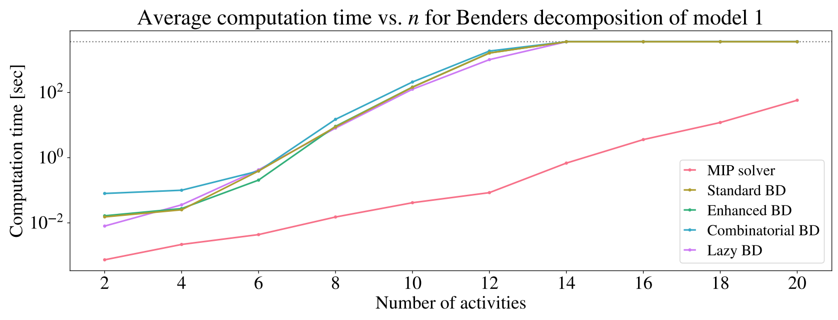


Figure 7.7: Average computation time as a function of n for Benders decomposition of model 1 with $K = 3$. Calculated using three randomly generated problem instances at each n.

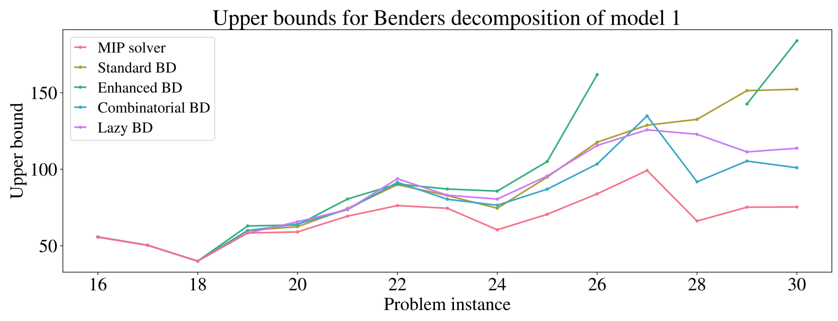


Figure 7.8: Upper bounds for problem instances with $n \geq 12$ solved with Benders decomposition in Figure 7.7 with $K = 3$. The individual problem instances, ordered by size, are shown on the x-axis, while the upper bounds of the problems are shown on the y-axis. Missing values indicate that no upper bound was found.

n	Model	Time [min]	UB	LB	FC	OC
14	MIP solver	0.02	69.36	69.36	–	–
	Standard BD	60	74.33	0	2868	3878
	Enhanced BD	60	80.55	0	4138	2012
	Combinatorial BD	60	73.66	0	3312	4166
	Lazy BD	60	73.66	0	2334	6556
20	MIP solver	0.27	75.38	75.38	–	–
	Standard BD	60	152.30	0	2007	6
	Enhanced BD	60	184.00	0	1320	1
	Combinatorial BD	60	101.01	0	6470	392
	Lazy BD	60	113.77	0	1903	511
60	MIP solver	240	175.60	174.90	–	–
	Standard BD	240	∞	0	96	0
	Enhanced BD	240	∞	0	92	0
	Combinatorial BD	240	∞	0	1276	0
	Lazy BD	240	∞	0	143	0

Table 7.1: Results from Benders decomposition of model 1 for problem instances with $n \in \{14, 20\}$ and $K = 3$, and with $n = 60$ and $K = 6$. The columns UB, LB, FC and OC represent the upper bound, lower bound, number of feasibility cuts and number of optimality cuts at termination, respectively.

cuts than either of the other methods, indicating that the normalized feasibility cuts are weaker than the unaltered cuts. The Combinatorial BD produces many feasibility and optimality cuts, indicating that the iterations of the method are generally faster than those of the Standard and Enhanced BD. It also implies that the combinatorial cuts are well suited for the MRP. The Lazy BD produces more optimality cuts than any other methods, implying that the lazy addition of cuts is beneficial for updating the upper bounds faster.

Methods like Benders decomposition usually include an overhead before becoming more efficient than more straightforward solvers. To test if this is the case, we solve a large problem instance with $n = 60$ and $K = 6$. The results are shown in the last rows of Table 7.1. After a time limit of 4 hours, the Gurobi MIP solver reaches an optimality gap of 0.40 %, while none of the BD methods manage to find an incumbent solution. The decrease in feasibility cuts indicates that the master problem becomes too computationally demanding to solve to optimality at each iteration in the implemented BD algorithm.

For Benders decomposition of model 2, the average computation time as a function of n, with $K = 3$ and $S_{\max} = 100$, is shown in Figure 7.9. The figure indicates similar results for model 2 as for model 1. The MIP solver finds the optimal solutions for almost all instances and is considerably faster than the BD methods on small problems. The Standard BD method is generally the fastest of the decomposition methods, likely due to the fact that it does not have to solve any additional linear programs in each iteration.

Table 7.2 shows a selection of results for $n \in \{16, 20, 24\}$. The BD methods do not find optimal solutions for $n = 20$ or $n = 24$ within the 1-hour time limit. Because of how the Benders decomposition of model 2 is implemented, no optimality cuts are generated during the optimization. The Combinatorial BD method converges with the fewest feasibility cuts when a solution is found, though it uses more time on each iteration compared with the other methods. Similarly to model 1, the number of generated cuts decreases with problem size when the optimal solutions are not found. In these cases, neither the implementations of Benders decomposition nor the MIP solver manages to find incumbent solutions with $q < n$. As no lower bounds are generated in the BD methods because of the lack of optimality cuts, no approximate solutions can be obtained by simply cutting off the optimization early. This indicates that the current implementation of the BD algorithm for model 2 is less than ideal, and other implementations which produce a lower bound should be explored.

Based on the results, we conclude that applying Benders decomposition to model 1 and model 2 is not beneficial for solving the Mass Relocation Problem more efficiently. The methods suffer from an inefficient implementation and spend too much time solving the master problem in each

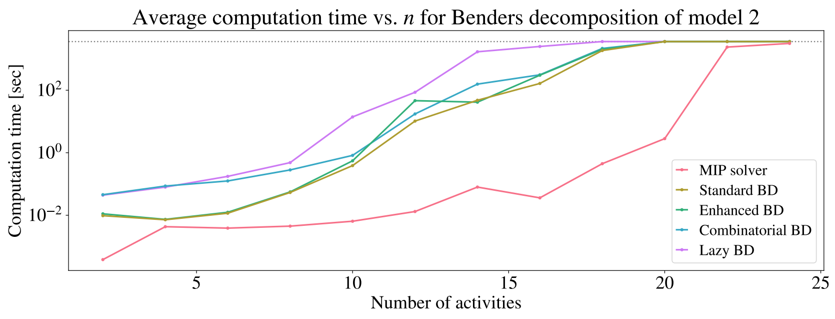


Figure 7.9: Average computation time as a function of n for Benders decomposition of model 2 with $K = 3$. Calculated using three randomly generated problem instances at each n.

n	Model	Time [min]	UB	LB	FC
16	MIP solver	0.00	16	16	–
	Standard BD	5.03	16	16	402
	Enhanced BD	4.63	16	16	301
	Combinatorial BD	5.48	16	16	279
	Lazy BD	43.18	16	16	508
20	MIP solver	0.08	20	20	–
	Standard BD	60	20	–	2062
	Enhanced BD	60	20	–	183
	Combinatorial BD	60	20	–	1257
	Lazy BD	60	20	–	338
24	MIP solver	60	24	22	–
	Standard BD	60	24	–	1034
	Enhanced BD	60	24	–	107
	Combinatorial BD	60	24	–	759
	Lazy BD	60	24	–	122

Table 7.2: Results from Benders decomposition of model 2 for problem instances with $n \in \{16, 20, 24\}$ and $K = 3$. The columns UB, LB and FC represent the upper bound, lower bound and number of feasibility cuts at termination, respectively.

iteration. Time-consuming iterations lead to fewer cuts being generated, and upper and lower bounds update slowly. How the algorithm is implemented is crucial to its success, and effort should go into making an efficient implementation that accounts for our specific problem structure. With the current decomposition, the master problems do not become simple enough to redeem being solved multiple times in the iterative algorithm. Therefore, a two-tier Benders decomposition where the master problem is further simplified could be an option.

Because of its inefficiency, we refrain from using Benders decomposition to solve further scheduling problems in this thesis and solely rely on the branch-and-cut algorithm implemented in Gurobi.

7.3 Inexact methods

In this section, the inexact scheduling methods developed in Chapter 6 are tested. Inexact schedules made with the location mapping, schedule concatenation, and the greedy scheduling heuristic are compared to schedules made from solving model 2 with a cut-off time. The schedules are compared in terms of computation time, number of included activities q, sum of prioritized activities q_w, and vehicle idle time. The goal is to identify which methods can quickly provide feasible schedules with as little inactive time for the vehicles as possible. Schedules are made using data from the simulated construction site in Figure 7.10. The site consists of four pickup locations and three delivery locations. The travel times are randomly generated using a uniform distribution on the interval $[5, 20]$ minutes. We assume the travel times are symmetric; thus t_{ij} and t_{ji} are equal.

Ideally, we would like to test the inexact methods on many different problem instances, as we know performance varies depending on construction site topography. As an exhaustive study is too time-consuming for this thesis, the generated example is simulated to emulate a typical construction site. Hence, the performance of the methods is hopefully generalizable

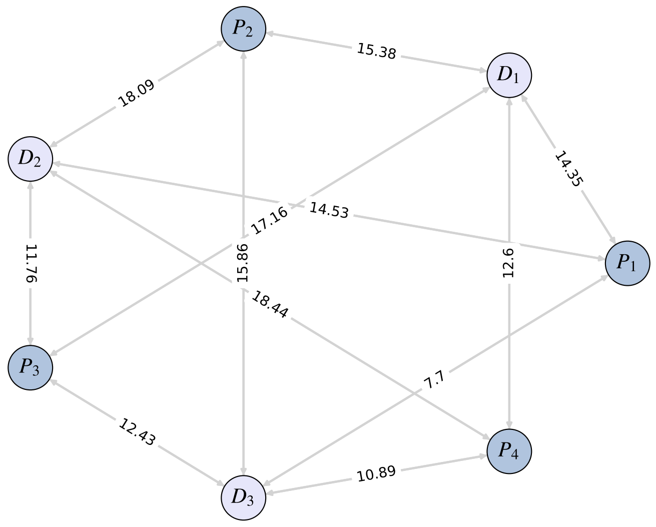


Figure 7.10: Simulated construction site with four pickup and three delivery locations. The weights on the edges show the symmetric travelling times in minutes.

to other Mass Relocation Problem instances. We focus on finding schedules for 90, 180, and 360 minutes using $K = 6$ dump trucks. As in the last sections, we assume every pickup and delivery duration takes 5 and 3 minutes, respectively, and that the mass preparation takes 4 minutes.

We start by making schedules with separate pickup and delivery activities, using model 2 with and without priorities. Approximate solutions are found using the Gurobi MIP solver with a cut-off time. Applying the location mapping, schedules are produced in a similar fashion for the new activity type defined in Section 6.1. These are compared to schedules created with the greedy scheduling heuristic, both with and without priorities. Finally, the schedule concatenation algorithm is used to concatenate the schedules made with the other methods. Schedules of 180 minutes are created by concatenating two 90-minute schedules, and schedules of 360

Location	Activities	Due date
P_1	20	15
P_2	40	10
P_3	50	5
P_4	30	8
D_1	40	10
D_2	30	8
D_3	70	10

Table 7.3: Randomly simulated plan for how many activities we have to complete within a due date on the construction site in Figure 7.10 without the location mapping. The plan is used to create priorities for the activities.

minutes are created by concatenating two 180-minute or four 90-minute schedules.

The location mapping we use is chosen by pairing locations randomly, and is given by: P_1-D_3, P_2-P_1, P_3-D_3 and P_4-D_2. As we have more pickup locations than delivery locations, two pickups are mapped to the same delivery. This means that we cannot enforce the precedence relations between the delivery activities at this location and that these have to be added in an additional post-processing step. Plans for the activities with and without the location mapping are given in Tables 7.3 and 7.4. The plans are constructed such that activities at the same locations get approximately the same prioritized ordering in both systems. In this way, we can make a fair comparison between the schedules and the prioritized sums q_w. The priorities for the individual activities are defined as described in Section 4.4.1.

When making schedules with model 2, a cut-off time equal to the length of the schedule is chosen. For the problems without the location mapping, an arbitrary activity overhead n is chosen such that we have the same number of activities at each location. The reasoning behind this is that we would not have time to find an approximate activity overhead at each location in a trial and error manner when utilizing the scheduling method on a real construction site. For the prioritized activities, the same activity

Location	Duration [min]	Rounds	Due date
P_1-D_3	7.70	20	15
P_2-D_1	15.38	40	10
P_3-D_3	12.43	50	5
P_4-D_2	18.44	30	8

Table 7.4: Randomly simulated plan for how many activities we have to complete within a due date on the construction site in Figure 7.10 with the location mapping. The durations indicate how long it takes to complete an activity at the locations. The plan is used to create priorities for the activities.

overheads are chosen.

When using model 2 with the location mapping, we can use the activity estimation from Section 6.1 to give an initial activity overhead. The estimation allows us to calculate how many rounds each vehicle can complete between the mapped pickup and delivery locations within $S_{\max}$. These estimates only represent how many activities one vehicle can complete. As we have more dump trucks than locations, we multiply the location estimates by three. This number is chosen because it is unlikely that more than three vehicles will work at the same location at a time, as this would induce delays. The variable γ is set to zero as we do not worry about overestimating the number of activities at each location. A similar procedure is completed to estimate the number of activities with priorities, except more activities from the prioritized types are added and fewer of the less prioritized ones. After producing the initial schedules, they are post-processed to enforce the precedence relations between the delivery activities at D_3.

The resulting schedules from model 2, with and without the location mapping and priorities, are shown in Table 7.5. The column M indicates if the activities are from the location mapping or not. Similarly, the column P signals whether the schedules are made with or without priorities. The time represents the cut-off time of the optimization, and the optimality gap represents the difference between the upper and lower bounds at termina-

$S_{\max}$	M	P	Time [min]	Opt. gap	n	q	q_w	Idle time
90	–	–	90	157.14 %	72	28	142.90	25.00 %
	–	✓	90	85.60 %	72	30	205.58	16.24 %
	✓	–	90	87.50 %	60	32	170.40	17.68 %
	✓	✓	90	19.77 %	48	32	219.28	14.52 %
180	–	–	180	54.84 %	96	62	311.00	16.81 %
	–	✓	180	49.06 %	96	54	328.43	16.59 %
	✓	–	180	93.55 %	120	62	394.80	12.86 %
	✓	✓	180	52.29 %	100	62	371.88	9.54 %
360	–	–	360	38.46 %	144	104	521.38	21.96 %
	–	✓	360	15.71 %	144	108	584.73	14.14 %
	✓	–	360	118.64 %	258	118	567.55	9.08 %
	✓	✓	360	34.30 %	216	118	676.95	12.18 %

Table 7.5: Results for schedules made with model 2 with a cut-off time equal to $S_{\max}$. Schedules for 90, 180 and 360 minutes are made with and without location mapping and priorities, as indicated by the columns M and P, respectively.

tion. The variable n gives the initial activity overhead counted in separate pickup and delivery activities, even when the location mapping is utilized. The variables q and q_w, calculated using the definitions in (4.19) and (4.32), give the number of separate pickup and delivery activities included in the schedules and the sum of the activities' priorities.

The idle time represents the inactive time in a schedule. The idle time for an individual vehicle is calculated by summing up its total active time, consisting of the activity durations and the travel times, and subtracting this from $S_{\max}$. The total percentage of idle time for all vehicles is then given by:

$$\frac{\sum_{k=1}^{K} I_k}{K \cdot S_{\max}}, \tag{7.1}$$

where I_k is the idle time of vehicle $k \in \mathcal{R}$. The number provides a simple metric to compare how well activities are fitted into the schedules. The variable q is not a good estimator of this in the situations where we add prioritized activities, as a high q might not be equivalent to a high q_w.

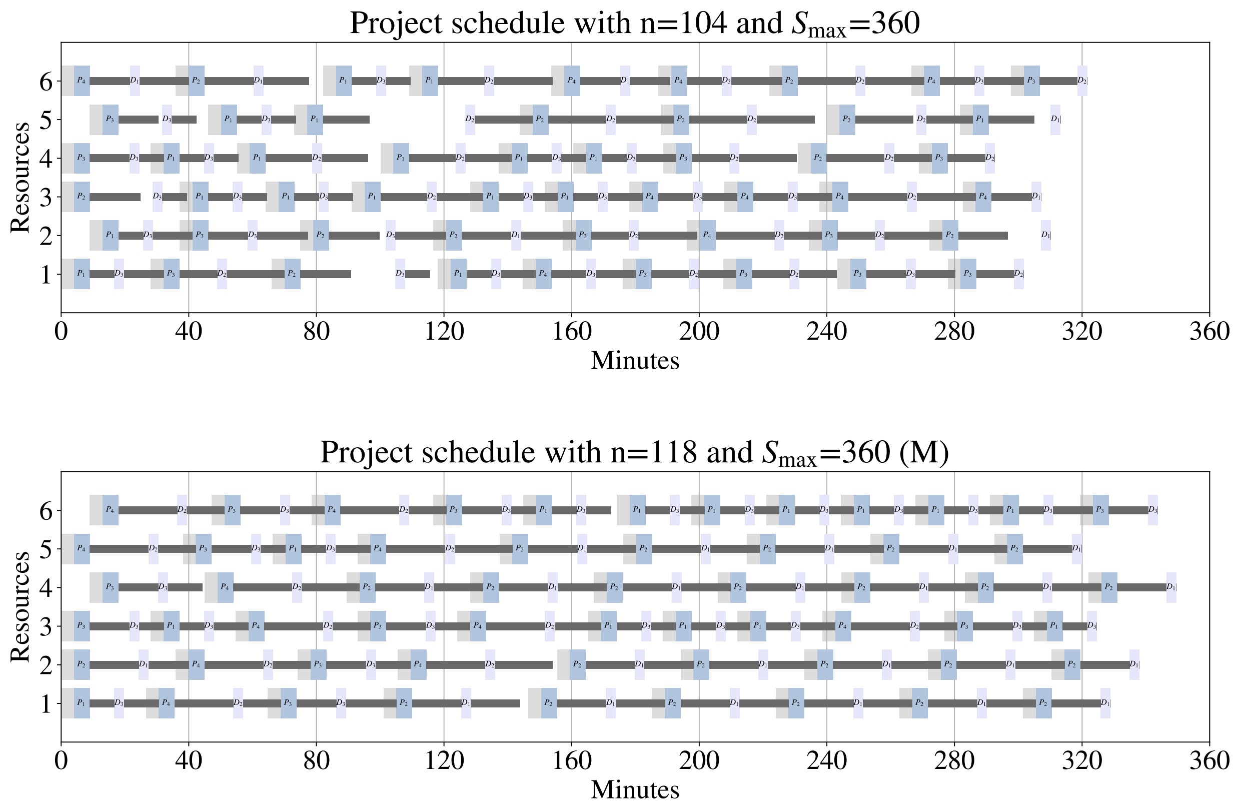


Figure 7.11: Schedules of 360 minutes created with model 2 and a cut-off time of 6 hours. *Top:* Schedule made without the location mapping. *Bottom:* Schedule made with the location mapping. Schedules with higher q-values are obtained with the location mapping.

Most of the idle time stems from the end of the schedules when vehicles do not have time to complete additional activities before $S_{\max}$.

The results from Table 7.5 indicate that schedules made without the location mapping generally include fewer activities and more idle time than the schedules made with the new activity type. This is partly due to the difficulty of estimating a suitable activity overhead at the locations with separate pickups and deliveries, as too few or too many of certain activities can make the scheduling difficult. Since the models with the location mapping only include $\frac{n}{2}$ activities during the optimization, it is also expected that better solutions are obtained because of the smaller problem sizes. An example of two 360-minute schedules with and without the location mapping is shown in Figure 7.11. The first schedule made with the old activity type includes 104 activities and has 21.96 % idle time for the vehicles. On the other hand, the schedule made with the mapped locations fits a total of 118 activities and has an idle time of just 9.08 %. This illustrates the effectiveness of reducing the size of the MRP instances with the location mapping.

Comparing the schedules with and without priorities, the optimality gaps in the prioritized schedules are smaller than in their unprioritized counterparts. The optimization becomes easier when we want to maximize the prioritized sum q_w, compared to simple maximizing q, as some of the symmetries in the problems disappear when the activities have weights. The idle times of the prioritized schedules are also generally lower, both for schedules made with and without the location mapping. This indicates that the chosen activities better fill out the scheduling horizons. As the value of q_w is generally higher in the prioritized schedules, the prioritization of activities by simply changing the objective of model 2 seems promising.

In general, we would expect the number of activities q to be lower in schedules with prioritized activities, as is seen in the 180-minute schedule without the location mapping. Even though the number of activities added to the schedule is lower, the vehicles spend more of their time active com-

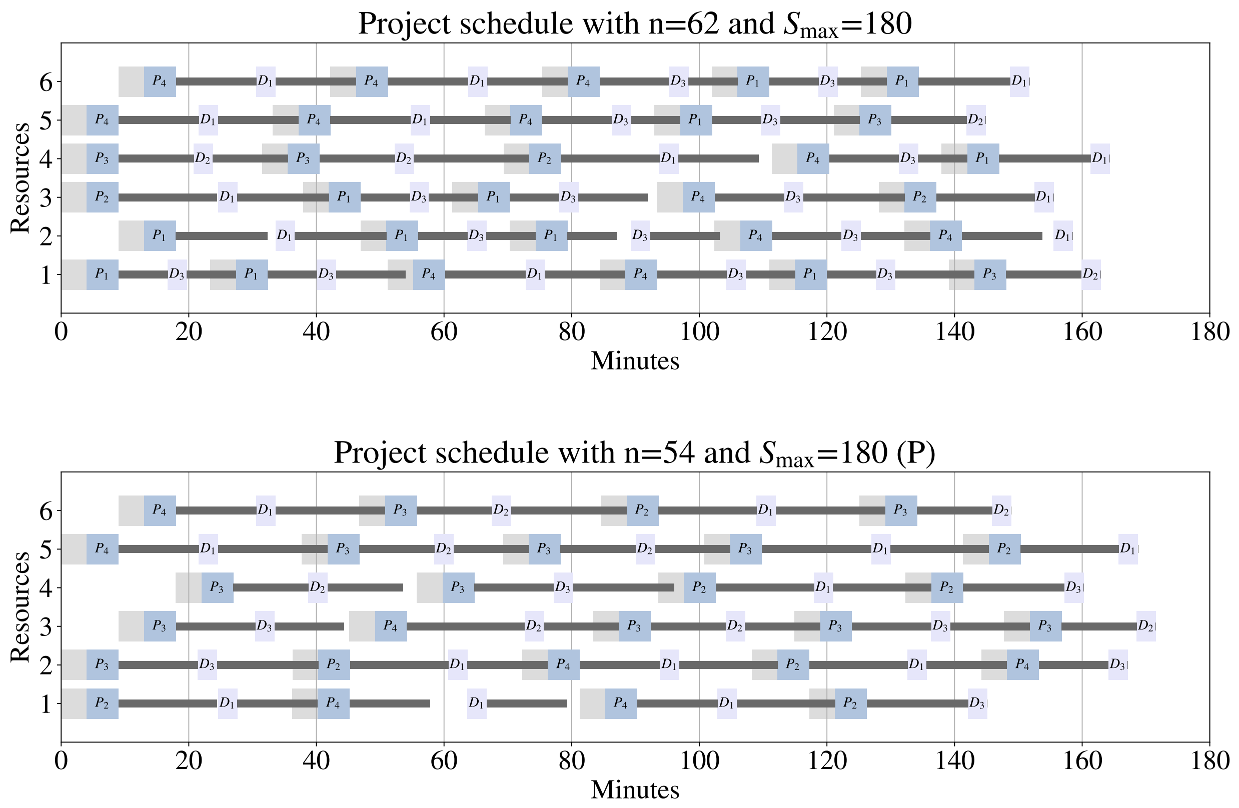
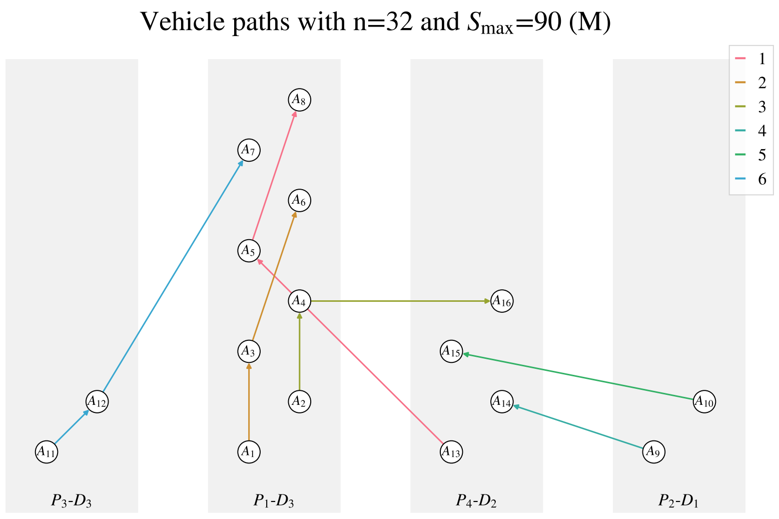


Figure 7.12: Schedules of 180 minutes created with model 2 and a cut-off time of 3 hours. *Top:* Schedule made without the location mapping and priorities. *Bottom:* Schedule made without the location mapping and with priorities. The schedule with prioritized activities has a lower q but a smaller idle time as the driving distances between the activities are longer.



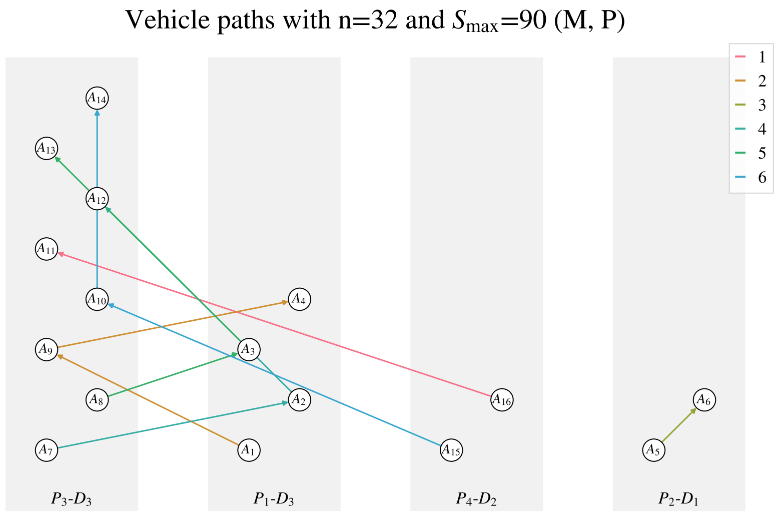


Figure 7.13: Driving paths of the $K = 6$ dump trucks in the 90-minute schedules created with model 2, with the location mapping and a cut-off time of 1.5 hours. *Top:* Driving paths without priorities. *Bottom:* Driving paths with priorities. The driving paths illustrate that the addition of the priorities changes the activities included in the schedules.

pared to the schedule without priorities. The reason for this is that the prioritized schedule includes activity paths with longer travelling times to maximize q_w. This is shown in Figure 7.12. For the schedules of 90 and 360 minutes, the number of activities in the prioritized schedules is higher or equal to the non-prioritized ones. This is because the optimization becomes easier once several of the symmetries in the problem formulations are removed, and thus more activities are proven to fit in the schedules within the cut-off time.

To illustrate how priorities change the activity composition of the schedules, we compare the number of different activity types in the two 90-minute schedules made with the location mapping. Based on the duration of the activities given in Table 7.4, the activities at P_1-D_3 are the fastest to complete. We thus expect many of these activity types in the schedules where we maximize q. According to the plan, the activities with the highest priorities are at P_3-D_3. For schedules maximizing q_w, more of these activities should thus be priorities. This is observed in Figure 7.13. The plots show the vehicles' driving paths between the activities in the schedules. Even though we complete the same number of activities in the two schedules, the first schedule favors the quicker activity type. In contrast, the second schedule favors the activities with the highest priorities.

Table 7.6 shows the results from using the greedy scheduling heuristic to make schedules with and without priorities. Time limits of 10, 20 and 30 minutes are chosen for the 90, 180 and 360 minute schedules, respectively. The time limit is increased to consider the growing difficulty of proving infeasibility for long schedules. In this simulation study, the time limits are chosen somewhat arbitrarily. However, when regularly creating schedules on a construction site, more reasonable limits can be chosen in a trial and error manner based on how many activities we usually include in a schedule. The column "infeasibility proved" indicates which activity types are proven infeasible when added to a schedule and which are just discarded because of the time limit. These are marked by a one and a zero, respectively.

$S_{\max}$	P	Time [min]	Infeasibility proved	q	q_w	Idle time
90	–	0.04	1—1—1—1	32	188.75	13.56 %
	✓	0.01	1—1—1—1	32	218.48	14.07 %
180	–	85.50	0—0—0—0	64	358.60	5.60 %
	✓	23.24	1—1—1—0	60	406.00	10.09 %
360	–	132.74	0—0—0—0	120	614.75	9.21 %
	✓	127.73	0—0—0—0	112	668.25	11.63 %

Table 7.6: Results from the greedy scheduling heuristic on schedules of 90, 180 and 360 minutes with and without priorities. The presence of priorities is indicated by the column P. Time limits τ of 10, 20 and 30 minutes are chosen for the schedules, respectively. The column "infeasibility proved" indicates how many of the four activity types are proven infeasible when added to the schedule (1), and how many are removed due to reaching the time limit (0).

The results show that short schedules are produced very quickly with the greedy heuristic. The 90-minute schedules include as many activities as the best 90-minute schedules from Table 7.5 and have equal or lower idle times. The schedules from the heuristic are found in seconds with the confidence that no further activities can be added to the schedules. We do not get the same security with the approximate schedules found with model 2.

The time limit is reached for almost all activity types for the longer schedules, but this does not seem to impair the results. The heuristic still uses less time than model 2 and achieves similar or even better values for q, q_w, and the vehicle idle time, even though the activities are added greedily. In the 180-minute schedules, the presence of priorities makes proving infeasibility easier, as the schedules become completely saturated with certain activity types. The scheduling heuristic thus shows promising results compared with the schedules from model 2.

The results from concatenating schedules made with model 2 and the greedy scheduling heuristic are shown in Tables 7.7 and 7.8. The column "units" indicates the building blocks used to make the concatenated schedules, and n indicates the maximum number of activities that can be included based on the unit schedules' number of activities. When merging schedules, a quick left-shifting of the activities is completed to condition the schedules the best way for the concatenation. As we base the merging on the remaining time in the activity paths in the first schedule, it is crucial that the activities are as left-shifted as possible. Still, we do not want to spend too much time on this step. Therefore, a left-shifting is completed using model 1 for five minutes with the objective (6.7) and the added constraint (6.8). The total time for the schedule concatenation thus includes the original time for making the schedules and the time for left-shifting them, in addition to the time spent on the procedure in Algorithm 2. When concatenating schedules without priorities, the same schedules are concatenated with themselves, but when concatenating prioritized schedules, different schedules with updated priorities must be created and merged. Thus, making concatenated schedules with priorities is more time-consuming.

The concatenated schedules in Table 7.7 generally produce similar or better results than the non-concatenated schedules in Table 7.5, especially for the 360-minute plans. When several schedules do not have to be computed to maximize q_w, the method also produces results faster. We observe that for most concatenated schedules, the difference between n and q is between 0-2, with one example of four. This means that the concatenation process generally manages to retain most of the activities in the merged schedules, even when a perfect matching of the activity paths is not used.

The results of the concatenation of schedules from the greedy scheduling heuristic are shown in Table 7.8. When combining the schedule concatenation and the heuristic, the resulting schedules are fed through the greedy scheduling algorithm after the concatenation to see if we can further increase the number of activities included. The time column in the table indicates the time it takes to make both the concatenated schedules and

$S_{\max}$	Units	M	P	Time [min]	n	q	q_w	Idle time
180	90	–	–	100	56	56	271.45	19.59 %
		–	✓	195	62	58	368.13	18.44 %
		✓	–	100	64	62	313.50	15.89 %
		✓	✓	195	62	62	401.38	11.49 %
360	90	–	–	110	112	112	485.90	16.59 %
		–	✓	395	124	120	673.06	12.28 %
		✓	–	110	128	126	561.70	12.45 %
		✓	✓	395	128	124	688.15	9.84 %
360	180	–	–	190	124	124	551.80	13.15 %
		–	✓	375	108	108	595.82	16.56 %
		✓	–	190	124	124	680.23	10.74 %
		✓	✓	375	122	120	670.63	8.75 %

Table 7.7: Results from schedule concatenation using the schedules made with model 2 in Table 7.5. The use of location mapping and prioritized activities are indicated by the columns M and P, respectively. The column ”units” indicates the size of the building blocks used to make the concatenated schedules.

$S_{\max}$	Units	P	Time [min]	n	q	q_w	Extra	Idle time
180	90	–	$5.04 + 45$	64	64	359.08	3	9.24 %
		✓	$5.06 + 45$	64	64	406.25	1	10.29 %
360	90	–	$15.04 + 85$	128	122	619.33	0	10.93 %
		✓	$15.04 + 85$	128	124	673.13	0	11.13 %
360	180	–	$90.50 + 125$	128	124	627.08	0	7.25 %
		✓	$51.49 + 125$	120	118	687.58	0	9.46 %

Table 7.8: Results from schedule concatenation using the inexact schedules produced by the greedy scheduling heuristic in Table 7.6. The use of prioritized activities is indicated by the column P. The column ”units” indicates the size of the building blocks used to make the concatenated schedules. The column ”extra” indicates how many additional activities the heuristic adds after the concatenation.

apply the extra heuristic step. The column "extra" indicates how many additional activities were added. We see that we can only add additional activities to the schedules of 180 minutes, and not the 360-minute ones. Thus, the extra time spent adding additional activities is wasteful for long schedules. Nevertheless, the schedules from the concatenated heuristic generally have a higher q and q_w, as well as less idle time, than the schedules made purely from the heuristic. This indicates that the heuristic's best use is to quickly create short schedules, which can be concatenated into longer ones.

Compared to the results in Table 7.7, the results from the concatenated heuristic are comparable to the best schedules from model 2, both when it comes to the number of included activities q, the prioritized activity sum q_w, and the vehicle idle time. The additional strength of the heuristic compared to the other methods is the efficiency, especially when disregarding the unnecessary time spent trying to add additional activities in the end. Examples of two of the best schedules considering q_w and idle time are shown in Figure 7.14. The schedule on top is made with the concatenated heuristic by merging two 180-minute schedules, and one on the bottom is created with model 2 and the location mapping by concatenating four 90-minute schedules. The schedules both have a high q_w and a low idle time of around $9 - 10\,\%$. The concatenated heuristic schedule takes approximately 50 minutes to create, while the other one takes 395 minutes.

Generally, the methods based on the location mapping and the greedy scheduling heuristic, both with and without concatenation, outperform the schedules purely based on model 2 with the old activity definition. The addition of the new activity type does not seem to produce schedules of worse quality, even though the movement of the vehicles is more restricted. The addition of the precedence relations on the delivery activities after scheduling never caused any schedules to surpass $S_{\max}$. This can, of course, be a coincidence for the MRP instance we have chosen to solve and should be tested on several different problems. For efficient scheduling, the simulation study shows that the concatenated heuristic produces the best results fastest by making short schedules and merging them.

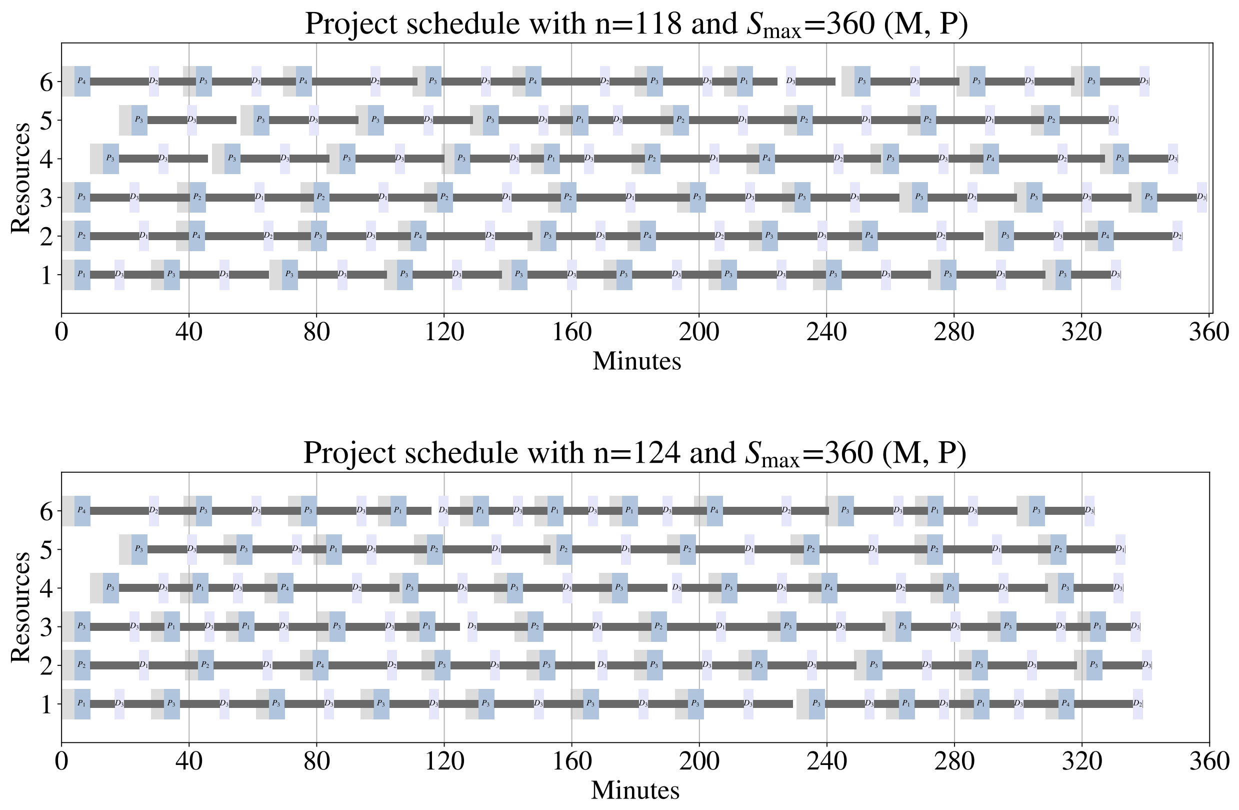


Figure 7.14: The best schedules of 360 minutes created by the schedule concatenation, considering q_w and idle time. *Top*: Schedule created by concatenating two 180-minute schedules from the heuristic with priorities. *Bottom:* Schedule created by concatenating four 90-minute schedules from model 2 with the location mapping and priorities.

Chapter 8

Case study: Skanska construction site

The simulation study in Chapter 7 showed promising results for the concatenation of schedules from the greedy scheduling heuristic with the location mapping. This section employs these methods on data from a real construction site to make 6-hour schedules with and without priorities. The inexact schedules are compared to approximate schedules made with model 2, and the practicability and quality of the results are discussed. The computations are completed on NTNU's computation server Markov using 10 threads.

8.1 Data

The data we utilize in this chapter originates from a Skanska construction site near Hønefoss, Norway. An aerial view of the site is shown in Figure 8.1. When solving the Mass Relocation Problem at the construction site, the previously discussed assumptions from Chapters 2 and 4 still hold. Thus, we assume we have a fleet of K homogeneous dump trucks to complete as many pickup and delivery activities as possible within 360 minutes. We also assume that there are no one-way roads or traffic congestion on the construction site and that there is only one possible travel path between each pickup and delivery location.

With the coordinates of the roads and the points of interest, we model

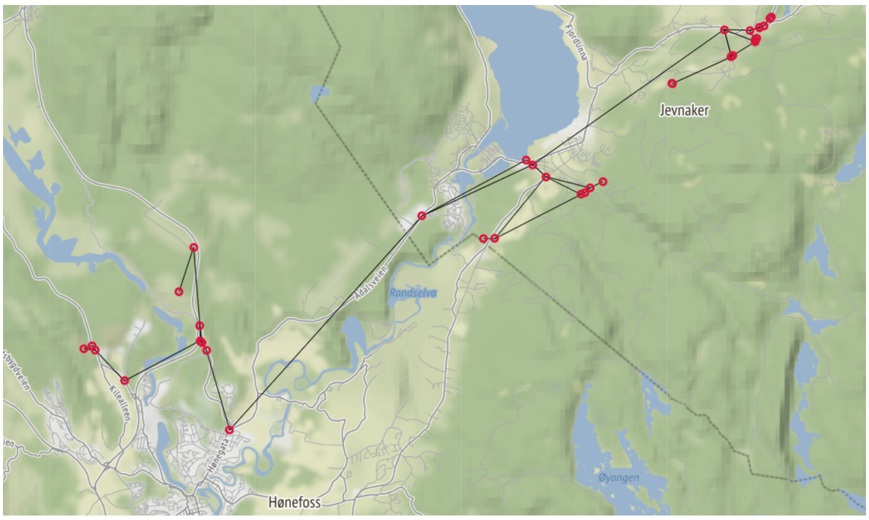


Figure 8.1: An aerial view of the Skanska construction site near Hønefoss, Norway, where the data we use is collected. The red marks indicate the points of interest: pickup locations, delivery locations, and road intersections.

the construction site as a graph, shown in Figure 8.2. The nodes represent all pickup locations, delivery locations, and road intersections on the construction site. When solving the MRP, we only consider the active pickup and delivery locations. The active pickup locations are recognized as those with excavators present to dig up mass, and the active delivery locations as those in the proximity of the active pickups. The current construction site includes six active pickup and seven active delivery locations. These are marked by red in Figure 8.2.

The dump trucks' travel times along each road on the construction site are automatically logged during their workday. Each distance thus has a set of historical durations that can be used to derive an average travel time. The data does not differentiate between the direction of travel, and therefore, the travel times will be symmetric. The automatic logging also does not take breaks or unexpected stops into account. Hence, the travel times include outliers which are often considerably larger than the other values. As including these would give a false impression of the average travel time, we remove the most prominent outliers using the interquartile range (IQR). The IQR is defined as the difference between the third and the first quartile of the data:

$$\mathrm{IQR} = \mathrm{Q3} - \mathrm{Q1}, \tag{8.1}$$

where Q1 and Q3 denote the first and third quartiles, respectively. We use the interquartile range to remove all travel times larger than the third quartile plus the IQR:

$$\mathrm{Q3} + \mathrm{IQR}. \tag{8.2}$$

After removing the outliers, we take the average of the remaining times to obtain a single travel time estimate for each road. Roads missing travel data are removed. As a consequence, two of the delivery locations are also removed as the roads leading into them are missing.

Since our model only allows for one path between the pickup and delivery locations, we calculate the shortest distance between each location with Dijkstra's algorithm. We are left with the simplified construction site in Figure 8.3. Figure 8.3(a) shows the resulting six pickup locations and five delivery locations where we want to solve the Mass Relocation Problem. Figure 8.3(b) shows the calculated travel distances between the locations. In the future, more thought should be put into the data preprocessing to obtain a more robust modelling of the construction site. In this preliminary effort at investigating the usefulness of the scheduling methods, using the average travel times and the shortest paths between the locations is enough to get a sense of the effectiveness of the methods.

To use the greedy scheduling heuristic, we need to define a location mapping. In a realistic scenario, people with insight into the construction process would choose a mapping that considers both the different mass types, the construction site topography, and the travel times between the locations. Lacking this information, we map the pickup and delivery loca-

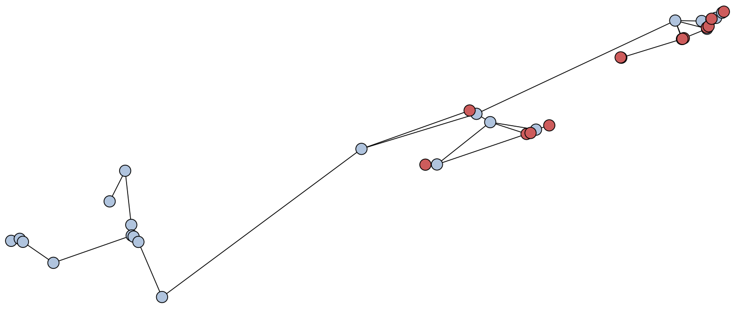
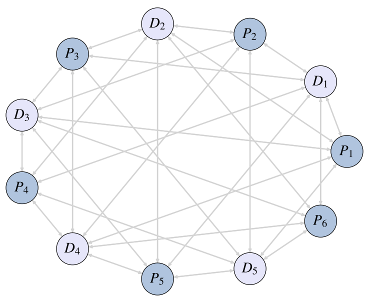
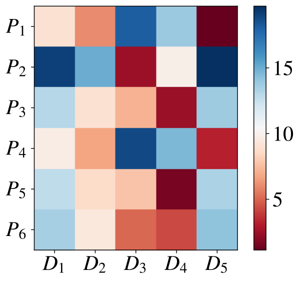


Figure 8.2: The Skanska construction site from Figure 8.1 illustrated as a graph. Nodes represent pickup locations, delivery locations, and road intersections. The edges indicate the network of roads. The red nodes represent the active pickup and delivery locations.



(a) Active pickup and delivery locations and driving patterns between them.



(b) Symmetric travel times between the pickup and delivery locations.

Figure 8.3: Simplified construction site with six pickup and five delivery locations. The only legal travel paths between the locations are the shortest paths found by Dijkstra's algorithm.

P_1	P_2	P_3	P_4	P_5	P_6
D_2	D_4	D_1	D_2	D_5	D_3

Table 8.1: Location mapping for the construction site. Each pickup location is paired with a delivery location. Both pickup locations P_1 and P_4 are mapped to delivery location D_2.

tions as shown in Table 8.1. The pairing is chosen such as to avoid very short or very long activity durations, as these might not be beneficial for the workflow. Since we have six pickup and five delivery locations, two pickup locations are mapped to the same delivery location. We also lack estimates of the pickup and delivery durations. Thus, as in the other sections, we assume that a pickup activity takes 5 minutes and a delivery activity takes 3 minutes. The mass preparation time at a pickup location is assumed to take 4 minutes. We use these numbers to solve the MRP on the construction site with $K = 7$ vehicles.

8.2 Schedules without priorities

The results from solving the Mass Relocation Problem using data from the real construction site without priorities are presented in Table 8.2. The column "type" refers to how the schedules are made; E refers to schedules made with model 2, H to schedules made with the heuristic, and C to schedules made by concatenation. The remaining columns are explained in Section 7.3. We choose to only concatenate 90-minute schedules, as these can be made quicker than the 180-minute ones, and we try to keep the schedule-making below 3 hours for all methods. Thus, the 6-hour approximate schedules made with model 2 are run for 180 minutes, while the 90-minute approximate schedules used for the concatenation are run for 90 minutes. This is implemented for the schedules both with and without the location mapping.

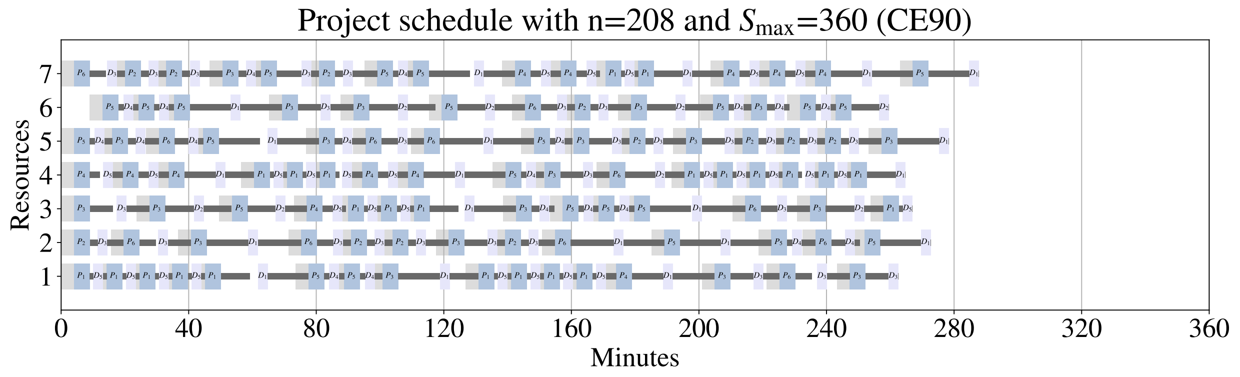
For the heuristic, a time limit of $\tau = 20\,\text{min}$ is chosen for making sched-

Type	Units	M	Time [min]	n	q	Idle time
E	–	–	180	420	124	51.48 %
CE	90	–	110	208	208	26.41 %
E	–	✓	180	516	194	15.02 %
CE	90	✓	110	208	204	13.30 %
H	–	✓	171.36	–	182	17.50 %
CH	90	✓	147.63	216	212	5.40 %

Table 8.2: Results of 360-minute schedules created with model 2 and the greedy scheduling heuristic, with and without concatenation. The column "type" indicates if the schedules are made with model 2 (E), the heuristic (H), or concatenation (C). The column "units" indicates the size of the building blocks used to make the concatenated schedules, and M represents if the schedules are made with or without the location mapping.

ules of both 360 and 90 minutes. The schedule concatenation includes left-shifting the schedules with model 1 with objective (6.7) and the additional constraint (6.8) for five minutes. This is performed both prior to concatenation and for the intermediate schedules during the process, thus adding 20 minutes to the total time. When concatenating the heuristic solutions, we do not attempt to add additional activities into the schedules after concatenation, as this did not work for the long schedules in the simulation study.

The results confirm the findings from the simulation study, namely that the schedules made with the concatenated scheduling heuristic outperform the others, both when it comes to the number of included activities and idle time. The concatenated heuristic schedule includes 212 activities and an idle time of just 5.40 %. In contrast, the approximate schedule without the location mapping includes only 124 activities and has an idle time of 51.48 %. The concatenated schedule without the location mapping comes closest to including as many activities as the concatenated heuristic but has a high vehicle idle time. The two schedules are compared in Figure 8.4.



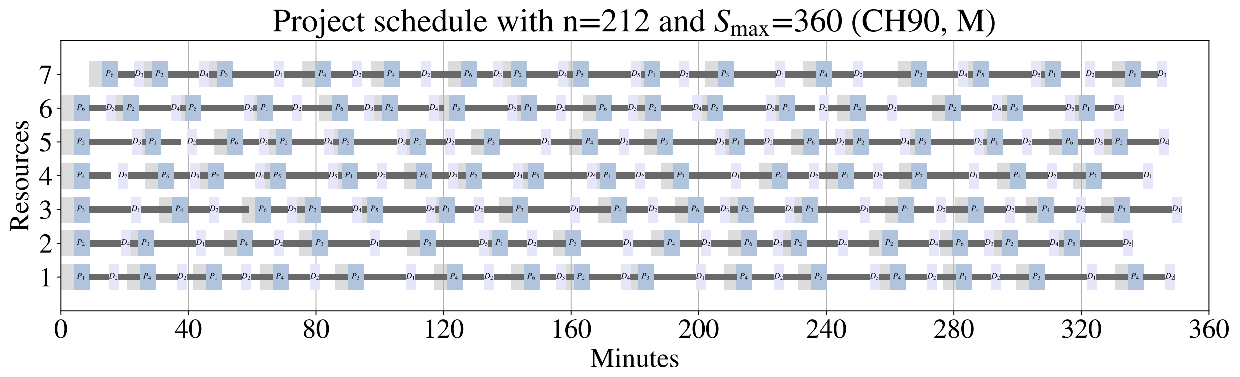


Figure 8.4: Schedules of 360 minutes. *Top:* Schedule created by concatenating four 90-minute schedules made with model 2 without the location mapping and with a cut-off time of 1.5 hours. *Bottom:* Schedule created by the concatenated greedy scheduling heuristic with $\tau = 20\,\text{min}$ and unit schedules of 90 minutes.

We see that while the concatenated heuristic manages to fill almost the entire 6-hour schedule with activities, the schedule made with model 2 includes much idle time for the vehicles. The idle time appears towards the end of the schedule because of the left-shifting of the activities after concatenation. It illustrates that there is more than enough room for more activities. Model 2, without the location mapping, should theoretically be able to find better schedules than the heuristic as it does not restrict the movement of the dump trucks. Still, it is unable to do so because of the computational burden associated with solving large problem instances. As model 2 simply tries to maximize q, it favors the locations with short travel times. This is why it can fit the activities much tighter than the concatenated heuristic. The heuristic has the advantage of alternating between the activities, which is preferable from a practical point of view.

The schedules made with the location mapping generally produce better results than those made without considering a combination of included activities and idle time, even though we have reduced the solution space considerably. This indicates that problem size and complexity are more decisive to the solution quality than retaining complete travel flexibility. The concatenated schedules also perform better than their non-concatenated counterparts, implying that utilizing the computation time to find good short schedules and merging them is better than scheduling all activities at once.

One of the disadvantages of the heuristic is the loss of precedence relations between the delivery activities, as discussed in Section 6.3. This can become an issue in short schedules. An example is shown in Figure 8.5 for the 90-minute schedule from the heuristic used in the concatenation. The figure shows that several delivery activities exceed the fixed scheduling horizon $S_{\max}$ when precedence relations between activities at location D_2 are enforced. When merging the schedule with itself into a longer one and enforcing the precedence relations, enough slack between the activities is present to avoid the activities exceeding the scheduling horizon, as shown in the bottom schedule in Figure 8.4.

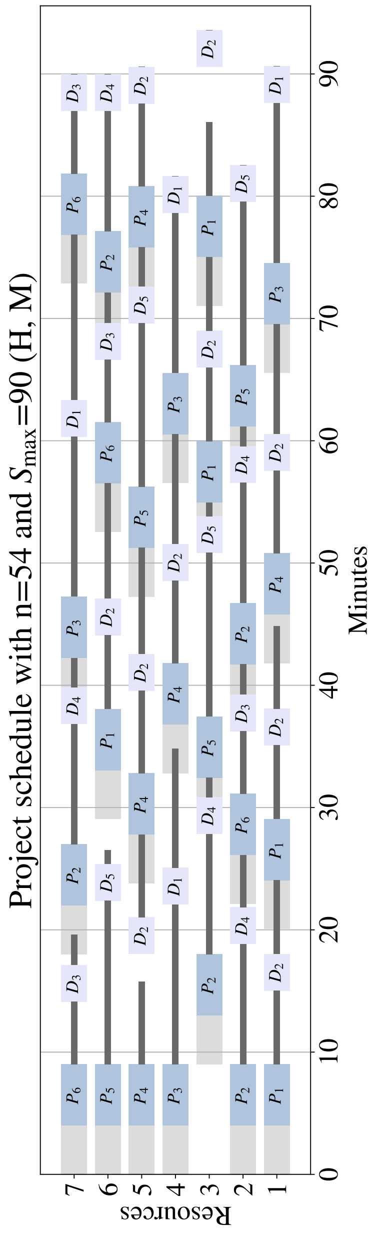


Figure 8.5: Example of a 90-minute schedule from the greedy scheduling heuristic made with $\tau = 20$ min. Imposing precedence relations between the delivery activities at D_2 causes the schedule's length to exceed $S_{\max}$.

8.3 Schedules with priorities

To make schedules with priorities, we need information about the total amount of mass to be moved from and to the pickup and delivery locations. As we currently have no such data from the construction site, we construct simulated long-term plans. The number of activities at each location is determined by randomly drawing numbers from the set $\{10, 20, \ldots, 100\}$ with a discrete uniform distribution. Similarly, the due dates are chosen by drawing numbers from the interval $[5, 30]$ with equal probability. The resulting plans are presented in Tables 8.3 and 8.4. For the activities to get approximately the same ordering both with and without the location mapping, the same activity demands and due dates are chosen for the pickup and delivery locations in the two plans.

The results from solving the Mass Relocation Problem on the real construction site with prioritized activities are presented in Table 8.5. The methods are implemented as in the last section, except with the objective of maximizing q_w instead of q. The variable q_w, defined in Section 4.4.1, represents the prioritized sum of the activities included in the schedules. When making the 90-minute schedules used in the concatenation, four different schedules must be created as each schedule needs to update its priorities based on the previous schedule's included activities. Thus, to keep the computational time low, only 30 minutes are used to create each individual 90-minute schedule with model 2. For the creation of the 90-minute schedules with the heuristic, the time limit $\tau = 5\,\text{min}$ is chosen.

Similar to what was observed in the simulation study, the results show that schedules made with model 2 without the location mapping can fit more activities when maximizing q_w than q. For the remaining schedules, the addition of the priorities decreases q, which is natural. For the heuristic, the addition of priorities allows the 90-minute schedules used in the concatenation to be made in a few seconds, as all activity types can be proven infeasible to add quickly. This makes the concatenated heuristic

Location	Activities	Due date
P_1	70	8
P_2	50	9
P_3	70	26
P_4	20	6
P_5	90	22
P_6	10	15
D_1	70	26
D_2	90	7
D_3	10	15
D_4	50	9
D_5	90	22

Table 8.3: Randomly simulated plan for how many activities we have to complete at each location on the Skanska construction site within a due date, without the location mapping. The plan is used to create priorities for the activities.

Location	Duration [min]	Rounds	Due date
P_1-D_2	14.03	70	8
P_2-D_4	17.84	50	9
P_3-D_1	21.08	70	26
P_4-D_2	14.76	20	6
P_5-D_5	21.38	90	22
P_6-D_3	13.11	10	15

Table 8.4: Randomly simulated plan for how many activities we have to complete at each location on the Skanska construction site within a due date, with the location mapping. The durations indicate how long it takes to complete an activity at a location. The plan is used to create priorities for the activities.

Type	Units	M	Time [min]	n	q	q_w	Idle time
E	–	–	180	420	126	768.42	47.92 %
CE	90	–	155	220	220	1161.87	28.30 %
E	–	✓	180	442	182	1126.30	18.74 %
CE	90	✓	155	198	196	1208.66	11.04 %
H	–	✓	170.54	–	182	1237.95	17.29 %
CH	90	✓	15.36	202	198	1239.23	12.04 %

Table 8.5: Results of prioritized 360-minute schedules created with model 2 and the greedy scheduling heuristic, with and without concatenation. The column ”type” indicates if the schedules are made with model 2 (E), the heuristic (H), or concatenation (C). The column ”units” indicates the size of the building blocks used to make the concatenated schedules, and M represents if the schedules are made with or without location mapping.

the fastest scheduling method of them all.

The table shows that the concatenated schedule from model 2 without the location mapping can do the most activities but has a low q_w compared to other schedules. It also has a high vehicle idle time. The schedules from the concatenated heuristic and model 2 made with the location mapping perform the best when it comes to both q_w and idle time. The two schedules are shown in Figure 8.6. The tie-breaker between them is the time usage. With the addition of priorities, the concatenated heuristic produces the 360-minute schedule in only 15 minutes.

The addition of priorities generally makes all methods converge faster and provides a certain control over which activities are included in the schedules. Priorities should thus be added to the models to ensure a reasonable number of each activity type is completed during a workday and to make the optimization easier by removing symmetries from the programs. We conclude that the concatenated heuristic is the most successful in creating feasible schedules with low idle time, both with and without priorities. Because of the quick scheduling, the method is easy to utilize on a construction site for day-to-day scheduling.

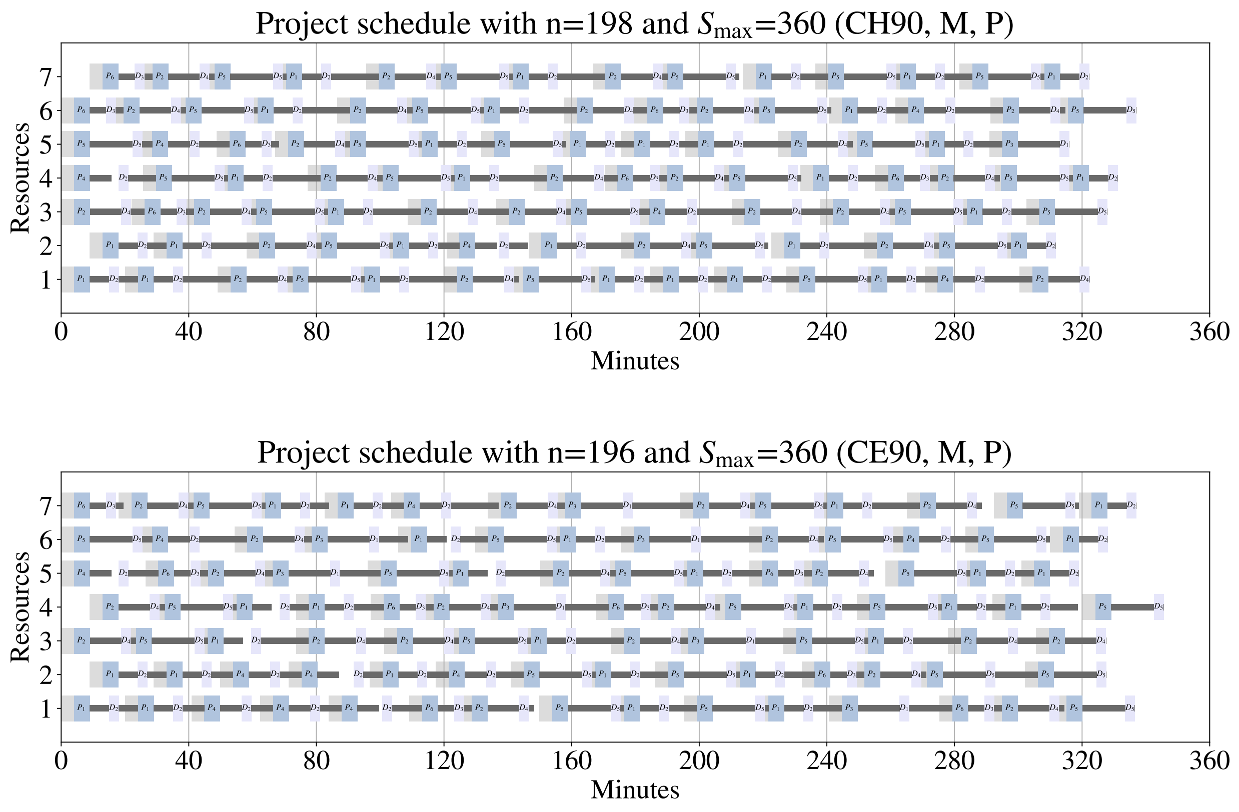


Figure 8.6: Prioritized schedules of 360 minutes. *Top:* Schedule created by concatenating four 90-minute schedules made with the greedy scheduling heuristic with $\tau = 5\,\text{min}$. *Bottom:* Schedule created by concatenating four 90-minute schedules from model 2 with the location mapping, with a time limit of 30 minutes for each unit schedule.

CHAPTER 9

CLOSING REMARKS

Section 7.1 demonstrated the inefficiency and computational complexity of solving the Mass Relocation Problem exactly with the RCPSP formulation. The intractability of the problem and the induced symmetries make using exact solution methods based on the branch-and-bound algorithm challenging. The introduction of the inexact methods in Chapter 6 allowed us to explore practical scheduling algorithms constructed to quickly build feasible schedules with many activities and little idle time, at the cost of losing optimality.

When creating 6-hour schedules in both the simulation study and the case study, schedules produced by the greedy scheduling heuristic with the location mapping and the schedule concatenation performed best considering a combination of q, q_w, idle time, and computation time. The advantage of the greedy scheduling heuristic, compared to model 2, is the complete control over which activities are added to the schedules. By increasing the time limit τ, the heuristic can create schedules of any length, though at the cost of increased computation time. The method performs best when used to create short schedules, such that the infeasibility of adding additional activities can be proved before τ.

The greedy element of the scheduling algorithm makes it efficient but also allows for the possibility that the addition of individual activities does not work towards the ultimate goal of increasing q and q_w. Another disadvantage is being locked to the location mapping, which can be detrimental to the schedule quality if not chosen with care. As previously discussed, we might also lose the precedence relations between the delivery activities when using the location mapping. As the number of deliveries we can do simultaneously at a location depends on the layout of the construction sites, the necessity of adding the precedence relations in a post-processing step has to be considered based on each individual site.

Given the assumptions made in Chapters 2 and 4, the schedules produced in the case study could theoretically be used to route vehicles for 6 hours on the Skanska construction site. No activities are processed at the same locations simultaneously, and the vehicles respect the network of roads on the site. Based on the estimation that dump trucks can spend up to 40 % of the workday idle, utilizing the schedules we have produced would decrease the inactive time to $5 - 10\,\%$. This reduction can reduce emissions, lessen unnecessary fuel usage, and contribute to finishing road construction projects more efficiently.

However, the schedules we have made do not consider road capacities, possible delays, or breaks. The lack of robustness and traffic modelling makes the implementation of the schedules infeasible in real-world applications. Thus, further study of the Mass Relocation Problem should focus on efficiently incorporating these complicating elements into the model. The most imminent change to the scheduling algorithms should be including uncertainty in all problem parameters. This includes the travel times, activity durations, and mass preparation times. Promising work on including robust optimization into the MRP formulation was conducted by Johnsen [8] by allowing for a certain degree of conservativeness in the coefficients when modelling the problem. Similar approaches could be implemented into the RCPSP formulation.

To more accurately model the Mass Relocation Problem, some of our imposed assumptions should also be reconsidered. In a real-world scenario, we cannot assume that all roads have infinite capacity and that no traffic congestion will occur. Including uncertainty in the travel times based on the historical data can help account for traffic delays without explicitly modelling the traffic. Using asymmetric travel times can also provide better travel estimates, as these would take into account the topography of the construction sites and the additional weight of the vehicles when carrying mass.

It is also unrealistic that the vehicles can start and stop wherever they like. When making schedules for a complete workday, imposing that the dump trucks have to start where they left off the previous day and allowing them to end wherever they like might be a solution. For half-day schedules, we have to impose both start and end locations for the possibility of continuity between schedules. Enforcing start and end locations in model 1 and model 2 can be implemented by changing the travel graph and adding some additional constraints. The scheduling algorithm should also consider the vehicles' different capacities, speeds, and abilities to handle different types of mass. To allow for this, we would have to reintroduce the multiple modes in the RCPSP formulation and allow the flow-variable f_{ij} to take on integer values.

Considering the complexity of the Mass Relocation Problem, it is doubtful that an exact mathematical formulation can be solved to optimality efficiently enough to create schedules for real-world use. Thus, going forward, the focus should be on developing efficient heuristics and inexact methods which take advantage of the specific problem formulation. Making methods that are not reliable on a location mapping and which are not greedy in nature can be beneficial for maximizing the movement of the mass and utilizing the dump trucks as efficiently as possible. The use of machine learning techniques and artificial intelligence for routing the dump trucks should also be explored.

Even though the methods we have developed in this thesis solve a simplified version of the MRP, they yield valuable insight into the problem and can be used as a benchmark for more sophisticated scheduling approaches in the future. Through the work of this thesis, we have also shown that approaching the MRP from a scheduling perspective by the Resource-Constrained Project Scheduling Problem is possible.

Bibliography

[1] Kenneth Løvold Rødseth et al. "Effektivitet og produktivitet i norsk veibygging 2007-2016". *Concept-rapport nr. 57* (2019).

[2] Byggenæringens Landsforening. "Bygg- og anleggssektorens klimagassutslipp". Unpublished. 2019.

[3] Skanska. *Skanska vil kutte utslipp med kunstig intelligens.* https://www.skanska.no/hvem-vi-er/media/aktuelt/skanska-vil-kutte-utslipp-med-kunstig-intelligense/. [Online; accessed 06.12.2021].

[4] Erik Leuven Demeulemeester and Willy S. Herroelen. *Project Scheduling - A Research Handbook.* 1st ed. Springer, 2002. ISBN: 978-0-306-48142-0.

[5] J. Zhou et al. "A review of methods and algorithms for optimizing construction scheduling". *Journal of the Operational Research Society* 64.8 (2013), pp. 1091–2013.

[6] Peter Brucker and Sigrid Knust. *Complex Scheduling.* 2nd ed. Springer, 2012. ISBN: 978-3-642-23929-8.

[7] Sintef. *Datadrevet anleggsplass.* https://www.sintef.no/prosjekter/2020/datadrevet-anleggsplass/. [Online; accessed 06.12.2021].

[8] Ella Frederika Johnsen. "The Mass Relocation Problem with Uncertainty". MA thesis. Norwegian University of Science and Technology, 2021.

[9] G. Clarke and J. W. Wright. "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". *Operations Research* 12.4 (1964), pp. 568–581.

[10] Christian Artigues, Sophie Demassey, and Emmanuel Néron, eds. *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications.* Wiley-ISTE, 2008. ISBN: 978-1-848-21034-9.

[11] Linet Özdamar and Gunduz Ulusoy. "A Survey on the Resource-Constrained Project Scheduling Problem". *IIE Transactions* 27.5 (1995), pp. 574–586.

[12] Sönke Hartmann and Dirk Briskorn. "A survey of variants and extensions of the resource-constrained project scheduling problem". *European Journal of Operational Research* 207.1 (2010), pp. 1–14.

[13] Sönke Hartmann and Dirk Briskorn. "An updated survey of variants and extensions of the resource-constrained project scheduling problem". *European Journal of Operational Research* 297.1 (2021), pp. 1–14.

[14] Gurobi Optimization. *Gurobi Optimizer*. https://www.gurobi.com/products/gurobi-optimizer/. [Online; accessed 06.12.2021].

[15] J.F. Benders. "Partitioning procedures for solving mixed-variables programming problems". *Numerische Mathematik* 4 (1962), pp. 238–252.

[16] Arno Sprecher, Rainer Kolisch, and Andreas Drexl. "Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem". *European Journal of Operational Research* 80.1 (1995), pp. 94–102.

[17] Oumar Koné et al. "Event-based MILP models for resource-constrained project scheduling problems". *Computers & Operations Research* 38.1 (2011), pp. 3–13.

[18] A. Pritsker, Lawrence J. Waiters, and P. Wolfe. "Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach". *Management Science* 16.1 (1969), pp. 93–108.

[19] Christian Artigues, Philippe Michelon, and Stéphane Reusser. “Insertion techniques for static and dynamic resource-constrained project scheduling”. *European Journal of Operational Research* 149.2 (2003), pp. 249–267.

[20] Aristide Mingozzi et al. “An Exact Algorithm for the Resource Constrained Project Scheduling Problem Based on a New Mathematical Formulation”. *Management Science* 44.5 (1996), pp. 595–741.

[21] Ramón Alvarez-Valdés Olaguíbel and JoséManuel Tamarit Goerlich. “The project scheduling polyhedron: Dimension, facets and lifting theorems”. *European Journal of Operational Research* 67.2 (1993), pp. 204–220.

[22] F. Brian Talbot. “Resource-Constrained Project Scheduling with Time-Resource Tradeoffs: The Nonpreemptive Case”. *Management Science* 28.10 (1982), pp. 1197–1210.

[23] Klaus Neumann and Christoph Schwindt. “Activity-on-node networks with minimal and maximal time lags and their application to make-to-order production”. *Operations-Research-Spektrum* 19.3 (1997), pp. 205–217.

[24] Alain Quilliot and Hélène Toussaint. “Resource Constrained Project Scheduling with Transportation Delays”. *IFAC Proceedings Volumes* 45.6 (2012), pp. 1481–1486.

[25] Jens Poppenborg and Sigrid Knust. “A flow-based tabu search algorithm for the RCPSP with transfer times”. *OR Spectrum* 38.2 (2016), pp. 305–334.

[26] Philippe Lacomme et al. “Integration of routing into a resource-constrained project scheduling problem”. *EURO Journal on Computational Optimization* 7.4 (2019), pp. 421–464.

[27] Ramteen Sioshansi and Antonio J. Conejo. *Optimization in Engineering - Models and Algorithms*. Springer, 2017. ISBN: 978-3-319-56769-3.

[28] François Margot. "Symmetry in Integer Linear Programming". *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art.* Ed. by Michael Jünger et al. Springer, 2010, pp. 647–686. ISBN: 978-3-540-68279-0.

[29] Steven Edwards et al. "Symmetry breaking of identical projects in the high-multiplicity RCPSP/max". *Journal of the Operational Research Society* 72.2 (2019), pp. 1–22.

[30] Gurobi Optimization. *Mixed-Integer Programming (MIP) – A Primer on the Basics.* https://www.gurobi.com/resource/mip-basics/. [Online; accessed 06.12.2021].

[31] Ragheb Rahmaniani et al. "The Benders decomposition algorithm: A literature review". *European Journal of Operational Research* 259.3 (2017), pp. 801–817.

[32] Lucas Agussurja, Akshat Kumar, and Hoong Chuin Lau. "Resource-Constrained Scheduling for Maritime Traffic Management". *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (2018).

[33] Haitao Li. "Benders Decomposition Approach for Project Scheduling with Multi-Purpose Resources" (2015).

[34] S.E. Terblanche and J.H. van Vuuren. *Benders decomposition of the resource constrained project scheduling problem.* http://www.optimization-online.org/DB_HTML/2017/08/6150.html?fbclid=IwAR0bkhG4RrXhSXtTQyvxl-qMcIdOdc3CL7KwfH9cFojgiCOyyn5n5TJSaH0. Technical Report FABWI-N-RA-2017-530, Centre for Business Mathematics & Informatics, North-West University, South Africa. [Online; accessed 04.11.2021]. 2017.

[35] Pierre Bonami, Domenico Salvagnin, and Andrea Tramontani. "Implementing Automatic Benders Decomposition in a Modern MIP Solver". *Integer Programming and Combinatorial Optimization.* Ed.

by Daniel Bienstock and Giacomo Zambelli. Springer, 2020, pp. 78–90.

[36] Paul A. Rubin. *Benders Decomposition Then and Now.* https://orinanobworld.blogspot.com/2011/10/benders-decomposition-then-and-now.html. [Online; accessed 05.11.2021]. Oct. 2011.

[37] T.L. Magnanti and Richard Wong. "Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria". *Operations Research* 29.3 (1981), pp. 464–484.

[38] Matteo Fischetti, Andrea Lodi, and Andrea Tramontani. "On the separation of disjunctive cuts". *Mathematical Programming* 128 (2011), pp. 205–230.

[39] Gianni Codato and Matteo Fischetti. "Combinatorial Benders' Cuts for Mixed-Integer Linear Programming". *Operations Research* 54.4 (2006), pp. 756–766.

[40] Curtiss Luong. "An Examination of Benders' Decomposition Approaches in Large-scale Healthcare Optimization Problems". MA thesis. University of Toronto, Mechanical and Industrial Engineering Department, 2015.

[41] Z. Taşkın and Mucahit Cevik. "Combinatorial Benders cuts for decomposing IMRT fluence maps using rectangular apertures". *Computers & Operations Research* 40.9 (2013).

[42] Matteo Fischetti, Ivana Ljubić, and Markus Sinnl. "Benders decomposition without separability: A computational study for capacitated facility location problems". *European Journal of Operational Research* 253.3 (2016), pp. 557–569.